

**T E X T , S P E E C H
A N D L A N G U A G E
T E C H N O L O G Y**

ADVANCES IN OPEN DOMAIN QUESTION ANSWERING

Edited by Tomek Strzalkowski and Sanda Harabagiu

“centered caps” type=slogan> War is
peace</q> <q rend=“centered caps”
type=slogan>Freedom is slavery</q> <q
rend=“centered caps” type=slogan>Ignorance
strength.</q>

</p>

<note place=foot id=N1>



Newspeak was the official
language of Oceania

Advances in Open Domain Question Answering

Text, Speech and Language Technology

VOLUME 32

Series Editors

Nancy Ide, *Vassar College, New York*
Jean Véronis, *Université de Provence and CNRS, France*

Editorial Board

Harald Baayen, *Max Planck Institute for Psycholinguistics, The Netherlands*
Kenneth W. Church, *AT & T Bell Labs, New Jersey, USA*
Judith Klavans, *Columbia University, New York, USA*
David T. Barnard, *University of Regina, Canada*
Dan Tufis, *Romanian Academy of Sciences, Romania*
Joaquim Llisterri, *Universitat Autònoma de Barcelona, Spain*
Stig Johansson, *University of Oslo, Norway*
Joseph Mariani, *LIMSI-CNRS, France*

The titles published in this series are listed on www.springer.com

Advances in Open Domain Question Answering

Edited by

Tomek Strzalkowski

University at Albany, SUNY, NY, U.S.A.

and

Sanda Harabagiu

University of Texas at Dallas, TX, U.S.A.



A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN 978-1-4020-4745-9 (PB)
ISBN 978-1-4020-4744-2 (HB)
ISBN 978-1-4020-4746-6 (e-book)

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springer.com

Printed on acid-free paper

All Rights Reserved

© 2008 Springer

No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, microfilming, recording or otherwise, without written permission from the Publisher, with the exception of any material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work.

CONTENTS

Contributors	ix
Preface	xv
Introduction	xxiii
PART 1: Approaches to Question Answering	1
D. MOLDOVAN, M. PAŞCA AND M. SURDEANU SOME ADVANCED FEATURES OF LCC'S POWERANSWER	3
ABRAHAM ITTYCHERIAH A STATISTICAL APPROACH FOR OPEN DOMAIN QUESTION ANSWERING	35
JOSE L. VICEDO AND ANTONIO FERRÁNDEZ COREFERENCE IN Q&A	71
PART 2: Question Processing	97
SANDA M. HARABAGIU QUESTIONS AND INTENTIONS	99
TOMEK STRZALKOWSKI, SHARON SMALL, HILDA HARDY, PAUL KANTOR, WU MIN, SEAN RYAN, NOBUYUKI SHIMIZU, LIU TING, NINA WACHOLDER AND BORIS YAMROM QUESTION ANSWERING AS DIALOGUE WITH DATA	149

B. GRAU, O. FERRET, M. HURAUT-PLANTET, C. JACQUEMIN, L. MONCEAUX, I. ROBBA AND A. VILNAT COPING WITH ALTERNATE FORMULATIONS OF QUESTIONS AND ANSWERS	189
PART 3: Question Answering as Information Retrieval	227
L. GRUNFELD AND K.L. KWOK SENTENCE RANKING USING KEYWORDS AND META-KEYWORDS	229
CHARLES L. A. CLARKE, GORDON V. CORMACK, THOMAS R. LYNAM AND EGIDIO L. TERRA QUESTION ANSWERING BY PASSAGE SELECTION	259
DRAGOMIR R. RADEV, HONG QI, ZHIPING ZHENG, SASHA BLAIR-GOLDENSOHN, ZHU ZHANG, WEIGUO FAN AND JOHN PRAGER QUERY MODULATION FOR WEB-BASED QUESTION ANSWERING	285
PART 4: Answer Extraction	305
JOHN M. PRAGER, JENNIFER CHU-CARROLL, ERIC W. BROWN AND KRZYSZTOF CZUBA QUESTION ANSWERING BY PREDICTIVE ANNOTATION	307
ROHINI K. SRIHARI, WEI LI AND XIAOGE LI QUESTION ANSWERING SUPPORTED BY MULTIPLE LEVELS OF INFORMATION EXTRACTION	349
ABDESSAMAD ECHIHABI, ULF HERMJAKOB, EDUARD HOVY, DANIEL MARCU, ERIC MELZ AND DEEPAK RAVICHANDRAN HOW TO SELECT AN ANSWER STRING?	383
PART 5: Evaluating Question Answering Systems	407
ELLEN M. VOORHEES EVALUATING QUESTION ANSWERING SYSTEM PERFORMANCE	409
WILLIAM HERSH EVALUATING INTERACTIVE QUESTION ANSWERING	431
WILLIAM OGDEN, JAMES McDONALD, PHILIP BERNICK, ROGER CHADWICK HABITABILITY IN QUESTION-ANSWERING SYSTEMS	457
PART 6: Perspectives on Question Answering	475
STEVEN J. MAIORANO QUESTION ANSWERING: TECHNOLOGY FOR INTELLIGENCE ANALYSIS	477

ELLEN RILOFF, GIDEON S. MANN AND WILLIAM PHILLIPS REVERSE-ENGINEERING QUESTION/ANSWER COLLECTIONS FROM ORDINARY TEXT	505
MARK MAYBURY NEW DIRECTIONS IN QUESTION ANSWERING	533
INDEX	559

CONTRIBUTORS

1. **Philip Bernick**, Arizona State University, Box 870302, Tempe, AZ 85287, USA. Email: philip.bernick@asu.edu
2. **Sasha Blair-Goldensohn**, Columbia University, 1214 Amsterdam Avenue, New York, NY 10027, USA. Email: sashabg@cs.columbia.edu
3. **Eric W. Brown**, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA. Email: ewb@us.ibm.com
4. **Roger Chadwick**, Department of Psychology, New Mexico State University, PO Box 30001, Las Cruces, NM 88003-8001, USA. Email: rchadwic@nmsu.edu
5. **Jennifer Chu-Carroll**, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA. Email: jencc@us.ibm.com
6. **Charles L. A. Clarke**, School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada. Email: claclark@plg.uwaterloo.ca
7. **Gordon V. Cormack**, School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada. Email: gvcormac@plg.uwaterloo.ca
8. **Krzysztof Czuba**, Google, Inc., 1440 Broadway, 21st Floor, NY, New York 10018, USA. Email: kczuba@google.com
9. **Abdessamad Echihabi**, USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA. Email: echihabi@isi.edu

10. **Antonio Ferrández**, Departamento de Lenguajes y Sistemas Informáticos, University of Alicante, 03008 Alicante, Spain. Email: antonio@dlsi.ua.es
11. **Olivier Ferret**, LIMSI (CNRS), BP 133, 91 403 Orsay Cedex, France. Email: Olivier.Ferret@limsi.fr
12. **Brigitte Grau**, LIMSI (CNRS), BP 133, 91 403 Orsay Cedex, France. Email: Brigitte.Grau@limsi.fr
13. **Laszlo Grunfeld**, Queens College of CUNY, Flushing, NY 11367, USA. Email: grunfeld@ir.cs.qc.edu
14. **Sanda M. Harabagiu**, University of Texas at Dallas, Richardson, TX 75083, USA. Email: sanda@cs.utdallas.edu
15. **Hilda Hardy**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: hhardy@albany.edu
16. **Ulf Hermjakob**, USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA. Email: ulf@isi.edu
17. **William Hersh**, Oregon Health & Science University, BICC, 3181 SW Sam Jackson Park Rd., Portland, OR 97239, USA. Email: hersh@ohsu.edu
18. **Hong Qi**, 33F Citigroup Tower No. 33 Hua Yuan Shi Qiao Road, Lu Jia Zui Finance and Trade Zone, Shanghai, 200120, P.R.China. Email: hong.hqi@gmail.com
19. **Eduard Hovy**, USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA. Email: hovy@isi.edu
20. **Martine Hurault-Plantet**, LIMSI (CNRS), BP 133, 91 403 Orsay Cedex, France. Email: Martine.Hurault-Plantet@limsi.fr
21. **Abraham Ittycheriah**, IBM Thomas J. Watson Research Center, 1101 Kitchawan Road, Building 801, Yorktown Heights, NY 10598, USA. Email: abei@us.ibm.com
22. **Christian Jacquemin**, LIMSI (CNRS), BP 133, 91 403 Orsay Cedex, France. Email: Christian.Jacquemin@limsi.fr
23. **Paul Kantor**, Rutgers University, New Brunswick, NJ 08901, USA. Email: kantor@scils.rutgers.edu

24. **Kui-Lam Kwok**, Queens College of CUNY, Flushing, NY 11367, USA. Email: kwok@ir.cs.qc.edu
25. **Liu Ting**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: tingliu@cs.albany.edu
26. **Thomas R. Lynam**, School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada. Email: trlynam@plg.uwaterloo.ca
27. **Steven J. Maiorano**, ARDA AQUAINT Technical Steering Committee, LF-7, Washington, D.C. 20505, USA. Email: stevemai@mac.com
28. **Gideon S. Mann**, Computer Science Building, 140 Governors Drive, University of Massachusetts, Amherst, MA 01003, USA. Email: gideon.mann@gmail.com
29. **Daniel Marcu**, USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA. Email: marcu@isi.edu
30. **Mark Maybury**, Information Technology Division, The MITRE Corporation, Mail Stop K-325, 202 Burlington Road, Bedford, MA 01730, USA. Email: maybury@mitre.org
31. **James McDonald**, Department of Psychology, Box 3452, New Mexico State University, Las Cruces, NM 88003, USA. Email: jemcdon@nmsu.edu
32. **Eric Melz**, USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA. Email: melz@isi.edu
33. **Dan Moldovan**, Language Computer Corporation, Richardson, TX 75080, USA. Email: moldovan@languagecomputer.com
34. **Laura Monceaux**, LIMSI (CNRS), BP 133, 91 403 Orsay Cedex, France. Email: Laura.Monceaux@limsi.fr
35. **William Ogden**, Computing Research Laboratory, New Mexico State University, Box 30001/MSC 3CRL, Las Cruces, NM 88003, USA. Email: ogden@nmsu.edu
36. **Marius Paşa**, Google Inc., 1600 Amphitheatre Parkway, Mountain View, California 94043, USA. Email: mars@google.com
37. **William Phillips**, University of Utah, Salt Lake City, UT 84112, USA. Email: phillips@cs.utah.edu

38. **John Prager**, IBM T.J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598, USA. Email: jprager@us.ibm.com
39. **Dragomir R. Radev**, University of Michigan, Ann Arbor, MI 48109, USA. Email: radev@umich.edu
40. **Deepak Ravichandran**, USC Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292, USA. Email: ravichan@isi.edu
41. **Ellen Riloff**, University of Utah, Salt Lake City, UT 84112, USA. Email: riloff@cs.utah.edu
42. **Isabelle Robba**, LIMSI (CNRS), BP 133, 91 403 Orsay Cedex, France. Email: Isabelle.Robba@limsi.fr
43. **Sean Ryan**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: sryan@albany.edu
44. **Nobuyuki Shimizu**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: shimizu@cs.albany.edu
45. **Sharon Small**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: small@albany.edu
46. **Rohini K. Srihari**, State University of New York at Buffalo and Janya Inc., 1408 Sweet Home Road, Amherst, NY 14228, USA. Email: rohini@janyainc.com
47. **Tomek Strzalkowski**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: tomek@csc.albany.edu
48. **Mihai Surdeanu**, Language Computer Corporation, Richardson, TX 75080, USA. Email: mihai@languagecomputer.com
49. **Egidio L. Terra**, School of Computer Science, University of Waterloo, 200 University Avenue West, Waterloo, Ontario N2L 3G1, Canada. Email: elterra@plg.uwaterloo.ca
50. **Jose L. Vicedo**, Departamento de Lenguajes y Sistemas Informáticos, University of Alicante, 03008 Alicante, Spain. Email: vicedo@dlsi.ua.es
51. **Anne Vilnat**, LIMSI (CNRS), BP 133, 91 403 Orsay Cedex, France. Email: Anne.Vilnat@limsi.fr
52. **Ellen M. Voorhees**, National Institute of Standards and Technology, 100 Bureau Drive, STOP 8940, Gaithersburg, MD 20899, USA. Email: ellenv@nist.gov

53. **Nina Wacholder**, Rutgers University, New Brunswick, NJ 08901, USA. Email: wacholder@scils.rutgers.edu
54. **Wei Li**, Janya Inc., 1408 Sweet Home Road, Amherst, NY 14228, USA. Email: wei@janyainc.com
55. **Weiguo Fan**, Virginia Polytechnic Institute and State University 3007 Pamplin Hall, Blacksburg, VA 24061, USA. Email: wfan@vt.edu
56. **Yorick Wilks**, University of Sheffield, Regent Court 211, Portobello Street, Sheffield, S1 4DP, UK. Email: yorick@dcs.sheffield.ac.uk
57. **Wu Min**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: wumin@cs.albany.edu
58. **Xiaoge Li**, Janya Inc., 1408 Sweet Home Road, Amherst, NY 14228, USA. Email: xli@janyainc.com
59. **Boris Yamrom**, State University of New York at Albany, 1400 Washington Avenue, Albany, NY 12222, USA. Email: yamrom@optonline.org
60. **Zhiping Zheng**, 15322 NW Dane Lane, Portland, OR 97229, USA. Email: zheng@answerbus.com
61. **Zhu Zhang**, The University of Arizona, 1130 E. Helen Street, Tucson, AZ 85721, USA. Email: zhuzhang@u.arizona.edu

PREFACE

WHAT IS QUESTION ANSWERING?

Automated question answering—the ability of a machine to answer questions, simple or complex, posed in ordinary human language—is perhaps the most exciting technological development of the past few years. We are talking about only the past 6 or 7 years thus far, not long at all. Nonetheless, the expectations are already tremendous, reaching beyond the discipline itself, which is a subfield of Natural Language Processing, a branch of both Artificial Intelligence and Computer Science. This excitement is well justified because finding the right answer to a question often requires non-trivial knowledge and intelligence, and occasionally it calls for superhuman powers. Already the ancient Greeks had their oracles, which communicated with the gods and could foretell the future; the Celts in Northern Europe deferred to Druids in all weighty matters and disputes; and the Azande in sub-Saharan Africa inquired of the *benge* or the *iwa*, but only yes/no questions were allowed. Then, in 1900 Frank Baum created the Wizard of Oz.

The arrival of the computer age and artificial intelligence research in the 1970's has opened the possibility of building a question answering machine, a device with boundless capacity to store information and then retrieve it on demand. However, even simple fact (or information) retrieval, turned out to be a surprisingly hard problem unless one could *precisely* specify what information was needed. Doing so necessitated the use of awkward, unwieldy database access languages, such as SQL, which had limited expressive power and were hard to translate questions into. Early research into natural language processing was largely devoted to overcoming this linguistic barrier by developing algorithms which would convert human language questions into some formal representation that computers could deal with (Scha, 1982). In spite of these difficulties, the QA research in the 1960's and 1970's has resulted in a number of interesting prototypes including Baseball (Green, 1963), PARRY (Colby, 1971), Lunar (Woods *et al.*, 1972), Protosynthex (Simmons 1965),

QUALM (Lehnert, 1978), and PHLIQA1 (Bronnenberg *et al.*, 1980). Some of these systems, like Lunar, were built to interface with structured databases containing well-formed information in a narrow domain (e.g., lunar rock geology); other systems, like QUALM, answered questions from textual documents which needed to be understood first (i.e., reading comprehension). Still others, like PARRY, were primarily concerned with maintaining human-machine dialogue, with fact recall being only one, and not necessarily the most important aspect of the conversation.

However intriguing, these early systems were rapidly eclipsed by their own glaring inadequacies and impracticalities, and AI research has moved on to consider more fundamental questions of knowledge, meaning, and reasoning. It wasn't until nearly two decades later (an enormous gap in the computer age!) that QA research returned on the center stage. It did not happen overnight. The collapse of symbolic approaches to language understanding around the end of the 1980's has given rise to a wave of empirical research. In a radical departure from previous work, this research viewed language as an information medium, and thus considered all communication as a problem of information retrieval and exchange. The initial goals of the empiricist movement were as modest as they were practical. The aim was to produce robust automated systems that would do an adequate job at whatever they were designed for: document retrieval, named entity extraction, part-of-speech tagging, speech recognition, etc. A "perfect" performance was no longer expected and few claims were made that the empirical approaches had anything to do with how humans process information. Instead, the superior speed with which modern computers could perform simple and mundane tasks had been successfully harnessed to produce the results that eluded early AI work.

It took another decade for the empiricist movement to mature and gain confidence before we dared, once again, to take on question answering research. The technology landscape is very different now and the demand for question answering technology is real and growing. The major changes of the last decade and a half were of course the rise of the internet and the success of document retrieval, particularly on the web. Equally stunning, though far less popularized were the advances in named entity extraction (*Who? Where? When? How much?*), automated speech recognition, and human-computer dialogue. None of these technologies are anywhere near matching human linguistic capabilities; nonetheless, the most advanced systems became practically and commercially viable, e.g., BBN Identifinder (Miller *et al.*, 1999), Communicator (Walker *et al.*, 2000), AMITIÉS (Hardy *et al.*, 2004), and occasionally they became indispensable, e.g., *Google*TM. This practical dimension of otherwise still emerging technologies brought on a different kind of pressure, something researchers rarely reckoned with: customer demand.

The stage for QA return has thus been set. While information retrieval is workable, customers now demand a better tool. First, they want to cut time and effort involved in formulating effective queries, and then they want their results to be real answers—not the list of "relevant" links. They want to spend less time searching and more time thinking about what they found and using it for whatever purpose they started the search in the first place. They want to perform transactions quickly and efficiently; in other words, they want QA that is more efficient than

internet search, but at the same time is flexible, robust, and not fussy, just like Google. In its current incarnation QA is thus primarily viewed as an improvement of document retrieval: QA becomes *information* retrieval, as originally intended. In a traditional document retrieval system, the task of extracting the answer from the retrieved documents falls squarely upon the user, and it becomes a significant cognitive burden. QA technology aims to reduce this burden by following document retrieval with a series of advanced processing steps to locate and return the answer.

Today's automatic question-answering systems are revolutionizing the processing of textual information. By combining complex natural language processing techniques, sophisticated linguistic representations, and advanced machine learning methods, automatic question-answering systems can find exact answers to a wide variety of natural language questions in unstructured texts. The process of providing a brief and precise answer to a question is quite different from the task of information retrieval or information extraction, but it depends on both as important components. Furthermore, current progress in QA owes much to the development of an independent framework that goes beyond the simple combination of IR and IE techniques. Modern automatic QA systems normally consist of three separate components: a question processing module, a document processing module, and an answer extraction module. In question processing, a natural language question is transformed into a query which can be used to retrieve documents (using an IR module) that may contain answers to the question. Once the retrieval step is complete, the document processing component identifies specific text passages in retrieved documents where the answer is most likely to be found. Finally, the answer extraction component returns the actual text snippet that represents the exact answer to the user's question.

Research in open-domain QA was catalyzed by competitive evaluations conducted by the question answering track of the Text Retrieval Conference (TREC)¹, an annual event sponsored by the U.S. National Institute of Standards and Technology (NIST). Starting in 1999, the TREC QA evaluations initially focused on factoid (or fact-recall) questions, which could be answered by extracting phrase-length passages; for example: "*Who is Tom Cruise married to?*", "*When was Wolfgang Amadeus Mozart born?*" etc. Some of the TREC systems achieved remarkable accuracy: the best factoid QA systems can now answer over 70% of arbitrary, open domain factoid questions. Moving beyond factoids, the current TREC QA systems can also provide composite answers to definition-style and biography questions, such as "*What is a velociraptor?*" or "*Who is Billy Strayhorn?*", list questions such as "*Which countries has Pope John Paul II visited?*", and even relationship questions that look for connections between specific entities, e.g. "*What is Suresh Premachandran's connection to the Tamil Tigers?*"

Huge advances have been made in answer extraction accuracy and dependability over the first several years of TREC QA, but QA does not stop with answer recall. Answering analytical, exploratory, hypothetical and other open-ended questions requires a two-way dialogue between the user and the system. It requires producing answer estimates that can support political, business or military decisions. In order

¹ <http://www.trec.nist.gov>

to provide useful answers, an analytical QA system will be expected to consider user's questions in a larger context of an analytical task, as well as the implications of that task on the user's intentions and information needs. It will require helping users ask the right questions and guiding them to the information that best satisfies their needs. Rather than answering individual questions in isolation, QA systems are now expected to work through a series of interrelated questions, employing various contextual devices to keep track of co-references, elliptical expressions, and maintaining complex discourse representation models to support intelligent dialogue with the user. By analyzing user interactions with an analytical QA system, dynamic interaction models can be developed that would anticipate user information needs and their likely next move, even before a question is asked. Finally, the next generation of QA systems will need to tackle the problem of justifying the correctness of the answers they return, using methods derived from formal logic, deterministic, probabilistic, or abductive inference (e.g., Harabagiu, Pasca, and Maiorano, 2000; Moldovan *et al.*, 2003), as well as by harnessing textual entailment, which was the focus of the PASCAL (Pattern Analysis, Statistical Modeling and Computational Learning) project in Europe.²

The Advanced Question Answering for Intelligence (AQUAINT) Program sponsored by the U.S. Advanced Research and Development Activity (ARDA)³ is addressing these and other issues of advanced QA for the U.S. Intelligence Community. Some of the work presented in our book has been done within the AQUAINT Program. The AQUAINT program has played a central role in the development of much of the groundbreaking QA research conducted in recent years. In addition to providing financial support, AQUAINT has sponsored innovative new evaluations of QA systems (including workshops focusing on dialogue, inference, and the answering of complex question types such as definition questions and relationship questions) that have led to new QA technologies for intelligence applications.

This volume comprises 18 original contributions from leading researchers and developers in the field. The chapters have been assembled into a highly authoritative handbook on how to build practical question answering systems. Most of the systems described here are quite complex and consist of multiple components that can be mixed and matched to construct even more, different systems. We certainly encourage the reader to treat this volume as a practical guide, or as advanced teaching material from which several different graduate level courses can be derived. The book should also be reasonably self-contained in supplying essential characteristics of key supporting natural language processing technologies such as information extraction, dialogue management, word-sense disambiguation, anaphora resolution, and so forth.

² <http://www.pascal-network.org/>

³ <http://www.ic-arda.org>

CONTENTS OF THIS VOLUME

This volume is divided into six sections addressing different approaches and viewpoints on how QA can be done. Many systems described here aim at answering factoid questions, which could be compared to fact recall in databases. Other systems are designed to provide support for open-ended questions, those that professional information analysts or researchers may ask. Yet others aim at providing specialized support tools, such as in medical or legal work. We have attempted to organize this book to expose the variety of perspectives as much as possible, while also trying to discern trends and directions. This is not an easy task and the final organization inevitably reflects the editors' own biases; nonetheless, the order of sections and chapters should not be taken as endorsement of any particular approach over another. The volume can serve as a textbook for advanced graduate studies in question answering and natural language processing. There is certainly enough material here for several courses, and instructors may wish to make selections for semester-long classes.

Section 1 (Approaches to Question Answering) sets the tone for the volume by presenting three quite different methods of building a practical factoid question answering. The chapter by Moldovan, Pașca and Surdeanu gives a detailed account of how definition-style questions can be automatically answered using a combination of deep language processing and logic theorem proving. A radically different approach proposed by Ittycheriah in the second chapter centers around maximum entropy models and other machine learning approaches to QA. In the final chapter of this section, Vicedo and Ferrández give a broad and detailed account of the anaphora resolution problem in natural language and its impact on QA performance.

In Section 2 (Question Processing) Harabagiu discusses the problem of question understanding including user intentions and their implications. She argues that questions can only be understood in a larger context of other activities in which the user may be engaged. Strzalkowski and others describe an interactive approach to question answering where the user and the system negotiate the scope and shape of information being sought and then cooperate in locating and assembling the answer. Finally, Grau and others discuss how their QA system can cope with alternative formulation of questions, and how to reconcile different forms and variants of the expected answer.

Section 3 (Question Answering as Information Retrieval) explores several approaches where QA is cast as a variant of document retrieval, namely answer retrieval. In the opening chapter, Grunfeld and Kwok share their early TREC QA experiences with answer retrieval using *sentence* ranking based on keywords. Clarke and others follow with a detailed account of University of Waterloo's MultiText system that returns short text snippets extracted from relevant document *passages* retrieved from a large database. Finally, Radev and others discuss how to form effective web search queries from user questions in order to fetch the material which is likely to contain the answer they expect.

Section 4 (Answer Extraction) focuses on methods of locating and returning the answer to the user, particularly for well-defined factoid questions. In the opening

chapter, Prager, Chu-Carroll, Brown, and Czuba describe a method of predictive annotation of the database for items that have the potential to become answers to some future factoid questions. In this approach, a set of named entities and other linguistic units is used to index the data, so that the answer strings can be easily picked up later, should a particular text be determined to contain an answer. In the next chapter, Srihari, Li and Li discuss the use of information extraction techniques to obtain the answer directly from the retrieved documents. In their approach, multiple levels of information extraction are utilized to generate a precise, literal answer to a factoid question. Echihabi and others have a similar objective of selecting the correct answer string, but take a radically different tack of utilizing statistical and information theoretic algorithms similar to those used in machine translation. In their thought-provoking approach, QA is cast as a language translation problem where the answer is viewed as a “translation” of the question.

Section 5 (Evaluating Question Answering Systems) is devoted to methods and techniques employed to evaluate performance of QA systems. Text Retrieval Conference (TREC) has been a bellwether of QA evaluations since 1999 when the first TREC QA competition was held. Voorhees’ chapter discusses the rationale behind TREC QA, its scope and success in ranking factoid QA systems. Hersh describes his experience in running a parallel (and TREC-endorsed) evaluation of interactive question answering, as an extension of interactive information retrieval. Hersh’s evaluation centers on the effectiveness of finding answers in medical documents. It involves no actual QA systems and instead focuses on efficiency and effectiveness of interaction. Finally, Ogden, McDonald, Bernick, and Chadwick look at issues of utility and habitability of future QA systems, that is, things that the users are likely to expect from the emerging technology, some of which go well beyond the raw answer extraction performance.

Section 6 (Perspectives on Question Answering) takes a look at some of the immediate context surrounding the current QA research activities. Maiorano presents an analyst’s perspective on question answering and discusses the expected impact of this technology on the U.S. intelligence community. Much of the recent advances in QA research have been sponsored through the ARDA AQUAINT program, which considers QA a vital tool for the analyst. In the following chapter, Riloff, Mann and Phillips discuss creation of data resources that could facilitate rapid development of specialized QA systems and also evaluation metrics. These resources, including corpora of question-answer pairs, are particularly useful for statistical and machine learning approaches to QA. In the final chapter, Maybury takes a longer look into the future of QA through the end of the current decade. He notes that future question answering systems will need to demonstrate significantly improved capabilities to answer broader and deeper questions, extract answers from more varied sources, provide richly justified answers, and support a broader range of users of differing backgrounds, ages, levels of expertise, and interest.

This future starts now. The present volume provides but a snapshot of QA research and technology as it exists today. In the next few years, conversational and cognitive augmentation systems will eclipse the factoid QA and simple interactive QA. Capable artificial assistants will take away the mundane and error-prone activities of searching and organizing information and leave the person with more

time and resources for crucial intellectual and creative activities. More research will be needed to deliver this vision, but we are already part of the way there.

ACKNOWLEDGEMENTS

The plans for production of this volume have been drafted in early 2001, with the initial call for submission going out in June 2001. After evaluating a series of responses, the editors issued detailed guidelines for chapter submission in October 2001. The first round of contributions started arriving in summer 2002 followed by a round of reviews and revisions. Final contributions were collected through early 2004, with additional authors invited to join, to provide a more balanced content. The editors would like to take this opportunity to thank all the authors who responded to our calls, for their outstanding contributions and closely following the schedule imposed on them. The final contributions have undergone a grueling and detailed review followed by more revisions and final decisions regarding the book organization. Our thanks go to the reviewers who helped to assure the highest level of quality for this publication.

The editors would like to thank Ms. Jolanda Voogd, our Springer editor for her support and assistance throughout the publication process. This book would not be possible without the continuing support for QA research by the Advanced Research and Development Activity (ARDA) under the AQUAINT Program.

Tomek Strzalkowski (State University of New York, Albany)
Sanda Harabagiu (University of Texas, Dallas)
 February 2005

REFERENCES

- Bronnenberg, W., H. Bunt, S. Landsbergen, R. Scha, W. Schoenmakers and E. van Utteren: *The Question Answering System PHLQA1*. In: L. Bolc (ed.): *Natural Language Question Answering Systems*. München, Wien: Hanser. London, Basingstoke: Macmillan. 1980.
- Colby, K. M. *Artificial Paranoia. Artificial Intelligence*, Volume 2, 1971.
- Green, B. F. Jr., Wolf, A. K., Chomsky, C., Laughery, K. *Baseball: an automatic question answerer*. Proceedings of the Western Joint Computer Conference, 1961, pp. 219-224.
- Harabagiu, S., M. Pasca and S. Maiorano. *Experiments with Open-Domain Textual Question Answering*. Proceedings of COLING-2000, 292-298.
- Hardy, H., T. Strzalkowski, M. Wu, C. Ursu, N. Webb, A. Biermann, B. Inouye, A. McKenzie. *Data-Driven Strategies for an Automated Dialogue System*. Proceedings of the ACL 2004 Conference, Barcelona, Spain. 2004.
- Lehnert, W. *The Process of Question Answering*. Erlbaum, NJ, 1978.
- Miller, D., R. Schwartz, R. Weischedel, and R. Stone. *Named entity extraction from broadcast news*. Proceedings of DARPA Broadcast News Workshop, Herndon, VA. 1999.
- Moldovan, D., C. Clark, S. Harabagiu, and S. Maiorano. *Cogex: A logic prover for question answering*. Proceedings of the North American Association for Computational Linguistics/Human Language Technology Conference. Edmonton, Alberta. 2003.
- Scha, R. *English words and data bases: how to bridge the gap?* Proceedings of the 20th Annual Meeting of the ACL, Toronto, 1982. pp. 57-59.
- Simmons, R. F. *Answering English questions by computer: a survey*. *Communications of the ACM*, 8(1), 1965. pp. 53-70.

- Walker, M. A., L. Hirschman and J. Aberdeen. *Evaluation for DARPA Communicator Spoken Dialogue Systems*. Proceedings of Language Resources and Evaluation Conference (LREC), Athens, 2000.
- Woods W. A., Kaplan R. M., Nash-Webber B. L.: *The Lunar Sciences Natural Language Information System: Final Report*, BBN Report 2378, Cambridge, MA, 1972.

INTRODUCTION

Question Answering (QA) is a classic natural language processing (NLP) application of obvious practical importance and one presenting special intellectual challenges. Like Information Extraction (IE), with which it is often associated and from which one form of it derives (see Srihari *et al.*, this volume), it has been the subject of special US Government sponsored competitions, the results of which are reported in papers in this book.

QA is an NLP application much transformed, like virtually all NLP, by the shift over the last fifteen years to corpus-based methods and statistical methodologies. The classic QA book was for many years the one derived from Lehnert's thesis work (1978) in which she proposed a combination of semantics- and reasoning-based algorithms. The program behind that book was very small, fragile and, in the spirit of those days, quite untested and untestable beyond the examples it cited. Now, of course, we see a range of QA systems, exemplified by the main papers in this volume, and usually built by extending the established NLP technology of IE, applied after an Information Retrieval (IR) phase which retrieves relevant candidate passages containing answers from a large corpus. Not everything is changed, however, since many IE systems still incorporate linguistic (i.e. not wholly statistical) elements (such as, Humphreys *et al.*, 1999) as well as reasoning components that attempt to verify the answers found.

Curiously, the last book I wrote a brief introduction for, was (Small *et al.*, 1988): a collection of papers on Word Sense Disambiguation (WSD) that appeared just as the empirical movement mentioned above was reaching out to tackle tasks as complex as WSD. The work in that book was perfectly good and interesting, but seemed utterly out of date soon after publication when, against the expectations of many, very high levels of success were being obtained by WSD methods with strong machine learning and other statistical components (e.g., Stevenson and Wilks, 2000). Though again, and like QA, such methods still incorporated linguistic data bases like dictionaries and thesauri in their computations. With the present volume,

I have been lucky enough, I hope, to be invited to write an introduction to a volume that will not be subject to a technological wave hitting it soon after publication.

This narrative can sound like an inevitable march of triumph of statistical and empirical methods through the whole range of NLP areas, but in fact the picture is much more complex. It is quite unclear where the ultimate balance will settle between unsupervised methods, on the one hand, which tend to produce uninterpretable structures of representation for language, and supervised methods, on the other, which are the statistical recapitulation, usually by machine learning, of established symbolic linguistic representations. This dispute is not over even (see Clough and Stevenson, 2003) in IR, with its long tradition of unsupervised clustering and hostility to semantic structures, a point I shall return to below. Much less is it over in the whole range of NLP interests and applications. However, one of the side-effects of the return of statistical and empirical methodology has been to change the perception of what the list of NLP applications is, and that is as evident in QA as elsewhere, a point I shall now turn to but only after noting that, whatever the outcome of all this turmoil, we can be quite sure that evaluation methods in QA, and NLP as a whole, will continue to be empirical and explicit; that is a revolution that cannot be overturned.

QA has had its own particular problems with evaluation, and very sophisticated methods have been derived for this purpose (see Hersh and Voorhees, this volume) in part because the semantic and truth content of the notion of an answer sits somewhat uneasily with the IR base on which empirical QA has, until now, rested. Consider questions whose answers might be, respectively, {string1 AND string 2} and {string1 OR string 2}. The main tradition of QA evaluation comes directly from its initial IR phase, i.e., the retrieval of documents, in this case short string documents corresponding (hopefully) to answers and drawn from larger documents.

Doing this has meant setting arbitrary maximum limits to the string length of candidate answers: usually 250 or 50 bytes in classic US Government TREC competitions. But consider a case where the answer was one of the above and there was a length cut off equivalent to, say, 7 characters. In that case, the truncated answer {string1} of 7 characters, would be right in the OR case and quite wrong in the AND case above. There are ways of easing this problem, but one can see, in general terms, the potential clash between an answer as consisting of interpreted content and a retrieved document of fixed length that may or may not express that answer.

I wrote earlier of modern approaches to QA as being an extension of existing work in IE (with a prior IR component, like most IE), but one could equally well say that QA is a proper part of human-machine dialogue research. The former link is one of methodology, the latter one of functionality since, in general, human dialogues do, at least sometimes, contain questions and answers. Some have gone further: Cowie (Cowie and Nirenburg, 2000) argued that all human dialogue interactions can be seen as forms of QA, so that in the most implausible case, such as the greetings below, could be glossed as QA, as shown in italics:

Good Morning!

[Are you prepared to cooperate in the initiation of a conventional conversational sequence of, as yet, unknown purpose?]

Good Morning!

[Yes]

This is not Cowie's own example, but can be used to show how his thesis, that all dialogue is really QA, could be demonstrated by an appropriate translation even in the most unpromising case. But, again, the situation is more complex because the word "translation" above could be considered literally significant: some have argued (Berger *et al.*, 2000) that answers to questions can be learned as paired-up parallel texts—as in the classic cases of parallel text learning for Machine Translation (MT) (Brown *et al.*, 1990) thus treating questions and paired answers as being like two separate languages, whose "translation", or what we would normally call answer, can be learned like any other phenomenon that presents data in pairs. On that view, then, QA becomes a form of MT, unlikely as that may sound, and this conjuring trick can be seen as part of a general trend to call all methods that involve the parallel text methods of 1980s statistical MT "MT methods", such as "Machine Translation methods in IR" (Berger and Lafferty, 1999). Perhaps it is no more implausible to call "365 days" a translation of "What is the length of a year" than to translate greetings as questions and answers as Cowie (almost) advocated.

Not all these cross-application analogies are equally good, but what is evident is that the introduction of empirical methodologies into NLP, and certainly QA, may well have the effect of redrawing the boundaries between traditional NLP areas; we do not know what shape our familiar landscape will have when the revolution has finally run its course.

Yorick Wilks

University of Sheffield

July 2003

REFERENCES

- Berger, A., R. Caruana, D. Cohn, D. Freitag, and V. Mittal (2000). Bridging the lexical chasm: Statistical approaches to answer-finding. In Proceedings of the 23rd Annual Conference on Research in Information Retrieval (SIGIR2000).
- Berger, A. and Lafferty, J.(1999)_Information retrieval as statistical translation. In Proc. ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99).
- Clough, P., and Stevenson, M. (2003) Evaluating the contribution of Eurowordnet and Word Sense Disambiguation to Cross-language Retrieval in Proc. Twelfth International Conference on Information and Knowledge Management (CIKM-2003), New Orleans.
- Cowie, J. and Nirenburg, S. (2000) Dialogue as question-answering. In Proc 3rd Workshop on Human-Computer Conversation, Bellagio, Italy.
- Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C., Mitchell, B., Cunningham, H., Wilks, Y. (1999) Description of the University of Sheffield LaSIE-II System as used for MUC-7. In Proceedings of the Seventh Message Understanding Conference (MUC-7). Morgan Kaufmann. (1999).
- Brown, P., Cocke, J., Della Pietra, S., Della Pietra, V., Jelinek, F., Lafferty, J. Mercer, R., Roossin, P., (1990) A Statistical Approach to Machine Translation, Computational Linguistics.

- Lehnert, W. (1978) *The Process of Question Answering: a computer simulation of cognition*, Lawrence Erlbaum.
- Small, S., Cottrell, G., & Tanenhaus, M. (Eds.), (1988). *Lexical ambiguity resolution*. San Mateo, CA: Morgan Kaufmann.
- Stevenson, M. and Wilks, Y. (2000) Large Vocabulary Word Sense Disambiguation, In Ravin, Y. and Leacock, C. (eds.) *Polysemy: Theoretical and Computational Approaches*.

1

APPROACHES TO QUESTION ANSWERING

SOME ADVANCED FEATURES OF LCC'S POWERANSWER

Abstract: Key factors determining the accuracy and usefulness of a question answering system span across several dimensions: depth of linguistic techniques, ability to answer general or domain-specific questions, coverage of textual and other multimedia document sources, support for static local collections and dynamic collections like the Web. This chapter illustrates how advanced features can be added to LCC's state-of-the-art question answering system to support a wide range of questions along all these dimensions. The impact of each feature is assessed by extracting answers from large text collections and from Web documents.

1. INTRODUCTION

Accurate information has always been a highly sought-after resource. Somewhere on the East Coast, a team of analysts working under hard deadlines needs to identify which militia leaders ever lived within 100 miles from a recently discovered terrorist base. Their only guess is that the information must be scattered within a few of the fifty thousand electronic reports collected over the past two years. Simultaneously, a graduate student on the other side of the continent is trying to find which music retrieval conferences are going to be held in Europe next year. With a different goal in mind, a leading movie critic is struggling to remember which film received the first Academy Award for Best Picture, realizing that seeing a few "classic" images from that movie would certainly fix the momentary memory lapse.

The tangible benefits for automated question answering (QA) from large collections are virtually unlimited. People formulate questions on a daily basis to find information of interest. They obtain answers from friends, experts, and other authoritative sources (books, dictionaries, encyclopedias) – a process which usually involves expensive information gathering and structuring. Frequently, however, the same pieces of information are already available elsewhere, in small fragments of documents created by anonymous authors and "hidden" in a sea of other documents. Large repositories of unstructured information are growing everywhere, whether they are email archives, internal documents of a specific organization, secret intelligence reports or publicly accessible sites like those found on the World Wide Web. In theory, once the repositories are indexed and a search interface added to access them, information becomes available at a click of a button. In practice, it is challenging yet rewarding to develop automated methods for finding within a large

collection the few document fragments that precisely answer a question posed in natural language.

To fully develop the boundless potential of QA requires linguistic and knowledge sources that are not always available. There are however some resources that are frequently used in advanced QA systems. An example is the well-known WordNet lexical database (Fellbaum, 1998), which was recently released in version 2.0. The Extended WordNet (Extended WordNet, 2003) enriches the information encoded in WordNet to make it even more useful for text-based inference. Various other large gazetteers and lexicons are available and have already been used in related tasks such as information extraction. In addition, tools have been developed to build ontologies pertaining to a specific domain from specialized document collections. These structured resources are complemented by a huge number of Web documents (text or multimedia), which QA systems can access easily via Web search engines.

This chapter illustrates how multiple resources can be exploited and advanced features added to Language Computer Corporation (LCC)'s PowerAnswer question answering system to support a wide range of questions along several dimensions. The *depth of linguistic processing* varies from shallow answer ranking heuristics, to the use of a logic prover for automated reasoning (Section 2). The *types of supported questions* range from open-domain to specialized, e.g., definition seeking questions (Section 3) or domain-specific questions by exploiting ontologies (Section 4). Different *collection types* like static (local) and dynamic (like the Web) are equally supported by adapting the architecture to extract answers from Web documents via search engines (Section 5). The system architecture can be further extended to cover both textual and multimedia documents (Section 6).

2. EXTRACTING ANSWERS WITH A LOGIC PROVER

2.1 Motivation

Most QA architectures are organized around the tasks of classifying questions, identifying relevant documents within large collections, and searching for a few short passages likely to contain answers. As QA systems are faced with stricter requirements and higher user expectations, several shortcomings of the current technology become apparent. In order to process increasingly complex questions, systems must bridge the gap between question and answer words, pinpoint exact answers, take into consideration syntactic and semantic roles of words, rank answers better, justify the answers, and so forth. Results from the Text REtrieval Conference (TREC) (Voorhees, 2002) demonstrated that many systems reached a plateau; the systems ranked from 4th to 14th answered correctly between 38.4% to 24.8% of the total number of questions in TREC-2002. It is clear that new ideas based on a deeper language understanding are necessary to push further the QA technology.

Our solution to the need for deeper, yet effective and scalable QA is to use automated reasoning by integrating a logic prover into the QA system. This section

describes a logic prover for QA, COGEX, presented in (Moldovan *et al.*, 2003), and illustrates how deep and intelligent representations of text can be used to effectively and efficiently re-rank candidate answers by their correctness, extract the exact answers, and ultimately eliminate incorrect answers.

2.2 Approach

The overall approach is to transform questions and candidate answer passages into logic representations, and use world knowledge and linguistic axioms to find which answers are actually correct, the degree of correctness, and the reasons for which the answers are found to be correct. Naturally, performing automated reasoning in the context of QA is very complex. Challenges include logic representation of open text, need of world knowledge axioms, and logic representation of semantically equivalent linguistic patterns. Moreover, logic proofs are accurate but costly, both in terms of high failure rate due to insufficient input axioms, as well as long processing time. We address these issues by adding the logic prover to other modules such as question processing, document retrieval and answer extraction. Therefore, the role of the logic prover is to *augment* (rather than replace) other previously implemented answer extraction techniques.

2.2.1 Architecture of the Logic Prover

As shown in Figure 1, the inputs to COGEX consist of logic representations of questions, candidate answer paragraphs, world knowledge and lexical information. The term Answer Logic Form (ALF) refers to the candidate answers in logic form. Candidate answers returned by the original answer extraction module are classified as open text due to the unpredictable nature of their grammatical structure. The term Question Logic Form (QLF) refers to the questions posed to the Question Answering system in logic form.

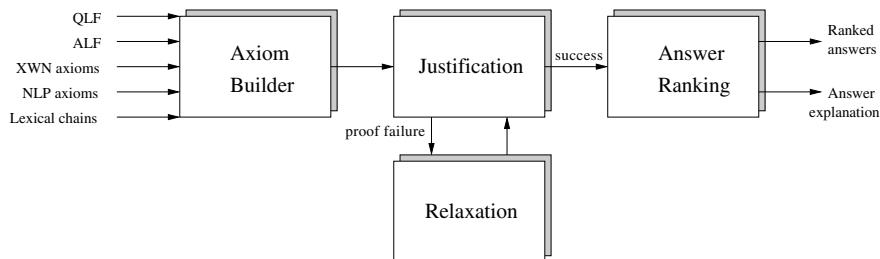


Figure 1. Architecture of the COGEX logic prover.

The prover also needs world knowledge axioms supplied by the WordNet glosses (the actual definitions associated with lexical concepts) transformed into logic representations. Additionally there are many other axioms representing

equivalence classes of linguistic patterns, called NLP axioms. All these are described below.

The Axiom Builder converts the Logic Forms for the question, the glosses, and its candidate answers into axioms. Based on the parse tree patterns in the question and answers, other NLP axioms are built to supplement the existing general NLP axioms. Once the axioms are complete and loaded, justification of the answer begins.

If a proof fails, the relaxation module is invoked. The purpose of this module is twofold: (1) to compensate for errors in the text parsing and Logic Form transformation phase, such as prepositional attachments and subject/object detection in verbs, and (2) to detect correct answers when the NLP and XWN (Extended WordNet) axioms fail to provide all the necessary inferences. During the relaxation, arguments to predicates in the question are incrementally uncoupled, the proof score is reduced, and the justification is re-attempted. The loop between the Justification and the Relaxation modules continues until the proof succeeds, or the proof score is below a predefined threshold. When all the candidate answers are processed, the candidate answers are ranked based on their proof scores, with the output from COGEX being the ranked answers and the answer justifications.

2.2.2 Conversion of Text into Logic Forms

A text logic form (LF) is an intermediate step between syntactic parse and the deep semantic form. The LF codification acknowledges syntax-based relationships such as: (1) syntactic subjects, (2) syntactic objects, (3) prepositional attachments, (4) complex nominals, and (5) adjectival/adverbial adjuncts. Our approach is to derive the LF directly from the output of the syntactic parser which already resolves structural and syntactic ambiguities.

Essentially there is a one to one mapping of the words of the text into the predicates in the logic form. The predicate names consist of the base form of the word concatenated with the part of speech of the word. Each noun has an argument that is used to represent it in other predicates. One of the most important features of the Logic Form representation is the fixed-slot allocation mechanism of the verb predicates (Hobbs *et al.*, 1993). This allows for the logic prover to see the difference between the role of the subjects and objects in a sentence that is not answerable in a keyword based situation.

Logic Forms are derived from the grammar rules found in the parse tree of a sentence. There are far too many grammar rules in the English language to efficiently and realistically implement them all. We have observed that the top ten most frequently used grammar rules cover 90% of the cases for WordNet glosses. This is referred to as the 10-90 rule (Moldovan and Rus, 2001). Below we provide a sample sentence and its corresponding LF representation. More details regarding the transformation of text into logic forms are presented in (Moldovan and Rus, 2001).

Example:

Bin Laden reportedly sent representatives to Afghanistan opium farmers to buy large amounts of opium, probably to raise funds for al-Qaeda.

LF: Bin_NN(x14) & Laden_NN(x15) & nn_NNC(x16, x14, x15) & reportedly_RB(e2) & send_VB(e2, x16, x17) & representative_NN(x17) & to_TO(e2, x21) & Afghanistan_NN(x18) & opium_NN(x19) & farmer_NN(x20) & nn_NNC(x21, x19, x20) & buy_VB(e3, x17, x22) & large_JJ(x22) & amount_NN(x22) & of_IN(x22, x23) & opium_NN(x23) & probably_RB(e4) & raise_VB(e4, x22, x24) & funds_NN(x24) & for_IN(x24, x27) & al_NN(x25) & Qaeda_NN(x26) & nn_NNC(x27, x25, x26).

2.2.3 Sources of Knowledge

The question and answer logic forms constitute two of the inputs to the logic prover shown in Figure 1. The other inputs provide the knowledge that guides the proof and bridges the logic forms. The logic prover has access to two main types of knowledge:

- *world knowledge axioms* are used to link question concepts to answer concepts. These axioms are provided by the WordNet glosses represented in logic forms.
- a separate set of *NLP axioms* correspond to rewriting procedures for semantically equivalent lexical patterns.

World Knowledge Axioms

Logic Representation of WordNet Glosses

A major problem in QA is that often an answer is expressed in words different from the question keywords. World knowledge is necessary to conceptually link questions and answers. WordNet glosses contain a source of world knowledge. To be useful in automated reasoning, the glosses need to be transformed into logic forms. Taking the same approach as for open text, we have parsed and represented in logic forms all WordNet glosses. For example, the gloss definition of concept sport_NN#1 is *an active diversion requiring physical exertion and competition*, which yields the logic representation:

active_JJ(x1) & diversion_NN(x1) & require_VB(e1,x1,x2) & or_CC(x2,x3,x4) & physical_JJ(x3) & exertion_NN(x3) & competition_NN(x4)

Lexical Chains

A much improved source of world knowledge is obtained when the gloss words are semantically disambiguated (Moldovan and Novischi, 2002). By doing this, the connectivity between synsets is dramatically increased. Lexical chains can be established between synsets in different hierarchies. These are sequences of semantically related words that link two concepts.

Lexical chains improve the performance of question answering systems in two ways: (1) increase the document retrieval recall and (2) improve the answer extraction by providing the much needed world knowledge axioms that link question keywords with answers concepts.

We developed software that automatically provides connecting paths between any two WordNet synsets S_i and S_j up to a certain distance (Moldovan and Novischi,

2002). The meaning of these paths is that the concepts along a path are topically related. The path may contain any of the WordNet relations augmented with a GLOSS relation which indicates that a certain concept is present in a synset gloss.

Examples

Below we provide some relevant lexical chains that link a few selected TREC-2002 questions with their answers.

Q1404: How many chromosomes does a human zygote have?

Answer: [...] at the 46 chromosomes that lie in the nucleus of every normal human cell without perceiving what has now been discovered.

Lexical chains:

- (1) zygote:n#1 → HYPERNYM → cell:n#1
- (2) zygote:n#1 → HYPERNYM → cell:n#1 → HAS_PART → nucleus:n#1

Q1512: What is the age of our solar system?

Answer: The solar system is 4.6 billion years old.

Lexical chains:

- (1) age:n#1 → RGLOSS → aged:a#1 → SIMILAR → old:a#1

Q1570: What is the legal age to vote in Argentina?

Answer: Voting is mandatory for all Argentines aged over 18.

Lexical chains:

- (1) legal:a#1 → GLOSS → rule:n#1 → RGLOSS → mandatory:a#1
- (2) age:n#1 → RGLOSS → aged:a#3
- (3) Argentine:a#1 → GLOSS → Argentina:n#1

NLP Axioms

In addition to world knowledge axioms, a QA logic prover needs linguistic knowledge. This is what distinguishes an NLP prover from a traditional mathematical prover. General axioms that reflect equivalence classes of linguistic patterns need to be created and instantiated when invoked. We call these NLP axioms and present below some examples together with questions that call them.

Complex Nominals and Coordinated Conjunctions

A question may refer to a subject/object by its full proper name, and the answer will refer to the subject/object in an abbreviated form. For example in the correct candidate answer for the question, “*Which company created the Internet browser Mosaic?*”, Internet browser Mosaic is referred to as Mosaic.

Using abduction, an axiom is built such that the head noun of the complex nominal in the question implies the remaining nouns in the complex nominal:

- all x1 (mosaic_nn(x1) → internet_nn(x1) & browser_nn(x1))

An additional axiom is built such that all the nouns in the complex nominal imply a complex nominal:

- all $x1(\text{internet_nn}(x1) \ \& \ \text{browser_nn}(x1) \ \& \ \text{mosaic_nn}(x1)) \rightarrow \text{nn_nnc}(x1, x1, x1, x1)$

So as not to restrict the ordering of the nouns in the noun phrase from which the complex nominal is built, the same argument is used for each of the noun predicates in the complex nominal. Similar to the above issue, a question may refer to the subject/object in an abbreviated form, while the answer will refer to the subject/object in its full, proper form. For example in the correct candidate answer for the question, “*When was Microsoft established?*”, Microsoft is referred to as Microsoft Corp. An axiom is built such that each noun of the complex nominal takes on the identifying argument of the complex nominal:

- all $x1 \ x2 \ x3 \ (\text{microsoft_nn}(x1) \ \& \ \text{corp_nn}(x2) \ \& \ \text{nn_nnc}(x3, x1, x2)) \rightarrow \text{microsoft_nn}(x3) \ \& \ \text{corp_nn}(x3)$

Similar axioms are used for coordinated conjunctions detected in the answer and the question. These are considered weak axioms, and any proof that uses them will be penalized by being given a lower score than those that do not.

Appositions

A candidate answer for a question may use an apposition to describe the subject/object of the answer. The question may refer to the subject/object by this apposition. For example in the question, “*Name the designer of the shoe that spawned millions of plastic imitations, known as jellies*”, the candidate answer, “...*Italian Andrea Pfister, designer of the 1979 ‘bird cage’ shoe that spawned millions of plastic imitations, known as jellies ...*” uses an apposition to describe the designer. An axiom is built to link the head of the noun phrases in the apposition such that they share the same argument:

- all $x12 \ x13 \ x14 \ x15 \ x17 \ x18 \ x19 \ (\text{italian_nn}(x12) \ \& \ \text{andrea_nn}(x13) \ \& \ \text{pfister_NN}(x14) \ \& \ \text{nn_nnc}(x15, x12, x13, x14)) \rightarrow \text{designer_nn}(x15) \ \& \ \text{of_in}(x15, x17) \ \& \ \text{1979_nn}(x17) \ \& \ \text{bird_nn}(x18) \ \& \ \text{cage_nn}(x19))$

Possessives

A question/answer substitutes the use of a possessive by using an *of* or *by* preposition. For example, in the question, “*What was the length of the Wright brothers’ first flight?*” the candidate answer, “*Flying machines, which got off the ground with a 120 - foot flight by the Wright brothers in 1903...*” implies ownership using the preposition *by* to connect the Wright brothers to flight. An axiom is built to connect *by* to the possessive:

- all $x1 \ x2 \ (\text{by_in}(x1, x2) \rightarrow \text{pos}(x1, x2))$

Equivalence Classes for Prepositions

Prepositions can be grouped into equivalence classes depending on the context of the question, which is determined by the expected answer type. In location seeking questions the prepositions *at* and *in* are often interchangeable. Similarly for *in* and *into*, and *from* and *of*. In date seeking questions *in* and *of* have interchangeable meanings as do *at* and *in*. For example, in the question, “*What body of water does the Colorado River flow into?*”, the candidate answer, “...*the Colorado River flowed*

in the Salton trough about 130 miles east of San Diego”, the preposition *in* and *into* in the answer take in the same meaning. An axiom is built to link *in* to *into*:

- all $x_1 \ x_2 \ (\text{in_in}(x_1, x_2) \rightarrow \text{into_in}(x_1, x_2))$

Part-of Relations in Location Questions

A location seeking question may have a candidate answer that identifies a location by referring to a part-of the location. For example, in the question, “*Where is Devil’s Tower?*”, the answer, “*American Indians won another court battle over their right to worship without interference at Devils Tower National Monument in the northeast corner of Wyoming*”, identifies Wyoming as the location of Devil’s Tower by referring to the part of Wyoming in which it lies. An axiom is built to connect Wyoming to its part:

- all $x_1 \ x_2 \ x_3 \ (\text{corner_nn}(x_1) \ \& \ \text{of_in}(x_1, x_2) \ \& \ \text{wyoming_nn}(x_2) \rightarrow \text{wyoming_nn}(x_1))$

Attribute-of Relations in Quantity Seeking Questions

A question seeking a quantity may have a candidate answer that implies quantity of subject by prefixing the quantity to the subject. For example in the question “*What is the height of the tallest redwood?*” the answer is “*329 feet Mother of Forest’s Big Basin tallest redwood...*” An axiom is built to connect the quantity to its subject, redwood:

- all $x_1 \ x_2 \ (\text{quantity}(x_1) \ \& \ \text{redwood_NN}(x_2) \rightarrow \text{of_in}(x_1, x_2))$

This is a weak axiom since the proximity of redwood to quantity in the answer text is not guaranteed. As mentioned for the complex nominal and coordinated conjunction axioms, any proof that uses these axioms should be penalized and ranked lower than those that do not. Note that for this axiom to be effective, an axiom linking the heads of the apposition is built:

- all $x_8 \ x_9 \ x_{10} \ x_{11} \ x_{12} \ (\text{mother_nn}(x_8) \ \& \ \text{of_in}(x_8, x_9) \ \& \ \text{forest_nn}(x_9) \rightarrow \text{big_nn}(x_{10}) \ \& \ \text{basin_nn}(x_{11}) \ \& \ \text{nn_nnc}(x_{12}, x_{10}, x_{11}) \ \& \ \text{_s_pos}(x_8, x_{12}) \ \& \ \text{tall_jj}(x_8) \ \& \ \text{redwood_nn}(x_8))$

2.3 Evaluation

COGEX was implemented and integrated into PowerAnswer and tested on the TREC-2002 evaluation questions. All questions are attempted by the prover, but if the proof fails the QA system resorts to other answer extraction methods that were part of the system before the prover. Thus, some questions are answered by the QA system without the prover, some only by the prover and some by both the non-prover system and the prover. The complete system answered 415 questions out of 500 TREC-2002 questions (Moldovan *et al.*, 2003). Of these, 206 were answered by COGEX but some of these answers were also provided by QA system without COGEX. A careful analysis indicates that the QA system without logic prover answered 317 questions and the prover can answer only 98 additional questions for which the system without prover failed. Table 1 summarizes these results.

Table 1. Performance over 500 TREC-2002 questions.

Questions answered by the complete system	415
Questions answered by COGEX	206
Question answered only by COGEX	98
Questions answered without COGEX	317

The added value of automated reasoning to the QA system is 30.9% (98/317). This represents a significant improvement in the performance of the logic prover for QA over the one reported in (Moldovan *et al.*, 2002). The failures of the prover are due primarily to the lack of linguistic axioms.

3. ANSWERING DEFINITION QUESTIONS

3.1 Motivation

There are categories of questions for which even complex system modules, like the logic prover described in the previous section, have difficulties extracting precise answers. Questions whose expected answer type does not fall into a well-defined semantic category constitute a challenge to otherwise successful question answering systems. Among them, *Definition* questions are a significant subset.

Table 2. Examples of Definition questions from the TREC QA track.

Q#	Question	Q #	Question
241	What is a caldera?	1033	What are platelets?
528	What are geckos?	1188	What are Aborigines?
917	What is bipolar disorder	1232	What is Ursa Major?

Two recent evaluations on large text collections acknowledged the importance of Definition questions by including them in the question test set. The combined set of evaluation questions used in the TREC QA track (Voorhees, 1999) over the last years contains about 170 Definition questions (see Table 2). The Aquaint pilot evaluation (Weischedel *et al.*, 2002) is a smaller-scale test-bed using 25 Definition questions (see Table 3).

Table 3. Examples of Definition questions from the Aquaint pilot evaluation.

Q #	Question	Q #	Question
1	Who is Gunter Blobel?	14	What are nanoparticles?
2	Who was Joseph Pilates?	15	What is the Nature Conservancy?
3	What is a femtosecond?	20	What is Yahoo!?
6	Who is Ellen Sauerbrey?	24	What is fibromyalgia?

This section explores the distinguishing properties of Definition questions, and describes alternative techniques for extracting answers from large text collections. The techniques are validated by answering approximately 200 test questions from two large document collections and from Web documents.

3.2 Approach

3.2.1 Analysis of Definition Questions

Even though structurally simpler than the more general class of fact-seeking questions, Definition questions have a few properties which complicate their processing.

First, there are few available keywords in the question. Consequently the output from the passage retrieval module is large, especially if the question phrase is a common (“*What is a star?*”) rather than specific term (“*What is phlebotomy?*”). To make things worse, candidate answers seldom fall into a clear-cut semantic category like *Person* or *Location*.

The candidate answers may span a word, a phrase, a sentence or even a few sentences, which further complicates the extraction methods. Multiple answers are the common case. For example, “*What is anorexia nervosa?*” is answered by “*illnesses such as anorexia nervosa*” as well as “*eating disorders, such as anorexia nervosa*”. The system should return as many distinct, correct answers as possible rather than only one answer.

Another difficulty is the variation of the relations between answers and the question phrases. A casual user who finds the term *M60A3* by hazard and formulates the corresponding Definition question may be satisfied by the answer *tank*. Comparatively, a military expert is likely to reject that answer as non-informative and prefer “*US Army main battle tank*”. Moreover, there is no unique relation type, even though most question phrases are in the IsA relation with their respective answers. We have identified several relations of interest to real users and assessors (the list is not exhaustive):

- IsA, genus: company for the question “*What is Yahoo?*”; disease for “*What is aspergillosis?*”;

- IsA, differentia: “on-line company” for the question “*What is Yahoo?*”; “electricity-producing device” for “*What is a battery?*”;
- Alternative name (synonymy): salt for “*What is sodium chloride?*”;
- KindOf: “such as AltaVista” for “*What is a search engine?*”;
- PartOf: “ingredient of gunpowder” for “*What is saltpeter?*”.

3.2.2 Techniques for Answer Extraction

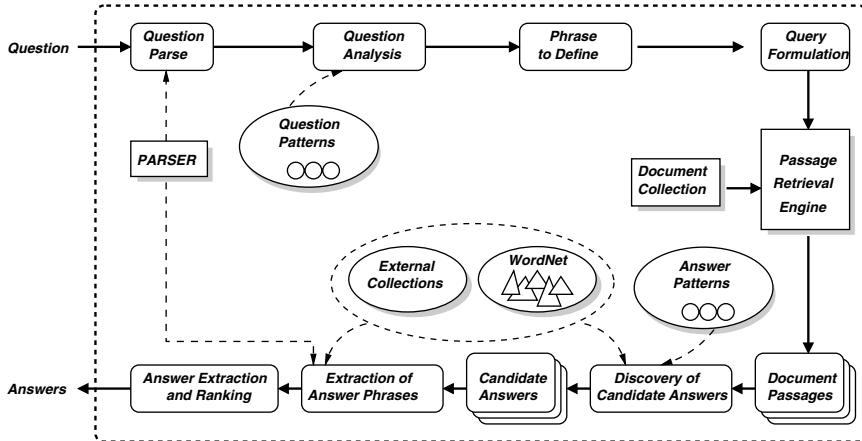


Figure 2. Architecture for processing definition questions.

Figure 2 illustrates an architecture adapted to answering Definition questions. Questions are parsed and then matched against a set of question patterns to identify the question phrase. A query formulation module transforms the questions into conjunctive Boolean queries, e.g. (sodium AND chloride). The passage retrieval engine selects the subset of document passages containing all keywords.

The discovery of candidate answers in the retrieved passages cannot rely on named entity recognition. Instead each candidate answer is related to the question phrase in the passage, either through a semantic relation such as WordNet (Miller, 1995) hypernymy, or through a lexico-semantic pattern.

Finding candidate answers

The first source for finding answer sentences in the retrieved passages is the matching of the passages on answer patterns (Paşa, 2001c). Table 4 illustrates the application of answer patterns on Aquaint questions. The answer patterns capture word sequences that are likely to introduce definitions or explanations of the phrase to define. Their simplicity provides robustness on unrestricted text.

Table 4. Correct answers identified for Aquaint questions with pattern matching.

Answer pattern	Phrase to define (QP)	Sentence answer
<AP> such as <QP>	What is <u>Yahoo!</u> ?	As for as others finding your Web sit, you can start the marketing process by adding it to search engines, such as <i>Yahoo</i>
<QP>, a <AP>	Who was <u>Joseph Pilates</u> ?	The exercise method was invented by a man, Joseph Pilates, a boxer and physical therapist from Germany
<QP> is the <AP>	What is a <u>femtosecond</u> ?	A femtosecond is the fraction of a second represented by a 1 preceded by 14 zeros and a decimal point.

Each pattern is assigned one of three priority classes. The sentence answers that were obtained with a higher-priority pattern are assigned a higher relevance. For example, the pattern “*AP, also called QP*” has a higher priority than “*QP is a AP*”.

Table 5. Using WordNet to find answers to TREC questions.

Phrase to define	WordNet hypernym	Sentence answer
What is a <u>nematode</u> ?	{worm}	Nematodes, microscopic worms, were released in 1985 and 1986 in several Alachua pastures
What are <u>geckos</u> ?	{lizard}	For the national Zoo came a par of red wolves, an endangered species that once roamed the South-eastern United States, 30 geckos –lizards-from Madagascar, and a male Sumatran tiger

The second source for finding answer sentences is the integration of a general-purpose knowledge source (Paşca, 2001c), namely the WordNet database (Miller, 1995). If the question phrase is in WordNet, it is expanded with concepts linked through semantic relations (Paşca, 2002). Currently, the question phrase is expanded with its immediate WordNet hypernyms. Others (Lin, 2002) experimented with expansion from WordNet glosses, in which case the answer extraction is closer to projecting the candidate answers from one document collection into another (Brill *et al.*, 2001). The sentences containing an expanded term and the question phrase are added to the set of candidate answers identified with patterns.

Table 5 illustrates relevant answer strings extracted after the WordNet-based automatic expansion of Definition questions.

Extracting Exact Answers

Exact answers are extracted after parsing the sentences in which answer patterns or WordNet concepts are found. To this effect, the parse tree of the whole sentence is processed to derive the phrase that gives the exact answer. The exact answer is selected from the enclosing phrase around the question phrase to search. In turn, the enclosing phrase is determined by iteratively selecting the parent of the current node in the parse tree as long as the parent is a noun phrase.

The resulting noun phrase is post-processed to remove the question phrase, as well as the stop words (determiners, pronouns) from the extremities. If the remaining phrase does not contain any noun, the associated sentence is rejected from the set of candidate answers. Finally, duplicate answers are removed except for the first occurrence. Duplicate checking currently consists of simple substring comparison.

3.3 Evaluation

3.3.1 Experimental Setting

The two sets of Definition questions are evaluated on two document collections. For each of the 174 TREC Definition questions, the system extracts at most five sentence and exact answers respectively from the 3-gigabyte text collection denoted Trec2001. The same collection was used in the TREC-9 (2000) and TREC-10 (2001) QA track evaluations (Voorhees, 1999). Comparatively, the 25 Aquaint questions are evaluated against a different document collection of similar size, denoted Trec2002. This collection was used in the TREC-2002 QA track evaluation.

For a stricter evaluation, the system does not return a candidate answer unless it found the candidate in the corresponding passage. In other words, there is no answer “guessing”. Note that the scores of this evaluation would be higher if the system always returned the top 5 answers.

Three metrics are used to measure the answer accuracy: MRR or the Mean Reciprocal Rank, the official metric of the TREC QA evaluations (Voorhees, 1999); Q-top5 (a more relaxed metric), i.e. the percentage of questions with a correct answer returned among the first five; and Q-top1 (a stricter metric), similar to Q-top5 but using only the first answer returned.

3.3.2 Results on TREC Questions

The overall scores on TREC Definition questions, shown in Table 6, confirm that it is easier to extract longer, sentence answers than exact answers. The MRR scores in the first two rows are computed according to the official TREC answer keys, with a script that marks an answer string as correct if it contains one of the answer keys. Therefore the exact answers marked as correct may be actually non-minimal, i.e. they may contain extraneous information.

Table 6. Performance over TREC Definition questions. Exact answers are considered correct if they are verbose (Vrb.) or minimal (Min.) respectively.

Run type	MRR	Q-top5	Q-top1
Sentence answers (Vrb.)	0.65	139/174 = 0.79	97/174 = 0.55
Exact answers (Vrb.)	0.37	96/174 = 0.55	45/174 = 0.26
Exact answers (Min.) baseline	0.20	39/174 = 0.22	30/174 = 0.17
Exact answers (Min.).	0.34	89/174 = 0.51	40/174 = 0.23

To assess the ability of the QA system to extract *minimal* answers (a task much harder than the extraction of relevant sentences), we also evaluated the returned answers manually. Each answer was marked as correct if it matched a TREC answer key *and* it was minimal. The scores are shown in the last two rows of Table 6. The third row corresponds to a baseline run in which the techniques for identifying the exact answers (described in Section 3.2.2) were disabled and a more relaxed approximation of exact answers was used instead. The approximation consists of estimating the position of the exact answer from the pattern matches without parsing the answer sentences. The fourth row corresponds to the more precise extraction technique described in Section 3.2.2.

The use of syntactic information to derive the exact answer accounts for most of the improvement in the fourth row over the third row baseline in Table 6. On the other hand, some of the relevant extracted answers are still not minimal even though the difference is relatively smaller. This explains the difference between the second and fourth rows of Table 6. Table 7 shows a few relevant exact answers.

Table 7. Examples of relevant answers returned for TREC question from the TREC-2001 collection.

Question	Sentence answer (exact answer shown in italics; not minimal if marked as *)
What are chloroplasts?	Photosynthesis occurs in chloroplasts, the <i>tiny structures in plant cells that contain chlorophyll</i>
What is a micron?	A micron is a <i>millionth of a meter, about one-hundredth the thickness of a human hair</i>
What is viscosity?	Magma that is 65 percent to 75 percent silica has <i>silica-rich molecules which tend to chain together and drastically increase the resistance to flow*</i> , or viscosity

Table 8. Performance over Aquaint questions. Sentence answers are considered correct if they are verbose (Vrb.). Exact answers are considered correct if they are minimal.

Run Type	MRR	Precision	Q-top5
Sentence answers (Vrb.) baseline	0.55	0.53	13/25
Sentence answers (Vrb.)	0.84	0.85	23/25
Exact answers (Min.)	0.72	0.58	23/25

Table 9. Answers extracted from Web documents for Aquaint questions from Table 3.

Q#	Sentence answer (exact answer shown in italics)
1	Porters and File Cabinets – Too Few Prospects Gunter Blobel, <i>The German Nobel Prize Winner in Medicine who has worked for decades at New York's Rockefeller University</i> , said it this easy, “In Germany back then, I would have had to work as a senior assistant for a professor” (extracted from www.campus-germany.com/english/2.60.264.html)
6	In this issue, we talk with Ellen Sauerbrey, the <i>Republican gubernatorial nominee in 1994 and the former minority leader in the Maryland House of Delegates</i> (extracted from www.calvertinstitute.org/news/Vol13-1/cnvol3_1f.htm)
20	Black rocket understood what few others agencies did: Yahoo is a <i>consumer brand, not a technology company</i> (extracted from www.trinityventures.com/team/articles/yahoo.html)
24	But their use has become more popular in recent years, especially with people who have <i>chronic illnesses</i> , such as fibromyalgia (extracted from www.mayoclinic.com/invoke.cfm?id=DS00079)

3.3.3 Results on Aquaint Questions

Table 8 illustrates the performance over all Aquaint questions. The precision is computed as the percentage of returned answers that are correct (regardless of their rank), and then as the average of the resulting values on all questions. Two of the 25 evaluation questions could not be answered correctly, namely Q19: “*What is asceticism?*” and Q23: “*What is the religious right?*”

Table 9 illustrates some of the answers that are extracted from Web documents rather than local text collections. The QA system fetches and analyzes locally the top N Web documents returned by an external search engine (Pașca, 2001a). Table 10 shows the overall scores from Web documents for N set to 20 and 100. There are no correct sentence or exact answers extracted for the questions Q18: “*Who is*

Treasury Secretary Robert Rubin?” and Q23: “*What is the religious right?*” from either AltaVista or Google. For the question Q19: “*What is asceticism?*”, which could not be answered from the TREC2002 collection, the QA system finds a relevant answer on the Web (www.orthodoxinfo.com/praxis/ascesis_modern.htm) via AltaVista. The extracted sentence answer is:

“*Asceticism is a strict, purposeful life, expressed in spiritual labors, that is, in prayer, contemplation, the directing of the mind toward God, frequently in conjunction with corresponding physical undertakings, and simultaneously with abstention from any [...].*”

Table 10. Performance (MRR) over Aquaint questions from Web documents.

Scores	via Google				via AltaVista			
	(top 20 docs)		(top 20 docs)		(top 20 docs)		(top 100 docs)	
	Sent.	Exact	Sent.	Exact	Sent.	Exact	Sent.	Exact
MRR	0.51	0.43	0.62	0.49	0.61	0.54	0.69	0.55
Q-top5	15/25	13/25	20/25	18/25	17/25	16/25	19/25	17/25

4. DOMAIN SPECIFIC QUESTION ANSWERING

4.1 Motivation

The TREC competition, which has stimulated a considerable momentum in the design of QA systems, aims at open text, that is, questions can come from any domain. However, in many applications and settings with documents describing a limited number of products and procedures, there is need for QA to perform well but in narrow domains.

Thus the question is how to boost the QA performance when the domain is restricted. One idea is to insert into the QA more information about that domain. This means ontologies, or knowledge bases that are domain specific such as weapons of mass destruction, several chemistry books, etc. Subsequent questions are how to rapidly build domain specific ontologies, and how to take advantage of domain ontologies in QA.

4.2 Approach

4.2.1 Building Domain Ontologies

In order to move quickly from one domain to another, we need a tool to build specific concepts, as well as relations among them from open text. Our approach is to acquire new concepts starting from a set of user-selected seed concepts, and then

integrate the resulting ontologies into the WordNet linguistic knowledge structure. The ontology building process has four main steps: discovery of new concepts; discovery of new relationships via lexico-semantic patterns; and classification and integration with a WordNet-like knowledge base (Moldovan and Girju, 2001).

Table 11. Illustration of ontologies acquired from local and Web documents.

Ontology	Doc. source	Nr. concepts	Max. depth	Examples of concepts
grenade	CNS	92	3	Antipersonnel grenade, carbon tet grenade, frag grenade, incendiary grenade
WMD	CNS	144	4	Biological weapon, atom bomb, Aryan Nations, ETA, Islamic Jihad, SEndoro Luminoso, Jewish Defense League, Hizballah
terrorist group	CNS	143	4	Fundamentalist terrorist group, Aryan Nation, ETA, Islamic Jihad, Sendoro Luminoso, Jewish Defense League, Hizballah
search engine	Web	60	3	Web search engine, CUSI, AltaVista, Fast, Google, Ixquick, meta search engine
sports car	Web	41	3	British sports cars, 914s, Nissan 300zx, luxury sports car, Ferrari Testarossa

The discovery of new concepts starts around some seed concepts that a user considers important for the domain. In the financial domain, for instance, a possible set of seed concepts is *interest rate*, *stock market*, *inflation*, *economic growth* and *employment*. The first step is to create a corpus of sentences containing each of the seed concepts. To this effect, queries are submitted to search engines to retrieve relevant documents, which are then filtered such that only sentences containing seed concepts are retained. The noun phrases that contain the seeds form a pool of potential new concepts. The potential concepts are WordNet-reduced, by identifying existing WordNet concepts. Thus *long term interest rate* becomes *long_term interest_rate*, since *long term* and *interest rate* are valid WordNet concepts. Furthermore, potential concepts whose modifiers are adjectives are discarded, if they are formed with descriptive (e.g., *heavy package*, *high speed*) rather than relational (*musical instrument*) or participial (*boiling water*) adjectives.

The new concepts are used iteratively to discover relations that link them with other concepts from the documents. The approach is to search for lexico-semantic

patterns corresponding to the type of the relation of interest. Note that relation types are either “standard” like IsA (Hypernymy), or specialized types like Space, Cause, Part-Whole, Manner. For example, the pattern “NP1 cause NP2” identifies the relation “Cause (NP1, NP2)” from the text “*Phillips, a British economist, stated in 1958 that high inflation causes low unemployment rates*”. Similarly, the pattern “NP1 reflect NP2” identified the relation “Influence(NP1, NP2)” from the text “*As a rough approximation, at least, nominal interest rates reflect rates of inflation*”.

New concepts are classified with respect to one another based on the subsumption principle. Each new concept is thus inserted in the correct position into partially built ontologies. This is how ontologies are built iteratively from a variety of document sources as shown in Table 11. In our tests, the sources of the ontology were both Web documents, and a specialized corpus denoted CNS.

In the final step, the ontologies must be anchored, or linked to other pre-existing, general-purpose knowledge in the QA system. In our case, they are integrated under WordNet hierarchies to supplement open-domain knowledge with focused, domain-specific knowledge.

4.2.2 Integrating Ontologies in QA

The QA system exploits domain-specific ontologies in three key tasks: question processing, through the recognition of the expected answer type and detection of candidate answers; passage retrieval, through ontology-based query expansion; and answer extraction, through automated reasoning with the logic prover.

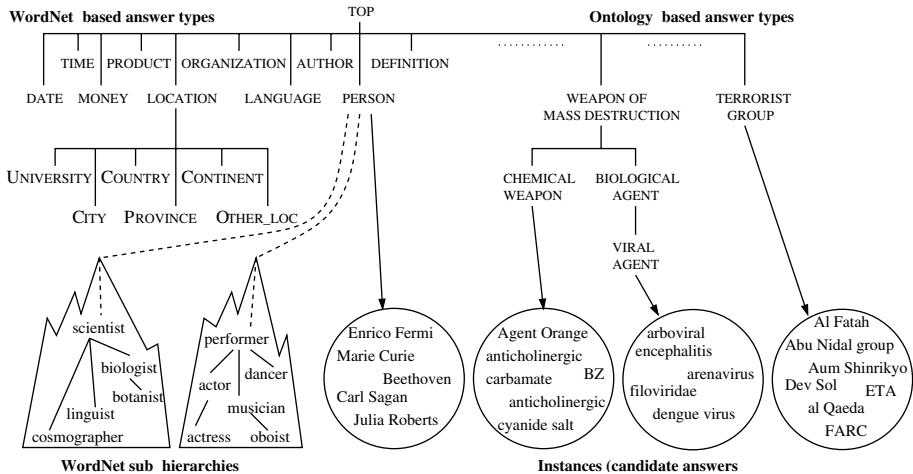


Figure 3. Using domain-specific ontologies for answer finding.

Figure 3 shows how an integrated answer type hierarchy enhances the recognition of the expected answer type. In the case of open-domain questions, WordNet sub-hierarchies map various semantically-related questions, such as “*What biologist received the Nobel Prize for his studies of protein signals?*” and “*What*

scientist discovered the insulin?", into the same category (e.g., *Person*). Concurrently, the ontology-based hierarchy (shown on the right in Figure 3) addresses domain-specific questions. If the concept asked about in the question is found in any of the ontologies, all its sub-concepts in the ontology become candidate answers. For example, questions such as "*What chemical weapons were used as defoliants in Vietnam?*" are answered by instances like *Agent Orange*, whereas the instances *al Qaeda* or *Al Fatah* would be relevant for questions related to terrorist activities, e.g. "*What terrorist groups operate in Egypt?*".

The quality of the passages retrieved within the QA system is significantly improved through ontology-based query expansion. The module adds related concepts from the domain ontologies to passage retrieval queries. First, it determines which semantic relations should be followed for query expansion; IsA is an immediate option, but other relation types may be useful. The domain-specific concepts included in the query are selected based on their frequency of occurrence in the collection and relative semantic distance metrics. For example, for the question "*What biological agents does al Qaeda possess?*" the queries used for passage retrieval are:

- without ontology information: (*biological AND agents AND Qaeda*)
- with ontology information: ((*biological AND agents*) OR (*bacterial AND agent*) OR (*viral AND agent*) OR (*fungal AND agent*) OR (*toxic AND agent*) OR ... OR *botulism* OR *botulinum* OR *cholera* OR *ebola* OR *lassa* OR *encephalitis* OR *marburg* OR ... OR (*purpura AND variola*) AND (*Qaeda*))

Ontologies provide an additional knowledge in answer extraction, more precisely an additional input in the logic prover besides the ones shown in Figure 1. To this effect, each ontology relation and its associated concepts are converted into a domain axiom. For example, the fact that a *chemical weapon* is a kind of a weapon of mass destruction, and that the *dengue virus* is an instance of a *viral agent* are represented as:

- all x1 (*chemical_weapon_nn(x1)* → *weapon_of_mass_destruction_nn(x1)*)
- all x1 (*dengue_virus_nn(x1)* → *viral_agent_nn(x1)*)

4.3 Evaluation

The evaluation of the impact of domain ontologies on QA uses a document collection developed by the Monterey Center for Nonproliferation Studies (CNS). The collection comprises more than 20,000 articles detailing recent nuclear-related events in key countries throughout the world. The system was tested on 57 questions specific to the CNS domain.

The overall performance is illustrated in Table 12. The results show that in a domain-specific environment, where domain information drives the system execution from keyword selection to answer extraction, the ontology contribution is very high, almost quadrupling the scores.

Table 12. Impact of ontologies measured on sentence answers for 57 domain specific (CNS) questions.

Run type	MRR	Q-top1	Q-top10
without ontologies	0.22	0.15	0.40
with ontologies	0.59	0.49	0.82

Table 12a: shows that the ontologies acquired offline from the CNS collection improve the quality of the extracted answers. The impact is more pronounced for exact rather than sentence answers, because the ontologies because the ontologies indicate what instances correspond to a domain-specific answer type.

Table 12a. Sentence answers extracted with and without ontologies (exact answers shown in italics).

<i>Question:</i> What terrorist groups have connections with bin laden?
<i>Answer w/o onto:</i> Two other Pakistani nuclear scientists suspected of having <i>ties</i> with bin laden, Suleiman Asad and Mohammad Ali Mukhtar, been granted asylum in Burma at the request of the Pakistani government
<i>Answer w/ onto:</i> February 23, 1998 Second fatwa against the United States issued by the International Islamic Front for Jihad Against the Hews and Crusaders, a Bin Laden – let Islamic consortium reportedly including radical Muslim leaders from <i>Islamic Jihad</i> , Bangladesh, the Egyptian Al – Gama a al – Islamiyya, and the Pakistani Ansar
<i>Question:</i> Which chemical weapons does al Qaida have access to?
<i>Answer w/o onto:</i> Some of the more prominent claims of an al <i>Qaeda</i> chemical weapons capability are given below
<i>Answers w/ onto:</i> November 11, 2001 American intelligence officials believe that Al Qaeda may have produced <i>cyanide</i> gas at a crude chemical weapons research lab in Derunta, near Jalalabad
<i>Question:</i> What weapons of mass destruction did bin laden attempt to buy?
<i>Answer w/o onto:</i> U.S. intelligence has concerns about Osama bin <i>Laden</i> acquiring nuclear weapons or radioactive material that could be turned into a weapon, but they have no concrete evidence he has done so, officials said Wednesday
<i>Answer w/ onto:</i> The article says that bin Laden may have tried to purchase <i>nuclear weapons</i> in Russia after reading reports that lax security in the former Soviet nuclear weapons complex.

5. QUESTION ANSWERING FROM THE WEB

5.1 Motivation

The design of complex algorithms and the addition of advanced features to state-of-the-art QA systems lead to the goal of extracting more accurate answers from large document collections. The previous sections are concrete illustrations in this sense. But regardless of the complexity of the techniques used for answer finding, the range of questions with potentially correct answers is still limited by the contents of the collection. In principle, the larger and more diverse the document collection, the wider the range of questions for which relevant information exists in the collection. Conversely, the presence of large amounts of text tests the efficiency of the QA system to the limit, calling for lightweight extraction mechanisms based on natural language.

It turns out that there is another way to boost the accuracy of a QA system: increase the amount of unstructured, potential knowledge that it has access to, by adding more documents to the existing collection. When it comes to very large, unstructured, diverse document collections that are publicly accessible and continuously growing, the World Wide Web constitutes a top candidate. This section looks at the benefits of QA from the Web, and describes the architectural changes required to extract answers from Web documents rather than local document collections.

5.2 Approach

5.1.1 Web Search Engines versus Answer Engines

Web search engines offer access to billions of Web documents (Sullivan, 2002) covering virtually every topic of interest. A lot of information is thus sitting behind the search interface of the average search engine, waiting to be uncovered and returned in response to users' queries. Question answering provides a few solutions to limitations of current search engine technology, which include unusefulness of a sizable part of the output; limited support for specific information needs; restrictions on input syntax for query formulation; and coarse output granularity.

Frequently, search engines return hyperlinks to thousands of documents per query. Given that the majority of the Web search engine users view 10 documents or less (Jansen, 2000), more than 99% of the output is useless. Either relevant information is present among the very first hits, or the search is deemed to be a failure. In a more general context, finding accurate information easily and fast is becoming a de-facto requirement in both time-sensitive domains such as the stock exchange, and in regular tasks such as writing a report. Consequently, there is a need for specific (*What was the value of the Nasdaq at the New York Stock Exchange fifteen days ago?*) rather than general information (*find anything about the stock*

market). With present search technology, there is limited support for specific information needs. Many search engines try to address the problem by developing advanced interfaces to support complex, Boolean queries. However, few queries submitted on the Web contain Boolean operators, and usage of Boolean operators actually has very small impact on the results (Jansen, 2000).

Output granularity is another disadvantage of search engines. As the Web continues to grow, it is increasingly likely that a broad range of specific information needs are answered by a few short fragments of Web documents. Search engines return hyperlinks to full-length Web documents, possibly accompanied by a document fragment in which keywords are highlighted. To actually find relevant information, users must read the first documents. Even if relevant information is inside the first hit returned, finding it can still be time-consuming if the document is very long.

QA systems act as advanced interfaces for Web searching, permitting users to use natural language and thus concentrate on what they are actually looking for, rather than on the artificial constraints imposed by a particular syntax. Submitting written questions to an answer engine is easier than building an artificial query from a set of keywords organized around various operators. Furthermore, users are presented directly with a small set of short answers, which are easier to read and evaluate than a set of full-length Web documents.

5.1.2 Extracting Textual Answers from the Web

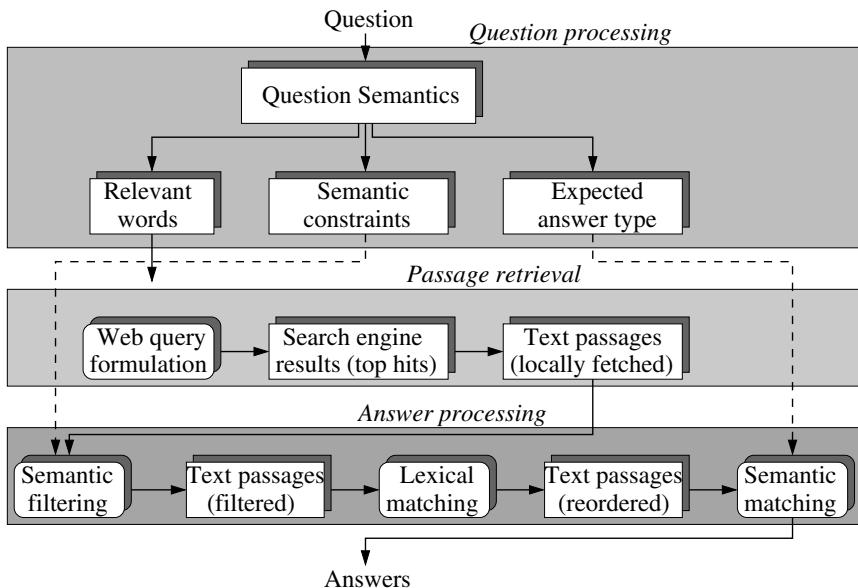


Figure 4. System architecture for answer extraction on the Web.

As shown in Figure 4, the Web QA system first derives the question semantics in terms of relevant words, semantic constraints expressed in the question, and expected answer type (Pasca, 2001a). To this effect, the submitted question string is transformed into a relational question representation, obtained from the question parse tree.

Relevant words are selected from the question terms, after examination of their part of speech and removal of stop words (prepositions, articles etc.). For instance, the relevant words for the question “*Who was the first American in space?*” are *first, American, space*. Semantic constraints include date constraints (“*How much did Manchester United spend on players in 1993?*”) or usage of very specific concepts (“*What two biochemists won the Nobel Prize in medicine in 1992?*”). Another task of the question processing module is to classify the expected answers into one or more known categories, i.e. *answer types*. For instance, the first question from above expects a *monetary value* as answer type (e.g., *Pounds 53,000, \$20,240*) while the second question expects a *person's name* (e.g., *Edwin Krebs, Peter Horne*).

The system builds queries to access the Web, e.g. (*first AND American AND space*) for the question “*Who was the first American in space?*”. Queries are submitted to online search engines. The passage retrieval module actually submits the queries, fetches the highest-ranking Web documents returned by search engines and processes the documents (e.g., to remove HTML tags). The set of Web documents is then reduced to a subset of relevant text passages.

Table 13. Example of sentence answers extracted by the QA system from the Web for TREC questions. Exact answers returned are shown in italic.

<i>Q899: How tall is the Sears Building?</i>
<i>A:</i> the 1,454-foot Sears Tower remained the world's tallest building for 23 years, when it was surpassed by the 1,483-foot Petronas Twin Towers in the Malaysian capital city Kuala Lumpur, although there is some debate (mostly sparked by people who live in Chicago)
<i>Source:</i> www.education-world.com via Google
<i>Q913: What year did the Titanic sink?</i>
<i>A:</i> The Titanic sank April 15, 1912 as a direct result of a collision with an iceberg
<i>Source:</i> www.webtales.com via AltaVista
<i>Q975: When was the first liver transplant?</i>
<i>A:</i> The first successful liver transplant was performed in 1963 by Dr. Thomas Starzl, of the University of Colorado Health Sciences Center, Denver, Colorado (USA)
<i>Source:</i> www.transweb.org via AllTheWeb

The answer extraction module consists of consecutive natural-language filters applied to the text passages. Semantic constraints derived from the question serve to

filter out those passages that cannot contain a correct answer. For instance, passages that do not contain any instance of the expected answer type are discarded. Immediate benefits include precision boosting and better time performance. The filtered passages are passed to a lexical matching component. The latter approximates question relational information through a set of lightweight proximity-based heuristics, which detect the passage fragments most likely to contain a relevant answer. Afterwards, a semantic matching component makes use of more advanced, yet lightweight processing based on the expected answer type. The output is a small set of short answer strings, extracted from the Web documents and presented to the user. Table 13 illustrates answers extracted from the Web in response to TREC questions via different search engines - Google, AltaVista and Fast/AllTheWeb.

5.3 Evaluation

To assess the impact of QA on the outcome of Web searches, we tested our system on the 500 questions from the TREC-10 QA track. Since the experiments are performed on the Web, the document collection is about 1000 times larger than the static collection used in the QA track. Furthermore, the heterogeneous nature of the Web, together with the limited indexing power of the search engines, make it impossible to ensure that all test questions are answered by at least one Web document. Broken links, communication failures, Web server maintenance procedures are technical aspects that hardly affect a static collection such as TREC-10, but cannot be ignored in a dynamic environment such as Web search.

The QA system accesses the Web separately via three search engines, namely AltaVista (www.altavista.com), Google (www.google.com), and Fast/AllTheWeb (www.alltheweb.com). In particular, Google introduced novel ranking algorithms based on page and link analysis (Brin and Page, 1998), and emerged as a leading search engine in terms of size (Sullivan, 2002). In response to a query, all engines currently return a list of hits (links to Web documents), together with a small context string under 200 bytes, extracted from each document. Our QA system fetches and explores the contents of the top hit documents from the search engines. Currently, the default number of hit documents thus fetched and explored is set to 20. The remaining hits are ignored, following the experimental observation that the majority of the Web users view 10 hit documents or less (Jansen, 2000). Table 14 illustrates the impact of our system on the quality of Web searches for the TREC-10 questions.

Table 14. Impact of QA on Web searches for TREC test questions.

	QA from Google	QA from Alta Vista	QA from AllTheWeb
MRR	0.37	0.33	0.30
Q-top5	256/500 = 0.51	236/500 = 0.47	203/500 = 0.40

Due to heterogeneity, the problem of the credibility of the information sources is more apparent on the Web than on local collections. Table 15 provides an illustration. Information is sometimes incomplete (e.g., the first and second answers) or even contradictory (e.g., the third, fifth and sixth answers). QA systems need to attack the task of choosing the correct answers in the presence of noisy or even deceiving documents, especially in intelligence gathering applications.

Table 15. Incomplete or contradictory answers retrieved from the Web for “How many Americans were killed in the Gulf War? ”.

A1:	Fewer than 300 Americans were killed in the Gulf War. (www.fpri.org)
A2:	By the way how many Americans were killed in the Gulf War? Just over 100, remember? (www.scrappleface.com)
A3:	148 Americans were killed in the Gulf War, and 458 were wounded. (www.geocities.com)
A4:	Approximately 350 US and allied military personnel were killed in the Gulf War. (www.mapw.org.au)
A5:	In all, only 313 American troops were killed in the Gulf War, while the fourth largest army in the world – that of Iraq – was left devastated. (www.nationalcenter.org)
A6:	In the Gulf War, less than one one-hundredth of one percent of the 537,000 Americans participants died. (www.allSoulsnyc.org)

6. MULTIMEDIA QUESTION ANSWERING

6.1 Motivation

In addition to textual information, recently the Web has grown to include a plethora of other useful multimedia information, such as images, sounds, movies. Following the common saying that “*a picture is worth a thousand words*”, it seems like a natural progression for our Web-based QA system to retrieve multimedia answers in addition to textual answers. For example, the textual answer to the TREC question: “*Who was the first Triple Crown Winner?*” is contained in the following text snippet retrieved by our QA system: “*War emerged as a two - year - old of frightening brilliance several months after Sir Barton became the first Triple Crown winner.*” While this textual answer is technically correct, the amount of information provided can be significantly increased with additional multimedia answers: the image retrieved by the QA system (see Table 18) shows not only a picture of Sir Barton, but also other additional details ranging from the winning horse to fashion details about the jockey costume.

The Web QA system is extended to extract image answers in addition to text. Unlike other image retrieval systems (e.g. Google Images), our QA system extracts multimedia objects as *enhancements* to textual answers not as stand alone objects.

This allows us to preserve all the advantages of our textual QA system and also use the textual answer as a justification for the multimedia objects retrieved. The approach can be immediately extended to other multimedia objects, such as sound or movie files.

6.2 Approach

6.2.1 Architecture for Multimedia Question Answering

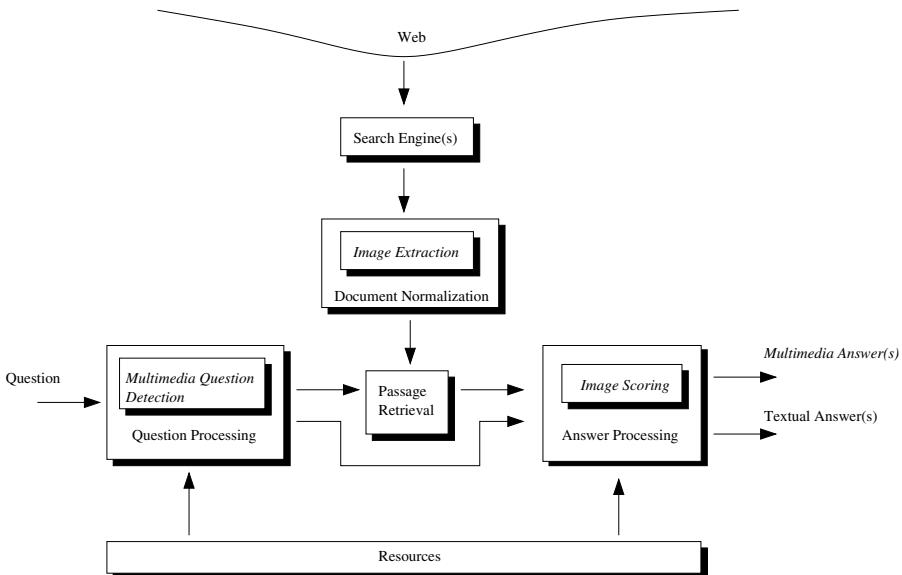


Figure 5. Multimedia Question Answering architecture.

The multimedia QA system architecture described here is a direct extension of the architecture introduced in the previous section. Figure 5 overviews the new modules added to the existing QA system. The *Multimedia Question Detection* module detects when a multimedia answer is appropriate for a question. For example, an image answer is useful for definition questions like: “Who is Sharon Watkins?”, but is useless for the question: “How many calories in a glass of wine?”. If the output of this module is positive, the answer data structure is extended with a new slot that will store the Uniform Resource Identifier (URI) of the multimedia answer next to the textual answer. The *Image Extraction* component extracts image URIs from HTML documents before they are converted to a canonical text representation. Finally, the *Image Scoring* module ranks candidate image answers and selects the best solution. The remaining of this section describes each of the introduced modules in detail.

6.2.2 Detection of Multimedia Question Types

The detection of multimedia questions translates into an analysis of which answer types *might* accept multimedia answers. We were surprised to realize that most answer types can be mapped to images, which is a further indication about the utility of multimedia QA systems. In fact, the only answer types in our ontology that are not relevant for a multimedia QA system are numerical answer types such as: NUMBER, QUANTITY, DATE, TIME, PRICE, and PERCENT. In order to keep the architecture scalable, each concept in the answer type ontology was assigned a Boolean flag which indicates if the system should search (see Subsection 5.2.2) for multimedia answers.

6.2.3 Image Extraction

The starting point for the multimedia QA system, the Web QA system, converted all retrieved HTML documents into a canonical text format, which initially did not allow for the inclusion of multimedia objects. To alleviate this problem we have included an *Image Extraction* module in the Document Normalization stage of the system. The Image Extraction works in the following three stages:

1. A *shallow HTML parser* extracts image information from the original HTML documents. We currently extract the following information: *image URL*, which uniquely identifies the image in the Web address space, and *image size*, including image width and height. Image position is implicitly preserved in the normalized document, because we store each extracted image at the same relative position in the normalized document.

2. All extracted images are passed through a series of filters that attempt to separate content images from images used for the HTML document layout. The following filters are currently used:

(a) Size filter. An image is discarded if any of its dimensions is smaller than a certain minimal threshold. We have obtained the best results when the threshold is 20 pixels. This filter identifies images that are typically used as “bullets” to mark the beginning or end of a paragraph.

(b) Ratio filter. An image is discarded if the ratio $\frac{\text{MAX(width, height)}}{\text{MIN(width, height)}}$ is greater

than a given threshold, e.g., 8. This filter detects images that are typically used to emphasize the borders of the various HTML canvases.

(c) Name filter. This filter discards images whose name contains specific strings we identified as typical layout image names. The most common sub-strings we discovered are: “dot”, “arrow”, “null”, “pixel”, “spacer”, “clear”, “blank”, “bullet”, “logo”, “banner”, “icon”, “bar” and “button”.

3. The images that passed the previous filter are stored in the normalized document, preserving the relative document position. Only these images are considered as candidate multimedia answers.

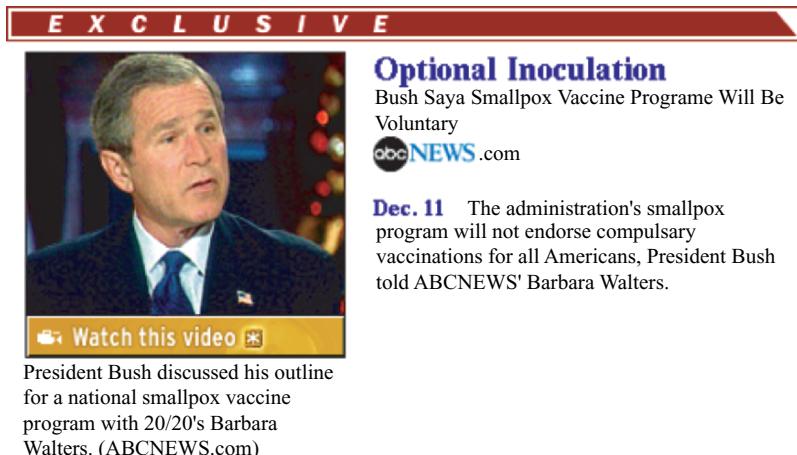


Figure 6. Image filter example.

Figure 6 shows an example of the image filters in action. The top banner (“Exclusive”) is eliminated by the ratio filter. The broadcasting company’s logo is discarded by the size filter, and the last layout image, the “Watch this video” button, is removed by the name filter (the image name contains the string “bar”).

6.2.4 Ranking Multimedia Answers

The ranking heuristic employed for multimedia answers is a direct extension of the scoring metrics we use for textual answers. Image answers are scored relative to their proximity to the corresponding textual answers using the following algorithm:

1. Find the textual answer center, which is defined as the *exact answer* if the question has an entity answer type (e.g., ORGANIZATION, DATE etc.), or as the token located at an equal distance from the outermost left and right keywords otherwise. For example, for the question: “Who was the abolitionist who led the raid on Harper’s Ferry in 1859?” the center of the textual answer: “On October 16, 1859, an armed band of abolitionists led by John Brown attacked the arms arsenal of Harper’s Ferry...” is the named entity “John Brown”.

2. Sort all image answers located within a certain span of the textual answer center based on their proximity to the textual answer center. The sort operation gives the ranking of the candidate image answers.

6.3 Evaluation

For the evaluation of the multimedia QA system we have selected 100 questions from the TREC-11 question set for which we found textual answers using the Web-based QA system. We limited our evaluation set to 100 questions only due to the complexity of the evaluation process, which required human verification for all image answers. The system was configured to report the top 10 answers found for

each question. For each answer position we have extracted the *sentence answer*, which consists of the sentence where the textual answer is located, the *exact answer*, which is only the text snippet that contains the actual answer, and the *image answer* which is the multimedia object associated with the textual answer.

Table 16 shows the Mean Reciprocal Rank scores for all these answer classes. The table indicates that the image answer score is quite close to the exact answer score, which indicates that the information provided by the textual exact answer is in fact doubled by an additional multimedia object.

Table 16. Performance on exact, sentence, and image answers over 100 TREC-11 questions.

	Exact	Sentence	Image
MRR	0.61	0.69	0.48

The extraction of multimedia answers is obviously influenced by the availability of multimedia information. An analysis of our test questions indicated that in many cases when the system did not return an image answer, there was in fact no image answer in the corresponding documents. To evaluate the accuracy of the system on actual multimedia documents, we removed from the original set of questions the questions for which the returned documents contained no multimedia answers. The system performance on the remaining 76 questions is shown in Table 17. Table 17 shows that the multimedia score is approximately 15% higher than the original score, which indicates that the image extraction performance is almost as good as the accuracy of the textual system for sentence answers.

Table 17. Performance over the questions which have image answers in the returned documents.

	Exact	Sentence	Image
MRR	0.59	0.72	0.65

To give an indication about the utility of a multimedia QA system we show some of our favorite textual answers with the corresponding image answers in Table 18. It is immediately obvious that the image answers complement well the textual answers. For example, the first textual answer in Table 18 contains the location name, but only the image answer shows that the battle place was actually a monastery. When our analyst evaluated the second image answer in Table 18, he remembered that he had actually seen that movie, a memory not triggered by the textual answer previously seen.

Table 18. Examples of textual and image answers.

<i>Question:</i> "Where was the battle of Alamo fought?"
<i>Answer:</i> "It was originally called San Antonio De Velaro, but later renamed the Alamo."

<i>Question:</i> "Which film received the first best picture Academy Award?"
<i>Answer:</i> "Wings received the first Best Picture/Production Academy Award in 1927."

<i>Question:</i> "Who was the first Triple Crown Winner?"
<i>Answer:</i> "War emerged as a two-year-old of frightening brilliance several months after Sir Barton became the first Triple Crown winner."


7. CONCLUSION

In order to cope with more difficult tasks, and reliably answer a wider range of questions, QA systems must be enhanced with an array of advanced features. The scope of the advanced features presented in this chapter demonstrates once again the complexity of the QA task. The addition of a new feature often entails pushing the QA task further into one of its many related domains, thus inheriting associated advantages and complexity issues. For example, one relies on knowledge representation and reasoning, to assess the correctness of the retrieved answers; and on the exploitation and development of new linguistic resources, to derive the knowledge on which other modules can operate. Thus, answer justification becomes possible by using the formalism of a logic prover. Similarly, one relies on information extraction, smart retrieval techniques and knowledge acquisition from open text, in order to process questions of different types; and on the exploration of alternative technologies, to move to document spaces like the Web. Future QA research will focus on multimedia documents, through systems that extract information from text, video and speech. Since in many practical applications QA systems must perform very well within narrow domains, another research path is the guided “tuning” of the QA systems for a fixed domain.

8. REFERENCES

- Brill, E., Banko, M., Dumais, S. and Ng., A. (2001). Data-intensive question answering. In *Proceedings of the 10th Text Retrieval Conference (TREC – 2001)*, Gaithersburg, Maryland. NIST.
- Brin, S. and Page, L. (1998). The anatomy of large scale hypertextual Web search engine. In *7th International World Wide Web Conference*, Brisbane, Australia.
- Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database and Some of its Applications*. MIT Press.
- Hobbs, J., Stickel, M., Appelt, D. and Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, 63: 69-142.
- Jansen, B. (2000). The effect of query complexity on Web searching results. *Information Research*, 6(1), October.
- Lin, C.Y. (2002) . The effectiveness of dictionary and web-based answer reranking. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*, Taipei, Taiwan, August.
- Miller, G., (1995). WordNet: a lexical database. *Communications of the ACM*, 38(11): 39-41.
- Moldovan, D., Girju, R. (2001). An interactive tool for the rapid development of knowledge bases. *International Journal on Artificial Intelligence Tools*, 10(1-2): 65-86.
- Moldovan, D., Novischi, A. (2002). Lexical chains for questions. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002)*, Taipei, Taiwan, August.
- Moldovan, D., Rus, V. (2001). Logic form transformations of wordnet and its applicability to question answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational linguistics (ACL-2001)*, Toulouse, France, July.
- Moldovan, D., Pașca, M., Harabagiu, S. and Surdeanu, M. (2002). Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, Philadelphia, Pennsylvania, July.
- Moldovan, D., Clark, C., Harabagiu, S. and Maiorano, S. (2003). COGEX: A logic prover for questions answering. In *Proceedings of the Human Language Technology Conference*, Edmonton, Canada, May.
- Pașca, M. (2001a). *High-Performance, Open-Domain Question Answering from Large Text Collections*. Ph.D. Thesis, Southern Methodist University, Dallas Texas, December.

- Paşa, M., (2001b). Unveiling next generation search technologies: Answer extraction on the Web. In *Proceedings of the 2nd International Conference on Internet Computing (IC-01)*, Las Vegas, Nevada, June.
- Paşa, M., (2002). Processing definition questions in an open-domain question answering system. In *2002 AAAI Spring Symposium Series*, Stanford, California, March.
- Sullivan, D. (2002). Search engine sizes. <http://www.searchenginewatch.com>.
- Voorhees, E.M. (1999). The TREC-8 Question Answering track report. In *Proceedings of the 8th Text Retrieval Conference (TREC-8)*, Gaithersburg, Maryland. NIST.
- Voorhees, E.M. (2002). Overview of the TREC-2002 Question Answering track. In *Proceedings of the 11th Text Retrieval Conference (TREC-8)*, Gaithersburg, Maryland. NIST.
- Weischedel, R., Moldovan, D. and Paşa, M. (2002). Overview of the Aquaint definition pilot evaluation. In J. Prange, editor, *AQUAINT Program Phase I Mid-year Workshop*, Arlington, Virginia, December. Advanced Research and Development Activity (ARDA). <http://www.ic-arda.org/InfoExploit/aquaint>.
- Extended WordNet (2003). Available at <http://xwn.hlt.utdallas.edu>.

A STATISTICAL APPROACH FOR OPEN DOMAIN QUESTION ANSWERING

Abstract: This chapter investigates a statistical approach to open domain question answering. Although the work presented in this chapter centers around maximum entropy models, the models required can be modelled using any machine learning approach. To perform question answering, as has been discussed in previous chapters, questions are first analyzed and a prediction is made as to what type of answer the user is expecting. Secondly, a fast search of the text database is performed and the top documents relevant to the query are retrieved. These documents have been annotated automatically using a named entity tagger. Finally, the answer tag prediction and the annotated documents are input to the answer selection stage. Results obtained from a trainable answer selection algorithm are reported.

1. INTRODUCTION

Automatic question answering systems offer the capability to derive an answer which is sufficient and complete without producing all of the surrounding text as well. The earliest reference of an automatic question answering system is (Green *et al.*, 1963). Surprisingly even that early system allowed natural language phrasing of the questions and retrieved the answer from a relational database. In the work done to date in the TREC question and answering track, we are concerned with questions that have a short answer found from a corpus of news articles.

Recent approaches at question answering have the basic structure as shown in Figure 1. Here, the question is input and classified as asking for a named entity from a small list of categories. Additionally, the question is filtered into query terms during which the query terms may themselves be expanded to include the most common co-occurring words with the query (Local Context Analysis (Xu and Croft, 1996). Then, the query is presented to the information retrieval (IR) engine for document retrieval. This engine, given the query, looks at the database of documents and outputs the most relevant documents or passages. The IR engine acts as a ‘fast’ match for the system, reducing the number of documents to be examined from about 1 million for the TREC database to about 70-200 documents. Performance on TREC queries saturates at about 70 documents so that increasing the number of documents significantly is not beneficial. Although the answer pattern is typically available in the top 70 documents, a good answer with proper support for the question may require many more documents. Some systems (Moldovan *et al.*, 2000) and more recently (Yang and Chua, 2002) have tackled this problem using feedback loops which are controlled by a complex mechanism which decides when to continue the

search for good passages. In the system presented below, we limit our attention to the first 70 documents which are retrieved by the IR system.

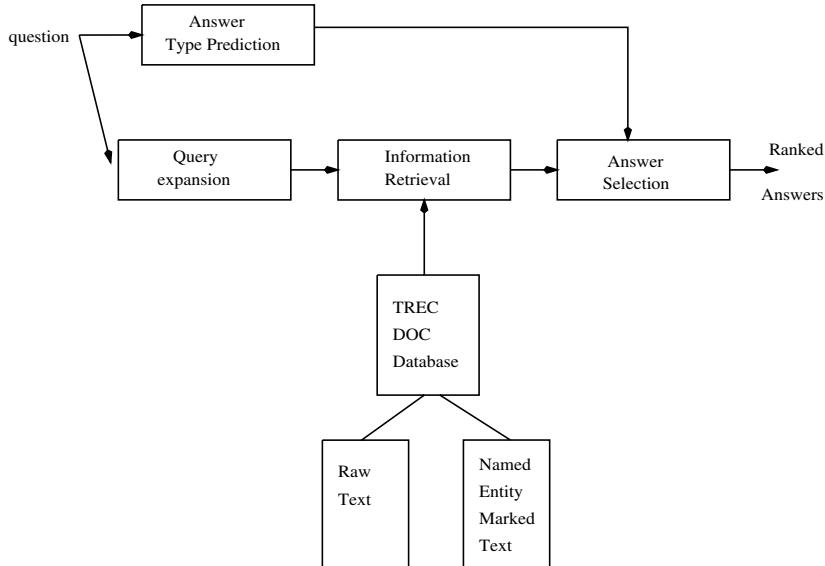


Figure 1. A Common Architecture for Question Answering.

The next stage is to select the exact answer, given the information about the answer class and the top documents. We examine closely the words of the question, their order, and their distances from each other and thus we term this stage as the detailed match. Minimizing the various distance metrics applied over phrases or windows of text results in the best scoring section that has a phrase belonging to answer class. This window represents the best scoring answer sentence, and finally we derive chunks which are ranked using a maximum entropy algorithm.

In this chapter we will discuss the answer tagging and answer selection models in detail, but there are significant other tools required for the question answering system presented here. These tools are:

- **Tokenizer** (or sentence and word segmenter). Generally for English, we can do with white-space word segmentation so that the real problem is segmentation of a document into sentences.
- **Indexer**. For efficient processing of documents, an inverted index is built. Details on index building and usage can be found in (Frakes and Baeza-Yates, 1992).
- **Query Expansion**. This tool does LCA based expansion of the query terms (Xu and Croft, 1996).
- **Query Engine**. This tool searches the document database using the inverted index and the expanded queries.

- **Parser.** The parser is a statistical parser similar to the one described in (Ratnaparkhi, 1999). It uses maximum entropy probability models to resolve ambiguities in parsing decisions. It has been trained on approximately 10,000 sentences of the Penn Treebank Wall St. Journal corpus (Marcus *et al.*, 1994). Given raw text, the parser produces phrase structure trees consisting of both part-of-speech tags and constituent labels.

Evaluations in question answering held by the National Institute of Standards and Technology (NIST) have provided documents relevant to a question so that researchers that do not have access to indexers and query engines can still participate. The use of the parser in this question answering system is to (a) identify the focus word of the question when it exists, (b) to provide dependency information for answer scoring and (c) to provide chunks that are potential answers.

1.1 Semantic Tags

The semantic tags that are used in the question answering system presented here are broken along five major categories:

- **Name Expressions.** Person, Salutation, Organization, Location, Country, Product
- **Time Expressions.** Date, Date-Reference, Time
- **Number Expressions.** Percent, Money, Cardinal, Ordinal, Age, Measure, Duration
- **Earth Entities.** Geological Objects, Areas, Weather, Plant, Animal, Substance, Attraction
- **Human Entities.** Events, Organ, Disease, Occupation, Title-of-work, Law, People, Company-roles

In addition, questions that do not fit these categories are categorized as desiring a “**PHRASE**” answer. Performance varies greatly among the classes with the highest error rate in the PHRASE class. Why are there only ~30 categories in our system? We have observed that semantic categories follow a trend similar to Zipf’s Law and the categories listed above capture about 50% of the questions as measured on a large set of internet question queries. Several approaches have been discovered as being effective against PHRASE type queries and these will be presented below.

1.2 Question Answering Model

Consider the following TREC-8 question answer:

<p>Q: Who is the author of the book, “The Iron Lady: A Biography of Margaret Thatcher”</p> <p>A: THE IRON LADY A Biography of Margaret Thatcher by Hugo Young (Farrar, Straus & Giroux).</p> <p>EXACT: Hugo Young</p>

The sentence answer is correct and often appropriate for display to the user but in TREC evaluations we consider the exact answer as “Hugo Young”. Generally, an answer can be decomposed as:

$$a = a_d + a_s \quad (0.1)$$

where a_d is the desired answer and a_s is the supporting evidence for the answer. Words comprising the answer support are generally found in the question.

In this chapter, a mathematical basis for open-domain fact-based question answering is developed. The mathematical framework is tractable and provides close to state-of-the-art performance without tuning to specific types of questions. Most systems use a component approach and these components can be interpreted in this framework and results in a clear understanding of the assumptions that are made in similar systems.

The framework should uniformly address all short answer fact type questions without resorting to heuristics for each question type. This is not to say that such heuristics are not the optimal method of answering a specific question type. Such heuristics work best in the absence of sufficient training material, but when training data becomes large, the task of writing question specific rules becomes burdensome. Take for example, the category of questions requiring a person as an answer – it is tempting to say its all questions starting with ‘Who’. Very quickly we run across: “Which doctor …”, “What actor …”, “In 2000, who was …”, “To whom was the ball passed …”. Also “who” can be used as a pronoun in a relative clause as in, “Where was the actor, who played Jake, buried?”. Automatic methods propose, evaluate and assign a weight to each rule. We are limited in this framework by the availability of data for training models. In creating the training data, the design choices of the number of classes to be used for classifying the question and answer will affect accuracy, not only in terms of confusion between the classes in each of the models, but also in the combination of the models.

The question answering architecture in Figure 1 is similar to other systems (Srihari and Li 1999; Moldovan *et al.*, 1999) participating in TREC-8. Note that the architecture presented above has three separate components which are derived from certain independence assumptions of the question, answer and a hidden variable relating the question and answer. We can derive a more formal mechanism which has absent these independence assumptions. To begin, we seek the conditional probability $p(a | q)$. This model would allow us to predict the answer given the question. However, this model is not tractable currently as there is insufficient data

to directly model this distribution. Instead, we will model the distribution, $p(c | a, q)$, which attempts to measure the “correctness”, c , of the answer and question. This modelling paradigm uses a classification point of view, as opposed to a generative model where we model $p(a | q)$ directly. Further, we can introduce a hidden variable representing the class of the answer, e , (answer tag/named entity) as follows,

$$p(c | q, a) = \sum_e p(c, e | q, a) \quad (0.2)$$

Using Bayes Rule we obtain,

$$p(c | q, a) = \sum_e p(e | q, a) p(c | e, q, a) \quad (0.3)$$

The distribution $p(c | q, a)$ measures the relevance of the configuration **(q,a)** and training data is naturally human judgments of such pairs. $p(e | q, a)$ is the answer tag model which predicts from the question and proposed answers, the “entity” that both satisfy. If we assume that the answer and question are independent and conditionally independent of the entity (note that this is a very poor assumption), we get

$$p(e | q, a) = \frac{p(e | q)p(e | a)}{p(e)} \quad (0.4)$$

where $p(e | a)$ is the Named Entity modelling problem, and $p(e | q)$ is the answer tag modelling problem. In addition, note that the sum in (0.3) is often replaced by a, ‘max’ operation and only a single entity is carried forward. For our current system, we make the assumption that the entity being sought by the question does not depend on the answer string yielding $p(e | a, q) \approx p(e | q)$.

$p(c | e, q, a)$ is the Answer Selection Model. Given the question, the answer and an entity as predicted by the answer tag model, we seek to model the correctness of this configuration. This model is tractable because we have judgment data on 1892 questions from the TREC competitions as well as locally collected question answer judgments from Trivia question answer pairs. In formulating this model, note that there is no shortage of negative data, although the most useful question-answer pairs are those that are nearly correct but judged as incorrect. In order to generate the nearly correct data, a basic question answering system (without the statistical answer selection model) is used to propose answer candidates. The output is then judged and these judgments form the training data for the statistical answer selection algorithm.

At this point, we deviate from our question answering discussion, and discuss the modelling paradigm we refer to as maximum entropy modelling in order to be complete.

2. MAXIMUM ENTROPY MODELS

Maximum entropy ideas are the simplest way of stating what we know, and assuming the least for what we do not know. In the limit of knowing nothing, the best we can do is to assume that each event is equiprobable and that the process is generated by a uniform distribution. By using the term ‘maximum entropy’, we seek to find the distribution that is closest to the uniform distribution that also satisfies some other constraints. These ideas have been pioneered and extended by (Jaynes 1983; Berger *et al.*, 1996; Della Pietra *et al.*, 1995; Ratnaparkhi, 1998a; Papineni, 2001) in the areas of statistical modeling and natural language processing. As also quoted in (Berger *et al.*, 1995), Jaynes’ writes:

“... the fact that a certain probability distribution maximizes entropy subject to certain constraints representing our incomplete information, is the fundamental property which justifies use of that distribution for inference; it agrees with everything that is known, but carefully avoids assuming anything that is not known. It is a transcription into mathematics of an ancient principle of wisdom.”

Maximum entropy principles have been applied to a variety of problems including:

- Parsing (Ratnaparkhi *et al.*, 1994)
- Sentence boundary detection (Reynar and Ratnaparkhi, 1997)
- Prepositional Phrase Attachment (Ratnaparkhi, 1998b)
- Answer type classification (Ittycheriah *et al.*, 2000; 2001)
- Phrase Chunking (Koeling, 2000)
- Named Entity detection (Borthwick *et al.*, 1998; Mikheev *et al.*, 1997)
- Machine Translation (Berger *et al.*, 1996)

A thorough description of maximum entropy is presented in (Berger *et al.*, 1996). For the purposes of this chapter, a brief presentation of the maximum entropy formulation is presented and proofs are left to the references.

2.1 Incorporating Constraints

The uses of maximum entropy modelling in this chapter will involve classification. In this problem, we are given the context of an event and are asked to predict the class. The context is usually referred to as the *history* and contains all the information about the event. For example, in detecting sentence boundaries, the history can be comprised of the current word, the previous word as well as the previous decision. Additional information whether from simple table lookups like

thesauri or complex annotations like word classes requiring a separate statistical models can all be added to the history of the current problem. Several examples of histories are presented in the statistical models developed in this chapter. Histories will be denoted by \mathbf{h} and the set of all distinct histories by H . The predicted classes we will refer to as *future*, are the output symbols. Futures will be represented by f and the set of all distinct futures by F . For example in sentence boundary detection, the outputs could be composed of {CONTINUE, END}. For each word in the sentence, we get a label out of the classifier indicating the status of the sentence boundary detection.

Suppose we wanted to incorporate the knowledge of the case of the next word into the sentence boundary detector. We could define a feature function ϕ as,

$$\phi(h, f) = \begin{cases} 1, & \text{if } w_{i+1} \text{ is capitalized} \\ 0, & \text{otherwise} \end{cases} \quad (0.5)$$

where w_{i+1} is the next token in the stream. This technique of incorporating any useful information into a feature is generic. In our training sample, we can compute the number times this feature fires and given the class information {CONTINUE, END} could form an impression of the probability to associate with each class. Formally, the mean of the feature is defined as,

$$E_{\tilde{p}}(f) = \sum_{h \in H} \sum_{f \in F} \tilde{p}(h, f) \phi(h, f) \quad (0.6)$$

where $\tilde{p}(h, f)$ is determined from the training data. This is not the traditional sense of the mean which requires the exact probability distribution $p(h, f)$, but rather an estimate of the mean using the observed distribution.

The conditional entropy of the model being built, $p(f | h)$ is defined as (Berger *et al.*, 1996),

$$H(p) = - \sum_{h, f} \tilde{p}(h) p(f | h) \log p(f | h) \quad (0.7)$$

where $\tilde{p}(h)$ is determined from the training sample. Then, what we seek is a model for the probability distribution $p(f | h)$ that maximizes this entropy subject to constraints that we observe the means of the features as defined above.

We can show that the only family of models that satisfies these requirements is the exponential family (Cover and Thomas, 1991) (solution to Chapter 11, problem 2),

$$p(f | h) = \frac{1}{Z(h)} \exp \sum_i \lambda_i \phi(h, f) \quad (0.8)$$

where $Z(h)$ is the normalizing constant,

$$Z(h) = \sum_f \exp \sum_i \lambda_i \phi_i(h, f) \quad (0.9)$$

The λ_i are the Lagrange multipliers used in the optimization and ϕ_i 's are the individual feature functions. We next deal with the problem of finding appropriate values for λ_i .

2.2 Parameter Estimation

In (Darroch and Ratcliff, 1972), the authors developed an algorithm to solve for the λ_i above when $\sum_i \phi_i(h, f) = C$. This means that for every class, f , the number of features that fire with context, h , remains a constant, C . A proof of this algorithm was given by (Csiszar, 1989). An improved version of this algorithm where the aforementioned sum need not be constant is given by (Della Pietra *et al.*, 1995). A sketch of the development given here is to take the difference of the new likelihood from that of the old and use the approximation $\log(x) \leq x - 1$ and Jensen's inequality and then show that the result is a polynomial which has one root in the positive real region which can be solved by Newton's method for instance. The full proof of this algorithm establishes the duality between maximum entropy and maximum likelihood which I skip here.

2.3 Two Inequalities

In order to prove the two inequalities that are required in the maximum entropy updates, first we need a mathematical definition of convexity. A real valued function, f , is defined to be convex on an interval, R , if and only if

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \quad (0.10)$$

for all $\{x, y\} \in R$ and $0 \leq t \leq 1$. Furthermore, for differentiable functions, the following theorem is useful.

THEOREM If $f''(x) \geq 0$ for all $x \in R$, then $f(x)$ is convex on R .

This is a fundamental result from calculus so we state it without proof. To establish $\log(x) \leq x - 1$, form a function $f(x) = -\log(x) + x - 1$. Setting the first derivative equal to 0 yields that the minimum occurs at $x = 1$, and the second derivative is everywhere positive. Thus, the function is convex and the minimum is $f(x) = 0$ from which we get that $f(x) \geq 0$ and the result is proved.

The second inequality is Jensen's Inequality (1906). This is stated in the following theorem.

THEOREM If $f(x)$ is a convex function, and $\sum_{i=1}^N t_i = 1$ where t_i are nonnegative real numbers, then

$$f\left(\sum_{i=1}^N t_i x_i\right) \leq \sum_{i=1}^N t_i f(x_i) \quad (0.11)$$

The conditions on t_i satisfy the requirements of a probability distribution so alternatively, $E(f(X)) \geq f(E(X))$ where X is a random variable with a discrete distribution given by t_i . The proof is stated in (Cover and Thomas 1991) so I omit it here.

2.4 Maximum Entropy Updates

We start from the dual of the optimization problem discussed in (Berger *et al.*, 1995), where we try to maximize the log-likelihood of the observed distribution as predicted by our conditional model, which is defined as,

$$L_{p_\lambda} = \sum_{h=H} \sum_{f=F} \tilde{p}(h, f) \log p(f | h) \quad (0.12)$$

Selecting the probability distribution $p(f | h)$ that maximizes this likelihood is also the distribution that maximizes the entropy as shown in (Della Pietra *et al.*, 1995). The mutual information between a random variables h and f is defined by

$$I(h; f) = \sum_{h=H} \sum_{f=F} p(h, f) \log \frac{p(h, f)}{p(h)p(f)} \quad (0.13)$$

where $p(h, f)$ is the joint distribution of the random variables. Normally for random variables we use upper case letters, but here upper case is reserved for the set of possible outcomes of the random variable. In our case we are estimating this joint distribution by the training data and using the relationship

$$\tilde{p}(h, f) = \tilde{p}(h)p(f | h) \quad (0.14)$$

and assuming a uniform prior, maximizing (0.12) is the same as maximizing the mutual information between the random variables H and F.

In the following derivation, we make λ a vector $\lambda = [\lambda_i]$ and $\lambda\phi = \sum_i \lambda_i \phi_i$. Now, for a δ_λ change in the λ vector, we would like to increase the likelihood of (0.12) or specifically:

$$L_{p_{\lambda'}} - L_{p_\lambda} = \delta_\lambda \geq 0 \quad (0.15)$$

Plugging in the distributions we get:

$$\delta L = \sum_{h \in H} \sum_{f \in F} \tilde{p}(h, f) \log \left[\frac{p_{\lambda'}(f | h)}{p_{\lambda}(f | h)} \right] \quad (0.16)$$

Using the form of the distribution given by (0.8) and that $\lambda' = \lambda + \delta_{\lambda}$

$$\delta L = \sum_{h \in H} \sum_{f \in F} \tilde{p}(h, f) \log \left[e^{\delta_{\lambda}\phi(h, f)} \frac{Z_{\lambda}}{Z_{\lambda'}} \right] \quad (0.17)$$

Bounding now with and rearranging the terms gives

$$\delta L \geq \sum_{h \in H} \sum_{f \in F} \tilde{p}(h, f) \left[\delta\phi(h, f) - \sum_{f' \in F} p_{\lambda}(f' | h) e^{\delta_{\lambda}\phi(h, f')} + 1 \right] \quad (0.18)$$

Following the development in (Della Pietra *et al.*, 1995), define a function

$$\phi^*(h, f) = \sum_i \phi_i(h, f) \quad (0.19)$$

Now $\frac{\phi_i(h, f)}{\phi_i^*(h, f)}$ behaves like a probability distribution and allows application of Jensen's Inequality. Next differentiating with respect to δ_{λ} to find the maximum change in likelihood, we finally arrive at

$$\frac{\partial \delta L}{\partial \delta_{\lambda}} = \sum_{h \in H} \sum_{f \in F} \tilde{p}(h, f) \left[\phi(h, f) - \sum_{f' \in F} p_{\lambda}(f' | h) e^{\delta_{\lambda}\phi^*(h, f')} \right] \quad (0.20)$$

Setting to 0 to find the maximum, and rearranging terms:

$$0 = \sum_{h \in H} \sum_{f \in F} \left[\tilde{p}(h, f) \phi(h, f) - \tilde{p}(h) p_{\lambda}(f | h) \phi(h, f) e^{\delta_{\lambda}\phi^*(h, f)} \right] \quad (0.21)$$

This is a polynomial in $e^{\delta_{\lambda}\phi^*(h, f)}$. The coefficients are all the same sign except the empirical probability term and by Descartes' Rule of Signs, this polynomial has a single positive root. The positive root is desired because the polynomial variable is defined as $e^{\delta_{\lambda}}$ which is always positive. Finding the root then gives us the increment for the parameter λ , which then gives a probability model which increases the likelihood, (0.12) on each iteration step.

2.5 Maximum Entropy Feature Selection

Feature selection is an orthogonal issue to the modelling problem. The support set of the probability distributions for the problems considered here are binary questions (or are converted into a number of bins and binary questions are posed

about membership in the bins). Feature selection refers to the process of selecting from the pool of features (which in most cases is very large), a subset which allows the algorithm to generalize. By generalize, we would like the algorithm to perform equally well on some unseen data as it does on the training data. Choosing more features generally improves training set performance but may not improve performance on test sets. Furthermore, having features which do not improve performance for either training or testing is simply inefficient. Feature selection methods rank the pool of features using some criteria and then selecting those features above a threshold. Incremental field induction algorithms perform this ranking many times, using the previous model as a prior for the model being estimated. In (Della Pietra *et al.*, 1995), the gain of a feature is defined to be the increase in the likelihood of the model from adding that feature. More details about feature selection for the maximum entropy framework is given in (Berger and Printz, 1998).

We now return to the problem of question answering. First, the answer tagging problem is discussed and then the answer selection problem.

3. ANSWER TAG MODEL

For our question answering system setting, a question necessarily seeks an answer. Such questions provide the following information:

- Question Word (who, what, where, when, which, why, how, list, identify, name)
- Question Focus – which is optional, and
- Question Support

The order of these items is not fixed, although a great majority of questions follow the order listed. As an example, consider the following question where the three parts of the question are identified.

Q: *What time of year is the annual balloon festival held?*
 Question Word: *What*
 Question Focus: *time_of_year*
 Question Support: *is the annual balloon festival held?*

Many different constructions of question words can essentially ask for the same answer. For example, “Where”, “In what city”, “Which city”, “What location” could all be used to ask a question about a city. A more detailed breakdown of question words and thus types is possible as is done in (Moldovan *et al.*, 1999). In our system, we directly model the answer tag and skip the step of determining the question type.

3.1 Question Focus

The definition of focus given by Foley in (Brown and Miller, 1999) is,

FOCUS: The nominal corresponding to the question word is the focus of the clause, expressing the new information which the utterance of the clause is expected to provide, while the remainder expresses what is taken for granted.

Foley centres the definition of focus as being an attribute of the answer to a question. Question focus is that word in the question that tells us what the question seeks. In this chapter, we will use *focus* to refer to the question focus and use *desired answer* to refer to the traditional linguistic use of focus.

The difficulty in creating the answer tag model is in the determination of the question focus, and if there exists a focus, in determining the boundary between the focus and the support. This boundary is important because answer tag classification depends on processing the focus to yield information which makes it easier to classify the answer tag as discussed below. Expanding words which are not part of the focus using WordNet (Miller, 1990) adds undesirable noise.

Question focus is not generally present with certain question words, for example, “who” or “when”. The question words for which a focus occurs most of the time are: what, which, how, list, identify and name. The other question words are less ambiguous about the answer they seek and commonly the focus is left out. The focus when present represents a type of class label of the answer. For example, in the question “*What doctor performed the first heart surgery?*”, the focus word is *doctor* and thus any answer must belong to this class.

Often in answer sentences, the focus may not be explicitly present and rather a specific member of the focus will be present (for example, “What disease has the symptom of dizziness?” and the answer “Sufferers of Tourette’s syndrome often complain of dizziness” – here the word ‘disease’ is the focus and is found in the answer as a specific instance of that class: ‘Tourette’s syndrome’).

3.2 Question Support

The support of a question is the remainder of the question that can be used to search for the answer. The support of the question is less important for answer tagging, but can not be ignored completely. In “who” type questions, the length of answer support, the capitalization of the words as well as the lexical words are clues in determining whether a PERSON, OCCUPATION or ORGANIZATION is being sought.

3.3 Question Focus

In order to determine the focus of a question, we use a syntactic parse of the question. Many such parsers have now been developed for English: (Brill, 1993; Collins, 1996; Ratnaparkhi, 1998a). We use the Ratnaparkhi parser because it is also

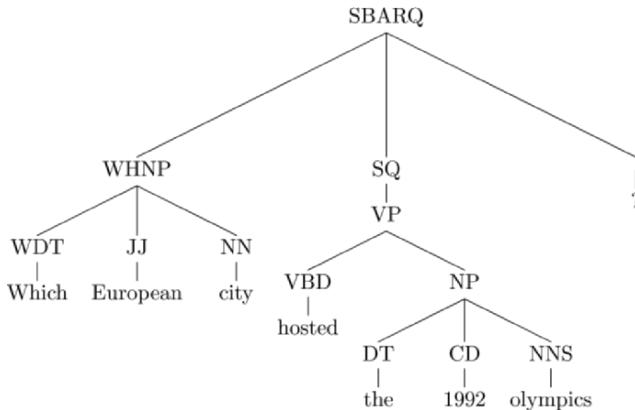


Figure 2. An example syntactic parse.

driven by maximum entropy principles as is most of this chapter and is readily available for use. A parses for “*Which European city hosted the 1992 Olympics?*” is shown below in Figure 2.

The headword of a noun phrase (NP) constituent is usually taken to be the last word in the noun cluster. Thus, the first headword following the question word for the question parse shown above is ‘city’. In questions like “What was the name of the first doctor to do heart surgery?” we must remove the common phrase “the name of”. The three common phrases removed before determining focus are “the name of”, “type of” and “kind of”. Such processing is only required because of the limited amount of data we have for such questions. When sufficient training data is available, a question focus selection model could predict which word or phrase is the question focus. Table 1 lists the processing done to extract the focus of the question.

Table 1. Rules used for generating the focus of a question.

Question Word	Processing
What, which, name, list, identify	question word + headword of first noun cluster
Who, why, whom, when	question word
where	question word + main verb
how	question word plus next word if it seeks an count attribute + headword of first noun cluster
	question word plus the next word if it seeks an attribute
	if question seeks a methodology, then just question word

Table 2. Examples of focus.

Question	Focus
What time of year is the annual balloon festival held?	what time_of_year
What animal jumps around on two legs?	what animal
What kind of animal jumps around on two legs?	what animal
Which metropolitan city houses the largest museum?	which city
Name two kinds of turtles.	name turtles
Where did the queen live?	where live
Where can I find information on palaces?	where find
How many types of dogs are there?	how many types
How long is the river?	how long
How will I fix my car?	how

Table 2 shows a few examples of focus derived from their questions. The first example highlights *WordNet* co-locations. Without the co-location, the focus would be ‘time’ which is clearly not appropriate.

3.4 Training Data

As in any pioneering field, there is significant work to be done to build a corpus driven system. Training data to perform answer tag classification consists of 4.1K questions that have been tagged manually. The TREC-8 queries are used as a validation set and the TREC-9 queries as a test set.

An important measure of the data is the distribution of classes among the question sets. The histogram of the answer tag categories in the data sets is shown in Figure 3. In the experiments to be described below, the training error rate is significantly higher than the test error rate. This effect is explained by the histogram which shows that there is a lot more PHRASE type questions in the training set than in the test set. These questions have the most variability among the classes, and consequently have the highest error rate. By variability, we mean that many question words can be used with this class and that there are numerous types of focus used with this class. The error rate on the TREC-9 test is closer to the error rate predicted in the training data since the histogram of classes in TREC-9 is closer to our training set.

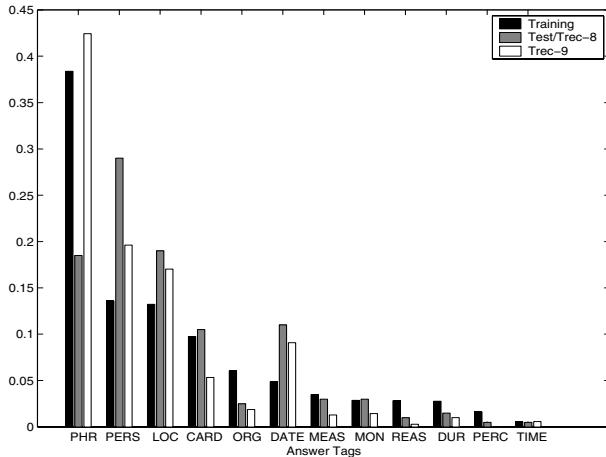


Figure 3. Histogram of Answer Tag Classes.

3.5 WordNet

WordNet (Miller, 1990) is lexical resource for English that is publicly available and widely recognized in the natural language processing community. The use of WordNet in this system is for the following tasks:

- a) thesaurus
- b) hierarchical categorization of words
- c) gloss definition of key terms in the question
- d) phrase detection by using WordNet co-locations

For our current purposes of answer tagging, we use the first sense listed in WordNet. In version 1.7, WordNet has 190K words and co-locations (phrases) arranged in 109K synsets. In a 500 question subset (the first 500 questions) of the TREC-9 test, there are 330 questions which have a focus (66%). Of these focus phrases, the number expanded through WordNet is 318 or 96.4%. The phrases that could not be expanded are mostly mistakes in finding the true focus and there was only one instance of a coverage problem: “cruise_line”. Even though the focus is found in WordNet the first sense is often not correct as in “Which star of *Sleepless in Seattle* acted in a TV series?” The first sense of ‘star’ is the celestial body and here the focus expansion fails due to the lack of word sense disambiguation being applied to the question focus.

3.6 A Maximum Entropy Answer Tag Model

The features used in answer tag classification are functions of the question words and other semantic information added to each question. A feature stream is a logical grouping of information. It is completely equivalent to have a single stream and annotate the information with a label so as to separate the question words from the annotations. However, the method of putting the information in separate streams results in much more readable features. Additionally, the method affords a stricter control in bigram formation since we can focus the bigrams to come from only a few preselected streams. The following feature streams were considered:

- a) Question Words And Annotations
- b) Part-Of-Speech
- c) Focus + WordNet Hypernyms

The annotations on the question words included both position of the question word in the question (bqw - begin question word, ...) as well as part of speech (POS) information of the following word. Addition of the POS information to the question word is a method to do some level of disambiguation. If a verb follows the word ‘when’ then very likely this question is about a DATE or TIME quantity. The numbers on the POS tags represent their distance from the question word. In this manner, occurrences of a tag far away from a question word will not influence the answer tag as much. As an example of the feature streams, consider the following streams for the questions discussed above:

Words	Which which_bqw which_JJ European city host 1992 olympics
Pos	WDT_0 which_WDT JJ_0 european_JJ NN_1 city_NN VBD_2 hosted_VBD DT_3 CD_4 1992_CD NNS_5 olympics_NNS
Wn	European city metropolis urban_center municipality urban_area geographical_area geographic_area geographical_region geographic_region region location entity metropolis urban_center city_center central_city financial_center hub civic_center municipal_center down_town inner_city

In this model, the primary features are the question words themselves. The prior information from the training data gives any algorithm 40% accuracy at least by guessing PHRASE for all questions. Two error measures are made: (a) the accuracy of the system and (b) the average F-measure. The best performance for the system using words only is selecting about 280 features yielding an error rate of 20%. The best performance when combined with POS information is selecting about 526 features yielding an error rate of 16%. Since the training data is limited, adding the *WordNet* projection of the focus word is found to be very useful. The application of the count cutoff in pruning features (i.e. any feature with a count below the threshold is not even considered) selects only those features that are well represented. As described earlier, there are a variety of ways to ask for a person. These can all be

efficiently collapsed by the *WordNet* hypernym expansion as they all have a node ‘person’ in the hierarchy. A few examples of the features selected are shown in Table 3. The best performance when combined with *WordNet* information is selecting about 570 features yielding an error rate of 10%.

Table 3. Features selected using Words, Part-of-Speech, and WordNet.

History Stream	History	Future	Weight
Wn	many	CARDINAL	6.85
Pos	why_WRB	REASON	33.10
Wn	region	LOCATION	1.80
Words	who_V	PERSON	5.15
Words	when_V	DATE	16.58
Wn	period	DURATION	6.76
Wn	mortal	PERSON	3.36
Words	what_bqw	PERSON	0.27
Wn	Number	CARDINAL	21.32
Words	what_V	LOCATION	0.29
Wn	decade	DATE	18.06

Addition of higher order features, for example bigrams between words in the ‘Word’ stream and those in the *WordNet* stream (Wn) improves performance further. The features of this type that are added are shown in Table 4. Essentially, here we are allowing the algorithm to tie information like ‘what person’ together. These features are stronger than two unigram features ‘what’ and ‘person’ to predict the PERSON answer tag. The performance curve is shown in Figure 4. The best performance is obtained by selecting 552 features yielding an error rate of 9%.

Table 4. Additional Features selected using Bigrams of Words and WordNet.

History Stream	History	Future	Weight
Words,Wn	many COUNT0	CARDINAL	6.87
Pos	why_WRB	REASON	33.04
Wn	region	LOCATION	5.75
Words	who_V	PERSON	4.09
Words,Wn	when_V DEFN0	DATE	17.31
Wn	period	DURATION	7.66
Wn	government	LOCATION	9.56

History Stream	History	Future	Weight
Wn	soul	PERSON	0.91
Words,Wn	what_bqw number	CARDINAL	20.73
Words	what_bqw	PERSON	0.06
Words,Wn	what_V DEFN0	LOCATION	0.27
Words,Wn	how_JJ COUNT0	CARDINAL	1.57

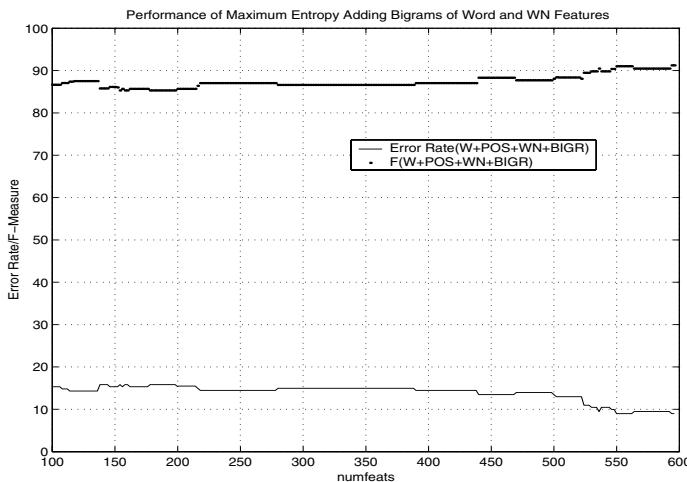


Figure 4. Performance using bigrams of words and WordNet.

3.7 Error Analysis of the Answer Tagger

A few of the errors of the best system are presented in this section. The streams are prefixed with the name of the stream. The future and the best guess have their probability displayed below.

Words	How how_bqw how_RB far be Yaroslavl from Moscow
Pos	WRB_0 how_WRB RB_0 far_RB VBZ_1 is_VBZ JJ_2 yaroslavl JJ IN_3 from_IN NP_4 moscow_NP_5 ._5
Wn	far DEFN0 COUNT0 QLENMED
Future	MEASURE 0.164192
Best Guess	PHRASE 0.237099

Words	Name name_bqw name_DT designer shoe that spawn million plastic imitation know jelly
Pos	VB_0 name_VB DT_0 NN_1 designer_NN IN_2 DT_3 NN_4 shoe_NN WDT_5 that_WDT
Wn	the designer interior_designer interior_decoration house_decoration room_decoration decoration specialist specializer expert individual someone somebody mortal human soul organism being living_thing entity causal_agent cause casual_agency DEFN0 COUNT0 QLENLARGE
Future	PERSON 0.294415
Best Guess	PHRASE 0.446929

The first error comes from insufficient data to model that ‘far’ indicates a measure. The count cut-off threshold set in this experiment is 6 and there were only 4 occurrences of the word ‘far’ with ‘How’ as the question word. The second error is a result of ‘Name’ question word co-occurring with the PHRASE class much more often than it occurs with PERSON. However, in both these cases we note that the probability estimate of the truth is significant and we can exploit this in the future. The modelling equation was a sum over all entities and it remains future work to estimate the effect of this on the overall performance.

3.8 Related Work

Several groups have undertaken either rule based or statistical techniques for answer tagging. In (Srihari and Li, 1999) they use an expanded list of categories to type questions as well as in tagging named entities. In (Moldovan *et al.*, 1999), they develop rules to extract a large hierarchy from WordNet tops. Closer to the approach discussed here is (Li and Roth, 2002) where a hierarchical approach of 6 major categories and 50 subcategories are used in a machine learning approach. A similar effort although rule based is (Yang and Chua, 2002), where they classify questions into 6 major types and then into 54 subtypes.

3.9 Maximum Entropy Answer Tag Conclusion

The results above indicate that *WordNet* expansion of the focus yields the greatest improvement in performance (reducing the error rate from 16% to 10%, a relative improvement of 37.5%). In this section we have examined the problem of modelling the answer tag prediction. Answer tag modelling is but one part of the question answering system. Errors caused by this component are not very significant at this time because for the entire system to fail due to answer tag modelling all the other components must have worked correctly.

4. ANSWER SELECTION

In this section, we create a probabilistic model to do answer selection. Two related efforts are (Berger *et al.*, 2000) where probabilistic models for answer finding are investigated, and (Ng *et al.*, 2000) where a decision tree model is derived for answering questions in a reading comprehension task. Answer finding differs from question answering because those systems seek to find the nearest question-answer pair to a given question and then provide the answer to the question. Answer finding is thus closest to the FAQ-finder system in (Burke *et al.*, 1997). Finally as discussed before, in reading comprehension it is the goal to find the sentence that answers the question from a single document that is known to contain the answer. The goal of extracting an exact answer chunk is significantly harder than extracting the best sentence, although as many have observed there is significant redundancy in large collections and this may be useful in improving question answering performance.

In terms of the system shown in Figure 1, the question has been processed and used for a search of relevant documents. It is now desired to find a more concise answer than the document. The particular type of question answering we are interested in this chapter requires the answer to be relatively short (less than 10 words). Also, note that the answer is *extracted* from the text and thus no modification is allowed on the chunk.

Conceptually, answer selection should operate on the entire database of text but this is neither practical nor necessary. The standard technique is to extract text from documents retrieved by a retrieval engine. Extending that reasoning, a logical document could be defined to be a sentence and then we seek to retrieve the most relevant sentences. This provides a rather simple system to rank the sentences. This notion will be used once more to rank chunks of a sentence. However, finding the answer support is often not possible and thus some variation is needed. The variation is to allow all chunks to inherit the rank and features of the parent sentence and additional features about the chunk are added in order to rank them. Thus, a rather simple system proposes chunks and then a model is created to measure the correctness of the chunk. This model will include features that were used in ranking the sentence and the chunk initially and thus two modelling steps have been removed and the problem is treated as one that requires a re-ranking model for the chunks proposed by a simple system. Each step in such a pipeline must ensure that the correct answer is in the list (otherwise it is termed a fast match error), although the ranking of the list may be poor. Depth requirements of the list will be investigated below.

Next we will discuss the answer sentence extraction, chunk extraction, and the statistical model for chunk ranking.

4.1 *Extracting the Answer as a Sentence*

The corpus (and thus retrieved documents) has already been sentence boundary detected. Figure 5 shows the process of ranking sentences. The documents returned

by the IR engine are read in order and each is processed in their sentence order. A window of three sentences, centered on the current sentence is formed. This window reduces the effects of various linguistic phenomena – anaphora, definite references and answer support that are found in multiple sentences. However, the window is a work-around for truly modelling and utilizing these linguistic phenomena. Anaphoric resolution for question answering has been studied by (Morton, 1999) and results there indicate a small improvement for question answering in terms of sentence extraction. In the next chapter, Vicedo and Ferrandez present an analysis of reference resolution for question answering.

A weight is applied to items found away from the center sentence and in the case of named entity match, the center sentence is given credit only if it has a named entity. Otherwise the overlapping sentences would all obtain the same scores thus defeating ranking them in a useful manner.

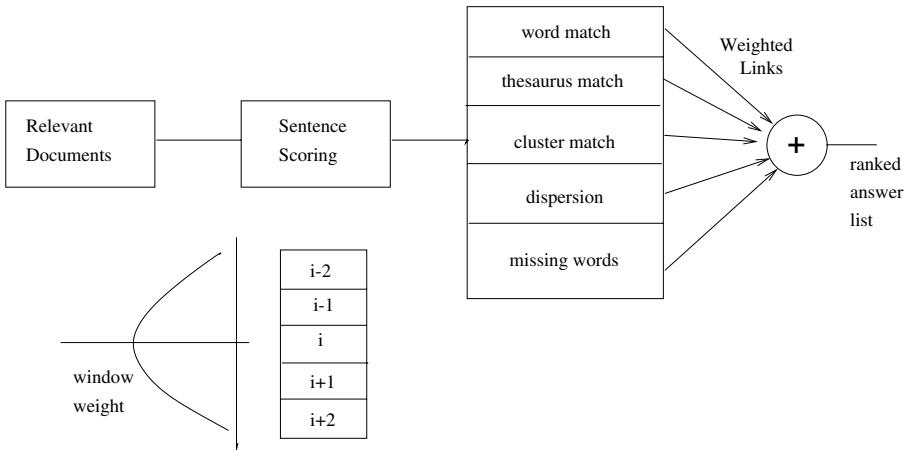


Figure 5. Sentence Ranking procedure for Answer Selection.

4.1.1 Sentence Ranking Features

The answer selection algorithm we report here was developed to address TREC 9, 10, and 11, and in this section we list the features used in the various systems. The features for the sentence ranking are presented in Table 5. In addition to these features, for each question word that is matched inside the answer, we measure the dispersion distance as how far away (in word count) is another matching question word. The dispersion distance for word i is d_i , and is capped at 20 words. Given this distance we compute the total dispersion as,

$$D = \log\left(\frac{1}{M} \sum_{i=1}^M d_i\right) \quad (0.22)$$

The ‘log’ function serves only to compress the distances as we want this feature to indicate those cases where the answer has only loose coupling to the question.

This feature is similar to the ‘Distance Score’ in Lasso (Moldovan *et al.*, 1999), although the details are slightly different.

We then compute the score of a window of sentences as the sum of the scores of the features minus the dispersion and the score of the missing words. This strategy of ranking windows of text is similar to that used in the LASSO system (Moldovan *et al.*, 1999). The key difference lies in the notion of using the IDF as a measure of weight on the type of match.

Table 5. Sentence Feature Definitions.

Feature Name	Description	Evaluations
Matching Word Score	The sum of the word IDFs (inverse document frequency) that matched identically, including case information, through stemming.	Trec9+
Missing Word Score	The sum of the word IDFs of question content words that are absent in the answer.	Trec9+
Matching Without Case	The sum of the IDFs of words that match ignoring the case	Trec10+
Thesaurus Match	The sum of the word IDFS that matched using a thesaurus match using <i>WordNet</i> .	Trec9+
Arc Match	The sum of the IDFs of answer words who have a parse link that matches parse tree of the question	Trec10+
Root Verb Match	The root verb of the question is matched in this answer	Trec10+
MisMatch Ordinal Score	Ordinals (example ‘first’) not found in the answer	Trec10+
Extra Words Score	The sum of the IDFs of words which link to question words but are not part of the answer (i.e. a word which is not the desired named entity or is not an encyclopedia expansion word)	Trec10
Expanding Word Score	The sum of the IDFs of words which are found on the LCA expansion list	Trec9+
Expanding Named Entity Word Score	The sum of the IDFs of words which are the desired named entity and also found in the encyclopedia	Trec10
Dispersion	A measure of how spread out the question words are in the answer support	Trec9+
Cluster Score	The number of pairs of words which are present in the question and answer sentence	Trec9+
Sentence Ranking Score	The score of the sentence ranking algorithm	Trec9+

The windows are then ranked using this score. To the top 100 windows, named entity matching and focus analysis is performed to arrive at the final ranking of the sentence. This ranking step relieves the subsequent analysis of a great computational burden as only 100 sentences need to be analyzed for named entities and the presence of the focus. Table 6 shows the number of questions which have no answers in Top N as more sentences are considered in the first 500 questions of TREC9.

4.1.2 Named Entity Matching

If the desired named entity is found in the current sentence then it is boosted in score. Additionally, the distance of the named entity to a content (meaning non-stopword) question word is measured. This ensures that the answer support is close to the desired answer. The score is computed as

$$S_{ne,i} = S_i + (E + (3 - d_{ne})D_{ne}) \quad (0.23)$$

where E is the entity boost, d_{ne} is the distance, and D_{ne} the distance penalty for the named entity. To test how sensitive the system is to the named entity boosting parameter, a series of experiments were conducted and the results are shown in Table 7. These results indicate that named entity boost improves performance but increasing its weight beyond 0.5 impairs performance. The reason for this degradation is the combination of the answer tag errors and the named entity errors and therefore we can not insist on finding the named entity always. In another viewpoint, this parameter controls the relative weight of the answer support match and named entity and too much reliance on the named entity hurts as we find sentences with little answer support.

Table 6. Performance versus depth of sentence window list.

Sentence List Depth	Number of questions missing answers (% of total)
Top 5	143 (28.6%)
Top 10	114 (22.8%)
Top 15	94 (18.8%)
Top 20	85 (17%)
Top 50	62 (12.4%)
Top 100	49 (9.8%)

Table 7. Effect of the Named Entity Boost parameter on sentence ranking.

NE Boost	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
TREC9 MRR	.574	.587	.587	.587	.588	.583	.580	.579	.577

4.1.3 Focus Matching

If the question has a focus, then answers which have a hypernym (parent) or hyponym (child) relationship in *WordNet* are boosted in score. For example, a question such as “*What river in the US is known as Big Muddy?*”, the focus is derived to be “river” and in *WordNet* we see that the Mississippi River has a child relationship with river. The distance of the focus word to the nearest content question word is measured as was done for the named entity. The score for the sentence is then computed as

$$S_{focus,i} = S_{ne} + (F + (3 - d_{focus})D_{focus}) \quad (0.24)$$

where F is the focus boost, d_{focus} is the distance, and D_{focus} the distance penalty for focus.

Focus matching is a back off method for the named entity matching. It allows any noun phrase known by *WordNet* to be used as an entity and can match any answer that has a hypernym or hyponym relationship. Additionally, a finer grain of matching is possible on named entities as the above example shows. “Mississippi River” should be tagged as a GEOLOGICALOBJ and even if the answer tag prediction made an error and required a PHRASE, this question would be handled correctly through focus matching. Keeping both named entity match and focus matching as features allows the answer selection model to weigh and utilize both.

4.2 Candidate Extraction

A general method of selecting candidate answers from a sentence is desired to further refine the answer and also satisfy length constraints on the answer. A most general method is to specify the overlap between candidates and select all possible chunks. The method used in this chapter is driven by the syntactic parse tree. Internal nodes, which satisfy the length constraint, are chosen as answer candidates. Initially, nodes that had a matching question word were skipped but it was noted quickly that this is a poor choice as many times the focus word may appear in the answer. An example of this effect is shown in the question above where Mississippi River has a matching word, ‘river’. Leaving off the ‘River’ has grievous consequences with automatically judging the answers using answer patterns written in Perl, although humans might judge the answer as being correct having read the supporting document.

(Ge, 2000) used the (Hobbs, 1976) algorithm for finding candidates for pronominal expressions. The algorithm used here for answer selection is a similar breadth-first traversal of the parse tree, and as described above selecting all interior nodes that meet the length constraint.

The more general method of deriving candidate answers, i.e. selecting windows of text, yields a large number of possibilities and there is little to discriminate between them. Reducing the number of answer chunks by using the parse tree yields candidates which are syntactically more appealing at the expense of some chunking errors. As an example, consider the question presented above “*What river in the US is known as Big Muddy?*” An answer sentence is “Known as Big Muddy, the Mississippi is the longest river in the US, draining all or part of 31 states on its winding, 2,500-mile journey from the hills of Minnesota to the Gulf of Mexico.” The parse of this sentence in bracketed notation is shown in Figure 6.

(S (NP (NP Known_NNP) (PPas_IN (NP(NP Big_NNP Muddy_NNP)
., (NP the_DT Mississippi_NNP)))) (VP is_VBZ (NP (NP the_DT
longest_JJS river_NN) (PP in_IN (NP the_DT US_NNP)))) ., (S (VP
draining_VBG (NP (NP all_DT or_CC part_NN) (PP of_IN (NP (NP
31_CD states_NNS) (PP on_IN (NPits_PRP\$ winding_NN)))) ., (NP
2,500_CD) .: (NP (NP mile_NN journey_NN) (PP from_IN (NP (NP
the_DT hills_NNS) (PP of_IN (NP Minnesota_NNP))) (PP to_TO (NP (NP
the_DT Gulf_NNP) (PP of_IN (NP Mexico_NNP)))))))))))))) .)

Figure 6. Parse of the example answer sentence.

The considered candidates derived by using the breadth-first search are shown in Table 8. Many of the candidates can be ruled out by insisting on named entity matches or other heuristics, but to do this in a principled manner we consider all such candidates and generate features about the candidate as discussed below.

Table 8. Proposed answer chunks for the example.

Known as Big Muddy, the Mississippi
the longest river in the US
draining
all or part
31 states on its winding
2,500
mile journey
from the hills of Minnesota to the Gulf of Mexico

All the processing described above can be considered as a necessary fast match to allow the answer selection problem to be modelled. These answer candidates are precisely the entities which we can form the answer selection model $p(c|e,q,a)$. The model will re-consider the features used in ranking them in each of the steps above.

4.3 Answer Candidate Feature Generation

The answer candidates derived above share some common features with others that lie in the same sentence. For example, this sentence is part of a document that has a document rank (its rank in the list of documents retrieved for this query). A feature that is shared among candidates of a sentence is the MatchingWordScore. The features at the sentence level have been described before in Table 5. In Table 9, we list the features that are applicable to the candidate exact answers.

Table 9. Answer Chunk Features generated for answer selection.

Candidate Feature Name	Feature Description
Entity Score	Score that reflects that the candidate has the desired entity
NE Mapping	String encoding of the NE and possibly its match from a confusion list
Focus Score	Score reflecting that <i>WordNet</i> has knowledge that this candidate is a hypernym or hyponym of the question focus
DB Focus Score	Candidate has a proximity relationship with the question focus
NP Score	Candidate has a proper noun
Digit Score	Candidate has a numeric entity
Date Score	Candidate has a date entity
<i>Definition Features</i>	
Defn Score	Candidate has words that match an encyclopedic definition of question terms
Hypon Score	Candidate has a hyponym relationship to the question focus
Expansion Score	Candidate has words that are in the LCA expansion list
Gloss Expansion Score	Candidate has words that are part of the <i>WordNet</i> gloss information

Candidate Feature Name	Feature Description
<i>Linguistic Features</i>	
Is Relationship Score	Candidate has an is-relationship with the term in the question
Arc Match Score	Candidate has a desired named entity that has a parse link (syntactic relationship) with a question word
Apposition Score	Candidate has an appositive relationship with the question words
Subject-Verb Score	Candidate is the subject of a major verb in the question
Verb-Object Score	Candidate is the object of a major verb in the question
<i>Miscellaneous Features</i>	
Max Match Score	The maximal string match measured in words between question and candidate including the named entity
Distance Score	The number of words from this candidate to the closest question word

Some of the features listed in Table 9 require further explanation and are examined in more detail next.

4.3.1 Entity Features

As described in the named entity section, these features (primarily the NP, Digit and Date) overcome failures in named entity marking as well provide a means for the system to correlate correctness with the firing of these features. The Candidate DB Focus is a feature that fires when a word that occurs next to the question focus is found in the answer candidate. In the example above, the feature fires for Mississippi because elsewhere in the text the word occurs next to “River”. The feature is most useful when an answer satisfies the focus somewhere in the text and then subsequently the answer is used without the focus.

4.3.2 Definition Features

Definitional questions request an elaboration on a concept or given a concept requires the term of the definition. These questions are largely outside the scope of named entity analysis and focus methods. The questions are simple to answer when there are only few instances of the term and the term is used primarily in the definitional context. However, this is often not the case and since the questions typically have only one major phrase that is being defined, there is very little the match type features can do. Using a dictionary resource such as WordNet can aid

greatly in answering these types of questions. The collection of these features isolates various types of matches from *WordNet* glosses to items found during LCA expansion.

4.3.3 Linguistic Features

- ***Is-Relationship*** This feature fires when the answer candidate is either the subject or object of a form of the verb “be” and has all the question words following.
- ***Apposition*** This feature applies to answer candidates which are immediately preceded or followed by a comma and then a group of matching question words are found. This is similar in function to the *Comma_3_words_score* of the LASSO system (Moldovan *et al.*, 1999) although in this case its even more restrictive in requiring all question words to be present either to the left or right of a noun phrase which is comma delimited.
- ***Subject-Verb/Verb-Object*** When the question has non-stop verb (meaning important and uncommon), and the answer candidate is either in subject or object position, this feature fires.

4.3.4 Web Feature

The web feature was used by a number of groups in TREC-10 by (Clarke *et al.*, 2001; Brill *et al.*, 2001) and an attempt was made to measure its impact on this system. We incorporated the feature as two indicators:

1. occurrence of the answer candidate in the top 10 documents retrieved from the web;
2. count of the number times the answer candidate occurred.

This method of using the web can be considered *Answer Verification* to differentiate it with other approaches which attempt to answer the question on the web and then look in the target document corpus for the same answer. The latter method can result in unsupported answers but seems more promising as other groups have reported much larger gains.

4.3.5 Statistical Machine Translation Thesaurus

As discussed earlier, words comprising the answer support are generally found in the question, words such as the focus of the question are sometimes deleted in the answer. Following our general approach of learning phenomena from training data, we used our question-answer corpus to train a Model 1 translation matrix (Brown *et al.*, 1993). Questions were tokenized with casing information folded and answers were both tokenized and name entity tagged. A question answer pair is presented below before and after the pre-preprocessing:

Q: How tall is Mt. Everest?

A: He started with the highest, 29,028-foot Mt. Everest, in 1984

Q: how tall is mt. everest?

A: he started with the highest, 29,028 - foot mt. everest, in 1984 MEASURE

We had 4K training pairs from the KM trivia database, 1.6K pairs from TREC-8 and 10.7K pairs from TREC-9. The latter were derived from correct judgments given to questions in those evaluations and which also came from unique sentences in the corpus. This data was split into two and separate translation models were derived. Entries which occurred in both translation models were retained; a few of the more interesting entries are shown below in Table 10. Each word is shown with the 5 top translation candidates. For the word “who”, the model prefers to see a named entity tag PERSON with a relative high probability. Even though the number of translation pairs is small (16.3K pairs), for the question answering application we are interested in only the most common words, which are potentially modified in the translated output of the question; rarer words have to appear identical to the form in the question. Using this additional thesaurus resource, we re-ranked the answer candidate windows (windows of text bounded by the question terms and the answer candidate) and quantized the rank into 5 bins (1, 2, high, mid and low) for use in the maximum entropy answer selection module. We will refer to these features as the SMT rank of the chunk.

Table 10. Translation entries for some question words.

who		haiti		river	
a	$t(a q)$	a	$t(a q)$	a	$t(a q)$
PERSON	0.125	haiti	0.076	river	0.217
,	0.010	Port-au-prince	0.048	the	0.081
the	0.051	Miami	0.034	water	0.060
.	0.046	People	0.021	LOCATION	0.039
“	0.042	Haitian	0.018	many	0.028

nuclear	
a	$t(a q)$
nuclear	0.183
atomic	0.020
at	0.013
soviet	0.010
site	0.010

tall	
a	$t(a q)$
MEASURE	0.056
foot	0.041
feet	0.027
-	0.017
i	0.012

team	
a	$t(a q)$
team	0.099
organization	0.056
game	0.030
;	0.029
their	0.023

4.3.6 Answer Patterns

The approach described in (Soubbotin, 2001) uses patterns for locating answers. In a related work, (Ravichandran and Hovy, 2002) has shown how to extract patterns

in an unsupervised manner from the web. In this work, we use the supervised corpus of question and answers to extract n-grams occurring in the answer. To specialize the pattern for a particular question type, the question was represented only by the question word and the first word to its right. To generalize the answer candidate window, it was modified to replace all non-stop question words with “<queryTerm>” and the answer candidate with “<answer>”. So for the example above:

QF: how tall

MW: he started with the highest, <answer> <queryTerm> measure_ne

where QF stands for the question focus and MW stands for the mapped answer candidate window. Ideally, the question would be represented by more than just the word adjacent to the question word but in most cases this suffices. To overcome some of the limitations of this choice, we also chose features relating the predicted answer tag and an answer pattern. An answer pattern consists of 5-grams or larger chosen with a count cut-off. The total number of pattern features incorporated was 8.5K out 15.3K features in the model.

It is not an object of this chapter to claim that these are all the possible features for question answering but to demonstrate the utility of the general framework in which useful features are used to make a statistical answer selection model. The modelling paradigm will weight these features in order to maximize the likelihood of our model. In order to improve performance, one avenue is to explore feature combinations. A second method is to improve the performance of each feature.

4.4 Training Data

The corpora used to train the answer selection model consists of TREC-8, TREC-9, and 4K questions from our KM database. This corpus represented an order of magnitude increase in size over the training data size used previously. For each question, a set of answer patterns was formed by judging several potential answer sentences in the TREC corpus. Using the answer patterns and sentences derived from the TREC corpus, we automatically labelled chunks as being correct or incorrect. The total number of chunks used in formulating the model was 207K. There were 20K instances of correct answers and 187K incorrect chunks.

4.5 Answer Selection Algorithm

The advantage of statistical systems is that the hard work is in defining features to tackle the problem. Sometimes these can be easy (for example, the context of the

current chunk: two words to the left and two words to the right) are easy to implement. Other features like the apposition detector using a parse tree are much more complex. Once implemented though, the features are easier to combine using the training data to set the weights of the combination.

The current answer selection model has 34K features. These features were generated using all training events and then applying a threshold on the feature set for a count cut-off of 10. The resulting features can be broken down into:

- Focus, Map Window, Answer Tag: Combination of the focus or the answer tag predicted of the question with an answer pattern.
- Unigrams, Bi-grams and Trigrams of Feats: These features are the amount of question word overlap, the named entity and answer tag match, the amount of dispersion.
- Answer Tag, Feats: Combination of the answer tag and the basic features.
- Answer Tag, Rank: Combination of the answer tag and the SMT rank of the chunk
- Rank: Features on SMT rank of the chunk

4.6 Rejection

For questions which are determined to have no answer in the corpus, the system was supposed to return ‘NIL’ as the document id. To determine which questions to reject, we employed the distribution $p(c|q,a)$ and used a threshold on the distribution. However, the system sometimes encounters events which are not sufficiently represented in the training corpus and to allow some level of control it was useful to smooth this probability with a decreasing function of chunk rank. This smooth estimate was computed as

$$p^* = (1 - \alpha)p(c|q,a) + \alpha(1 - 0.1(chunk_rank)) \quad (0.25)$$

where chunk rank was saturated at 10. This year the alpha was set to 0.2 and the rejection threshold to 0.3. The rejection threshold was optimized on the accuracy of TREC-10 questions using the TREC corpus of documents. We plot in Figure 7, the cumulative distribution function of questions with answers in the corpus and also 1.0 minus the cumulative distribution function for questions which should be rejected. The plot is for TREC-10 questions using the TREC corpus of documents for answers. We expected to reject about 80 answers in our base system and the actual run seems to have done approximately the same.

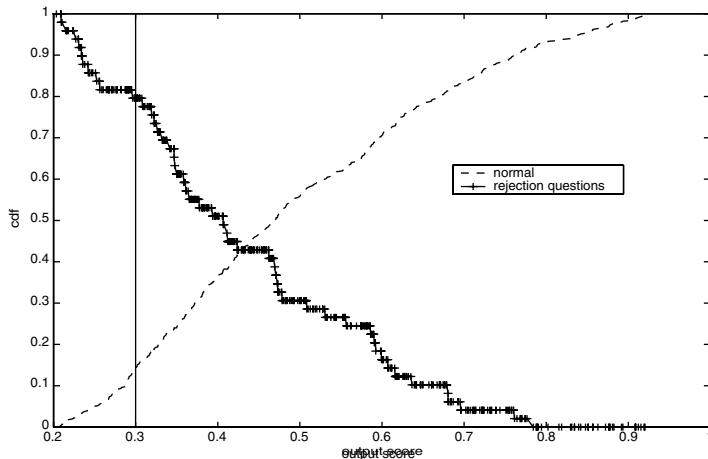


Figure 7. TREC-10 scores for normal and rejection questions.

4.7 Performance

The system has been run on TREC-9, 10 and 11. In Table 11, we quote only the official results in the TREC evaluations. Its very difficult to compare between these results because each test set is different and the conditions they were run under also changed.

Table 11. Evaluation results for the IBM statistical question answering system.

System	Description	Metric	Result	Wrong (Total)	Reject
trec9 (ibmhlt00250)	250 byte sentence	Top5 MRR	0.457	263 (692)	NA
trec9 (ibmhlt00050)	50 byte phrase	Top5 MRR	0.457	394 (692)	NA
trec10 (ibmsqa01a)	50 byte phrase - No query expansion in IR, encyclopedia query expansion in answer selection	Top5 MRR	0.390	218 (500)	28/192

trec10 (ibmsqa01b)	50 byte phrase - Encyclopedia query expansion in IR and answer selection	Top5 MRR	0.390	220 (500)	30/193
trec10 (ibmsqa01c)	50 byte phrase - Encyclopedia and WordNet query expansion in IR and answer selection	Top5 MRR	0.375	231 (500)	30/209
trec11 (ibmsqa02a)	exact answer - system described in this chapter	CWS	0.454	312 (500)	12/83
trec11 (ibmsqa02b)	exact answer - No web or rejection	CWS	0.403	328 (500)	0
trec11 (ibmsqa02c)	50 byte phrase - Feedback loop on ibmsqa02a with confident answers	CWS	0.455	293 (500)	11/49

The performance of the system in TREC-11 was degraded because the system was trained on inexact answers and 44 questions were regarded as inexact in the evaluation. Despite the larger training set used in TREC-11, the system needs many more examples of question answer pairs before the performance becomes reasonable.

5. CONCLUSIONS

In this chapter a statistical question answering system was described. The system offers flexibility in adding and weighting features while providing a mathematical formulation for question answering. The system description given in this chapter has evolved over a number of evaluations and will continue to change as we learn more about question answering, but the methodology is now firm and we believe that the relevancy formulation for question answering ($p(c|q,a)$) is superior to other adhoc methods.

6. ACKNOWLEDGEMENT

This work was partially supported by the Defense Advanced Research Projects Agency and monitored by SPAWAR under contract No. N66001-99-2-8916. The views and findings contained in this material are those of the author and do not necessarily reflect the position of policy of the Government and no official endorsement should be inferred.

7. REFERENCES

- Berger, A., Caruana, R., Cohn, D., Freitag, D., and Mittal, V. (2000). Bridging the lexical chasm: Statistical approaches to answer finding. In *Research and Development in Information Retrieval*, pages 192-199.
- Berger, A. and Printz, H. (1998). A comparison of criteria and maximum divergence feature selection. *Proceedings of the Third Conference on Empirical Methods in Natural Language Processing*, pages 97-106.
- Berger, A. L., Pietra, V. D., and Pietra, S. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1): 39-71.
- Borthwick, A. et al. (1998). Exploiting diverse knowledge sources via maximum entropy in named entity recognition. *Proceedings of the COLING-ACL 98, Sixth Workshop on Very Large Corpora*, pages 152-160.
- Brill, E. (1993). Transformation-based error-driven parsing. In *Proceedings of the 31st Annual Meeting of the ACL, Columbus, Ohio*, pages 543-565.
- Brill, E., Lin, J., Banko, M., Dumais, S., and Ng, A. (2001). Data-intensive question answering. *TREC-10 Proceedings*, pages 393-400.
- Brown, K. and Miller, J. (1999). *Concise Encyclopedia of Grammatical Categories*. Elsevier Science Ltd.
- Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19: 263-311.
- Burke, R. et al. (1997). Question Answering from Frequently-Asked Question Files: Experiences with the FAQ Finder System. *University of Chicago Technical Report TR-97-05*.
- Clarke, C., Cormack, G., Lynam, T., Li, C., and McLearn, G. (2001). Web reinforced questions answering (multitext experiments for trec 2001). *TREC-10 Proceedings*, pages 673-679.
- Collins, M. (1996). A new statistical parser based on bi-gram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 184-191.
- Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. John Wiley and Sons, Inc.
- Csiszar, I. (1989) a Geometric Interpretation of Darroch and Ratcliff's Generalized Iterative Scaling. *The Annals of Statistics*, 17(3): 1409-1414.
- Darroch, J. N. and Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annual of Mathematical Studies*, 43(5): 1470-1480.
- Frakes, W. B. and Baeza-Yates, R. (1992). *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall.
- Ge, N. (2000). *An Approach to Anaphora Resolution*. PhD Thesis Department of Computer Science, Brown University.
- Green, B., Wolf, A., Chomsky, C., and Baseball, L. (1963). An automatic question answerer. *Computers and Thought*, pages 207-216.
- Harabagiu, S et al. (2000). Falcon: Boosting knowledge for answer engines. *TREC-9 Proceeding*, pages 50-59.
- Hobbs, J. (1976). Pronoun resolution. Dept. of Computer Science, City College, CUNY, Technical Report TR76-1
- Ittycheriah, A., Franz, M., Zhu, W.-J., Ratnaparkhi, A., and Mammone, R. (2000). IBM's statistical question answering system. *TREC-9 Proceedings*, pages 60-65.
- Ittycheriah, A., Franz, M., Zhu, W.-J., Ratnaparkhi, A., and Mammone, R. (2001). Question answering using maximum entropy components. *The Second Meeting of the North America Chapter of the Association of Computational Linguistics, Pittsburgh, PA*, pages 33-39.

- Jaynes, E. (1983). *Papers on Probability, Statistics, and Statistical Physics*. D. Reidel Publishing Co., Dordrecht-Holland.
- Koeling, R. (2000). Chunking with maximum entropy models. In *Proceedings of CoNLL-2000 and LLL-2000, Lisbon, Portugal*, Pages 139-141.
- Li, X. and Roth, D. (2002). Learning questions classifiers. In *COLING 2002, The 19th International Conference on Computational Linguistics*, pages 556-562.
- Marcus, M. P., Santoro, B., and Marcinkiewicz, M. A. (1994). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313-303.
- Mikheev, A., Grover, C., and Moens M. (1997). Description of the LTG System Used for MUC-7. *Online Proceedings of MUC-7*, pages 1-11.
- Miller, G. (1990). Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4): 235-244.
- Moldovan, D. et al. (1999). LASSO: A tool for surfing the answer net. *TREC-8 Proceedings*, page 65-73.
- Morton, T. (1999). Using coreference for question and answering. In *ACL Workshop, Coreference and Its Applications*, pages 173-180.
- Ng, H. T., Teo, L. H., and Kwan, J. L. P. (2000). A machine learning approach to answering questions for reading comprehension tests. In *Proceedings of the 2002 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 124-132.
- Papineni, K. (2001). Why inverse document frequency. The Second Meeting of the North American Chapter of the Association of Computational Linguistics, Pittsburgh, PA, 10(10): 25-32.
- Pietra, S. D., Pietra, V. D., and Lafferty, J. (1995). Inducing features of random fields. *Technical Report, Department of Computer Science, Carnegie-Mellon University, CMU-CS-95-144*.
- Ratnaparkhi, A. (1998a). *Maximum Entropy Models For Natural Language Ambiguity Resolution*. PhD Thesis, Department of Computer and Information Science, University of Pennsylvania.
- Ratnaparkhi, A. (1998b). Statistical models for unsupervised prepositional phrase attachments. In *COLING-ACL*, pages 1079-1085.
- Ratnaparkhi, A. (1990). Learning to parse natural language with maximum entropy models. *Machine Learning Journal*, 34: 151-175.
- Ratnaparkhi, A., Roukos, S., and Ward, R. T. (1994). Maximum entropy model for parsing. In *Proc. Of the 1994 International Conference on Spoken Language Processing (ICSLP 94)*, PAGES 803-806, Yokohama, Japan.
- Ravichandran, D. and Hovy, E. (2002). Learning surface text patterns for questions answering system. In *Proceedings of the 4th Annual Meeting of the ACL*, pages 41-47.
- Reynar, J. C. and Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentences boundaries. *Proceedings of the Fifth on Applied Natural Language Processing*, pages 16-19.
- Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., and Gatford, M. (1995). Okapi at TREC-3. In Harman, D., editor, *Proceedings of the Third Text Retrieval Conference (TREC-3)*. NIST Special Publication 500-225.
- Soubbotin, M. M. (2001). Patterns of potential answer expressions as clues to the right answers. *TREC-10 Proceedings*, pages 293-302.
- Srihari, R. and Li, W. (1999). Question answering supported by information extraction. *TREC-8 Proceedings*, pages 75-85.
- Xu, J. and Croft, W. B. (1996). Query expansion using local and global document analysis. In *Research and Development in Information Retrieval*, pages 4-11.
- Yang, H. and Chua, T.-S. (2002). The integration of lexical knowledge and external resources for question answering. *TREC-11 Notebook Proceedings*.

COREFERENCE IN Q & A

Abstract: The main aim of this work is to study the application of automatic anaphora or co-reference resolution techniques to Question Answering (QA) systems. Moreover, this chapter includes an overview of anaphora problem, a summary of approaches to anaphora resolution in Natural Language Processing and an analysis of their effectiveness and applicability to QA tasks. This work is complemented with a detailed study of current QA computational systems that apply this kind of techniques and a full evaluation for measuring the effects on QA systems performance when the information that is referenced by pronominal anaphora in documents is not ignored.

1. INTRODUCTION

It is widely agreed that anaphora resolution is among the toughest problems in Computational Linguistics and Natural Language Processing (NLP) that are still to be solved. On the other hand, anaphoric expressions are so frequently used and so readily understood that their importance is generally overlooked, but it sometimes becomes crucial to resolve anaphoric expressions accurately; above all, in tasks related to finding very specific information such as Information Extraction and Question Answering (QA) applications. In both cases, if anaphora remains unresolved, some important information may be missed. For example, critical information usually appears referenced pronominally throughout documents making it impossible to extract the referenced entity without accomplishing pronoun resolution.

This work tries to achieve several objectives. Firstly, we introduce anaphora problem from a perspective that facilitates analysing the different aspects of its application for QA tasks. Secondly, we analyse the complexity that introduces an anaphora resolution module in a QA system. This complexity is analysed from different points of view: from its precision, from the improvement or noise that it introduces, and from its speed that can allow us to use it in a real QA system. Thirdly, we present a detailed analysis of the state of the art concerning the application of anaphora resolution techniques in current QA systems. Finally, we study the need to use these algorithms for entity tracking in documents, where the evaluation that has been carried out demonstrates that the benefits of solving anaphora also depend on the density of information of the document collection where QA systems perform the search.

In section 2, we present an overview of anaphora problem, including a taxonomy that is used in the rest of the paper, the main algorithms that are used to

automatically resolve it and the different approaches applied in current QA systems. In section 3, we analyse the complexity introduced by an anaphora resolution module in a QA system. In section 4, we study the relation between anaphora resolution effectiveness and information density for QA. Finally, we present our conclusions.

2. ANAPHORA PROBLEM

This section gives an overview of anaphora problem. It starts with some definitions that can help us to understand the problem we are trying to face. After that, a general taxonomy of anaphora is proposed, on which we can base it on, in order to quote the different anaphors that are resolved in QA systems. Next, a brief description of the kinds of knowledge that are required to resolve anaphora is presented. All these kinds of knowledge are used in different anaphora resolution algorithms, which are briefly summed up in the backgrounds of anaphora resolution.

2.1 Anaphora definitions

The *anaphora* phenomenon is defined by (Hirst, 1981) as the device, in discourse, of making an abbreviated reference to some entity or entities, in the expectation that the receiver of the discourse will be able to dis-abbreviate it and determine the identity of the entity. The *reference* to it is generally called *anaphor* (e.g. a pronoun or a definite description) or *anaphoric expression*, and the *entity* to which the anaphor refers is its *referent* or *antecedent*, where the difference between both is that the former means the abstract entity to which the linguistic expression refers, i.e. the antecedent. For example, the linguistic expression *George Bush* as it appears in a text, could be an antecedent, and the entity corresponding to the person whose name is *George Bush* would be its referent. In this work, we will also refer to the possible antecedents of an anaphor as *candidates*.

Anaphora should be distinguished from *cataphora* (or *forward anaphora*), which occurs when the anaphor appears before its antecedent, as in: *Near him, John, saw a snake*. Moreover, cataphora, unlike anaphora, generally requires that both the pronoun and its antecedent be mentioned in the same sentence. We should note that we are using subscripts, e.g. *i* or *j*, to join an anaphor with its antecedent.

Furthermore, there are certain kinds of anaphors, called *exophors* that refer to entities in the external world rather than in the given text, i.e. entities that have not been previously introduced in the discourse. One example of this is the following sentence, which would generally be accompanied by pointing, or some similar gesture, to draw the listener's attention to what *that* refers to, and precisely where *there* is: *Pick that up and put it over there*. The phenomenon that produces exophors is usually called *deixis*.

Finally, we should define a concept that is usually quoted in papers about anaphora problem: *co-reference chain*. A co-reference chain is formed when the antecedent of an anaphor is also an anaphor that co-refer with another antecedent. That is to say, a set of linguistic expressions that refers with the same referent.

2.2 Anaphora taxonomy

Several classifications of anaphora could be presented according to different criteria. For example, classifications according to the kind of reference, or according to the different linguistic expressions that form the anaphor, or with reference to the different kinds of antecedent. Here, since the aim of this work is not to carry out a deep study of anaphora problem, we only try to propose a general classification on which we can base it on, in order to quote the different anaphors that are resolved in QA systems.

2.2.1 Classification according to the reference

Allen (1995) distinguishes between *surface* and *deep* anaphora. The former means referring expressions that introduce new objects related to objects in the discourse context. That is to say, antecedent and anaphor does not refer to the same referent. For example in (1), the objects that are referred to in (b) and (c) are not mentioned in (a), but the context created by (a) allows to interpret each one. Whereas *deep anaphora* means those cases in which the anaphor and antecedent have the same referent, that is to say, anaphor and antecedent *co-refer* and they are *co-referential*.

- (1) (a) Tell me John's grade in CS.
 (b) Give me it in MT as well.
 (c) Give me Mike's in GG too.

These types of anaphora are called by other authors in different ways. For example, (Hirst, 1981) called them as *identity of sense anaphora* and *identity of reference anaphora* respectively. The former is defined by Hirst as an anaphor that denotes not the same entity as its antecedent, but one of similar description, as in the example (2), in which *it* means the second man's paycheck, not the first man's. The latter is an anaphor that denotes the same entity as its antecedent, as it occurs in sentences in (3).

- (2) The man who gave his paycheck to his wife was wiser than the man who gave it to his mistress.
- (3) *Ross made a sandwich_i and ate it_i.
The price of a radio_i is six pounds. This price_i is very expensive.*

Finally, just to mention another common denomination of surface anaphora, the so-called *bound anaphora*, which is used for example in the Message Understanding Conferences (MUC) co-reference tasks, where it is explained that these anaphors usually occur when the antecedent noun phrases are quantified: *John lost a pen yesterday and Bill found one today.*

2.2.2 Classification according to the anaphor

In this classification, we distinguish between different anaphors according to their linguistic category: pronouns, definite descriptions, proper nouns, acronyms, verbal phrases and temporal and locative expressions.

2.2.2.1 Pronouns

There are different classes of pronouns that can produce references: *personal, demonstrative, reflexive, reciprocal, possessives* and *zero-pronouns*. All of them are briefly presented below:

- Personal and demonstrative pronouns: *Ross made a sandwich_i and ate it_i.*
- Reflexive and reciprocal pronouns: *Peter_i hauled himself_i up onto the wall. John_i himself_i had become ill. Peter and John often borrow each other's clothes.*
- Possessives and the possessive forms of pronouns used as determiners: *The book is hers. His aunt.*
- Zero-pronouns. We mean with this kind of anaphor, those pronouns that are omitted in the text. In MUC, these pronouns are called *implicit pronouns*. Therefore, they should be detected in a previous stage. After that, they are resolved in a similar way to a personal pronoun. In English texts, this sort of pronoun occurs far less frequently, as the use of subject pronouns is generally compulsory in the language (e.g. *Ross_i, carefully folded his trousers and Ø_i climbed into bed*). Whereas in other languages, such as Japanese (Nakaiwa & Shirai, 1996) or Spanish (Ferrández & Peral, 2000), it is quite usual.

Pronouns usually take as antecedent a noun phrase, but in some cases their antecedents may be a sentence or a verbal phrase, such as in example (4), where *it* does not refer to any of the preceding noun phrases, but rather, to the entire situation. One special case has been quoted as *gerunds* in MUC, in which the antecedent of the pronoun is a verbal form, as it occurs in: *Slowing the economy is supported by some Fed officials; it is repudiated by others.*

- (4) The president was shot while riding in a motorcade along a major Dallas boulevard, today. It has caused panic on Wall Street.

With regard to the kind of reference that a pronoun can accomplish, as it was shown in examples (1) and (3), it can perform both deep and surface references.

Finally, it should be noted that in addition to the 1st and 2nd person pronouns, some pronouns can be non-anaphoric. For example, the English pronoun *it* or the Spanish complement pronouns *se, la, lo*. Non-anaphoric uses of *it* are also referred to as *pleonastic* or *semantically empty*. These kinds of pronouns should be detected in order to avoid their resolution since they are quite usual in English (examples in (5)).

- (5) It is obvious. It should be pointed out. It is teatime. It is summer.

2.2.2.2 Definite descriptions

Definite descriptions are those noun phrases whose pre-modifying word is a determiner (e.g. an article or demonstrative), which usually refers to discourse referents already mentioned, as it occurs in example (6).

- (6) Fred was discussing an interesting book_i in his class. This book_i...

There is an important distinction with reference to other kinds of anaphors, since they can be used *referentially* or *existentially*. The referential reading uses the description to refer to a previously known object, just as pronouns do, as it occurs in (6). Whereas the existential reading asserts the existence of a unique object satisfying the description, as in: The moon rose. Moreover, definite descriptions, unlike pronouns, do not have such a strong tendency to have antecedents in the immediate local context, since they have more lexical knowledge than pronouns.

In Poesio & Vieira (1998) several kinds of definite descriptions are distinguished:

- Anaphoric same head, in which the definite description and its antecedent share the same head noun: *Bill bought a new car_i. The car_i is red.*
- Associative definite descriptions, which include those that are not identified by the same head noun as their antecedent. These anaphors are also called by some authors as *bridging references*. Here, the relation between head nouns is through synonym, hypernym, meronym, verb nominalizations, etc. Some examples could be the following: *Peter cut a flower_i. That plant_i was very beautiful.... John_j sold his old car. The seller_j...*
- Idiomatic expressions, which do not have referential properties: *It went back into the soup.*

Some other authors also distinguish between some special cases of definite descriptions:

- Proadjectives: I was looking for a black man that spoke English, but I couldn't find such a man.
- Epithets: Ross used his bankcard so much, the poor guy had to declare bankruptcy.
- Surface count anaphora, whose anaphors are formed by a determiner and a linguistic expression that has a meaning of order: the former, the latter, the first, etc. It requires not only that the antecedent be in consciousness, but also that the surface structure of the sentence be retained or at least the order of possible referents. For example: Sue stared at the pumpkin_i and the turnip, and declared that she preferred the former_i.
- One-anaphora, which means an anaphor formed by a determiner, the pronoun one, and some modifiers. These anaphors could refer to a description rather than a specific entity: Wendy didn't give either boy a green tie-dyed T-shirt, but she gave Sue a red one". Here one is either T-shirt or tie-dyed T-shirt, but not green

tie-dyed T-shirt. Otherwise, they can co-refer with an antecedent as in: Peter has bought a blue jean and a dark jean. I like the blue one best.

2.2.2.3 Proper nouns and acronyms

Here, we include anaphors formed by a noun phrase, whose head is a set of proper nouns. For example in (7), *Peter* stands for the referent called *Peter McWilliams*, whereas *TREC-9* is an acronym of the whole noun phrase *Text REtrieval Conference in 2000*, which has been defined by means of an apposition.

- (7) Peter McWilliams attended the Text REtrieval Conference in 2000, TREC-9. Peter enjoyed it very much. In TREC-9...

2.2.2.4 Verbal phrase anaphora

These anaphors are formed by auxiliary verbs, and their antecedents are verbal phrases, as it occurs in: *When Ross orders a sweet, Nadia does too*. When they are used in conjunction with *so, it* or demonstratives, they can reference actions in a manner that is almost prosentential: *John drinks it in one gulp. Sue does it too*.

2.2.2.5 Temporal and locative expressions

Here, we mean those expressions of time and place that usually refer to the most recent temporal or locative expression of the text. These expressions could co-refer with their antecedent or could introduce new entities. For example, in (8) the temporal expression *at that time* co-refer with *the mid-sixties*, whereas *many years later* introduces a new time entity, which is generated from its antecedent. In (9), an example of a locative reference is presented.

- (8) In the mid-sixties, free love was rampant across campus. Many years later Sue turned to Scientology. At that time ... It was then that Sue turned to Scientology.
- (9) The church was behind the bookshop. Sue went there after lunch.

2.3 Kinds of knowledge to resolve anaphora

It is widely agreed that anaphora resolution needs to employ different kinds of knowledge (Mitkov, 2002), such as lexical, morphological agreement, syntactic and semantic. Here, we present a brief review on these kinds of knowledge.

2.3.1 Lexical knowledge

An example of this knowledge is the *implicit verb causality* or *causal valence*, which can affect the antecedents assigned to nearby anaphors, as it occurs in example (10). In this case, in general, with sentences of the form: $NP_1 \text{ VERB } NP_2$ because {he/she}... there is a distinct tendency for people to construct and interpret the sentence as such that the pronoun refers to NP_1 in the case of some verbs, and NP_2 in the case of some others (some verbs are neutral).

- (10) Ronald scolded Joe because he was annoying.

2.3.2 Morphological agreement

Here we mean the morpho-syntactic agreement in person, gender and/or number that is usually required between anaphor and its antecedent. For example, in pronominal anaphora resolution, it is used as a filter in order to restrict the search space that a discourse system of anaphora resolution must consider. However, we should bear in mind that these are strong constraints on reference, but they are not absolute:

- A plural anaphor can have a singular antecedent: John_i told Bill_j that they_{i,j} should do the right thing.
- A singular anaphor can have a plural antecedent (reference to elements of sets): At the zoo, a monkey scampered between two elephants_i. One_i snorted at it.
- A feminine one a masculine antecedent, and so forth: Who is this Bresson_i? Is she_i a woman? In this sentence, she refers to a person, who is probably marked as male in the listener's world knowledge.

Here it is important to remark on the problems caused by quantification, as it occurs in *Ross gave each girl_i a crayon. They_i used them_j to draw pictures of Daryel in the bath.* In this example, the quantifier *each* introduces a set of singular candidates: *each girl* and *the crayon* of each girl. Although both candidates appear as singular noun phrases in the sentence, they are referred to by the plural pronouns *they* and *them*. Several works attempt to solve this problem, e.g. (Webber, 1979).

2.3.3 Syntactic knowledge

This knowledge comprises both the syntactic structure of the anaphor and its possible candidates to antecedent in order to detect them, and the syntactic structure of the text in which they appear, in order to find out some rules or heuristics that make the resolution process easier. With reference to the latter, two examples can be presented:

- *C-command restrictions* (Reinhart, 1983) (Lappin & Leass, 1994), i.e. the syntactic filter on pronominal co-reference that restricts the search space that a discourse system of anaphora resolution must consider. For example, a non-pronominal noun phrase must be interpreted as non co-referential with any noun phrase that c-commands it.

- *Syntactic parallelism*: an anaphor resolver needs knowledge of parallelism, because the anaphor and its antecedent tend to be in parallel syntactic structures. That parallelism is particularly important in resolving surface count anaphora, which requires that the surface structure of the sentence (or at least the order of possible referents) be retained.

2.3.4 Semantic knowledge

In this knowledge, we include knowledge about words and their uses, world knowledge, and inference mechanisms to apply to the previous ones and to the meaning of the discourse itself.

For knowledge about words and their uses, we mean verb restrictions, relations of synonym, hyponym, meronym and troponym between words. For example in (11), *them* co-refer with *these laws* thanks to the need of keeping the semantic consistency between the verbs *improve* and *modernize*. Otherwise, if we substitute these verbs by *fulfil*, then *them* would co-refer with *Fulton legislators*.

- (11) *The jury recommended that Fulton legislators act to have these laws, studied and revised to the end of modernizing and improving them.*

For world knowledge, we can present the example in (12), where the ambiguity appears about whom the ugliest person is, *Peter* or *John*? It could be *Peter* because insulting behaviour is more common than openly self-critical behaviour with respect to personal appearance in western English-speaking cultures.

- (12) *John told Peter he was the ugliest person he knew of.*

As an example of inference mechanisms, Carbonell & Brown (1988) proposed the *pragmatic restrictions*, where the inference is applied on the action that is performed in the discourse. In (13), the verb *to give* forces that the person who carries out the action, after that, does not possess the object that is given. Moreover, the agent of the verb *to eat* must possess what is eaten. Therefore, the pronoun *he* should co-refer with *Peter*, since *John* does not have the apple in that moment.

- (13) *John gave Peter_i an apple. He_i ate the apple.*

2.4 Backgrounds of anaphora resolution algorithms

Common among all languages is the fact that the anaphora phenomenon requires similar strategies for its resolution, in spite having to resolve different kind of anaphors (e.g. pronouns or definite descriptions), or if it is resolved in different languages (Harabagiu & Maiorano, 2000) (Mitkov, 1999) (Ferrández, Palomar & Moreno, 1999). All languages employ different kinds of knowledge, but their

strategies differ only in the manner in which this knowledge is coordinated. For example, in some strategies just one kind of knowledge becomes the main selector for identifying the antecedent, with other kinds of knowledge being used merely to confirm or reject the proposed antecedent. In such cases, the typical kind of knowledge used as the selector is that of discourse structure. Centering theory, as employed by (Strube & Hahn 1999) (Tetreault, 2001), uses this type of approach. Other approaches, however, give equal importance to each kind of knowledge and generally distinguish between constraints and preferences (Lappin & Leass, 1994) (Carbonell & Brown, 1988). Whereas constraints tend to be absolute and therefore discard possible antecedents, preferences tend to be relative and require the use of additional criteria (e.g. the use of heuristics that are not always satisfied by all antecedents). Some examples of definite description resolution algorithms that are based on constraints and preferences could be the following: (Poesio & Vieira, 1998), (Humphreys *et al.*, 1998), and those based on Centering (Garigliano, Urbanowicz & Nettleton, 1998).

Still other approaches to anaphora resolution are based either on machine learning techniques (Connolly, Burger & Day 1994) (Paul, Yamamoto & Sumita, 1999) (Bean & Riloff, 1999) (Wee, Hwee & Daniel, 2001), or on Maximum Entropy model (Morton, 2000), or on statistical information (Ge, Hale & Charniak, 1998), or on genetic algorithms (Orasan & Mitkov, 2000).

Computational processing of semantic and domain information is relatively expensive when compared with other kinds of knowledge (Chan & T'Sou, 1999). Consequently, current anaphora resolution methods mainly rely on constraint and preference heuristics, which employ morpho-syntactic information or shallow semantic analysis (Palomar *et al.*, 2001). Such approaches have performed notably well. Lappin & Leass (1994), describes an algorithm for pronominal anaphora resolution that achieves a high rate of correct analyses (85%). Their approach, however, operates almost exclusively on syntactic information. More recently, Kennedy & Boguraev (1996) proposed an algorithm for anaphora resolution that is actually a modified and extended version of the one developed by Lappin & Leass (1994). It works from the output of a part-of-speech (POS) tagger and achieves an accuracy rate of 75%.

Most of the previously quoted works perform the evaluation of their approaches. However, there is not a comparative evaluation of all these systems on the same texts and languages since co-reference evaluation was carried out in some of the Message Understanding Conference Evaluations, specifically the MUC-6 and MUC-7 in 1995 and 1998 respectively. The MUC co-reference task was designed for capturing the information of co-referring expressions, in the context of the information extraction tasks. In these evaluations, a corpus manually and anaphorically annotated was created, independent of the MUC extraction task. The annotation covers only the co-referential relations for noun phrases; it does not include co-reference among clauses, nor does it cover other kinds of co-reference relations.

Since MUC evaluations, several efforts have been carried out in order to set common evaluation measures (Barbu and Mitkov, 2001) (Byron, 2001), but it is clear that a comparative evaluation between approaches on the same texts is

desirable. That is to say, it requires an *evaluation workbench* that allows the comparison of different algorithms, although they use different pre-processing tools, but on the same data. Moreover, we also agree with Mitkov (2002), that evaluation in anaphora resolution should be addressed from two different perspectives depending on whether the evaluation only focuses on the “anaphora resolution algorithm” or if it covers the performance of the “anaphora resolution system”, i.e. the whole implemented system that processes input at various levels (morphological, syntactic, etc.) with the corresponding levels of errors that will affect the anaphora resolution precision.

3. BACKGROUNDS OF ANAPHORA RESOLUTION IN QA

Interest in QA systems is quite recent from Information Retrieval (IR) research community perspective. MURAX (Kupiec, 1993) was the first QA system that combined traditional IR techniques with superficial NLP to obtain what his author denominated a high precision retrieval system. This system acted like an interface between a traditional IR system and the user. The process of this interface included understanding user’s questions, locating passages of text that were likely to contain the answer, and the application of shallow NLP techniques such as POS tagging and syntactic patterns comparison in order to detect and extract the expected answers. Since then, research in QA systems has been continuously growing, fostered mainly by the inclusion of a specific task for the evaluation of QA systems within the annual TREC Conference series.

There is almost a general agreement about the importance of applying Natural Language Processing for QA tasks. In fact, the majority of current systems integrate different levels of NLP processing such as lexical, syntactic, semantic and contextual analysis (Moldovan *et al.*, 2002) (Yang & Chua, 2002) (Hovy, Hermjakob & Lin, 2001) (Prager, Chu-Carroll & Czuba, 2001) (Ittycheriah & Roukos, 2002) (Harabagiu *et al.* 2001).

Several NLP techniques (such as POS tagging, parsing, name-entity tagging and question type classification) have already demonstrated their effectiveness. However, there are other NLP tools whose application for QA tasks needs to be deeply investigated in order to determine if they are really effective and to what extent. Co-reference resolution is currently considered one of these “other” techniques.

Although QA research community finds it reasonable to think that solving anaphora would substantially improve systems performance, co-reference phenomena have been investigated very little in relation to QA tasks. This fact relies mainly on the following aspects:

1. The low effectiveness of most current computational approaches. These algorithms need to achieve a minimum level of effectiveness for compensating the “noise” introduced by wrong solutions.
2. The high quantity of different kinds of anaphora to be solved. It would be important to know much more about the conditions that effectively solve each of them.

3. The high computational complexity of these algorithms. This fact limits its application on large-scale QA. Therefore, it would be very helpful to determine in which processing stage it is worth applying anaphora resolution techniques and the minimum amount of data to be treated.

Table 1. Co-reference in current QA Systems.

CO-REFERENCE	SYSTEMS											
	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
Pronouns												
Personal third person	x	x	x	x	x	x	x		x	x	x	x
Demonstrative		x	x	x	x	x	x		x	x	x	x
Reflexive and Reciprocal	x	x	x	x	x	x		x		x		x
Possessives	x	x	x	x		x	x		x	x	x	x
Zero-Pronouns		x	x			x		x				
Definite descriptions (DD)												
Same head	x		x	x		x	x		x	x		x
Associative relations (bridging)												x
Verb nominalization												x
Proper nouns and Acronyms												
Temporal Expressions												
Stage												
Answer selection	x	x	x	x	x	x	x	x		x	x	x
Question Analisys									x	x	x	x

We can find several QA systems that model some co-reference relations (as they are quoted in Table 1: [1] Morton, 1999; [2] Oard *et al.*, 1999; [3] Breck *et al.*, 1999; [4] Humphreys *et al.*, 1999; [5] Vicedo & Ferrandez, November 2000; [6] Lin & Chen, 2000; [7] Hovy *et al.*, 2001; [8] Alpha *et al.*, 2001; [9] Lin & Chen, 2001; [10] Oh *et al.*, 2001; [11] Litkowski, 2001; [12] Harabagiu *et al.*, 2001). Nevertheless, only a reduced number of works have performed a specific study about the application of anaphora resolution techniques for QA tasks. Particularly, a detailed analysis is accomplished in Vicedo & Ferrández (2000) where the authors analyse the effects of applying pronominal anaphora resolution using a simple sentence-based QA system.

Table 1 details the different kinds of anaphora whose resolution has been attempted by current approaches, as well as the processing stage in which these tools have been applied. As we can see, pronominal references, same head definite descriptions, proper nouns and acronyms are the kinds of anaphora solved by most of the systems. Other kinds of references such as bridging relations, ellipsis and temporal expressions are rarely solved and no system attempts to solve verbal phrase and more complex types of definite descriptions.

Co-reference resolution is usually applied at two different processing stages: answer selection and question analysis. For answer selection, co-reference is solved in relevant text extracts or documents to help entity detection and extraction. In the

second case, systems use anaphora resolution to track discourse objects across series of individual questions that request different aspects within the same context.

3.1 Solving co-reference at question analysis stage

Individual questions posed to a QA system rarely contain anaphoric references. Nevertheless, they are commonly used when questions are not asked in isolation, but in a series of related questions that ask for different aspects in the same context (*contextual questions*). In this case, the interpretation of a question later in the series could depend on the meaning or the answer of an earlier question.

- (14) (a) Who invented the telephone?
- (b) Where was he born?
- (c) How much money did he earn with this invention?

As we can observe in (14), understanding (b) and (c) questions needs solving several anaphoric references: *he* refers to the answer of the first question and *this invention* refers to *the telephone*, in the first question. As it is quoted in Harabagiu and Maiorano (2002), these questions require the resolution of several forms of anaphora. A question may use:

1. Demonstrative pronouns, like this, these or there (temporal expression); e.g. *On what day did this happen?* or *Where were these people located?* or *Name a company that flies there.*
2. Third person pronouns, like he or it; e.g. *What California winery does he own?* or *In what facility was it constructed?*
3. Possessive pronouns, like his or its; e.g. *What was his first radio song?*
4. Definite nominals, in which the definite article or the demonstrative pronoun indicate that the concept was already introduced by a previous question or answer; e.g. *What executive from the company was a member of the Supreme Council in 1994?* or *This city's name was later changed to what?*
5. Nominalizations of verbs used in previous questions; e.g. *When was construction begun?* following *In what facility was it constructed?*
6. Elliptical reference, in which the expected answer type is inherited from the previous question; e.g. *How many are poisonous to humans?* following *How many species of spiders are there?*
7. Cause-effect relational reference; e.g. explosive from *How much explosive was used?* is the cause of explosion from its preceding question *Which museum in Florence was damaged by a major bomb explosion in 1993?*
8. Meronymic reference, e.g. galleries from *Which galleries were involved?* are referenced as a part of the museum from the preceding question *Which museum in Florence was damaged by a major bomb explosion in 1993?*

Although anaphora resolution techniques are very complex and far away from an optimum performance level, their application at question analysis stage is usually

effective according to the following circumstances. First, time consumed for anaphora resolution is insignificant due to the small number of references to be solved and the reduced search space where the antecedents are located (previous questions and their respective answers). And second, this small search space reduces greatly the possibility of erroneous co-reference resolutions even if these algorithms lack a high accuracy rate. Nevertheless, processing context questions not only depends on solving co-references correctly but on a correct selection of the keywords that define the context and the answer selection techniques applied (Harabagiu *et al.*, 2001; Lin and Chen, 2001; Oh *et al.*, 2001; Litkowsky, 2001).

3.2 Solving co-reference at answer selection stage.

At this stage, co-reference resolution usually consists in replacing anaphoric references with its respective antecedents in text passages or sentences that are likely to contain the correct answer. This way, relevant passages are explicitly completed with all the information that can only be known by tracking these references. This process achieves two main objectives:

1. When computing question and relevant sentences similarity, systems usually compute the existence of matching terms between them. In this process, co-references are not taken into account and therefore, sentences that contain anaphoric references to question terms are wrongly evaluated. Solving these references means replacing them with their corresponding referring terms and consequently, allowing a correct evaluation of these sentences, which increase their similarity value with the question. This improvement allows detecting very relevant sentences that, in other cases, would have probably been discarded by the system. See example in (15).
- (15) (Question) What did Democrats propose for low-income families?
 (Answer) “*They* want to provide small subsidies for low-income families in which both parents work at outside jobs.”
 (Anaphora resolution) *They* = *Democrats*
2. Sometimes, the information required as answer may not appear explicitly in relevant text since it is expressed via anaphoric references. In this case, the system can only detect the answer when co-references are solved and anaphoric references are replaced with their corresponding antecedents. See example in (16).
- (16) (Question) Who is the head man of Digha?
 (Answer) “*He* is the sarpanch, or village head man of Digha, a hamlet or mud-and-straw huts”
 (Anaphora resolution) *He* = *Ram Bahadu*

The application of co-reference resolution techniques at this stage has encountered a lot of difficulties in terms of effectiveness and computational complexity. These difficulties are mainly caused by a great increase of the number of references to solve and their respective search space. This fact has probably caused that although several QA systems apply anaphora resolution techniques (see Table 1), only four of them have tried to compare their performance with and without applying co-reference resolution (Vicedo and Ferrández, 2000; Vicedo, Ferrández and Llopis, 2001; Lin and Chen, 2000; Vicedo and Ferrández, 2000b). Table 2 shows the main characteristics of the evaluations performed in these works. The text collection used, the number of questions processed and the percentage of correct answers provided by the system with and without solving third person pronominal anaphora. The last column shows the increment of performance obtained when applying anaphora resolution.

Table 2. Comparing QA performance when applying co-reference resolution at answer extraction stage.

SYSTEM	COLLECTION	QUESTIONS	BASELINE	ANAPHORA SOLVED	BENEFIT
(Vicedo & Ferrández, 2000)	LAT	83	32.5	55.4	22.9
(Lin & Chen, 2000)	TREC-9	682	44.7	42.2	-2.5
(Vicedo & Ferrández, November 2000)	TREC-9	682	51.6	52.9	1.3
(Vicedo, Ferrández & Llopis, 2001)	TREC-10	496	39.2	39.6	0.4

As we can see, results obtained when applying these tools for answer extraction are contradictory. In (Lin and Chen, 2000), the percentage of correct answers retrieved decreases 2.5 points when using pronominal anaphora resolution. On the contrary, the remaining tests showed better performances (22.9, 1.3 and 0.4 points respectively). All these tests and results could be considered similar except the one described in (Vicedo and Ferrández, 2000). In this case, the number of questions processed and the size of the document collection were smaller.

A deep analysis of all these works, their test conditions and their different results, gave us the notion that the benefits of solving anaphora in QA systems could also depend on the density of information of the document database. That is, as document collection increases, the number of correct answers to a question in the collection would probably increase and then, finding an answer without solving co-reference could be easier. The evaluation performed in section 5 will focus on analysing this hypothesis.

4. ANALYSIS OF COMPLEXITY INTRODUCED BY ANAPHORA RESOLUTION IN A QA SYSTEM

In this section, we present a study of the complexity introduced by the anaphora resolution module that we have used in our QA application (Vicedo and Ferrández, 2000b; Vicedo, Ferrández and Llopis, 2001). This module is embedded in a whole NLP system called *Slot Unification Parser for Anaphora Resolution (SUPAR)* (Ferrández, Palomar and Moreno, 1999).

SUPAR is described graphically in Figure 1. It works on the output of a part-of-speech tagger, and partial or full parses the text. Parsing is performed by means of the syntactic information stored for each language and the *Slot Unification Grammar (SUG)* formalism. At present, SUPAR can deal with Spanish or English texts, although it can be easily extended to other languages by representing the syntactic information in the SUG formalism, and using the proper POS-tagger.

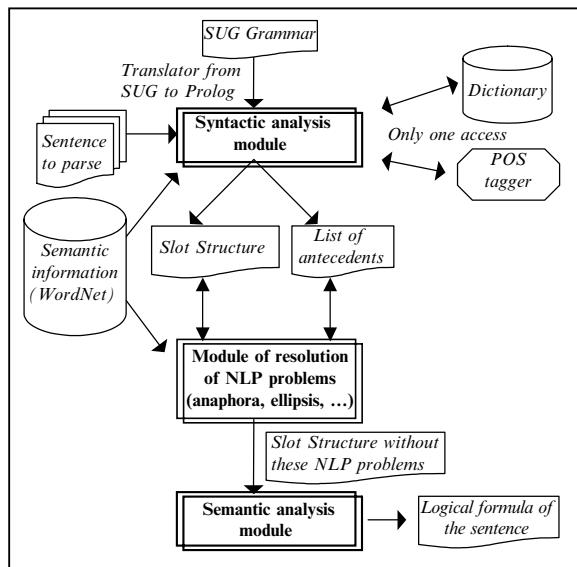


Figure 1. The architecture of the SUPAR system.

The **syntactic analysis module** takes as input the SUG grammar and the output of the POS tagger. It allows one to carry out either partial or full parsing of the text. With partial parsing, just one syntactic-tree per sentence is obtained, and SUPAR allows us to determine which constituents we wish to parse. Concretely, for QA and anaphora resolution, we partial parse coordinated NPs, coordinated PPs, verbal phrases and conjunctions, where NPs can include relative clauses, appositions,

coordinated PPs and coordinated adjectives. Conjunctions are used to split sentences into clauses.

The following **module of resolution of NLP problems** deals with anaphora resolution as well as other NLP problems such as extraposition, ellipsis or PP-attachment. This module builds a list of candidate antecedents from previous sentences in order to solve intersentential anaphora. This list stores knowledge obtained from previous stages, such as morphological (number or gender of the antecedent), syntactic (e.g. the head and modifiers), and knowledge about the position of the antecedent in the text (e.g. the identifier of the antecedent or the position of the antecedent with reference to the verb of the clause). In Figure 2, an example of the information stored for two antecedents is shown, and how anaphora resolution is solved. The output of this module is a structure (SS) where all the anaphors have been resolved. The anaphora resolution means that all the information about the anaphor and its antecedent is stored in the SS: entity identifier, morphological and syntactic information, and knowledge about the position of the antecedent in the text. In this way, co-reference chains are stored in the whole SS of the text, and it allows one to evaluate SUPAR with or without anaphora resolution as we have carried out in the last two TREC Question Answering tracks. This SS is then used in the last module of the system, in which the final logical formula of the sentence is obtained.

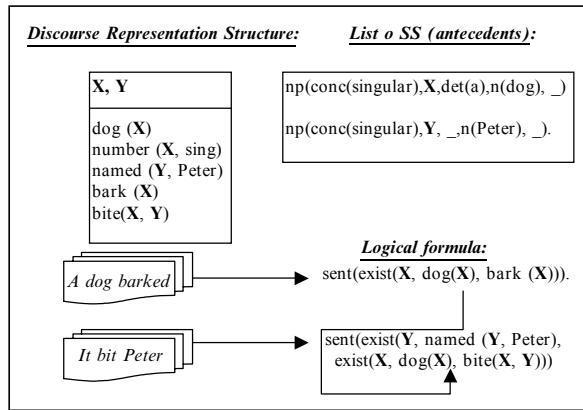


Figure 2. An example of anaphora resolution in the SUPAR system.

The SUPAR anaphora resolution module distinguishes between constraints and preferences as it is described in (Carbonell and Brown, 1988). Whereas constraints tend to be absolute and therefore discard possible antecedents, preferences tend to be relative and require the use of additional criteria (e.g. the use of heuristics that are not always satisfied by all antecedents). It uses as input the morphological knowledge of the POS tagger, and the partial syntactic structure of each sentence. Therefore, we should remark that it does not use semantic knowledge. Moreover, it

carries out a clause segmentation of each sentence in order to run c-command restrictions, in spite of the partial parsing (Palomar *et al.*, 2001). Furthermore, it detects English pleonastic pronouns, e.g. *It's teatime*, with a precision of 91%, evaluation that has been run on 970 pronouns of the TREC Federal Register collection. With all this knowledge, when we resolve personal, demonstrative, reflexive and Spanish zero-pronouns on texts of different genres (newspapers, technical manuals, novels, etc.), we have obtained the following precision, measured as the number of correctly solved pronouns divided by the total number of solved pronouns: English pronouns: $835/1.163 = 74\%$, Spanish pronouns: $921/1.144 = 81\%$. The resolution failures have been caused by the input errors (i.e. POS tagger, partial parser, sentence and clause segmentator) and mainly by the lack of semantic knowledge.

Next, we present some figures that can give us an idea of the complexity that SUPAR anaphora resolution module introduces in a QA system. The following experiments have been carried out on 887 randomly selected documents of the TREC collections: the Los Angeles Times (LAT) and the Foreign Broadcast Information Service (FBIS), as it is described in Table 3. In this Table, we can see that the SUPAR's total running time is 3,389 seconds (up to 256 words per second), which includes parsing process (up to 2,001 words per second) and anaphora resolution module. It does not include the POS tagging time, which has been carried out by the Tree-Tagger². Anaphora resolution module takes about 89% of the total running time. These measures have been obtained on a Pentium III, 1000 GHz, 128 Mb RAM. The anaphora module takes such a long time because it has to segment each sentence into clauses and it has to create the list of possible candidates, which contains every noun phrase in the text. The list of candidates contains all configurational knowledge, i.e. the verb of the clause, position with reference to the verb of the clause, if they are included in a prepositional phrase or in another noun phrase, the number of times that the noun phrase has appeared in the text and/or with the verb of the clause, etc. The time SUPAR spends on processing the list of candidates is 89.7% of the anaphora resolution time in the LAT collection. In this evaluation, SUPAR has resolved 216 reflexive pronouns and 8,722 personal and demonstrative pronouns. For all kinds of pronouns, we have only considered the noun phrases in the same sentence as the pronoun or in the previous four sentences, and 396,977 candidates have been found, which means an average of 44.4 candidates per pronoun. After constraints (c-command and morphological agreement), there are 17.8 candidates per non-reflexive pronoun, and 1.3 candidates per reflexive pronoun, on average. This means that a high degree of ambiguity has to be finally resolved by preferences. In (17) an example extracted from the LAT collection is presented. It shows the SUPAR's segmentation into candidates and clauses. In this example, 10 candidates have been numerated and delimited between square brackets, and two clauses are divided by the conjunction *that*.

Table 3. SUPAR evaluation.

	<i>N. Doc.</i>	<i>Total Words</i>	<i>Average Words per sentence</i>	<i>Total Time (second)</i>	<i>SUPAR Speed (w/sec.)</i>	<i>Parsing Speed (w/sec.)</i>	<i>% Time anaphora</i>
<i>LAT</i>	370	281,149	20.9	1,580	178	1,939	89.1%
<i>FBIS</i>	517	462,221	26.1	1,809	256	2,001	89.4%

- (17) [[David R. Marples's]₁ new book, his second on [the Chernobyl accident of [April 26, 1986]₂]₃]₄, is [a shining example of [the best type of [non-Soviet analysis into [topics]₅]₆]₇]₈ **that** only recently were [absolutely taboo in [Moscow official circles]₉]₁₀.

Lastly, we can extract the following conclusions from the analysis that we have carried out in this section. Pronominal anaphora resolution is necessary because of the high number of pronouns (216 reflexive pronouns and 8,722 personal and demonstrative pronouns), although it is a hard task since there is a high ambiguity (there are 44.4 candidates per pronoun and 17.8 candidates per non-reflexive pronoun after constraints). Finally, anaphora resolution takes about 89% of SUPAR running time, which makes a global speed of up to 256 words per second. This may make it computationally expensive to resolve anaphora but as we will show in the following section, QA benefits from this resolution.

5. ANAPHORA RESOLUTION EFFECTIVENESS AND INFORMATION DENSITY FOR QA

As we have introduced before, using co-reference resolution techniques seems essential for a correct processing of contextual questions. On the other hand, the contradictory results obtained at answer extraction stage do not allow us to confirm if its application is convenient at this stage or not (Table 2).

The different conditions in which those evaluations were developed led us to think that probably, their different results could be related to the density of information in the document collection. That is, as document collection increases, the number of correct responses to a question will probably increase too, making it much easier to find a correct response without solving co-references.

In order to verify this hypothesis, we planned to extend the test developed by Vicedo and Ferrández (2000), by repeating this experiment, under the same conditions, but increasing progressively the size of the document collection. This work has been organised as follows. First, we introduce the experiment and results obtained by Vicedo and Ferrández (2000). Second, we detail the experiments developed for measuring the effects of information density for QA tasks. Finally, we present and analyse the results obtained.

5.1 Effectiveness of anaphora resolution in QA

This section summarizes the work developed by Vicedo and Ferrández (2000) since both, the test performed and the QA system applied will be used for the experiments carried out later.

5.1.1 The QA system

The QA system is made up of three modules: document retrieval, document analysis and sentence ranking. The first module is a standard IR system that retrieves relevant documents for each query. The second module manages (optionally) with anaphora resolution in queries and retrieved documents. The third module evaluates and ranks relevant sentences depending on their likelihood of containing the answer to the question.

Document retrieval is performed by a standard document ranking IR system based on the approach described by Salton (1989). Each term in the query and documents is assigned an inverse document frequency (*idf*) score based on the document collection. This measure is computed as follows (1):

$$idf(t) = \log\left(\frac{N}{df(t)}\right) \quad (1)$$

Where N is the total number of documents in the collection and $df(t)$ represents the number of documents which contains term t . Query expansion consists of stemming terms using a version of Porter's stemmer. Document similarity to the query is computed using the cosine similarity measure.

- *Document Analysis*. This step is applied over the 50 best matching documents retrieved at document retrieval stage. This analysis consists on detecting sentence boundaries and solving pronominal references. As a result, relevant documents are split into sentences and each pronoun is associated with the noun phrase it refers to in the documents. Co-reference resolution is only applied when the system is required to. Finally, all the sentences in relevant documents are directed to the sentence extraction stage.
- *Sentence Extraction*. Each term in the query is weighted with its *idf* value. Each document sentence is weighted the same way. When anaphora resolution option has been activated, pronouns are given the weight of their antecedents. Sentences are then ranked depending on their similarity to the query and the top 5 sentences are returned as final answer. Similarity between a query and a sentence is obtained by computing the cosine similarity measure.

5.1.2 The evaluation

The evaluation was performed using the TREC corpus *Los Angeles Times* (LAT) due to its high level of pronominal references. The test question set was built with the collaboration of 6 people. Each of them read 30 different documents from a set of 180 randomly selected documents pertaining to the LAT collection. After that, they proposed 150 queries requiring factual information in relation to the topics they

had read about. Later on, these queries were filtered based on their expressing the user's information need clearly in a natural way. Moreover, questions that presented grammatical mistakes and those including unusual or very specific expressions that seemed to be extracted directly from the sentences that contained the answer were discarded. Finally, the test set was made up of 83 questions whose correct answer appeared at least once in the LAT collection. Questions in (15) and (16) are examples of this set.

Table 4 summarizes the results obtained in this evaluation. The "baseline" run performed QA task without solving pronominal references while the "anaphora solved" run processed the same 83 questions applying pronominal anaphora resolution. As we can see, the second run answered correctly 46 questions while first run only answered 27. Therefore, pronominal anaphora resolution improved the percentage of correct answers retrieved by the baseline run by 22.9 points.

Table 4. Results in Vicedo & Ferrández (2000).

<i>Questions</i>	<i>83</i>	<i>Correct Answer</i>	<i>%</i>
Baseline		27	32.5
Anaphora solved		46	55.4

5.2 *Information density tests*

As we have stated before, the only difference between the tests developed in all previous works related to co-reference resolution lies basically in the size of their respective document collection. The evaluation performed in this article investigates if the benefits of applying anaphora resolution techniques depend on the density of information contained in the document database.

Our experiment starts from the previous one (Vicedo and Ferrández, 2000). We use the same QA system and the questions that were correctly answered when anaphora resolution was performed. This way, using only these 46 questions allows investigating accurately the effects of pronominal anaphora resolution by isolating the effects of increasing document collection over the 37 queries that were not answered in the test performed in (Vicedo and Ferrández, 2000).

From this departure point, we have designed several tests whose only difference resides in the size of the document used. The different tests were performed using the document collections included in TREC-9 QA track. This document set consists of 978,952 documents from the following collections:

1. Los Angeles Times (LAT).
2. Foreign Broadcast Information Service (FBIS).
3. Financial Times (FT).
4. AP Newswire (AP).
5. Wall Street Journal (WSJ).
6. San Jose Mercury News (SJMN).

We designed six different tests that processed the 46 selected questions without solving pronominal references. The first one used LAT as document collection. The second test was performed using LAT and FBIS collections. For each of the remaining tests, one more collection was added to the document database by following the order they had been enumerated before.

The decision to use TREC collections was based on several circumstances. First, the LAT collection was used in (Vicedo and Ferrández, 2000) and our experiments could start from the exact point this work concluded. Second, as related works that measure the effectiveness of co-reference resolution used the TREC document set, we would be able to compare the results obtained from this new evaluation with the results obtained previously in these works. And third, it would be interesting to evaluate system performance on a standard test collection in order to scale the results obtained and evaluate what would be the impact of applying these techniques to other QA systems. For this purpose we performed a final test that processes the 682 questions included in TREC-9 test collection without applying co-reference resolution techniques.

5.3 Results

Table 5 presents the results achieved in our tests. The second column shows the number of questions correctly answered by the system. The third column displays the percentage of questions answered in each test. This percentage is obtained by taking as base the percentage of correct answers achieved by the test developed in (Vicedo and Ferrández, 2000) when pronominal anaphora resolution was applied. The fourth column indicates the benefit it would have obtained if co-references had been solved. Finally, the fifth column details the number of documents the collection database contained in each test.

Table 5. Test Results

Tests	Correct Answer	%	Incr	Documents ($\times 10^6$)
Anaphora solved	46	55.4	0.00	0.132
1	27	32.5	22.9	0.132
2	30	36.1	19.3	0.267
3	35	42.2	13.3	0.477
4	44	53.0	2.40	0.720
5	44	53.0	2.40	0.893
6	44	53.0	2.40	0.983
TREC-9	327	47.9		0.983

As we can observe, the benefit of solving co-references decreases, as the size of the document collection increases. Figure 3 compares the increment of benefit obtained when pronominal anaphora is solved, with the number of documents included in the database collection. The percentage of questions that need anaphora resolution to be answered decreases to a minimum level. From this point, the application of anaphora resolution techniques is essential if the system wants to answer all the test questions.

These results are comparable to the ones obtained in related works because all the experiments have been performed over the same document collection (Table 2). The 2.4 points of benefit obtained are similar to the 1.3 and 0.4 points obtained in (Vicedo and Ferrández, 2000) and (Vicedo, Ferrández and Llopis, 2001) and even the (-2.5) points achieved in (Lin and Chen, 2000). These small differences may rely on the type of co-references solved and the question test set used in each case. Furthermore, this system performs reasonably well with TREC-9 standard test collection, obtaining the correct answer for a 47.9% of the questions without using other effective tools such as named-entities taggers or answer classifiers. This fact confirms that results obtained in this evaluation can be easily scaled to other QA approaches.

We have proved that solving co-references at answer selection stage allows the discovery of essential information for QA tasks. However, the high computational complexity of these tools and their relation to systems performance advise us to apply them when there is a low level of redundant information in the document collection.

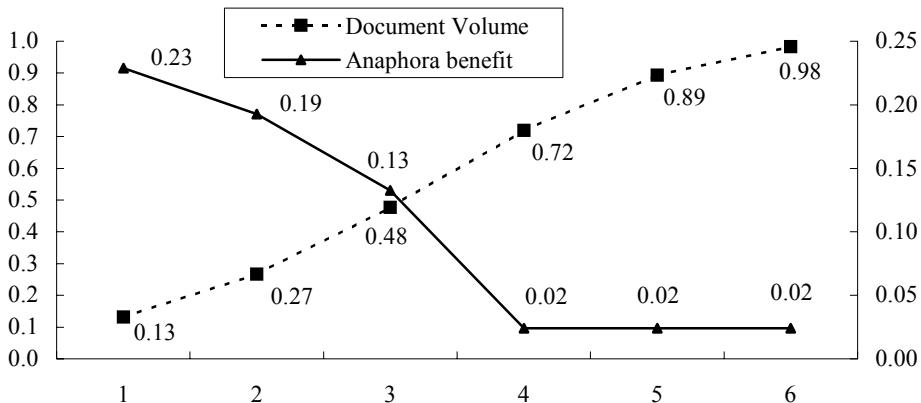


Figure 3. Test results comparison.

Nevertheless, this condition is difficult to evaluate since the density of information for a document collection may be very high for certain answers and very low for others. Consequently, an efficient use of co-reference resolution techniques at this processing stage would need to be related to the application of answer validation techniques (Harabagiu *et al.*, 2001; Ittycheriah, Franz and Roukos, 2001).

This way, co-reference resolution would only be used when needed. That is, when the correct answer is not found by applying low-cost answer selection techniques.

6. CONCLUSIONS

In this paper, we have given an overview of anaphora resolution problem and its influence in Question Answering (QA) systems. We have reviewed the different QA systems that use any kind of co-reference resolution as well as the specific works that investigate the effects of applying anaphora resolution algorithms for QA tasks.

Co-reference resolution algorithms have been applied in two QA processing stages: question analysis and answer selection. The need to solve co-reference at question analysis stage for contextual questions processing has been demonstrated in several of these works. However, QA community has not arrived at a consensus about the need to use these algorithms for entity tracking in documents at answer selection stage mainly due to the different results obtained in previous works.

Besides summarizing the state of the art of co-reference in QA, this paper succeeds in explaining the main reason for these contradictory results. The series of tests performed demonstrates that the benefits of solving pronominal anaphora also depend on the density of information of the document collection where QA systems perform the search. Results show that pronominal anaphora resolution improves system performance and this benefit may range approximately between 2.4 and 22.9 points depending on the density of information in the document collection.

Furthermore, an analysis of the complexity introduced by the SUPAR anaphora resolution module in a QA system has been performed. The precision, ambiguity, speed and QA precision improvement has been measured.

The future of co-reference resolution in QA systems is not restricted to current applications. There are many new aspects to be investigated. Particularly, the first step could be the study of inter document entity tracking for processing questions whose answer needs to relate information extracted from more than a single document.

7. NOTES

¹ This paper has been partially supported by the Spanish Government (CICYT) project number TIC2000-0664-C02-02 and (PROFIT) project number FIT-150500-2003-412.

² <http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html>

8. REFERENCES

- Allen, J. (1995). Natural Language Understanding. The Benjamin/Cummings Publishing Company, Inc.
- Alpha, S., Dixon P., Liao C. & Yang C. (2001). Oracle at TREC 10. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Barbu, C. & Mitkov, R. (2001). Evaluation tool for rule-based anaphora resolution methods. In *Proceedings of the 39th Annual Meeting and 10th Conference of the European Chapter of the Association for Computational Linguistics (ACL'2001)*. 34-41. Toulouse (France).

- Byron, D. (2001). The Uncommon Denominator: A Proposal for Consistent Reporting of Pronoun Resolution Results. *Computational Linguistics*. Special Issue on Anaphora and Ellipsis Resolution. 27(4): 569-577.
- Bean, D. L. & Riloff, E. (1999). Corpus-based Identification of Non-Anaphoric Noun Phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, 373-380, Maryland (USA).
- Breck E., Burger J., Ferro L., House D., Light M. & Mani, I. (1999). A Sys Called Qanda. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 499-506, Gaithersburg, Maryland (USA).
- Carbonell, J. G. & Brown, R. D. (1988). Anaphora resolution: A multi-strategy approach. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING)*, 96-101, Budapest (Hungary).
- Chan, S. W. K. & T'Sou, B. K. (1999). Semantic Inference for Anaphora Resolution: Toward a Framework in Machine Translation. *Machine Translation*, 14(3/4), 163-190.
- Connoly, D., Burger, J. & Day, D. (1994). A Machine learning approach to anaphoric reference. In *Proceedings of the International Conference on New Methods in Language Processing (NEMLP)*, 255-261, Manchester (UK).
- Ferrández, A., Palomar, M. & Moreno, L. (1999). An empirical approach to Spanish anaphora resolution. *Machine Translation*, 14(3/4), 191-216.
- Ferrández, A. & Peral, J. (2000). A Computational Approach to Zero-pronouns in Spanish. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, 166-172, Hong-Kong (China).
- Gariglano, R., Urbanowicz, A. & Nettleton, D. J. (1998). University of Durham: Description of the LOLITA System as used for MUC-7. Morgan Kaufman Publishers.
- Ge, N., Hale, J. & Charniak, E. (1998). A statistical approach to anaphora resolution. In *Proceedings of the Sixth Workshop on Very Large Corpora*, 161-170, Montreal (Canada).
- Harabagiu, S. & Maiorano, S. (2000). Multilingual coreference resolution. In *Proceedings of Conference on Applied Natural Language Processing/North American Chapter of the Association for Computational Linguistics (ANLP-NAACL)*, 142149, Seattle (USA).
- Harabagiu S., Moldovan D., Pasca M., Mihalcea R., Surdeanu M., Bunescu R., Gîrju R., Rus, V. & Morarescu P. (2000). FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 479-488, Gaithersburg, Maryland (USA).
- Harabagiu, S., Moldovan, D., Pasca, M., Surdeanu, M., Mihalcea, R., Gîrju, R., Rus, V., Lacatusu, F., Morarescu, P. & Bunescu, R. (2001). Answering complex, list and context questions with LCC's Question -Answering Server. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Harabagiu, S. & Maiorano, S. (2002). Three Ways to Customize Reference Resolution. In *Proceedings of 2002 International Symposium on Reference Resolution for Natural Language Processing*, 17-24, Alicante (Spain).
- Hirst, G. (1981). Anaphora in Natural Language Understanding. Berlin: Springer-Verlag.
- Hovy, E., Hermjakob, U. & Lin, C. (2001). The Use of External Knowledge in Factoid QA. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Humphreys, K., Gaizauskas, R., Azzam, S., Huyck, C. & Mitchell, B. (1998). University of Sheffield: Description of the LaSIE-II System as used for MUC-7. Morgan Kaufman Publishers.
- Humphreys, K., Gaizauskas, R., Hepple, M. & Sanderson, M. (1999). University of Sheffield TREC-8 QA System. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 707-716, Gaithersburg, Maryland (USA).
- Ittycheriah, A., Franz, M. & Roukos, S. (2001). IBM's statistical Question Answering System – TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Ittycheriah, A. & Roukos, S. (2002). IBM's Statistical Question Answering System-TREC 11. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, Gaithersburg, Maryland (USA).
- Kennedy, C. & Boguraev, B. (1996). Anaphora for Everyone: Pronominal Anaphora resolution without a Parser. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, 113-118, Copenhagen (Denmark).
- Kupiec, J.M. (1993). MURAX: A robust linguistic approach for question-answering using an on-line encyclopaedia. In *Proceedings of the 16th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 181-190, Pittsburgh, PA (USA).

- Lappin, S. & Leass, H. (1994). An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4), 535-561.
- Lin, C. & Chen, H. (2000). Description of NTU QA and CLIR Systems in TREC-9. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 389-398, Gaithersburg, Maryland (USA).
- Lin, C. & Chen, H. (2001). Description of NTU System at TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Litkowski, K.C. (2001). CL Research Experiments in TREC-10 Question Answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Mitkov, R. (1999). Multilingual Anaphora Resolution. *Machine Translation*, 14(3/4), 281-299.
- Mitkov, R. (2002). Anaphora resolution. Longman.
- Moldovan, D., Harabagiu, S., Gîrju, R., Morarescu, P., Lacatusu, F., Novischi, A., Badulescu, A. & Bolohan, O. (2002). LCC Tools for Question Answering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, Gaithersburg, Maryland (USA).
- Morton, T. S. (1999). Using Coreference in Question Answering. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 685-688, Gaithersburg, Maryland (USA).
- Morton, T. S. (2000). Coreference for NLP Applications. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, 173-180, Hong-Kong (China).
- Nakaiwa, H. & Shirai, S. (1996). Anaphora Resolution of Japanese Zero Pronouns with Deictic Reference. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, 812-817, Copenhagen (Denmark).
- Oard, D., Wang, J., Lin, D. & Soboroff, I. (1999). TREC-8 Experiments at Maryland: CLIR, QA and Routing. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 623-636, Gaithersburg, Maryland (USA).
- Oh, J., Lee, K., Chang, D., Seo, C. & Choi, K. (2001). TREC-10 Experiments at KAIST: Batch Filtering and Question Answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Orasan, C., Evans, R. & Mitkov, R. (2000). Enhancing preference-based anaphora resolution with genetic algorithms. In *Proceedings of NLP-2000*, 185-195, Patras (Greece).
- Palomar, M., Ferrández, A., Moreno, L., Martínez-Barco, P., Peral, J., Saiz-Noeda, M. & Muñoz, R. (2001). An Algorithm for Anaphora Resolution in Spanish Texts. *Computational Linguistics*. Special Issue on Anaphora and Ellipsis Resolution, 27(4), 545-567.
- Paul, M., Yamamoto, K. & Sumita, E. (1999). Corpus-based anaphora resolution towards antecedent preference. In *Proceedings of the ACL Workshop on Coreference and Its Applications*, 47-52, College Park, MD (USA).
- Poesio, M. & Vieira, R. (1998). A Corpus-Based Investigation of Definite Description Use. *Computational Linguistics*, 24(2), 183-216.
- Prager, J., Chu-Carroll, J. & Czuba, K. (2001). Use of WordNet Hyponyms for Answering What-Is Questions. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Reinhart, T. (1983). Anaphora and Semantic Interpretation. Croom Helm Backenham, Kent.
- Salton, G. (1989). Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison Wesley. New York.
- Strube, M. & Hahn, U. (1999). Functional centering: Grounding referential coherence in information structure. *Computational Linguistics*, 25(3), 309-344.
- Tetreault, J. R. (2001). A Corpus-Based Evaluation of Centering and Pronoun Resolution. *Computational Linguistics*. Special Issue on Anaphora and Ellipsis Resolution, 27(4), 507-520.
- Vicedo, J.L. & Ferrández, A. (2000). Importance of Pronominal Anaphora resolution in Question Answering systems. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, 555-562, Hong Kong (China).
- Vicedo, J.L. & Ferrández, A. (2000b). A semantic approach to Question Answering systems. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 511-516, Gaithersburg, Maryland (USA).
- Vicedo, J.L., Ferrández, A. & Llopis, F. (2001). University of Alicante at TREC-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland (USA).
- Webber, B. (1979). A formal approach to discourse anaphora. New York: Garland Publishing.

- Wee Meng Soon, Hwee Tou Ng, Daniel Chung Yong Lim. (2001). A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics*. Special Issue on Anaphora and Ellipsis Resolution, 27(4), 521-544.
- Yang, H. & Chua T. (2002). The Integration of Lexical Knowledge and External Resources for Question Answering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, Gaithersburg, Maryland (USA).

2

QUESTION PROCESSING

QUESTIONS AND INTENTIONS

Abstract: Before a question can be answered by an advanced Question Answering (QA) system, it must be understood at several different levels. First, the *complexity* of the question needs to be identified based on a combination of syntactic, semantic and pragmatic knowledge. Second, since questions are rarely asked in isolation, the question *context* needs to be determined for better understanding its request. Third, it is difficult to separate the question *intentions* from the question formulation, therefore plausible *implications* need to be coerced from each question. Fourth, mechanisms that either *accept* or *reject* the implied intentions are needed. All these different processes impact on the question understanding and implicitly on the accuracy of the returned answers.

1. THE PROBLEM

The rapid growth of the Internet has made an overwhelming amount of textual information available online. One solution to mining information of interest from these vast electronic resources is provided by *textual question answering*, a technology that returns short and exact answers to questions expressed in natural language. The answers are extracted from electronic documents either posted on the Web or from local Intranet collections.

Recent work in Textual Question Answering has been driven by a series of evaluations conducted as part of the Text REtrieval Conferences (TREC). Established in 1999, the TREC Question Answering (QA) track has tested the ability of systems to provide answers to questions, regardless of the domain of interest. Questions tested in the TREC QA track have traditionally been selected from a class of questions that targets only *trivia information*, e.g., *Who is the author of the book "The Iron Lady - A Biography of Margaret Thatcher"?* or *What is an atom?*. Although these questions may seem relatively simple linguistically, they exhibit a surprising range of syntactic, semantic and pragmatic complexities.

In particular, pragmatic knowledge has been shown to play an important role in answering questions. Understanding the intentions that lead to the asking of a question can help systems resolve lexicosemantic ambiguities and better identify the answers that should be returned to a particular question.

Users seek information for a number of reasons. For example, the answer to a question such as

Q1: Where is the Taj Mahal?

asked by someone sightseeing in Asia would necessarily have to be

A₁¹: Agra, India.

The same question asked by an investment banker reviewing the properties of Donald Trump is answered by:

A₂¹: Atlantic City, New Jersey.

Thus, the interpretation of the question cannot be separated by its context nor can the intention of the questioner be ignored.

One very simplistic treatment of question interpretation involves the assumption that the same intentions are shared by two people that ask the same question. Obviously, this assumption is not tenable. Imagine the surprise on the banker's face when he was presented with answer *A₁¹* after he asked question *Q1*. Incorrect answers are determined sometimes by the interpretation of the question in isolation, without processing their context and without modelling the user's intentions.

In natural language communications, intentions are not expressed directly. Uncovering the intentions of a question depends on the inference of the context and on the semantic and pragmatic knowledge available. For example, if the context of question *Q1* comprises several references to tourism attractions or travel planning, the ambiguity of the user's intentions may be resolved. However, such references need to be enforced by semantic knowledge, e.g., by deriving from the cohesive structure of the context conceptual and relational information that assigns the topic of *travel* instead of the topic of *property*. Semantic knowledge is available from several lexico-semantic resources, e.g., WordNet (Fellbaum, 1998), FrameNet (Baker *et al.*, 1998), or PropBank (Kingsbury *et al.*, 2002).

The interpretation of questions relies also on pragmatic knowledge, which is not directly derivable from a given source. For every specific topic, pragmatic knowledge complements the semantic knowledge and allows inferences to be drawn. For example, in the case of the TRAVEL topic, semantic concepts such as MONUMENT, TOURIST or SIGHTSEEING are organized into structures that define their functions. Such semantic structures may be automatically recognized in the questions or their candidate answers, highlighting the role of the TOURIST to SIGHTSEE and eventually to ADMIRE the artistic and historical MONUMENTS. (Gildea and Jurafsky, 2002) report on a promising method of identifying semantic structures similar to those encoded in FrameNet whereas (Gildea and Palmer, 2002) describe similar mechanism that operate with the semantic knowledge encoded in PropBank. However, even for simple questions like:

Q2: How difficult is it to visit the Taj Mahal?

More than semantic knowledge is required for the correct interpretation of the request. Question *Q2* does not ask only if the monument is easily accessible, but it also implicitly asks for information regarding the way the visit may take place. Thus the processing of *Q2* does not involve only the recognition of (1) a manner semantic relation, showing how one can organize the visit to the Taj Mahal; and (2) a quantification of the difficulty of the way the visit may be take place. Processing *Q2* also requires access to pragmatic information that encodes the following facts: (1). to visit a place, one needs to travel by riding a train or flying a plane; (2) the visit

involves a place of origin and a destination; (2) the visit may span a few hours, a full day or several days, in the latter case eventually one needs to book a hotel room. Answer *A2* returns the desired information by relying on the pragmatic knowledge implied by question *Q2*:

A2: This is a one-day tour from New Delhi to the Agra to visit the Taj Mahal and other ancient sites in the former capital. An early morning transfer to the New Delhi train station for the 2 hour journey on the Shatabdi Express A/C train to Agra, home of the Taj Mahal. The full day will be spent exploring the wonders of Agra including the Taj Mahal and the Red Fort of Agra. A late afternoon train will return you to New Delhi.

Answer *A2* implies that the visit is not at all difficult, because one-day organized tours in the comforts of express trains are typically enjoyable. This implication can be recognized if additional pragmatic knowledge is available. Such knowledge needs to contrast pleasant travel experiences with less tempting experiences. Since pragmatic knowledge is not readily available, its acquisition and representation is a challenging, yet much needed requirement for question processing.

Moreover, the interpretation of some questions is hindered by the fact that they contain figurative or idiomatic expressions, imposing interpretations that either reinforce the derived intentions or simply generate representations of new intentions. Pragmatic knowledge thus needs to also enable the coercion of non-literal expressions. Such coercions interact with the mechanisms of accepting or rejecting the question implications⁴. Questions *Q3* and *Q4* constitute examples of interrogations requiring metonymic coercion:

Q3: What groups or factions are tied to Usama Bin Laden's umbrella organization?

Q4: What is the penalty for cooking the books?

Finding the answer to *Q3* depends on the coercion of the expression *umbrella organization* into possible instances of organizations controlled by Bin Laden. Similarly, for *Q4*, the expression *cooking the books* needs to be coerced into *altering the accounting records*. The latter coercion relies on the idiomatic interpretation of the expression rather than on pragmatic knowledge. When question interpretation is based on pragmatic knowledge, different plausibility scores are assigned to the uncovered implicatures, depending on the available evidence. These plausibility scores enable the codification of mechanisms for accepting or rejecting implications.

In this chapter we describe several classes of questions that are better processed if the intentions and implications are identified. In Section 2 we describe several forms of complex questions that presuppose both semantic and pragmatic knowledge processing. Section 3 details the processing of questions in contexts whereas Section 4 presents the coercion of implicatures and the mechanisms for detecting question intentions. Section 5 shows the experimental results and Section 6 summarizes the conclusions.

⁴ H.R. Grice defines conversational implications as *implicatures* in (Grice 1972).

2. COMPLEX QUESTIONS

Real-world users of QA systems tend to ask complex questions which contain implied information that is not recognizable at the syntactic or semantic level. This happens because people do not separate their beliefs or intentions from the formulation of a question. For example, when asking:

*Q5: Biographical information needed on Elizardo Sanchez, Cuban dissident.*⁵

The user relies on the presupposition that information about Elizardo Sanchez must contain (*i*) typical biographical information, e.g., date and place of birth, facts about his family and friends; but also (*ii*) information typical for political dissidents, e.g., education, political views, a list of his protests or political actions, as well as (*iii*) information characteristic only to Elizardo Sanchez, e.g., what is he best known for.

In order for a QA system to process successfully question Q5, the system needs to be capable of inferring the same presupposition. As noted in (Levinson, 1983), presupposition is a form of pragmatic inference that is based on **presupposition triggers**. For question Q5 the verb *needed* is an implicative verb⁶.

However, the inference is not that biographical information on Elizardo Sanchez exists, but rather that such information needs to be classified according to the criterion that (*i*) Elizardo Sanchez is a PERSON, therefore his biographical information needs to include data about his birth, education, family; and (*ii*) Elizardo Sanchez is a unique individual, known as a Cuban dissident, therefore the information must contain also facts about his political activities and beliefs. Pragmatic knowledge reflecting the typical stages of a person's life is assembled in a list of linguistic patterns used for retrieving biographical information.

Examples of surface patterns enabling the recognition of birth place and date are:

PATTERN 1: PERSON was born on DATE [, in LOCATION]
 PATTERN 2: PERSON is a native of LOCATION
 PATTERN 3: PERSON saw the day of light in LOCATION
 PATTERN 4: PERSON's roots are in LOCATION

Figure 1. Example of Surface Patterns Used for Processing Complex Questions.

Patterns for place/date of birth, death, education, marriage, divorce, work experience and affiliation to organizations (e.g., religious, professional) can be used to acquire general biographical information. To gather information that is typical to a given occupation, activities typical for each occupation must be known. Filatova and Prager (2005) report on a method that discovers occupation-related activities by ranking verbs through a stochastic matrix that corresponds to the transition between

⁵ This question was extracted from the logs of the dialog performed in the Wizard-of-Oz experiments performed in the AQUAINT program in 2002. The Advanced QUestion and Answering for INTElligence (AQUAINT) program is sponsored by ARDA (www.ic-arda.org).

⁶ Example of implicative verbs are *manage*, as in “*John managed to open the door*”, or *forget*, as in “*John forgot to lock the door*”. The concept of implicative verbs was introduced in (Karttunen, 1983).

activities and entities that are involved in activities. By considering only triplets that involve (i) the name of a PERSON, (ii) a verb, and (iii) another name or entity, generalized patterns can be acquired. When characterizing the activities of dissidents, the following generalized patterns are discovered by the method reported in (Filatova and Prager, 2005):

PATTERN 5: PERSON critisized REGIME
PATTERN 6: PERSON released DATE
PATTERN 7: PERSON denied allegations
PATTERN 8: PERSON protests REGIME's ACTION

Figure 2. Generalized Patterns for Classifying Complex Questions.

One major limitation of this method stems from the need to recognize many types of names and semantic classes for virtually any occupation. For example, in the case of dissidents, names of political regimes need to be recognized, as well as names of political parties (e.g., “*Fidel Castro’s government*”, “*North Korea*”). Several recent attempts of enhancing the coverage of name classes have been reported (Morarescu and Harabagiu, 2004; Sekine *et al.*, 2002). Additionally, we need to allow for patterns that recognize common nouns as well, in order to better capture the typical activities of a given occupation. Pattern 7 is such an example. Patterns must deal not only with relations between events and entities, but also with relations between events, as it is illustrated by Pattern 8 in Figure 2.

Practically, a complex question such as Q5 need to be translated into a series of similar questions, e.g., Q5¹: “*When and where was Elizardo Sanchez born?*”, Q5²: “*Where did Elizardo Sanchez go to school?*”, and Q5³: “*What did Elizardo Sanchez criticize?*”. Such questions have the same degree of complexity as the questions evaluated in TREC. Their complexity is validated by the fact that Q5¹ asks about a DATE and a PLACE; Q5² about an EDUCATION INSTITUTION or a list of such semantic types; and Q5³ asks about a REGIME.

At the core of processing factual questions is the recognition of the **expected answer type (EAT)**, since the semantic category of the expected answer guides both the retrieval of paragraphs that contain candidate answers and the actual answer extraction. The recognition of the expected answer type is done by (a) identifying the question word/phrase that indicate the semantic category of the expected answer; and (b) mapping this word or phrase against an off-line ontology of possible answer categories. Sometimes the questions stem (e.g., *when*, *where*) indicates without ambiguity the expected answer type (e.g., DATE, LOCATION). However, more frequently, questions stems are ambiguous (e.g., *what*, *name*), and the expected answer type needs to be identified using another word or phrase. For example, the expected answer type of the TREC question: *Name the designer of the shoe that spawned millions of plastic imitations, known as “jellies”*. is derived from the semantic class of the word designer, which is a PERSON. Algorithms for detecting the expected answer type of TREC-like questions have been reported in (Harabagiu *et al.*, 2000; Radev *et al.*, 2000; and Srihari and Li, 2000).

If the expected answer type of factoid questions can be identified in almost 64% of the cases (as reported in Moldovan *et al.*, 2003) we expect that the ability to

detect the answer type of complex questions also depends on the recognition of the semantic structure of the complex answer. In order to successfully process complex questions, four enhancements have been considered:

1. representing answer types as complex semantic structures instead of simple semantic concepts;
2. identifying the intentional structure of the question and the candidate answers;
3. representing and using domain-specific knowledge for the question topic; and
4. generating questions that are relevant for a question topic.

2.1 Complex Answer Types

Complex questions are characterized by the fact that they typically ask about an answer that cannot be represented as a single semantic concept, but rather as a complex semantic structure. To process complex questions, QA systems must encode answer structures that involve: (a) semantic classes of entities and events that provide partial answers for the question; (b) linguistic patterns that facilitate the recognition of such entities or events in texts; and (c) a process of conceptual mapping between the complex answer type and the relevant entities and events. The set of semantic classes that characterize a complex answer type is called the **semantic signature** of the answer. To identify the semantic signature of a complex question we have considered (1) a *user-centric* procedure; and (2) a *knowledge-centric* procedure.

The User-Centric Procedure for Acquiring Semantic Signatures

The procedure is based on the assumption that there is some form of redundancy between several users that search information. As multiple users access a QA system, a *transition network* of questions and follow-up answers is generated. Figure 3(a) illustrates a *QA-transition network* that was produced by three different users trying to find information on the biography of Sherron Watkins. *QA-transition networks* have two kinds of nodes: questions and answers. To a given question node Q^i there may correspond several answer nodes $A_{i,a}^i, A_{i,b}^i, A_{i,c}^i, \dots$ that directly respond to the question Q^i . *QA-transition networks* also have two kinds of links: *direct links*, between a question and one or several answers that respond to it; and *indirect links*, between a question Q^i and an answer $A_{j,b}^i$ that is a direct answer to question Q^j , but it also contains information that answers question Q^i . For example in the *QA-transition network* illustrated in Figure 3(a), the direct link between question Q6 and the answer A6 is re-enforced by the indirect link between Q6 and A7, which, while providing an answer to Q7, also contains information regarding the age of Sherron Watkins, thus indirectly answering Q6.

We have considered and implemented three forms of recognizing indirect answers:

(a) Whenever an answer includes a concept that belongs to the same semantic class as the EAT of another question from the *QA transition network*. An example is the indirect link between Q6 and A7.

(b) Whenever the semantic properties of an answer qualify it as a member of the semantic class of the EAT of another question. For example, A2 has an indirect link to Q8 since husbands are members of families.

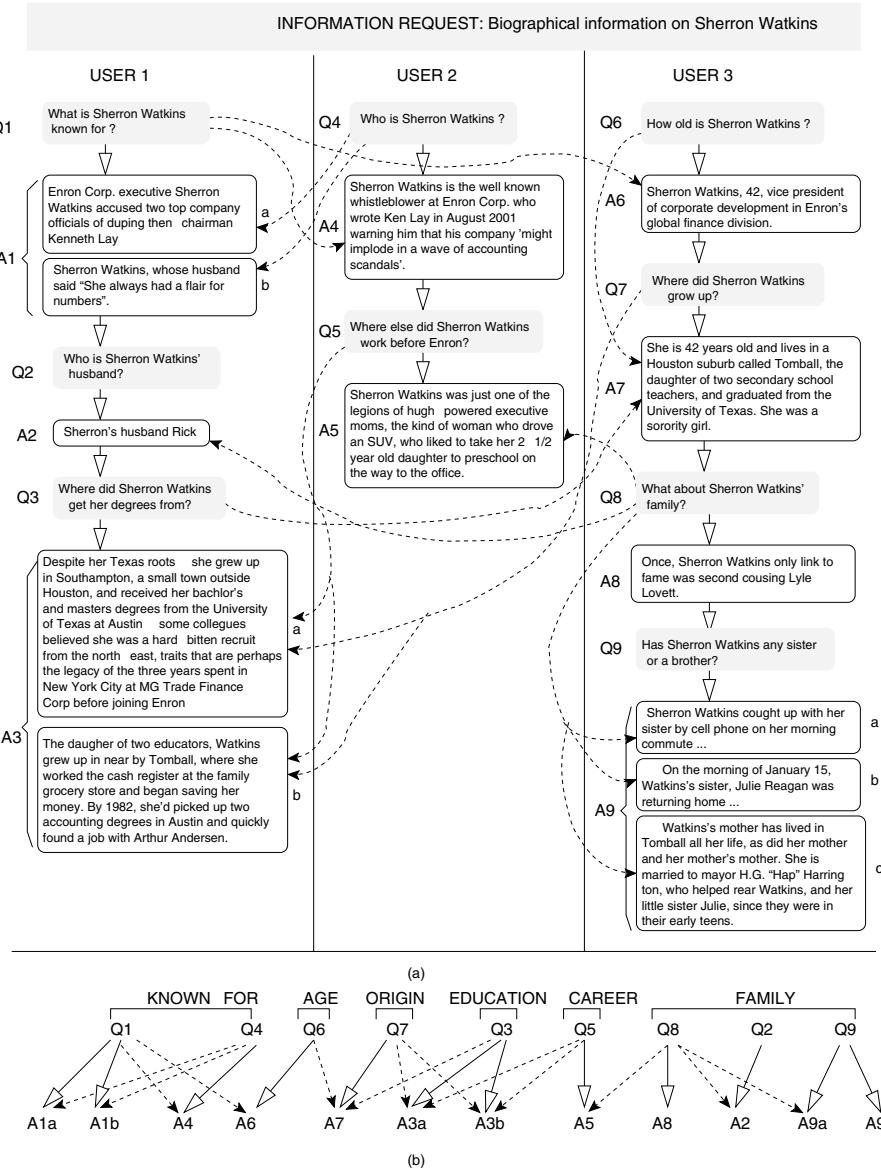


Figure 3. User-Centric Processing of Complex Questions.

This form of indirect link takes advantage of the different forms of meronyms, some available from WordNet, others discovered automatically with methods similar to those reported in (Girju *et al.*, 2003).

(c) Indirect links are also established based on the semantic similarity between the direct answer of a question and answers returned to different questions posed by other users when searching information on the same subject. To compute the semantic similarity, we have experimented with seven different metrics. The first four metrics were introduced by Lytinen and Tomuro (2002), while the final three metrics were introduced in (Harabagiu *et al.*, 2005).

Similarity Metric 1 is based on two processing steps:

(a) the content words of the answers are weighted using the *tf-idf* measure used in Information Retrieval $w_i = w(t_i) = (1 + \log(t_f))(log N)/df_i$, where N is the number of answers in the *QA-transition network*, df_i is the number of answers containing t_i and t_f is the number of times t_i appears in the direct answer. This allows the direct answer and any other answer to be transformed into two vectors, $v_d = \langle w_{d_1}, w_{d_2}, \dots, w_{d_n} \rangle$ and $v_o = \langle w_{o_1}, w_{o_2}, \dots, w_{o_m} \rangle$;

(b) the term vector similarity is used to compute the similarity between the two answers: $\cos(v_d, v_o) = (\sum_i w_{d_i} w_{o_i}) / \left(\left(\sum_i w_{d_i}^2 \right)^{\frac{1}{2}} \times \left(\sum_i w_{o_i}^2 \right)^{\frac{1}{2}} \right)$

Similarity Metric 2 is based on the percent of terms that appear in the direct answer and can also be found in other answers. It is obtained by finding the intersection of the terms in the term vectors of the two answers.

Similarity Metric 3 is based on semantic information available from WordNet. It involves:

(a) Finding the minimum path between WordNet concepts. Given two terms t_1 and t_2 , each with n and m WordNet senses $S_1 = \{s_1, \dots, s_n\}$ and $S_2 = \{r_1, \dots, r_m\}$. The semantic distance between the terms $\delta(t_1, t_2)$ is defined by the minimum of all the possible pair-wise semantic distances between S_1 and S_2 : $\delta(t_1, t_2) = \min_{s_i \in S_1, r_j \in S_2} D(s_i, r_j)$ where $D(s_i, r_j)$ is the path length between s_i and r_j

(b) The semantic similarity between the direct answer $T_d = \langle d_1, d_2, \dots, d_n \rangle$ and any other answer $T_o = \langle o_1, o_2, \dots, o_m \rangle$ to be defined as

$$\text{sim}(T_u, T_q) = \frac{I(T_u, T_q) + I(T_q, T_u)}{|T_u| + |T_q|}$$

$$\text{where } I(T_x, T_y) = \sum_{x \in T_x} \frac{1}{1 + \min_{y \in T_y} \delta(x, y)}$$

Similarity Metric 4 is based on the question type similarity. Instead of using the question class, determined by its stem, whenever we could recognize the answer type expected by the question, we used it for matching. As back-off only, we used a question type similarity based on a matrix akin to the one reported in (Lytinen and Tomuro, 2002).

Similarity Metric 5 is based on answer concepts rather than answer terms. In order to translate answer terms into concepts, we replaced (a) *exact direct answer* with expected answer types (taken from the answer type hierarchy) and (b) *named entities* with their corresponding classes. Remaining nouns and verbs were also replaced with their WordNet semantic classes, as well. Each concept was then associated with a weight: concepts derived from named entities classes were weighted heavier than concepts from answer types, which were in turn weighted heavier than concepts taken from WordNet classes. Similarity was then computed across “matching” concepts.⁷ The resultant similarity score was based on three variables: x = sum of the weights of all concepts matched between a direct answer (A_d) and another answer from the *QA-transition network* (A_o); y = sum of the weights of all unmatched concepts in (A_d); z = sum of the weights of all unmatched concepts in (A_o). The similarity between A_d and A_o was calculated as $x - (p_d \times y) - (p_o \times z)$, where p_d and p_o were used as coefficients to penalize the contribution of unmatched concepts in A_d and A_o respectively.⁸

Similarity Metric 6 is based on the fact that the questions and answers are clustered based on their mapping to a vector of important concepts in the *QA-transition network*. The clustering was done using the K-Nearest Neighbor (KNN) method (Dudani, 1976). Instead of measuring the similarity between the direct answer and any other answer, similarities are computed only between the direct answer and the centroid of each cluster.

Similarity Metric 7 was derived from the results of Similarity Metrics 5 and 6 above. In this case, if another answer (A_o) that was deemed to be most similar to a direct answer (A_d) under Similarity Metric 5 is contained in the cluster of answers deemed to be most similar to A_d under Similarity Metric 6, then A_o receives a *cluster adjustment score* in order to boost its ranking within its cluster. We calculate the cluster adjustment score as $score_{adj}(Q_b) = (sim_5 * (1-C_f)) + (sim_6 * C_f)$, where C_f represents the difference in rank between the centroid of the cluster and the previous rank of the answer A_o .

Figure 3(b) illustrates the result of processing the *QA-transition network* represented in Figure 3(a). It shows the direct and indirect relations between the *semantic answer types* of the questions that were asked and the answers that were returned. Typically, the semantic types of the expected answers are derived from off-line hierarchies. They play an important role in searching for the correct answer, as was reported in (Harabagiu *et al.*, 2000). Sometimes multiple questions share the

⁷ In the case of ambiguous nouns and verbs associated with multiple WordNet classes, all possible classes for a term were considered in matching.

⁸ We set $p_d = 0.4$ and $p_o = 0.1$ in our experiments.

same *semantic answer type*, e.g., Q², Q⁸ and Q⁹ ask about Sherron Watkins's family. Figure 3(b) shows that questions are answered both by direct and indirect answers. The *semantic answer types* of these questions represent the semantic signatures of the complex answer being searched. In the case of BIOGRAPHICAL INFORMATION or BIOGRAPHIES, connections to AGE, ORIGIN, KNOWN-FOR, EDUCATION and FAMILY are uncovered in this way. To be able to identify the semantic signature of a complex question we need to organize concepts into semantic networks. A baseline semantic network can be easily generated by accessing parts of the WordNet database.

The expected answer types of the questions, along with the patterns that enable the selection of the answers, represent the semantic signature of a complex question. This method for acquiring semantic signatures of complex questions relies on a model of answer similarity between responses to multiple users searching information on the same subject. In contrast, the acquisition method that is knowledge-centric relies on a model of knowledge refinement for selecting the semantic signature of a complex question.

Knowledge-Centric Method for Acquiring Semantic Signatures

Complex questions express an information need through lexemes. Decoding the information need is facilitated through information available in lexico-semantic resources such as WordNet. WordNet (Fellbaum, 1998) encodes a vast majority of English nouns, verbs, adjectives and adverbs, grouping synonym words in synsets. Sometimes synsets contain also multi-word expressions. For example, the synset for the noun *biography* contains also {*life*, *life story*, *life history*}. Moreover, synsets are associated with definitions, known as glosses. The gloss for *biography* is “*an account of the series of events making up a person’s life*”. The extended WordNet, available from <http://xwn.hlt.utdallas.edu>, provides the semantic disambiguations of the glosses, e.g., assigning the noun *account* from the gloss of *biography* the sense given by the synset {*history*, *account*, *chronicle*, *story*}. The presence of certain words in the gloss indicates the complexity of the concept, entailing the need for a semantic signature. In the case of the gloss for *biography*, the complexity is indicated by the words [*account*, *series*, *events*]. By scanning the entire extended WordNet, we have been able to assemble the set of synonyms that constitute *complexity hooks*.

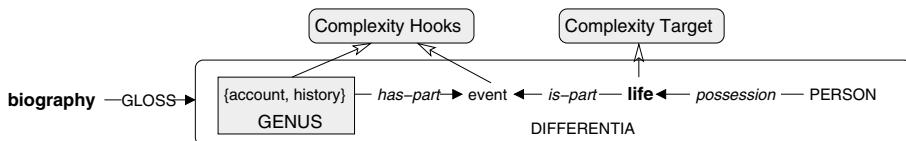


Figure 4. Complexity Hooks and Targets.

Extended WordNet also encodes relations between the disambiguated synsets of the glosses. For the gloss of the word *biography*, the relations are illustrated in Figure 4. The gloss relations are important since they help the identification of concept targets. The concept targets are used in mining the semantic signature of the complex answer from (1) WordNet (the knowledge base); (2) relevant documents

(the Web); and (3) the patterns used for extracting concepts and relations that define the complex answer type.

Given a complex answer type S , we need to identify a set of target concepts T which help us identify additional semantic information about S . In the case of complex questions, S is different from the question focus F , which identifies the concept (entity or event) about which the question is asking. For illustration, we consider the question $Q5$: “Biographical information needed on Elizardo Sanchez, Cuban dissident.” In the case of $Q5$, $S = \{\text{biographical information}, \text{biography}\}$ and $F = \text{Elizardo Sanchez}$, with the attributes $F_A = \{\text{Cuban}, \text{dissident}\}$. The semantic class of F is PERSON, which is a semantic class directly linked to the concept *life* in the gloss of *biography*. Furthermore, the relations between the genus of *biography* and *life* indicate compatible combinations that enable the selection of *life* as a target concept T for S .

The semantic signature that is acquired for S with the knowledge-centric method has three components:

- A. Acquisition of semantic information pertaining to the general aspect of the expected answer type. For example, in the case of $Q5$, it refers to information pertaining to any biography.
- B. Acquisition of semantic information that is particular only to the entities that share the same attributes as the focus of the question, i.e. biographies of dissidents, biographies of Cubans.
- C. Acquisition of semantic information that represents nuggets of knowledge about the specific focus of the question.

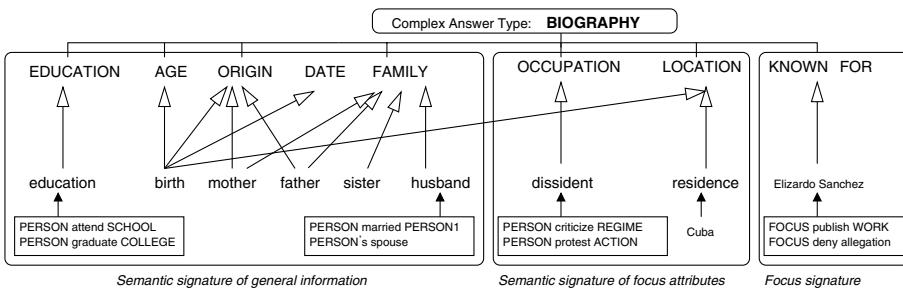


Figure 5. Semantic Signature of Complex Answer Generated by the Knowledge-Centric Approach.

Each of the acquisition components operates differently. General semantic information is acquired in the form of (i) semantic concepts, (ii) patterns for recognizing them in texts, (iii) mappings to classes of answers from the EAT hierarchy that is used when processing factoid questions. For example, Figure illustrates the semantic signature generated for $Q5$. Concepts from the EAT hierarchy are capitalized in the Figure. Also, patterns are illustrated only for concepts *education*, *husband*, *dissident* and the question focus. However, each concept is associated with a set of patterns in the semantic signature. Semantic information particular only to entities with the same attributes as the focus is obtained by (i) restricting some of the concepts to agree with the attributes and

(ii) by modifying the initial acquisition method in order to incorporate the attributes. Figure 5 illustrates the result of the signature when the attributes are considered. For example, the location constraint restricts the extractions for the concepts *residence* or *birth* to be included in *Cuba*. Finally, nuggets characteristic only to the focus are added by searching for patterns in which the focus is an argument of a predicate. The result is a semantic signature for the complex answer type $\mathcal{S} = \{\text{[Answer-Type List], [General Concept-List], [Concept Mappings], [Concept Extraction Patterns]}\}$. The procedure that generates the semantic signature has the following steps:

Step 1: Identify general concepts

Given the set of target concepts \mathbf{T} , we collect G_T , a set of WordNet concepts that have in their gloss at least one element from \mathbf{T} . For each concept $x \in G_T$, we assemble a corpus D_x^F as a result from retrieving documents relevant to meta-queries of the form $[S_F \text{ AND } (\text{GEN-OR}(x, \text{Deriv}(x)))]$, where S_F is the semantic class of the focus of the complex question, $\text{Deriv}(x)$ are all the morphological derivations of all the elements of the synset x and GEN-OR is a generalized OR operator that can take more than two arguments by generating a regular OR between each pair of elements from the set of its arguments. For example, when $\mathbf{T} = \{\text{life}\}$, $x = \text{birth}$ belongs to G_T and $\text{Deriv}(x) = \{\text{born, give birth, deliver}\}$. Finally, S_F is expanded into $N=100$ most frequent concepts that are closest to x or $\text{Deriv}(x)$ in the corpus C from which we want to extract the answer for the complex question.

Each document from D_x^F is processed with a parser that generates all the predicate-argument structures. The parser, which was described in (Surdeanu *et al.*, 2003) was trained on the PropBank annotations. PropBank is a one million word corpus annotated with predicate-argument structures on top of the Penn Treebank2 Wall Street Journal texts.

To determine which elements from G_T are selected for the General Concept-List in the semantic signature, we assign to each predicate-argument structure $\text{PAS}(x, S_F)$ from D_x^F which contains (i) x or an element from $\text{Deriv}(x)$, and (ii) any concept of semantic type S_F an association weight w_x . To compute the association weights we use the likelihood ratio to test the hypothesis that x is an important concept for the semantic signature of \mathcal{S} . The likelihood ratio is computed by assembling a corpus D^F which is relevant to the focus, but not to x . Assuming the following two hypotheses:

- Hypothesis 1 (H_1) The semantic signature of \mathcal{S} is independent of x .
- Hypothesis 2 (H_2) The presence of x in the signature of \mathcal{S} will indicate high relevance of the partial answers extracted based on x .

we compute the likelihood of each of the hypotheses $L(H_1)$ and $L(H_2)$ in order to evaluate $-2\log\lambda = -2\log(L(H_1) / L(H_2))$. The association weight of each concept x is given by the corresponding value of $-2\log\lambda$, as it was previously shown in (Manning and Schütze, 1999). To compute the likelihood of each hypothesis, we assume the binomial distribution $b(k; n, a) = \binom{n}{k} a^k (1-a)^{(n-k)}$ and:

- f_{11} , the frequency with which x or any $z \in \text{Deriv}(x)$ and any concept of semantic type S_F belong to the same predicate-argument structure in D_x^F ,

- f_{12} , the frequency with which x or any $z \in \text{Deriv}(x)$ and any concept of semantic type S_F belong to the same predicate-argument structure in D^F ,
- f_{21} , the frequency with which any concept $z \neq x$ and any concept of semantic type S_F belong to the same predicate-argument structure in D_x^F ,
- f_{22} , the frequency with which any concept $z \neq x$ and any concept of semantic type S_F belong to the same predicate-argument structure in D^F .

Then $L(H_1) = b(f_{11}; f_{11}+f_{12}, p) b(f_{21}; f_{21}+f_{22}, p)$

$$L(H_2) = b(f_{11}; f_{11}+f_{12}, p_1) b(f_{21}; f_{21}+f_{22}, p_2)$$

where:

- o $p = \text{NR-PAS}(S_F) / \text{NR-PAS}$, with $\text{NR-PAS}(S_F)$ counting all the predicate-argument structures involving any concept of class S_F in D_x and NR-PAS counts the total number of predicate-argument structures in D_x .
- o $p_1 = \text{NR-PAS}(S_F, x) / \text{NR-PAS}(S_F)$, with $\text{NR-PAS}(S_F, x)$ counting all the predicate-argument structures involving any concept of class S_F and x or an element in D_x^F of $\text{Deriv}(x)$.
- o $p_2 = \text{NR-PAS}(S_F, z) / \text{NR-PAS}(S_F)$, with $\text{NR-PAS}(S_F, z)$ counting the number of times any concept z in D_x^F different from x and not belonging to $\text{Deriv}(x)$ belongs to a predicate-argument structure that involves any concept y with semantic class S_F .

Concept seeds are selected based on a confidence level available from the χ^2 distribution table which determines the cut-off associated weight and the number of concepts to be included in the semantic signature. The semantic concepts that were initially selected for the BIOGRAPHY complex answer type were *birth* and *death*.

Step 2: Knowledge expansion of general concepts.

Each concept $x \in G_T$ which was selected by the likelihood ratio method is included in the target set T and is thus expanded into a new set G_x . Step 1 and Step 2 are repeated until one of the following conditions are met: (i) no new general concepts are added, or (ii) the loop of Step 1 – Step 2 continue until a pre-set maximum number has been reached. Typically we allowed only 20 such loops. Those loops acquired more general concepts, e.g., *mother*, *father*, *husband*.

Step 3: Discovery of semantic patterns.

The general concepts have been selected because they participate in predicate-argument structures that are representative for the complex answer. Since PropBank is known to encode shallow semantics (cf. Gildea and Palmer, 2002) it makes sense to try to discover semantic patterns induced from predicate-argument structures. For this purpose, we generate a semantic normalization of the arguments of each predicate. This is possible because many arguments have named entities as their syntactic heads. Classes of names are mapped into the tops of the EAT hierarchy for processing factoid questions, as was reported in (Harabagiu *et al.*, 2001). For all the other arguments we have used a combination of the EAT hierarchy with an ontology of 150 concepts that subsumes 22,000 English content words.

For each general concept, the semantic patterns have the format $[x \leftarrow \text{Predicate} \rightarrow \text{Argument}]$. To select the *Predicate* and the *Argument* for each pattern we have built an association matrix M for each predicate-argument structure that involved concept

$x.M$ has $k=150$ rows, one for each semantic class of an argument, and n columns, one for each base form of a verbal predicate. The \langle *Predicate-Argument* \rangle pairs were ordered according to the value of m_{ij} , representing the number of times *Predicate_j* had an argument of semantic class i . Whenever there was a tie, the predicates with the largest value for $\sum_j m_{ij}$ was preferred. If a tie for $\sum_j m_{ij}$ was noticed, the \langle *Predicate-Argument* \rangle that was selected had a higher argument number on the PropBank annotation⁹.

Step 4: Conceptual Mappings.

The semantic normalizations produced at Step 3 need to be generalized with semantic mappings between any general concept and selected tops from the EAT hierarchy. For this purpose, the EAT hierarchies, produced with the method reported in (Pasca and Harabagiu, 2001), are used for assigning a general semantic class to each general concept from the semantic signature. The EAT hierarchies consist of a selection of WordNet sub-hierarchies. For example, the general concept *birth* was mapped in AGE and DATE which are classes discovered by Named Entity Recognizers, and in ORIGIN, which is a top of the EAT hierarchy.

Step 5: Acquisition of Semantic Information Typical for the Focus Attributes.

To acquire semantic information typical for each of the attributes associated with the question focus, we use WordNet. For example, the list of hyponyms for *dissident* contains *Aleksandr Solzhenitsyn* as a representative. We enhance the list of representatives by retrieving from the text collection examples obtained when matching several forms of copulative knowledge (e.g., arguments of verbs *become*, *be*, or phrasal verbs *known for*, *recognized as*). The discovery of attribute-specific patterns follows the method reported in (Filatova and Prager, 2005). For each document that is retrieved when using a query that employs one representative of the attribute, a shallow semantic parser produces all predicate-argument structures. These structures are post-processed for semantic normalization in the same way as it was done in Step 3. To discover the most representative predicates for an attribute, we generate a bipartite graph $G = \{N, V, E\}$, where N are the nodes corresponding to semantic normalizations of entities, V are verb nodes denoting the predicates, and E are the arcs connecting the entities to the predicates. As noted in (Filatova and Prager, 2005), predicates typical for an attribute act like “hubs” whereas the entities act similarly to “authorities”. Based on this observation, algorithms developed for ranking web pages (e.g., Kleinberg, 1998) can be used to rank the most relevant predicates for a given attribute.

As in (Filatova and Prager, 2005), we define a stochastic transition matrix from entities to verbs $M_{N \rightarrow V}$ that has $k = 150$ rows and n columns, one for each verb from a predicate-argument structure. The elements of $M_{N \rightarrow V}$ are defined by:

$$m_{i,j} = (1 - c) \frac{f(n_i \rightarrow v_j)}{\sum_{v \in V} f(n_i \rightarrow v)} + c$$

⁹ PropBank annotates five arguments Arg0 – Arg5.

where f is the sum of the counts of arguments from an entity n_i to a predicate v_j , and c is a smoothing factor ($c = 0.01$ like in Filatova and Prager, 2005). In the same way, a stochastic transition matrix from predicates to arguments can be defined as $M_{V \rightarrow N}$. Using $M_{N \rightarrow V}$ and $M_{V \rightarrow N}$ we can define the transition matrix

$$M_{V \rightarrow V} = M_{V \rightarrow N} \times M_{N \rightarrow V}$$

The matrix $M_{V \rightarrow V}$ can be used for scoring predicates according to how important they are for the focus attribute. As in Step 3, the patterns have the format $[S_F \leftarrow \text{Predicate} \rightarrow \text{Argument}]$. By selecting only the most discriminating predicates indicated by the transition matrix $M_{V \rightarrow V}$, we can also select their corresponding Argument using the same method as in Step 3. For the attribute *dissident*, patterns that are based on the most frequent predicates are selected, e.g., *protest*, *criticize*.

Step 6: Discovery of Semantic Nuggets for the Question Focus.

All the procedure from Step 5 is repeated, but the search is performed by replacing S_F with the question focus itself. This time, not only predicate-argument structures are used, but also appositives and patterns used for answering definition questions. Such patterns have been reported in (Xu *et al.*, 2003; Katz *et al.*, 2004).

2.2 Intentional Structure

The basic elements of a compositional theory of discourse are, according to Grosz and Sidner (1986): (1) the structure of the sequence of utterances, also known as the linguistic structure; (2) the state of the focus of attention; and (3) the intentional structure of the text. When searching for the answer to a question, the linguistic structure represents an important factor in deciding the search space. Just as words in a question form the information request, the discourse paragraphs (or segments) of documents represent the natural elements for searching candidate answers. To retrieve the answer to a question from a text collection, a QA system needs to (1) find congruent passages of text where candidate answers may be found; (2) evaluate the relevance of those passages to the question; and (3) unify the intentions of the question with the intentions recognized in the answer passage. Throughout the search for the answer, the focus of the question is used for validating candidate answers and the entities from the focus of attention of the answer. In current QA systems answer selection is performed by considering different combinations of question classes, question keywords and answer types. If the question stem is *who*, the answer type is very likely to be PERSON. The keywords selected from the questions as well as the answer type guide the search for the answer. The answer is selected when it best matches the answer type against candidate text snippets. This simple technique is efficient because it operates under the simplified assumption for factual questions. This assumption originated in TREC evaluations where questions ask only about trivia-like facts. For factual questions the intentions are always the same, namely to find the factual information answering the question.

Complex questions sometimes have a linguistic structure of their own. Furthermore, they may have a focus space that needs to be processed, which in turn

determines the discourse segment purpose (DSP) of each of the question segments. Complex questions require domain expertise and express a wide variety of intentions. Figure 6 illustrates an example of a complex question which consists of (1) a background or the context of the question; and (2) an inquiry. The background has the role to introduce (a) the question topic (e.g., *Russia's biological weapons program*); (b) the question focus (e.g., *the status of Russia's biological weapons program*); and (c) a temporal grounding of the question focus (e.g., *since the collapse of the Soviet Union in the 1990s*). The inquiry is composed of (a) a request of evidence (e.g., *of the difficulty of finding information about Russia's biological weapons program*); (b) the question beliefs (e.g., *no hard information was found yet*); and (c) an evaluation of the implications (e.g., *of the international arms control*).

Since the collapse of the Soviet Union in the 1990s, there has been much speculation as to the status of Russia's biological weapons programs.

Question Background

Why has it proven so difficult to discover hard information about the current nature of Russia's biological weapons program and what are the implications of those difficulties for the international non-conventional arms control regime ?

Question Inquiry

Figure 6. Example of Complex Question.

In the case of factual questions, the expected answer type is identical to the question topic and to the question focus. For complex questions the *topic*, the *focus* and the *expected answer type* are different. The question topic defines the semantic domain or the context in which information is sought. The question focus identifies a specific aspect of the question topic. For the question illustrated in Figure 6, the question topic is biological weapons, whereas the question focus is the status of biological weapons. To identify the question topic and focus we rely on semantic information available from FrameNet (<http://www.icsi.berkeley.edu/~framenet>). The FrameNet project produced a lexico-semantic resource encoding a set of **frames**, which represent schematic situations characterized by a set of **target words**; or lexicalized predicates, which can be verbs, nouns or adjectives. In each frame, various participants and conceptual roles are related by case-roles or theta-roles which are called **frame elements** or FEs. FEs are local to each frame, some are quite general, while others are specific to a small family of lexical items and their corresponding frames.

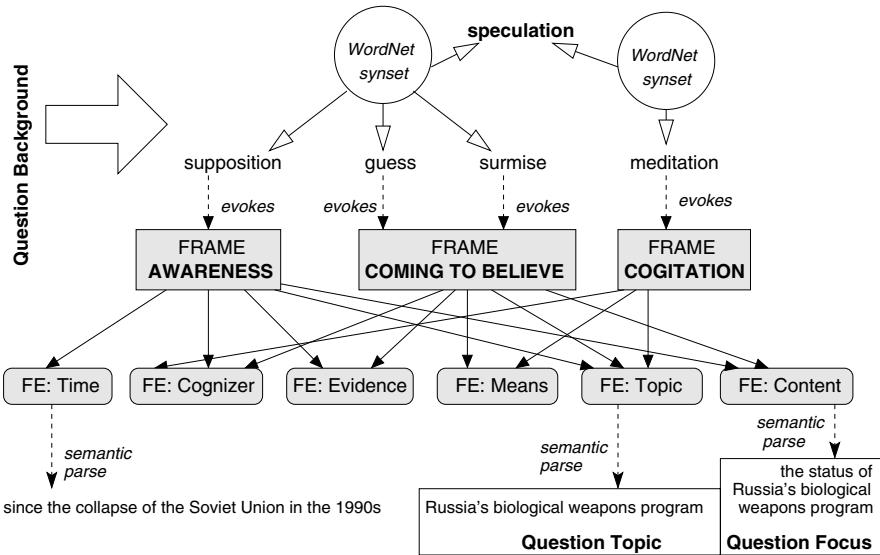


Figure 7. Semantic Characterization of the Background of the Complex Question Illustrated in Figure 6.

The semantic information encoded in frames is important because it enables the discovery of the question topic, focus and complex answer type. Certain target words from the complex question evoke frames that help us find the topic and the focus. The words from the question may belong to multiple *WordNet synsets*, as they may be semantically ambiguous. In turn, each word from a synset may evoke a different *FrameNet frame*. When the evoked frames have at least three common FEs, we believe that they illustrate some representative semantic schemes that are relevant for the semantic interpretation of the question. For example, Figure 7 illustrates the frames that are evoked by the words that are synonymous with the noun *speculation*, which is present in the background of the question illustrated in Figure 6. Semantic parsers trained on *FrameNet*, similar to the one reported in (Bejan *et al.*, 2004), recognize the FEs in the question background. The question topic corresponds to the Topic FE, whereas the question focus corresponds to the Content FE. Therefore, the question topic is recognized due to the Topic FE, whereas the question focus is recognized due to the Content FE.

It is to be noted that not all complex questions are parsed into frames that have Topic and Content FEs. To be able to identify the topic and focus, we have trained a classifier that considers other FEs as well. The FEs that are used for discovering the focus and topic were learned by training on a corpus of more than 10,000 complex questions.

The inquiry segment of the question illustrated in Figure 6 has also a linguistic structure. The structure is marked by: (i) the presence of the two question stems (e.g., *why* and *what*), (ii) a coordination between the two sub-questions (e.g., *and*), and (iii) anaphoric expressions in the second sub-question (e.g., *those difficulties*).

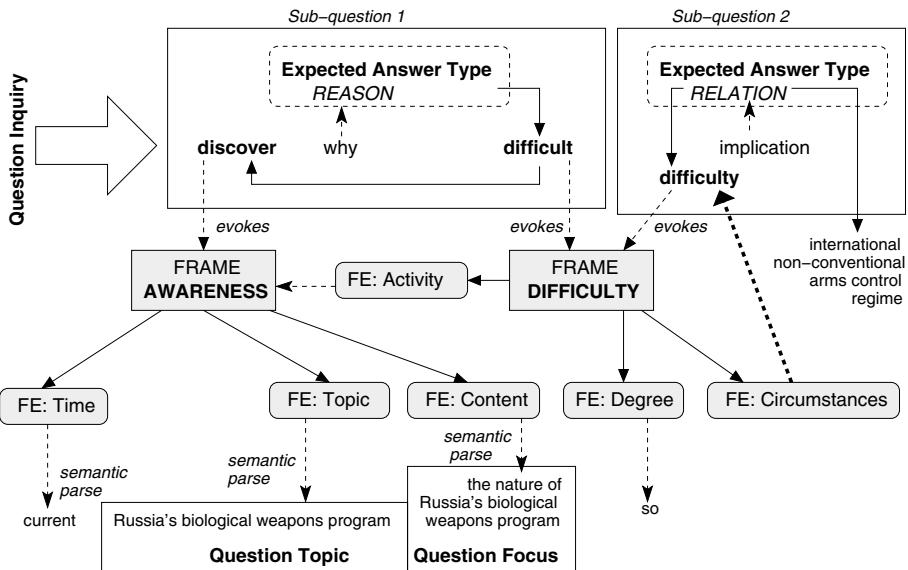


Figure 8. Semantic Characterization of the Inquiry of the Complex Question Illustrated in Figure 6.

As illustrated in Figure 8, the stem of the sub-question 1 of the inquiry determines the EAT, e.g., REASON, which has a relation to *difficult*. In turn, *difficult* is related to *discover*. The binary relations are produced as a result of the question processing first reported in (Harabagiu *et al.*, 2000). The noun *discover* evokes the AWARENESS frame, and the semantic parse determines the question topic which is identical to the topic discovered for the background. The question focus however has a different head than the focus discovered for the background. Both heads are hyponyms of the noun *attribute*, and thus are considered compatible and representing the same concept.

The adjective *difficult* evokes the DIFFICULTY frame. Since it modifies the verb *discover*, its FE Activity is assigned to the frame corresponding to the verb *discovered* instead of a text span identified by the semantic parser. Additionally, the FE Circumstances is used for resolving the anaphor from the sub-question 2. The anaphoric is *those difficulties*, it evokes the DIFFICULTY frame as well, and the semantic parser maps the anaphor to the FE Circumstances.

The segmentation of the question illustrated in Figure 6 is also spanned by **dominance** relations. Dominance and satisfaction-precedence relations have been defined in (Grosz and Sidner, 1986). Given two discourse segments *DS1* and *DS2*, an action that satisfies one intention from *DS1* may be intended to provide part of the satisfaction of an intention from *DS2*. In that case the purpose of *DS1* contributes to the purpose of *DS2*, and conversely, the purpose of *DS2* dominates the purpose of *DS1*.

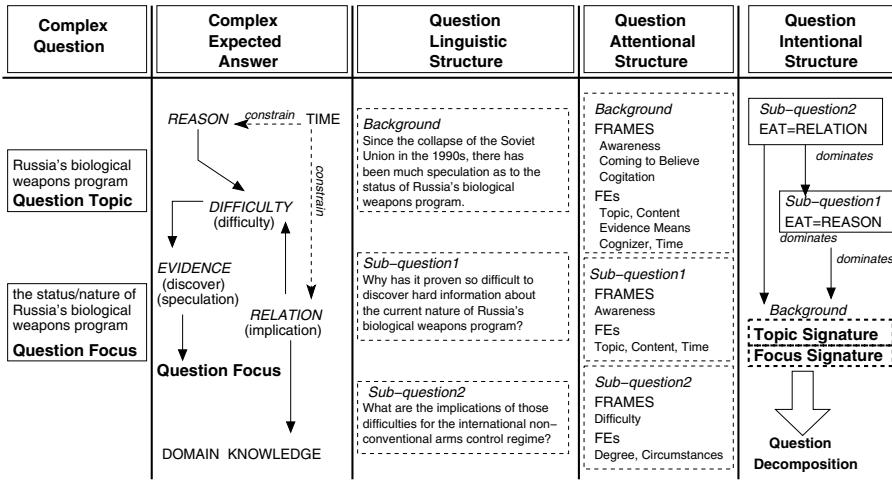


Figure 9. Processing Complex Questions.

For the question illustrated in Figure 6, the intentions of discovering information about the question focus, which are derived from the background, provide part of the satisfaction of the intention of finding the reason for the difficulties involved in the discovery. These difficulties may involve the evidence, the means of discovery, or the cognizer. In a way, the FEs of frames which are not mapped into text snippets can be used to derive possible intentions. Therefore sub-question 1 dominates the background. Similarly, sub-question 2 dominates the background, since the intention to find relations between the difficulties and another entity depends on the ability of first discovering the question focus, and then finding the difficulties. Sub-question 2 also dominates sub-question 1. This intentional structure is illustrated in Figure 9.

The intentional structure indicates the order in which the question intentions must be discovered. First, the intentions of the background need to be uncovered, followed by the intentions of the sub-question 1, and then those of sub-question 2. This structure motivated also the structure of the complex answer type, which is illustrated in Figure 9. The question decomposition corresponds to the signature of the question topic and question focus. As Figure 9 indicates, the intentional structure determines a question decomposition that aims at satisfying the intentions corresponding to each component of the linguistic structure of the question. The intentional structure consists of: (1) the dominance structure between the question segments; (2) the semantic signatures of the question topic and the question focus; and (3) questions associated with these signatures.

The background, which is dominated by the interrogation part of the question, is introducing the question topic and the question focus. The question topic determines the forms of domain knowledge that needs to be made available. For the question illustrated in Figure 9, the topic is a specialization of the more general topic $T_0 = \text{Biological Weapons Programs}$.

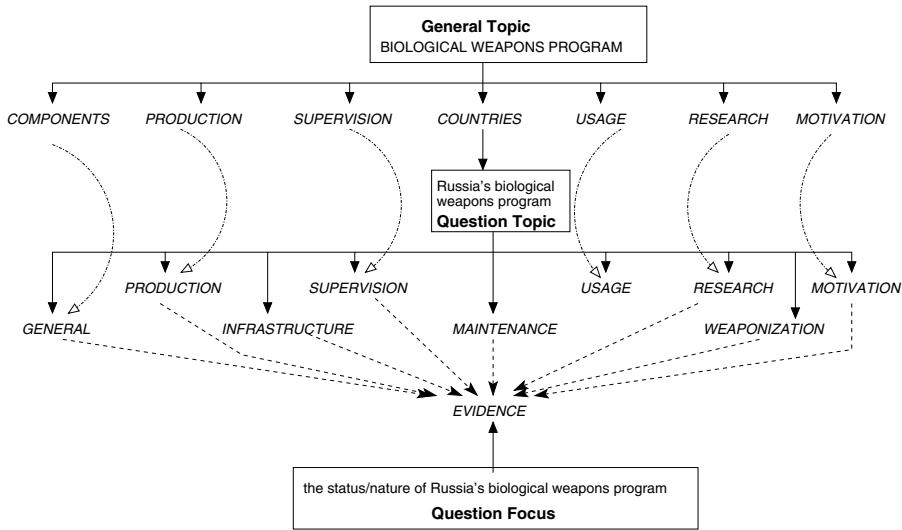


Figure 10. Semantic Connections between the Question Topic and the Question Focus.

Figure 10 illustrates the semantic signature associated with topics T_0 and T_1 . Topic semantic signatures have been initially introduced in (Lin and Hovy, 2000). Harabagiu (2004) reports on several incremental enhancements that generate semantic signatures similar to those illustrated in Figure 10. It is to be noted that since T_1 is specializing T_0 , all information pertaining to the same semantic class in T_0 is transferred to the corresponding class in T_1 . For example, general information regarding the production of bio-weapons is transferred into the information that relates only to Russia's production of bio-weapons. Details about the induction of the semantic signature of topics are given in Section 2.3. Figure 11 illustrates examples of questions that are generated for uncovering information related to the topic. The methodology for generating questions is detailed in Section 2.4. As Figure 11 shows, the information pertaining to the topic is semantically characterized not only by the topic signature, but also by the EAT of the question. It is to be noted that only those questions having EVIDENCE as a possible EAT are relevant also to the question focus.

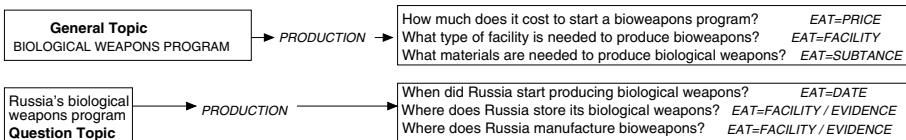


Figure 11. Examples of Decomposed Questions .

The question focus employs a semantic differentiator from the question topic in the case of the question illustrated in Figures 6, 7, 8 and 9 the differentiator is the semantic type EVIDENCE. Since this is a complex semantic type, procedures

described in Section 2.1 discover the semantic signatures for EVIDENCE. Figure 6 illustrates additional questions associated with the question focus.

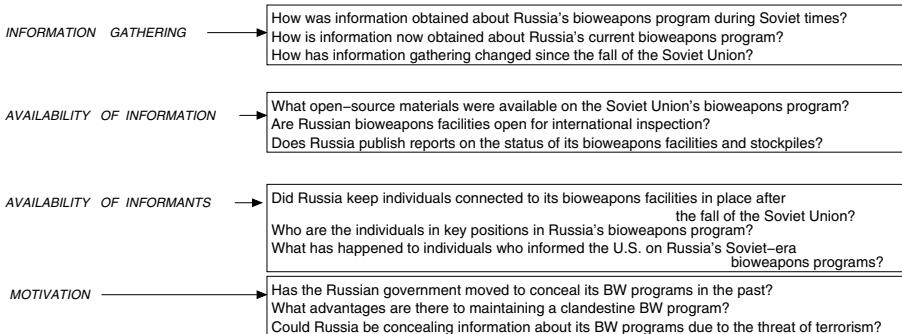


Figure 12. Additional Questions Associated with the Question Focus.

2.3 Question Interpretation with Domain Knowledge

The interpretation of a complex question relies often on domain knowledge related to the question topic. The notion of topic signatures was first introduced in (Lin and Hovy, 2000). For each subtopic in a scenario, given (a) documents relevant to the sub-topic and (b) documents not relevant to the subtopic, a statistical method based on the likelihood ratio is used to discover a weighted list of the most topic-specific concepts, known as the topic signature. Later work by Harabagiu (2004) demonstrated that topic signatures can be further enhanced by discovering the most relevant relations that exist between pairs of concepts. However, both of these types of topic representations are limited by the fact that they require the identification of topic-relevant documents prior to the discovery of the topic signatures. In our experiments, we were only presented with a set of documents relevant to a particular scenario; no further relevance information was provided for individual subject areas or sub-topics.

In order to solve the problem of finding relevant documents for each subtopic, we considered four different approaches:

Approach 1: All documents in the text collection were initially clustered using K-Nearest Neighbor (KNN) clustering (Dudani, 1976). Each cluster that contained at least one keyword that described the sub-topic was deemed relevant to the topic.

Approach 2: Since individual documents may contain discourse segments pertaining to different sub-topics, we first used TextTiling (Hearst, 1994) to automatically segment all of the documents in the text collection into individual text tiles. These individual discourse segments then served as input to the KNN clustering algorithm described in Approach 1.

Approach 3: In this approach, relevant documents were discovered simultaneously with the discovery of topic signatures. First, we associated a binary *seed relation* r_i for each sub-topic S_i . Seed relations were created both by hand and using the method presented in (Harabagiu, 2004). Since seed relations are by

definition relevant to a particular subtopic, they can be used to determine a binary partition of the document collection C into (1) a relevant set of documents R_i (that is, the documents relevant to relation r_i) and (2) a set of non-relevant documents $C - R_i$. Inspired by the method presented in (Yangarber *et al.*, 2000), a topic signature (as calculated by Harabagiu, 2004) is then produced for the set of documents in R_i . For each subtopic S_i defined as part of the dialogue scenario, documents relevant to a corresponding seed relation r_i are added to R iff the relation r_i meets the *density criterion* (as defined in Yangarber *et al.*, 2000). If D represents the set of documents where r_i is recognized, then the density criterion can be defined as: $|D \cap R| / |D \cap C| >> |R| / |C|$. Once D is added to R_i , then a new topic signature is calculated for R . Relations extracted from the new topic signature can then be used to determine a new document partition by re-iterating the discovery of the topic signature and of the documents relevant to each subtopic.

Approach 4: Approach 4 implements the technique described in Approach 3, but operates at the level of discourse segments (or text-tiles) rather than at the level of full documents. As with Approach 2, segments were produced using the TextTiling algorithm.

In modeling the domain knowledge, we considered three types of topic-relevant relations: (1) *structural relations*, which represent hypernymy or meronymy relations between topic-relevant concepts, (2) *definition relations*, which uncover the characteristic properties of a concept, and (3) *extraction relations*, which model the most relevant events or states associated with a sub-topic. Although structural relations and definition relations are discovered reliably using patterns available from our Q/A system (Harabagiu *et al.*, 2003), we found only extraction relations to be useful in determining the set of documents relevant to a subtopic. Structural relations were available from concept ontologies implemented in the Q/A system. The definition relations were identified by patterns used for processing definition questions.

Extraction relations are discovered by processing documents in order to identify three types of relations, including: (1) syntactic attachment relations (including subject-verb, object-verb, and verb-PP relations), (2) predicate-argument relations, and (3) salience-based relations that can be used to encode long-distance dependencies between topic-relevant concepts. Salience-based relations are discovered using a technique first reported in (Harabagiu, 2004), which approximates a Centering Theory-style approach (Kameyama, 1997) to the resolution of coreference.

Subtopic: Egypt's production of toxins and BW agents

Topic Signature:

produce – phosphorous trichloride (TOXIN)
house – ORGANIZATION
cultivate – non-pathogenic *Bacillus Subtilis* (TOXIN)
produce – mycotoxins (TOXIN)
acquire – FACILITY

Subtopic: Egypt's allies and partners

Topic Signature:

provide – COUNTRY
cultivate – COUNTRY
cooperate – COUNTRY
train – PERSON
supply – know-how

Figure 13. Example of Two Topic Signatures.

We made the extraction relations associated with each topic signature more general (a) by replacing words with their (morphological) root form (e.g., *wounded* with *wound*, *weapons* with *weapon*), (b) by replacing lexemes with their subsuming category from an ontology of 100,000 words (e.g., *truck* is replaced by VEHICLE, ARTEFACT, or OBJECT), and (c) by replacing each name with its name class (*Egypt* with COUNTRY). Figure 13 illustrates the topic signatures resulting for the scenario “Assessment of Egypt’s Biological Weapons”.

Once extraction relations were obtained for a particular set of documents, the resulting set of relations were ranked according to a method proposed in (Yangarber, 2003). Under this approach, the score associated with each relation is given by: $score(r) = (Sup(r)/|D|) * \log_2 Sup(r)$, where $|D|$ represents the cardinality of the documents where the relation is identified, and $Sup(r)$ represents support associated with the relation r . $Sup(r)$ is defined as the sum of the relevance of each document in D : $Sup(r) = \sum_{d \in D} Rel(d)$. The relevance of a document that contains a topic-significant relation can be defined as: $Rel(d) = 1 - \prod_{r \in TS} (1 - Prec(r))$, where TS represents the topic signature of the subtopic¹⁰. The accuracy of the relation, then, is given by: $Prec(r) = (1/|D|) (\sum_{d \in D} Rel^{S_i}(d) - \sum_{j \neq i} Rel^{S_j}(d))$. Here, $Rel^{S_i}(d)$ measures the relevance of a subtopic S_i to a particular document d , while $Rel^{S_j}(d)$ measures the relevance of d to another subtopic, S_j .

We use a different learner for each subtopic in order to train simultaneously on each iteration. (The calculation of topic signatures continues to iterate until there are no more relations that can be added to the overall topic signature.) When the precision of a relation to a subtopic S_i is computed, it takes into account the *negative* evidence of its relevance to any other subtopic $S_j \neq S_i$. If $Prec(r) \leq 0$, the relation is not included in the topic signature, where relations are ranked by the score $Score(r) = Prec(r) * \log(Sup(r))$.

Representing topics in terms of relevant concepts and relations is important for the processing of questions asked within the context of a given topic.

There are NLP applications for which topic representation as a collection of relevant relations is not sufficient. This is because topic relations capture only the most characteristic and repetitive information about a topic. We argue that additional information is of interest, especially for such applications as multi-document summarization. When processing complex questions, we would like to have access to the semantic signature of the topic. For this purpose, we considered a third representation of topic signatures. TS_3 is defined as $\{topic, \langle (Th_1, r_1), \dots, (Th_s, r_s) \rangle\}$, where Th_i represents one of the themes associated with the topic and r_i is its rank.

¹⁰ Initially, TS contains only the seed relation. Additional relations can be added with each iteration.

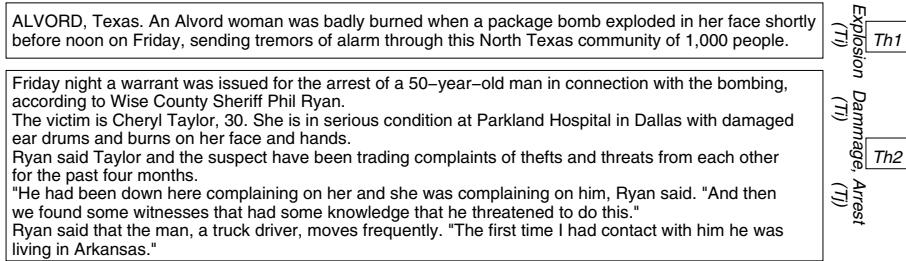


Figure 14. Topic Themes.

Figure 14 illustrates the segmentation of a document discussing topic $T_i = \text{BOMBINGS}$ into topic themes. The segmentation was produced by the TextTiling algorithm (Hearst, 1997) when term similarity was adjusted to consider identity for topic words belonging to the same set of trigger words or compatible concepts. The themes of topics are labelled automatically. There are four different cases for label assignment of a theme for topic T :

Case 1: A single topic relation pertaining to the topic T is recognized throughout the theme segment. One of the trigger words that facilitates the recognition of the relation becomes the theme label. For the example from Figure 14, theme Th_1 is labeled as *Explosion* because the relation [*explode-bomb*] was identified and the trigger word *explode* was more frequently responsible for the recognition of this relation in the topic collection than *detonate*.

Case 2: Several topic relations are recognized in the same theme. The label is determined by the topic relation which is ranked highest in the topic signature $TS_2(T)$.

Case 3: Topic relations pertaining to other topics are recognized as well. The theme receives additional labels, determined by the new topics recognized. For the example illustrated in Figure 14, the second theme has two labels, one pertaining to topic $T_i = \text{BOMBINGS}$, the other pertaining to topic $T_j = \text{COURT_CASES}$.

Case 4: The theme contains relevant concepts for the topic T but no topic relation of T is recognized. In this case the UNKNOWN label is assigned.

The ranking of the topic themes for topic T considers that (1) relations pertaining to any other topic $T' \neq T$ contribute less to the ranking than relations from the signature of T ; and (2) whenever no relation from T is identified in a theme, but only relevant concepts exist, the ranking of that theme is lower than that of themes containing such relations. Consequently we computed the rank of a theme as $R(th_i) = \sum w(r_j^T) + k_1 \times \sum w(r_k^{T'})$ whenever relations r_j^T pertaining to the topic T are identified in th_i and relations $r_k^{T'}$ of any other topic are also identified; k_1 represent the ratio between the largest weight of any relation $r_k^{T'}$ and the smallest weight of any relation r_j^T .

2.4 Generating Questions for a Given Topic

As we have seen in Section 2.2, it is very important to associate each sub-topic with a set of relevant questions, and eventually their answers. In order to generate question-answer pairs for each subtopic, we first identified relevant text passages in the document collection to serve as “answers” and then generated individual questions that could be answered by each answer passage.

Answer Identification: We defined an *answer passage* as a contiguous sequence of sentences with a positive *answer rank* and a *passage price* of ≤ 4 . To select answer passages for each sub-topic S_i , we calculate an *answer rank*, $rank(a) = \sum_{r_i} score(r_i)$, that sums across the scores of each relation from the topic signature that is identified in the same text window. Initially, the text window is set to one sentence. (If the sentence is part of a quote, however, the text window is immediately expanded to encompass the entire sentence that contains the quote.) Each passage with $rank(a) > 0$ is then considered to be a *candidate answer passage*. The text window of each candidate answer passage is then expanded to include the following sentence. If the answer rank does not increase with the addition of the succeeding sentence, then the *price* (p) of the candidate answer passage is incremented by 1, otherwise it is decremented by 1. The text window of each candidate answer passage continues to expand until $p = 4$. Before the ranked list of candidate answers can be considered by the Question Generation module, answer passages with a positive price p are stripped of the last p sentences.

Question Generation: In order to automatically generate questions from answer passages, we considered the following two problems:

Problem 1: Every word in an answer passage can refer to an entity, a relation, or an event. In order for question generation be successful, we must determine whether a particular reference is “interesting” enough to the scenario such that it deserves to be mentioned in a topic-relevant question. For example, Figure 15 illustrates an answer that includes two predicates and four entities. In this case, four types of reference are used to associate these linguistic objects with other related objects: (a) *definitional reference*, used to link entity (E1) *Anwar Sadat* to a corresponding attribute *Egyptian President*, (b) *metonymic reference*, since (E1) can be coerced into (E2), (c) *part-whole reference*, since *BW stockpiles* (E4) necessarily imply the existence of a *BW program* (E5), and (d) *relational reference*, since *validating* is subsumed as part of the meaning of *declaring* (as determined by WordNet glosses), while *admitting* can be defined in terms of *declaring*, as in *declaring [to be true]*.

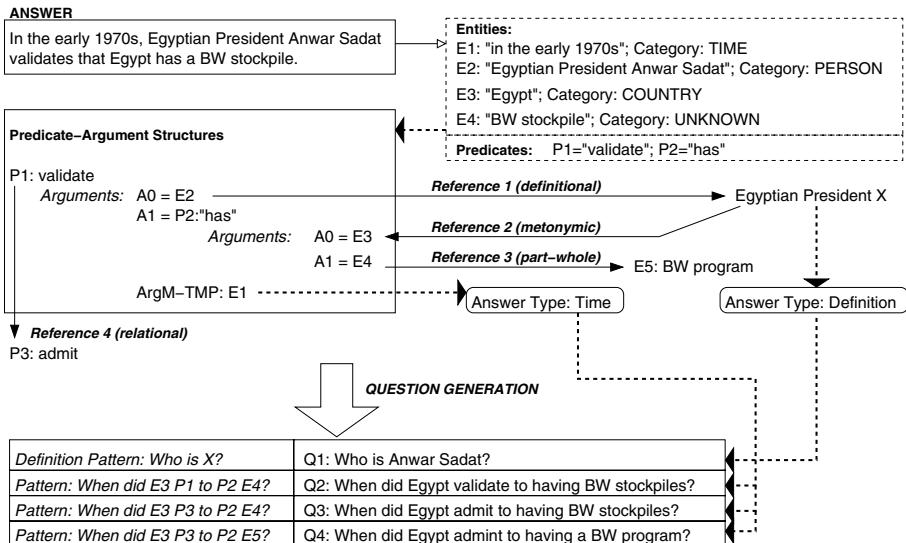


Figure 15. Associating Questions with Answers.

Additionally, the definitional reference indicates that entity E_2 is the answer to a question that has the answer type DEFINITION. Such a question has a definitional pattern “Who is X ?", where X corresponds to the PERSON name, stripped of all describing attributes (e.g., *Egyptian President*). The definition question generated is Q1, as illustrated in Figure 15. Another entity that is associated with an answer type is E_1 . The corresponding EAT is TIME, which corresponds to questions Q2, Q3 and Q4 illustrated in Figure 15. Each of them is generated based on a corresponding pattern which uses the predicates and entities identified in the answer, or references of them. Question patterns were acquired from a large database of questions and answers that contains more than one million such examples.

It is to be noted that all the generated questions illustrated in Figure 15 have the topic “*Egypt’s Biological Weapons Program*”. Questions Q1 and Q4 correspond to the GENERAL component of the semantic signature of the topic. Figure 10 from Section 2.2 illustrates the semantic signature of the topic. Questions Q2 and Q3 correspond to the PRODUCTION component.

Problem 2: We have found that the identification of the association between a candidate answer and a question depends on (a) the recognition of predicates and entities based on both the output of a named entity recognizer and a semantic parser (Surdeanu *et al.*, 2003) and their structuring into predicate-argument frames, (b) the resolution of reference (addressed in Problem 1), (c) the recognition of implicit relations between predication stated in the answer. Some of these implicit relations are referential, as is the relation between predicates P_1 and P_3 illustrated in Figure 15. A special case of implicit relations are the causal relations. Figure 16 illustrates an answer where a causal relation exists and is marked by the cue phrase *because*. Predicates – like those in Figure 16 – can be phrasal (like P'_3) or negative (like P'_2).

Causality is established between predicates P'_2 and P'_4 as they are the ones that ultimately determine the selection of the answer. The predicate P'_4 can be substituted by its nominalization since Arg1 of P_2 is *BW*, the same argument is transferred to P''_4 . The causality implied by the answer from Figure has two components: (1) the effect (i.e., the predicate P''_4) and (2) the result, which eliminates the semantic effect of the negative polarity item *never* by implying the predicate P_5 , *obstacle*. The questions that are generated are based on question patterns associated with causal relations and therefore allow different degrees for the specificity of the resultative, i.e., *obstacle* or *deterrant*.

We generated several questions for each answer passage. Questions were generated based on patterns that were acquired to model interrogations using relations between predicates and their arguments. Such interrogations are based on (1) associations between the answer type (e.g., DATE) and the question stem (e.g., *when*) and (2) the relation between predicates, question stem and the words that determine the answer type (Narayanan and Harbagiu, 2004). As Figure 15 and Figure 16 illustrate, we used patterns based on (a) embedded predicates and (b) causal or counterfactual predicates.

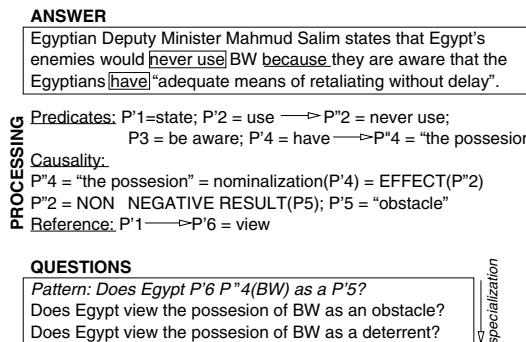


Figure 16. Questions for Implied Causal Relations.

3. PROCESSING QUESTIONS IN CONTEXT

Questions are rarely asked in isolation. When satisfied by the answer, a user may have follow-up questions, requiring additional information. If the answer is not satisfactory, a new question may clarify the user's intentions, thus enabling a better disambiguation of the question. In both cases, the follow-up question needs to be interpreted in the context of the previous interactions with the QA system. In 2001, the TREC QA track evaluated ten different groups of questions, each representing a different context. Figure 17 illustrates two examples of question series that were evaluated in the 2001 TREC.

Context	Question	Focus	EAT
CTX1a:	Which museum of Florence was damaged by a major bomb explosion in 1993?	TARGET	MUSEUM
CTX1b:	On what day did it happen?	BOMBING	DATE
CTX1c:	Which galleries were involved?	TARGET	GALLERIES
CTX1d:	How many people were killed?	VICTIMS	NUMBER
CTX1e:	Where were these people located?	VICTIMS	LOCATION
CTX1f:	How much explosive was used?	EXPLOSIVE	QUANTITY
CTX3a:	What grape variety is used in Chateau Petrus Bordeaux?	WINE	GRAPE VARIETY
CTX3b:	How much did the futures cost in the 1989 vintage?	VALUE	PRICE
CTX3c:	Where did the winery's owner go to college?	OWNER	UNIVERSITY
CTX3d:	What California winery does he own?	OWNER	UNIVERSITY

Figure 17. Example of question series asked in context.

Most series contained three or four questions, though one series contained nine questions. The evaluation of the questions asked in context tested the system's ability to track discourse objects through a series of questions. Generally, the discourse objects relate to a topic, while the question focus may be changing. For example, the series of questions illustrated in Figure 17 have two topics: the 1993 bomb explosion at the Uffizzi and the Chateau Petrus Bordeaux vineyard.

The intentions of the users can be recognized if the questions that are posed are found to be similar to the questions generated when the question topic and the question focus are processed. However, many times the topic and the focus can also be derived by processing the context of a series of questions.

3.1 Context representations

The context in which a question is posed plays an important role. To represent the context we consider (1) the question topic, (2) the question focus, (3) the EAT, and (4) the entities referred by the questions asked in the same context. It is typical for the first question from a series of interrogations to introduce the context topic. Figure 17 illustrates two series of questions asked in two different contexts. Both series of questions were evaluated in TREC 2000. The topic of CTX1 is "MUSEUM BOMBING" and the topic of CTX2 is "BORDEAUX WINERY". Figure 17 also illustrates the focus of each question as well as its EAT.

The detection of the context topic is a complex process that relies on several semantic resources. Similar to question topics, the semantic information encoded in FrameNet may help to indicate which words define the topic. In the case of context CTX1, the topic is selected based on the topic signatures that are available for (1) BOMBINGS and (2) MUSEUMS. In (Harabagiu, 2004), the enhanced topic signatures discover relations between explosions, their causes, and the damage they produce for the BOMBINGS topic. The target of the bombing is the museum that is thought by the question CTX1a.

As Figure 17 illustrates, each following question has a focus which belongs to the semantic signature of the topic. The same observation holds for context CTX3 as well. For this context, the topic is indicated by the semantic class of a name. For example, knowing that *Chateau Petrus Bordeaux* is the name of a wine or a winery may help in discovering the typical concepts for this topic and their semantic classes. They include the LOCATION; the PRODUCT, which are varieties of wines; the owner, which is a PERSON or an ORGANIZATION; and the vintage, which is expressed as the YEAR of the production. The typical relations pertaining to wineries associate them with vineyards, which have grape varieties as products. Typical relations of wineries may be inherited from topics referring to general commercial organizations that have futures, which can be acquired or sold. In this way, the topic of CTX2 is discovered.

Slot Name	Slot Description
BOMB	The description or type of bomb.
PERPETRATOR	The alleged, suspected, claimed, or known perpetrator.
DAMAGE	A description of the physical damage to objects other than persons.
INJURY	Identifiers of persons injured.
VICTIMS	Identifiers of persons killed.
LOCATION	The most specific event location.
DATE	Description or indication of the time of the bombing.
TARGET	Description of the entity targeted by the bombs.
EXPLOSIVE	Description of the explosive used for manufacturing the bomb.

(a) Components of the Semantic Signature of the Topic

Concept	Weight	Concept	Weight
bombing(N)	54223	people(N)	47479
kill(V)	32064	attack(N)	29932
official(N)	27781	government(N)	22975
police(N)	17503	explode(V)	16755

(b) Concept Signature of the Topic

Syntax-Based Relations	Context-Based Relations
1. VP(EXPLODE-word)-Subject → NP(ARTIFACT OBJECT)	1'. VP(INJURE-word) → NP(BOMB-word)
2. VP(EXPLODE-word)-Object → NP(BOMB-word)	2'. VP(KILL-word) → NP(BOMB-word)
3. VP(DAMAGE-word)-Object → NP(ARTIFACT VEHICLE BOMBING)	3'. VP(EVENT)-Object → NP(BOMB-word)
4. VP(INJURE-word)-Object → NP(NUMBER)	4'. VP(EXPLODE-word) → NP(ARTIFACT OBJECT)
5. VP(EXPLODE-word)-Prep_Attach{in outise} → NP(ARTIFACT)	5'. VP(EXPLODE-word) → NP(MALE HUMAN LIVING PERSON-Name)
6. VP(EXPLODE-word)-Object → NP(DATE TIME)	6'. VP(EXPLODE-word) → NP(CITY-Name COUNTRY-Name)
7. VP(CLAIM-word)-Object → NP("responsability")	7'. VP(EXPLODE-word) → NP(ORGANIZATION-Name)
8. VP(INJURE-word)-Object → NP(HUMAN LIVING PERSON-Name)	8'. VP(EXPLODE-word) → NP(DATE TIME)
9. VP(CLAIM-word)-Subject → NP(HUMAN LIVING PERSON-Name)	9'. VP(INJURE-word) → NP(HUMAN LIVING)
10. VP(KILL-word)-Object → NP(NUMBER)	10'. VP(KILL-word) → NP(NUMBER)

(c) Topic Relations

Figure 18. Topic Signature for BOMBINGS.

For the context in CTX1 illustrated in Figure 17, the topic is BOMBINGS, and the method reported in (Harabagiu, 2004) discovers the topic signatures illustrated in Figure 18. Figure 18(a) lists the components of the semantic signature of the topic,

giving also a description of each element. Figure 18(b) lists the most significant topic concepts discovered by the likelihood ratio method. This method was presented in Section 2.1, when describing the Knowledge-Centric Method for Acquiring Semantic signatures. Figure 18(c) presents the topic relations discovered for the BOMBING topic by the method reported in (Harabagiu, 2004). There are two kinds of relations: (i) relations motivated by syntactic information, especially by Verb-Subject, Verb-Object and Verb-Prepositional Attachment relations; and (ii) relations motivated by context, which approximate intra-sentential centering, as introduced in (Kameyama, 1997). Figure 19 lists the representation of the context CTX1 illustrated in Figure 17.

Question	Topic Signature	Focus	EAT	New Entities
CTX1a	BOMBINGS (illustrated in Figure 18)	TARGET	MUSEUM	Florence, 1993, explosion, bomb
CTX1b		BOMBING	DATE	day
CTX1c		TARGET	GALLERIES	galleries
CTX1d		VICTIMS	NUMBER	people
CTX1e		VICTIMS	LOCATION	—
CTX1f		EXPLOSIVE	QUANTITY	explosive

Figure 19. Representation of the Context CTX1.

To have a full representation of the context, reference resolution between entities needs to be solved. The reference accounts for the context cohesion.

3.2 Context cohesion

Questions are not asked in isolation, but rather in a cohesive manner that relates a sequence of interrogations. The cohesion is achieved by repetitions, semantic relatedness and reference. A sequence of questions posed in the same context is characterized by several forms of anaphoric reference or ellipsis. A question may use:

1. demonstrative pronouns, like this, these or there; (e.g., On what day did this happen?, Where were these people located?, or Name a company that flies there?)
2. third person pronouns, like he or it; (e.g., What California winery does he own?, or In what facility was it constructed?)
3. possessive pronouns, like his or its; (e.g., What was his first radio song?)
4. definite nominals, in which the definite article or the demonstrative pronoun indicate that the concept was already introduced by a previous question or answer; (e.g., What executive from the company was a member of the Supreme Council in 1994? or This city's name was later changed to what?)
5. nominalizations of verbs used in previous questions; (e.g., When was construction begun? following In what facility was it constructed?)

6. elliptical reference, in which the expected answer type is inherited from the previous question; (e.g., How many are poisonous to humans? following How many species of spiders are there?)
7. causal-effect reference; e.g., explosive from How much explosive was used? is the cause of explosion from its preceding question Which museum in Florence was damaged by a major bomb explosion in 1993?
8. meronymic reference, e.g., galleries from Which galleries were involved? are referenced as a part of the museum from the preceding question Which museum in Florence was damaged by a major bomb explosion in 1993?.

The resolution of all the forms of reference is performed by identifying the antecedent of the anaphora in (1) a previous question; or (2) the answer to a previous question; or (3) an anaphor used in a previous question. Before applying the reference resolution algorithm, the pleonastic usage of pronouns is identified, ruling out the resolution of pronouns like *there* in *How many species of spiders are there?*.

The reference resolution algorithm employed for QA is different from reference resolution algorithms used in discourse or dialog processing because the goal is not to resolve the reference, but to identify the question that either contains the antecedent of the reference or expects an answer that contains the antecedent. Consequently, when processing a question Q_i that contains a reference, by knowing which preceding question Q_{i-k} generates the antecedent, we can combine the keywords of Q_i with the keywords of Q_{i-k} to retrieve relevant paragraphs. Moreover, since question keywords are usually extracted in a predefined order, when keywords from two different questions are combined, the keywords from the previous question always preceded the keywords from the current question. For example, Table 1 illustrates the combination of keywords resulting from the reference resolution within context questions.

Table 1. Keyword extraction for context questions.

<i>Example 1</i>
<i>Question CTX1d:</i> How many people were killed?
<i>Keywords from CTX1d:</i> ($k_1=\text{killed}$)
<i>Reference of question CTX1d = question CTX1a:</i>
Which museum in Florence was <u>damaged by a major bomb explosion</u> in 1993?
<i>Keywords from CTX1a:</i> ($k_2=\text{Florence}$, $k_3=\text{bomb}$, $k_4=\text{explosion}$)
<i>Keywords used to process CTX1d:</i> ($k_2=\text{Florence}$, $k_3=\text{bomb}$, $k_4=\text{explosion}$, $k_1=\text{killed}$)
<i>Example 2</i>
<i>Question CTX7g:</i> How wide?
<i>Keywords from CTX7g:</i> ($k_1=\text{wide}$)
<i>Reference of question CTX7g = question CTX7a:</i>
What type of vessel was the modern Varyag?
<i>Keywords from CTX7a:</i> ($k_2=\text{Varyag}$)
<i>Keywords used to process CTX1g:</i> ($k_2=\text{Varyag}$, $k_1=\text{wide}$)

The algorithm that performs reference resolution for context questions is as follows:

Algorithm Reference Context Resolution (Q)

```

Input: LQ = precedence-ordered list of previous questions asked in the same context +
wQ, where wQ = the reference word from Q when we have an ellipsis wQ = ø
if (wQ repeats in a question Q' from LQ)
    return Q' if it does not contain a reference
else return Reference Context Resolution(Q')
if (wQ is a pronoun)
    CASE (wQ ∈ {he, his, she, her, they, their})
        return Q', the closest question from LQ that has the expected answer type = PERSON or
        has a PERSON mentioned
    CASE (wQ ∈ {it, its})
        if (wQ is the subject of one of the verbs {happen, occur}) return Q' the first question that
        mentions an event
        return Q', the closest question from LQ that has the expected answer type different
        than PERSON or mentions some non-PERSON entity
    CASE (wQ = there)
        return Q', the closest question from LQ that has the expected the expected answer type
        = LOCATION or has a LOCATION mentioned
    CASE (wQ = this or wQ = ø)
        return Q', the first question from LQ if it does not contain a reference
    else return Reference Context Resolution(Q')
if (wQ morphological-root(wQ)= morphological-root(wQ'))
    -where wQ' is a word from a question Q' ∈ LQ
        return Q'

```

An interesting by-product of this reference resolution algorithm is the way in which it allows the modelling of context through the passing of keywords from the antecedent question to the follow-up question. It is interesting to be noted that each time when a follow-up question is processed, the QA system operates on the same relevant paragraphs as for the antecedent question in 85% of the cases. However, every time it extracts different answers, since the expected answer type is dictated by each different question. The set of keywords used can be expanded to include topic concepts or words used in the topic relations.

4. INTENTIONS AND IMPLICATURES

A question, followed by its answer, can be viewed as a component of a conversation between a user and a QA system. As such, when expressing its intentions, the user employs what Grice (1975) has called *conversational implicatures*. The notion of conversational implicature is, as stated in (Levinson, 1983), one of the most important ideas in linguistic pragmatics. This is because implicatures provide an explicit account of how it is possible to mean more than what is actually ‘said’. With other words, in a conversation, the question or the answer may convey more information than what is literally expressed by the conventional sense of the linguistic expressions which are used.

The recognition of question implicatures is important, since they enable the identification of a user's purpose, or set of purposes. When using a Question Answering system to find information, the user cannot separate the intentions and beliefs from the formulation of the question. A direct consequence of this phenomenon is that the user incorporates his or her intentions and beliefs into the interrogation. For example, when asking the question:

Q6: What kind of assistance has North Korea received from the USSR/Russia for its missile program?

the user associates with the question a number of intentions, that may be expressed a set of *intended questions*. Each intended question, in turn generates implied information, which may be expressed as a set of *implied questions*. For question Q6, a list of intended questions and implied questions is listed in Table 2.

Table 2. Question decomposition associated with question Q6.

Intended Questions	Implied Questions
Q_1^i : What is USSR/Russia?	Q_1^{m1} : Is this the Soviet/Russian government? Q_2^{m1} : Does it include private firms, state-owned firms, educational institutions, and individuals?
Q_2^i : What is North Korea?	Q_1^{m2} : Is this the North Korean government only? Q_2^{m2} : Does it include private firms, state-owned firms, educational institutions, and individuals?
Q_3^i : What is assistance?	Q_1^{m3} : Is it the transfer of complete missile systems, licensing agreements, components, materials, or plans? Q_2^{m3} : Is it the training of personnel? Q_3^{m3} : What kind of training? Q_4^{m3} : Does transfer include data, and, if so, what kind of data? Q_5^{m3} : Does transfer include financial assistance, and, if so, what kind of financial assistance?
Q_4^i : What are the missiles in the North Korean inventory?	Q_1^{m4} : Are any based upon Soviet/Russian designs? Q_2^{m4} : If so, which ones? Q_3^{m4} : What was the development timeline of the missiles? Q_4^{m4} : Did any timeline differ significantly from others? Q_5^{m4} : Did North Korea receive assistance from other sources besides USSR/Russia to develop these missiles?
Q_5^i : When did North Korea receive assistance from the USSR/Russia?	Q_1^{m5} : Was any intended assistance halted, stopped or intercepted?
Q_6^i : What are the sources of information?	Q_1^{m6} : Are the sources reliable? Q_2^{m6} : Is some information contradictory?

Most of the intended questions are similar with the questions evaluated in TREC, and they are part of the questions generated for the intentional structure associated with Q6. For example questions Q_1^i , Q_2^i and Q_3^i are so-called *definition questions*, since they ask about defining properties of an object. However unlike the TREC definition questions, these questions express unstated intentions of the questioner

and need to be processed for finding implications that can be associated with them. The desired result for such processing are questions Q_1^{m1} and Q_1^{m2} , which are processed in the context of Q_1^i . The context processing involved, as it was detailed in Section 3, deals both with the representation of the context (i.e., the topic, focus and EATs) and the resolution of reference (i.e., *it* from Q_1^{m2} refers to *USSR/Russia*). Questions Q_4^i and Q_5^i are *factoid questions*, requesting information about facts or events. Q_6^i asks about the source of information that enables the answers of question Q_6 .

The implied questions associated with Q_4^i and Q_5^i further probe into the ASSISTANCE aspect of the PRODUCTION component of the topic. They reveal unstated implications made by Q_4^i and Q_5^i . For example, Q_1^{m4} reveals the implication that (a) missiles are designed based on some models; (b) the designs have been produced by the Soviet Union/Russia. To identify such implications, techniques used for metonymy coercion need to be employed. Furthermore, the implied information needs to be incorporated into a question generation process. The resulting series of questions can be further processed such that referring expressions generate a cohesive context, as it can be noticed for the implied question corresponding to Q_4^i .

The implied questions associated with Q_5^i and Q_6^i illustrate two other interesting aspects of implications. When the EAT of the intended question is DATE or TIME (e.g., Q_5^i), aspectual information related to the question predicates can be sought (e.g., halting or stopping an activity). The implication may also have to do with the information related to the activity (e.g., interception) or the sources of information. These aspects are covered by the EVIDENCE component of each topic of interest. In the case of the intended question Q_6^i , the implied information asks about the quality of the sources or of the information. Key to uncovering the implied information is methodology of coercing, thus expanding *sources of information* into *reliable sources* and *contradictory information*. The coercion technique is inspired by a recent method for metonymy coercion.

4.1 Metonymy coercion

Numerous complex questions are difficult to process because their answer type cannot guide the process of finding the answer. For example, the answer type of question Q_6 : “*What kind of assistance has North Korea received from the USSR/Russia for its missile program?*” has the answer type FORM-OF-ASSISTANCE, that can be mapped to a very large number of verbs and nominalizations encoded in the WordNet synonyms. A sentence like “*In 1992, 10 Russian scientists modernized North Korea’s missile program.*” provides a partial answer. The answer requires two metonymy resolutions:

1. USSR/ Russia, recognized as a country, needs to be coerced to “*10 Russian scientists*”, since generally people are offering assistance in the name of their countries; and
2. the form of assistance needs to be coerced to “*modernizing the missile program*”, since a viable missile program relies on up-to-date capabilities.

We have considered four different forms of metonymies and developed resolution methods corresponding to each of them. We have tackled Named Entity Metonymy, Predicate Metonymy, Argument Metonymy and Evidence Metonymy. For illustration we detail herein the methods used only for the first two forms of metonymies.

Named Entity Metonymy. For each topic, several classes of named entities are encountered. Names classified as PERSON, ORGANIZATION, or LOCATION are encountered in almost every topic. Names representing a WEAPON, TREATY, or a WAR may be specific only to some topics. Moreover, many classes of names are structured into a taxonomy. For example, Figure 20 illustrates the classes of names that are types of locations.

CITY_NAME	STREET_NAME	LAKE_NAME	COMPLEX-LOCATION_NAME
COUNTRY_NAME	HEMISPHERE_NAME	GULF_NAME	BEACH_NAME
PROVINCE_NAME	PLANET_NAME	WATERFALL_NAME	CANYON_NAME
COUNTY_NAME	OCEAN_NAME	CANAL_NAME	MOUNTAIN_NAME
CONTINENT_NAME	ISLAND_NAME	RIVER_NAME	PARK_NAME
SEA_NAME	BAY_NAME	PENINSULA_NAME	NEIGHBORHOOD_NAME

Figure 20. Classes of Names that Represent LOCATIONS.

As mentioned in (ACE, 2002) names of a certain class are used to reference an entity of another name class. For instance, capital city names are typically used for referring to the corresponding government. To discover and resolve all name-based metonymies, we rely on a double-stage classification. In the first stage, names are classified as literal or metonymic. A name has a metonymic interpretation if, when recognized as belonging to a name class NC_i , it can be interpreted as representing a name class NC_j as well. For example, in the sentence SE1: “*Beijing will not continue sales of anti-ship missiles to Iran.*”, the name *Beijing* is classified as the name of a city, which has the property that it is the capital of China. In the sentence, *Beijing* in fact refers to the Chinese government, and thus it should be classified as a special type of organization, namely a governmental organization. In contrast, in the sentence SE2: “*Secretary of Defence William S. Cohen has wrapped up a four-day visit to Beijing.*”, the name *Beijing* has a literal interpretation, referring to the capital city of China. The second stage of our name metonymy resolution approach consists of the decision of which new semantic class should also be attached to the name, to satisfy the non-literal reference. For example, in sentence SE1, the name *Beijing* receives two labels: CITY_NAME and GOVERNMENT_ORGANIZATION.

The idea of considering metonymy recognition and resolution as a two-stage classification problem is not new. In (Markert and Nissim, 2002), metonymies for LOCATION names are resolved by using features typically employed in word sense disambiguation. Such features include (a) co-occurrence features based on windows of size 0, 1, 2, 3, 4, 5, 10 and 25 words; (b) collocation features, derived from the two words to the left and right of the name; (c) grammatical features, based on the recognition of the *subject, object, prepositional phrases* and *NP modifiers*.

In contrast, our metonymy recognition and resolution method aims to first interpret the literalness of all 142 classes of names we are able to identify with our Named Entity Recognizer.

Classification Stage I: In the first stage of the classification we identify whether a name has a literal or a metonymic usage. To recognize metonymies we employ three kinds of features:

A. Semantic Features. Such features result from (i) the processing of names; (ii) the results of semantic parsers; (iii) lexico-semantic classes of predicates that have metonymies as arguments.

- The structure and context of a name may also give information about its literal / metonymic interpretation. Names that contain other names in their structure may indicate a license to metonymy. For example, *University of California*, classified as an EDUCATIONAL_ORGANIZATION, may indicate the possibility that it may be referred by the name *California*, recognized as the name of a PROVINCE/STATE. The features collect lists of name classes. The first element of the list represents the semantic class of the entire name, No, (e.g., EDUCATIONAL_ORGANIZATION) whereas the following elements of the list are pairs (name, class) for each name identified in the structure of No (e.g., (*California*, STATE)).
- We use three different semantic parsers. The first parser, described in (Surdeanu *et al.*, 2003), produces predicate-argument structures similar to those annotated in PropBank. The second parser uses the same techniques as those of the previous parser, but it generates predicate-argument structures for predicates expressed as nominalizations. Such predicate-argument structures result from training on NomBank (Meyers *et al.*, 2004). The third parser, detailed in (Bejan *et al.*, 2004), generates frame structures for every target word that was encoded in FrameNet. For each name we retain the following features: (1) the argument number in which the name appears, (2) the predicate of the predicate-argument structure coercing the name, (3) the FrameNet frame covering the name, and (4) the frame element to which the name belongs. For example, for sentence SE1, the features associated with the name *Beijing* are [Arg0, sales, Commerce_sell, seller]. For SE2, the features associated with the same name are [ArgM-LOC, visit, Arriving, goal].
- Predicates related to names can also be classified based on the WordNet lexico-semantic classes of verbs and nouns. Since many predicates are semantically ambiguous, when using the lexico-semantic class feature of the predicate, the feature vector contains the classes corresponding to each of the semantic senses of the predicates. For example, in the case of sentence SE2, the *visit* predicate has eight senses as a verb in WordNet 2.1. Two senses belong to the SOCIAL lexico-semantic group, two senses belong to the MOTION group, two to the COMMUNICATION group, one to the STATIVE group and one to the CHANGE group. The feature vector for SE2 will be [SOCIAL, MOTION, COMMUNICATION, STATIVE, CHANGE]. Since *Beijing* has the argument number ARGM-LOC, the value MOTION of the feature vector will most likely determine the literal reading of the name *Beijing* as a CITY_NAME. In contrast, in SE1, the feature set of verb classes

is [POSSESSION, STATIVE, SOCIAL]. The argument number is Arg0 and the FE is SELLER. The name classes associated with a SELLER are PERSON or ORGANIZATION. The second stage of the classification allows the selection of GOVERNMENT_ORGANIZATION as a metonymic resolution of the name.

B. Syntactic Features. There are several syntactic structures that may indicate the metonymic usage of a name. They include (i) appositions, (ii) NP modifiers, and (iii) coordinations.

- Appositions preceding or following a name may contain names themselves. In such cases, the name in the apposition may be used as an indicator of the literalness of the name. For example, the name *Beijing*, in “*Beijing, China*” will always have a literal meaning of a CITY_NAME.
- NP modifiers may also contain names. Metonymic references can be established between the name classes, similarly to the appositions.
- Coordinations between names propagate the metonymic interpretation of a name from one member of the coordination to the other.

The syntactic features take the form of a pair. The first element is the syntactic function (e.g., apposition), the second element is a list of names and their associated classes. Each name from the list was identified in the same syntactic constituent.

C. Morphological Features. Derivational morphology also helps in identifying the reference to another name class. For example, the NP *German Government* is connected morphologically to the name *Germany*, indicating that a COUNTRY (e.g., *Germany*) may be used to refer to a GOVERNMENT_ORGANIZATION (e.g., *German Government*).

D. Geographical and Political Information Features. Metonymy resolution also benefits from geographical or political information. For example, knowledge about which city is the capital of which country helps deciding whether to interpret the usage of a city name as metonymic or not. In a similar way, knowledge of the names of the officials representing a country or another organization also contributes to the classification of the name as literal or metonymic. Historical information about names may also be useful. Knowing the fact that the COUNTRY named *Russia* had the name *USSR* before 1991 is helpful. Such information is organized as quintuples generated with information available in off-line repositories for geographical, political or historical information. For example, for the name *Beijing* encountered in the sentences SE1 and SE2, the quintuple is (*Beijing*, CITY_NAME, Capital, *China*, COUNTRY_NAME); for *William S. Cohen*, the quintuple is (*William S. Cohen*, PERSON_NAME, Official, *USA*, COUNTRY_NAME); for *Russia* (*USSR*, COUNTRY_NAME, 1991, *Russia*, COUNTRY_NAME).

The features are used to decide whether a name has a metonymic interpretation or not. We have used decision trees to classify the names. The supervised method was based on a set of 414 documents that we extracted from the CNS corpus made available to the participants in the AQUAINT program. These documents contained 7040 names and we have performed on them 1970 annotations as metonymies. The annotations distinguished between literally and metonymic readings of the names.

Three linguists performed the annotations, obtaining an inter-annotator agreement of 87%, when measured with the kappa statistics (Carletta, 1996). All features encountered in the training data were used to train a decision tree based on C5. Only syntactic and semantic features were used in the first phase of the classification.

Classification Stage II: Names that are classified as metonymic needed a second stage classification. The second phase classifier is a multi-class classifier, which uses (1) the semantic class of the name, (2) the knowledge that it has metonymic interpretation, (3) the values of the syntactic and semantic features extracted for a new example, and (4) the values of the morphological, geographical, and political features when the same name and name class were encountered in the training data for metonymic names. The classifier, based on decision trees, decides which new name class label can be assigned to the metonymy.

Name coercions establish a reference between country names (e.g., *Russia*) and (a) several forms of organizations (e.g., *Russian Government*, educational institutions) or (b) persons. Such coercions are responsible for the associations of questions Q_1^i and Q_2^i into Q_1^{m1} and Q_2^{m1} or Q_1^{m2} and Q_2^{m2} respectively.

Predicate Metonymy. This form of metonymy requires the coercion of the predicates encountered in questions. For example, in question Q_6 : “*What kind of assistance has North Korea received from the USSR/Russia for its missile program?*”, the phrasal predicate *receive assistance* needs to be coerced into the various possible forms of assistance, e.g., *modernize*, *transfer technology*, or *train personnel*. This form of coercion is related to **logical metonymy**, i.e. constructions where the argument of a word in syntax appears to be different from the argument in logical form. For example, the sentence “*John enjoys the book*” may be coerced into “*John enjoys reading the book*” or “*John enjoys looking at the book*”. At the logical level, the enjoying event takes as arguments the EXPERIENCER = *John* and the EXPERIENCE = “*John reads the book*”. The EXPERIENCE argument can be further decomposed into the predicate *read* with its two arguments, the agent *John* and the object *the book*. The coercion involves the discovery of the predicate *read* that has *book* among its typical objects, unlike *enjoy* which has experiences as its typical syntactic object. In the case of predicate coercion for QA, the predominant form of coercion does not insert a new predicate, but it rather replaces a predicate with another. For this reason, we call it **troponymic metonymy**, since the coercion substitutes a predicate with one of its troponyms. For example, in the case of question Q_6 , this would involve the recognition of the predicate *modernize* as a troponym to the phrasal predicate *receive assistance*, or *assist*.

It is to be noted that the *troponymy* between *assist* and *modernize* is different than the one between the verbs *stride* and *walk*, as encoded in the WordNet database. The difference stems from the fact that regardless of context, striding is a way of walking with long steps. In contrast, by modernizing a program, one assists the program, but when modernizing a house, no assistance is provided to the house, but rather the house becomes more liveable or more valuable. This explains why in WordNet, there is no *troponymy* relation encoded between the *assist* and *modernize* verbs. Therefore, the resolution of troponymic metonymies relies on the recognition of context-dependent *troponymy* relations. We found that such relations can be

discovered similarly to the way in which interpretations required in logical metonymy are discovered.

Part of the question interpretation is the discovery of the question topic and question focus. In the case of question *Q6*, techniques described in Section 2 enable the recognition of the question topic as “*North Korea’s missile program*”. The question focus is “*USSR/Russia’s assistance to the North Korean missile program*”, and the EAT are the forms of assistance, or the troponyms of the assistance predicate.

The topic representation allows the discovery of the most representative concepts, e.g., *plutonium*, and the most representative relations, e.g., [acquire-OBJ → *plutonium*]. Some of these relations and concepts may be troponyms of the questions focus, others not. To discover troponyms we employ two techniques:

Technique 1: Discovering troponyms with semantic classification.

- A. We collect all the predicates that have the property that they have in the immediate context (1) representative concepts from the topic, (2) representative relations from the topic, (3) the arguments of the predicate metonymy, or their name or argument coercions. The immediate context is represented by the sentence containing the predicate and its preceding sentence. To evaluate the likelihood that a coercion *T* is a troponym of the question predicate *V*, we consider:
 - i. the arguments of the arguments of the question predicate, their PropBank / NomBank argument numbers, the FrameNet frame that covers them, and the corresponding FEs resulting from the semantic parse of the question.
 - ii. the arguments of the candidate coercion, their PropBank / NomBank argument numbers, the FrameNet frame that covers them and the corresponding FEs resulting from the syntactic parse of each candidate context.
- B. We classify the heads of the arguments encountered in the question and in the coercion context against the same concept taxonomy that was used to produce the topic signature.
- C. We generate an ad-hoc training corpus by using the WordNet troponym relations. For any candidate troponym *T*, we select its WordNet hypernym *H* for gathering positive examples by searching the World Wide Web. All other predicates from the context of *T* are negative examples of the classification. Again, we have implemented the classification with decision trees.
- D. The likelihood of candidate troponyms is decided by the decision trees trained on the ad-hoc corpus.

Technique 2: Discovering troponyms as logical coercions.

In (Lapata and Lascarides, 2003) a probabilistic method for solving logical metonymy was reported. Adapting this method to the troponymic metonymy, coercions can be modelled as the joint distribution $P(T, A, V)$ of three variables: the metonymic predicate *v* (e.g., assist/assistance), its argument list $A = [\text{ARG1}, \text{ARG2}, \dots]$, and the sought-after context-dependent *troponym T* (e.g., modernize). By

choosing the ordering $\langle T, A, V \rangle$ for variables, similarly to (Lapata and Lascarides, 2003) we factor $P(T, A, V) = P(T) \times P(A | T) \times P(V | A, T)$.

The probabilities $P(T)$, $P(A | T)$ and $P(V | A, T)$ can be estimated using maximum likelihood as follows:

$$\hat{P}(T) = \frac{f(T)}{N}; \quad \hat{P}(A | T) = \frac{f(T, A)}{f(T)}; \quad \hat{P}(V | A, T) = \frac{f(V, A, T)}{f(A, T)},$$

where N is the sum of all the verbs from the corpus of texts, excluding modals and auxiliaries and $f(T)$ amounts to the number of times the metonymic predicate T (as a verb or a nominalization) is present in the corpus. $f(T, A)$ counts the number of time the verb or nominalization T has any of the members of the list A as an argument, $f(A)$ counts the number of times at least two of the arguments from A occur in the same paragraph and $f(T, V, A)$ computes the number of times in which both predicates T and V have at least one common argument from the list A . The frequency $f(T, V, A)$ presupposes the co-occurrence of metonymic expressions and their interpretations in the same corpus, but as Lapata and Lascarides (2003) note, there is discrepancy between a metonymic interpretation and its usage. However, unlike the case of logical metonymies, the metonymy V and its interpretation T do not have to share a verb complement relationship, since V and T do not collocate, but rather T substitutes V in some predicate-relation with an argument from A . Moreover, in questions, metonymies occur more frequently, whereas their overt interpretations are more frequent in the corpus from which the answers are extracted. Consequently, the argument list A should contain only arguments that are recognized in the question. For this reason, we make the following approximation:

$$P(V | A, T) \approx P(V | A) \quad \text{and} \quad \hat{P}(V | A) = \frac{f(V, A)}{f(A)}$$

By substituting the probability estimations in the formula for computing the join probability $P(T, A, V)$ we obtain:

$$P(T, A, V) = \frac{f(T, A) \times f(V, A)}{N \times f(A)}$$

4.2 Accepting Implicatures and Intentions through Interactive QA

The correct interpretation of a question, including its implicatures and intentional structure processed from the question enables a QA system to provide more accurate and informative answers. By generating the intentional structure of a question, each complex question is associated with (i) a semantic signature of its topic and focus; (ii) a question decomposition that provides a list of questions and their answers to reflect the user's recognized intentions; and (iii) a dominance structure that provides an order on the decomposed questions. Furthermore, the coercion of implicatures generates new questions. All sentences that are classified as containing implied information are used as answers for implicative questions. Section 2.4 has described the methodology for generating questions for a given answer-sentence.

	Predictive Questions:
QU1: Does Iran have an indigenous CW program?	(QP1a) How did Iran start its CW program? (QP1b) Has the plant at Qazvin been linked to CW production? (QP1c) What CW does Iran produce?
Answer (A1):	<i>Although Iran is making a concerted effort to attain an independent production capability for all aspects of chemical weapons program, it remains dependent on foreign sources for chemical warfare-related technologies.</i>
	Predictive Questions:
QU2: Where are Iran's CW facilities located?	(QP2a) What factories in Iran could produce CW? (QP2b) Where are Iran's stockpiles of CW? (QP2c) Where has Iran bought equipment to produce CW?
Answer (A2):	<i>According to several sources, Iran's primary suspected chemical weapons production facility is located in the city of Damghan.</i>
	Predictive Questions:
QU3: What is Iran's goal for its CW program?	(QP3a) What motivated Iran to expand its chemical weapons program? (QP3b) How do CW figure into Iran's long-term strategic plan? (QP3c) What are Iran's future CW plans?
Answer (A3):	<i>In their pursuit of regional hegemony, Iran and Iraq probably regard CW weapons and missiles as necessary to support their political and military objectives. Possession of chemical weapons would likely lead to increased intimidation of their Gulf neighbors, as well as increased willingness to confront the United States.</i>

Figure 21. A Sample Predictive QA Dialogue.

The implied questions together with the questions from the intentional structures can be viewed as a set of questions that the QA system “predicts” to be asked. Experiments reported in (Harabagiu *et al.*, 2005) show that access to predictive questions enhances the quality of interactive question answering. For example, when a user asks a question like *QU1* illustrated in Figure 21, the QA system returns the answer and also proposes the predictive questions *QP1a*, *QP1b*, or *QP1c* as suggestions of possible continuations of the QA interaction. Users choose how to continue the interaction by either (1) ignoring the suggestions made by the system and proposing a different question; or by (2) selecting one of the proposed questions and examining its answer. In both cases, the user may grade the validity of the predicted questions. We used a three-valued validity scoring: (+1) for an acceptable prediction; (0) for a prediction for which the user is ambivalent; and (-1) for a prediction rejected by the user.

It is interesting to note that Figure 21 illustrates three turns from a real dialogue from a human user investigating Iran’s chemical weapons program. As it can be seen coherence can be established between the user’s questions and the system’s answers (e.g., *QU3* is related to both *A1* and *A3*) as well as between the predictive questions and the user’s follow-up questions (e.g., *QP1b* is more related to *QU2* than either *QU1* or *A1*). Coherence alone is not sufficient to analyze the quality of interactions, however, nor the validity of the intentions or implications recognized.

By accepting the validity of a predicted question, the user is able to confirm its intentions or implications (s)he made when asking a question. By selecting a predicted question, the user uses the QA system more efficiently. In our experiments, the availability of the predictive questions enables the users to submit an average of 12.25 questions each session. When no predictive questions are available, an average of 44.5 questions per session was asked.

Additionally, predictive questions also improve the overall accuracy of the answers returned by the QA system. To measure the effectiveness of QA, human annotators are used to produce a post-hoc analysis of how relevant the predicted questions were to each question entered by a user. Each predictive question was graded as “relevant” or “irrelevant” to a user question in a forced-choice task.

Aggregate relevance scores were used to calculate (1) the percentage of relevant QUAB pairs returned and (2) the mean reciprocal rank (MRR) for each user question. MRR is defined as $\frac{1}{n} \sum_i \frac{1}{r_i}$, where r_i is the lowest rank of any relevant answer for the i^{th} user query. We chose MRR as our scoring metric because it reflects the fact that a user is most likely to examine the first few answers from any system, but that all correct answers returned by the system have some value because users will sometimes examine a very large list of query results. When only answers from the QA system were available to users, only 15.7% of system responses were deemed to be relevant to a user's query. In contrast, when manually-generated predictive questions were introduced, as high as 84% of the system's responses were deemed to be relevant.

Finally, the acceptance of question intentions and implicatures was evaluated by measuring the user satisfaction. Users were consistently satisfied with their interactions with the QA system that processed the question intentions and implicatures. In all the experiments, respondents claimed that they found that the QA system (1) gave meaningful answers, (2) provided useful suggestions, (3) helped answer specific questions. Complete results of this study are presented in Table 3. Evaluation scale: 1-does not describe the system, 5-completely describes the system.

In the experiments, users were allotted two hours to interact with the QA system to gather information requested by a complex question (Table 3). In Experiment 1 (E1), eight expert users researched eight different complex questions related to chemical and biological weapons. Experiment 2 (E2) and Experiment 3 (E3) considered several of the same scenarios addressed in E1: E2 included 24 mixed teams of expert and novice users working with two complex questions, while E3 featured 4 USNR expert users working with six of the original eight complex questions, but no predictive questions were available in experiment E3.

Table 3. User Satisfaction Survey Results.

Factor	E1	E2	E3
Promoted understanding	3.40	3.20	3.75
Helped with specific questions	3.70	3.60	3.25
Made good use of questions	3.40	3.55	3.00
Gave good collection coverage	3.75	3.70	3.75
Stimulated user thinking	3.50	3.20	2.75
Easy to use	3.50	3.55	4.10
Expanded understanding	3.40	3.20	3.00
Gave meaningful answers	4.10	3.60	2.75
Was helpful	4.00	3.75	3.25
Helped with new search methods	2.75	3.05	2.25
Provided novel suggestions	3.25	3.40	2.65
Would speed up work	3.25	3.25	3.00
Overall like of system	3.75	3.60	3.75

5. EXPERIMENTAL RESULTS

To evaluate the performance of the QA system capable of processing complex questions, either in isolation or in the context of a dialog, we have relied on over 90 QA dialogs with human users. The QA system had available 9740 predictive questions. Out of these, 3210 have been hand-created, while 6530 of them were generated by the intentional structure or the metonymic coercions. In all these dialogs users were presented with a complex question, similar to the one illustrated in Figure 22.

It is to be noted that the *general background* consists of a list of subject areas, whereas the *scenario* is a narration in which several sub-topics are identified (e.g., *production of toxins* or *exportation of materials*). In addition to identifying a particular *domain of interest*, scenarios must specify the set of relevant *actors*, *outcomes*, and *related topics* that are expected to operate within the domain of interest, the salient *associations* that may exist between entities and events in the scenario, and the specific *timeframe* and *location* that bound the scenario in space and time. We assume the set of sub-topics mentioned in the general background and the scenario can be used together to define a topic structure that will govern future interactions with the QA system. In order to model this structure, the topic representation that we create considers separate *topic signatures* for each sub-topic.

GENERAL BACKGROUND

Serving as a background to the scenarios, the following list contains subject areas that may be relevant to the scenarios under examination, and it is provided to assist the analyst in generating questions.

- 1) Country Profile
- 2) Government: Type of, Leadership, Relations
- 3) Military Operations: Army, Navy, Air Force, Leaders, Capabilities, Intentions
- 4) Allies/Partners: Coalition Forces
- 5) Weapons: Chemical, Biological, Materials, Stockpiles, Facilities, Access, Research Efforts, Scientists
- 6) Citizens: Population, Growth Rate, Education
- 7) Industrial: Major Industries, Exports, Power Sources
- 8) Economics: Growth Domestic Product, Growth Rate, Imports
- 9) Threat Perception: Border and Surrounding States, International, Terrorist Groups
- 10) Behaviour: Threats, Invasions, Sponsorship and Harboring of Bad Actors
- 11) Transportation Infrastructure: Kilometers of Road, Rail, Air Runways, Harbors and Ports, Rivers
- 12) Beliefs: Ideology, Goals, Intentions
- 13) Leadership:
- 14) Behaviour: Threats to use WMDs, Actual Usage, Sophistication of Attack, Anecdotal or Simultaneous
- 15) Weapons: Chemical, Biological, Materials, Stockpiles, Facilities, Access

SCENARIO: Assessment of Egypt's Biological Weapons

As terrorist Activity in Egypt increases, the Commander of the United States Army believes a better understanding of Egypt's Military capabilities is needed. Egypt's biological weapons database needs to be updated to correspond with the Commander's request. Focus your investigation on Egypt's access to old technology, assistance received from the Soviet Union for development of their pharmaceutical infrastructure, production of toxins and BW agents, stockpiles, exportation of these materials and development technology to Middle Eastern countries, and the effect that this information will have on the United States and Coalition Forces in the Middle East.

Figure 22. Example of a Complex Question.

In our experiments, users asked a total of 200 complex questions. For each of the complex questions we evaluated the accuracy and completeness of the expected answer type, of the intentional structure and of the implied coercions.

5.1 *Recognition of Answer Structures, Predicate-Argument Structures and Question Topic*

As it was reported in (Moldovan *et al.*, 2003), the ability to recognize the expected answer type has the largest impact on the performance of a QA system operating on factoid questions. In our results, we have found that when processing complex questions, the recognition of the answer structures is also very important.

The experiments allowed the comparisons of the derivation of the complex answer types based on the user-centric procedure and the knowledge-centric method. The comparison is performed by counting the common concepts in the semantic signatures obtained by the two methods. To assess the precision, we assemble the set A of all answers containing a concept from the answer signature or matched by a pattern generated by the concept from the signature. The precision P is given by the fraction of answers from A which are deemed relevant from the total number of answers in A . The recall R is measured by the fraction of the number of answers from A which are relevant from the total number of answers which are relevant, regardless of them having any matching of the semantic signature of the expected answer. Since all the relevant answers are hard to find, we consider only those answers returned by the QA system and the answers available from the intentional structure. The precision, recall and F1-score are illustrated in Table 4. The F1 measure is computed as $F1 = \frac{2P \times R}{P + R}$.

Table 4. Results of Identification of Answer Structures.

Identification of Topic and Focus Signatures		<i>P</i>	<i>R</i>	<i>F1</i>
	Topic	82%	73%	77.2%
	Focus	80%	69%	74.1%
Identification of Answer Structure		<i>P</i>	<i>R</i>	<i>F1</i>
	Overlapped	78%	75%	76.5%
	User-Centric	63%	52%	57.0%
	Topic-Centric	68%	44%	53.4%

The recognition of the question topic and question focus was based on the result of shallow parsing based on FrameNet, as well as the acquisition of topic signatures with the methods presented in Section 2.2. To evaluate the precision and recall of the Topic and Focus signatures, three linguists have inspected the signatures and decided whether (a) the presence of each semantic descriptor (topic theme label) from the signature was acceptable; and (b) if new descriptors should have been discovered. The decision was made by having access to (*i*) the concepts and

relations associated with the descriptor; and (ii) the questions generated for each descriptor. The precision measured the number of descriptors deemed acceptable by a majority of the linguists out of all the descriptors discovered. The recall measured the fraction between the accepted descriptors and the number of descriptors which were newly proposed or accepted by the linguists. The results are listed in Table 4.

5.2 Evaluating the Intentional Structure

The intentional structure plays an important role in processing the intentions of a question because (1) it determines the dominance relations that parallel the relations in the complex answer structure; (2) it generates decomposed questions based on the topic and focus signatures; and (3) it determines whether further implications can be processed. To evaluate the intentional structure, we have evaluated only the quality of the decomposed questions and their impact on the QA accuracy. By evaluating the efficiency, effectiveness, and user satisfaction we were able to find that, due to the predictive questions originating from the intentional structure, the QA process was more efficient since users could find more relevant information by asking fewer questions. When predictive questions were available in experiments E1 and E2, users submitted an average of 12.25 questions each session. When no predictive questions were available (E3), users entered a total of 44.5 questions per session. Table 5 lists the number of predictive questions selected by users and the number of user questions entered by users during the 8 scenarios considered in E1 and the two questions considered in E2. In E2, freed from the task of writing a research report, users asked significantly fewer questions and selected fewer predictive questions than they did in E1.

Table 5. Efficiency of Dialogues in Experiments E1 and E2.

Complex Question Regarding the Bio-Chemical Program of:	Nr. Users	Nr. Predictive Questions Accepted (avg.)	Nr. Predictive Questions from the Intentional Structure	Nr. Predictive Questions from Metonymy Coercions	User Q (avg.)	Total (avg.)
Experiment 1 (E1)						
India	2	21.5	10.0	2.0	13.0	34.5
Lybia	2	12.0	7.5	1.5	9.0	21.0
Iran	2	18.5	9.0	1.0	11.0	29.5
North Korea	2	16.5	7.5	1.0	7.5	34.0
Pakistan	2	29.5	16.0	2.5	15.5	45.0
South Africa	2	14.5	8.0	2.0	6.0	20.5
Russia	2	13.5	5.5	1.0	15.5	29.0
Egypt	2	15.0	7.7	1.5	20.5	35.5
TOTAL (E1)	16	17.63	8.88	1.56	12.25	29.88
Experiment 2 (E2)						
Russia	24	8.2	4.2	0.7	5.5	13.7
Egypt	24	10.8	5.0	0.9	7.6	18.4
TOTAL (E2)	48	9.50	4.60	0.80	6.55	16.05

We also noticed that the dominance order between questions impacts the quality of the answers presented to a user. The dominance order also depends on the approach used for finding relevant documents for each element from the topic or focus signature. The questions generated for the intentional structure were submitted to human assessors who annotated as relevant or irrelevant each of the generated questions. Aggregated scores are presented for two scenarios in Table 6 which shows the results obtained when each of the four methods for finding relevant documents presented in Section 2.3 were used.

Table 6. Quality of Questions from the Intentional Structure.

Scenario:	Egypt's Biological Weapons		Russia's Nuclear and Biological Programs	
Approach	% of Top 5 Responses Relevant to User Question	MRR	% of Top 5 Responses Relevant to User Question	MRR
Approach 1	40.01%	0.295	60.25%	0.310
Approach 2	36.00%	0.243	72.00%	0.475
Approach 3	44.62%	0.271	60.00%	0.297
Approach 4	68.05%	0.510	68.00%	0.406

5.3 Evaluation of Metonymic Coercions

Some of the predictive questions used in our experiments were generated because of metonymic coercions. For name coercions we were able to evaluate the precision and recall of the interpretations on the annotated data. We trained on 80% of that total of 1970 annotations and tested on 20% of them. The precision was 92% and the recall was 87%. More important was the evaluation determined by the user selected validity of the questions generated from coercions. In the case of name coercions, the users marked as valid 68% of the questions, but selected only 15% of them. The two methods for coercing troponymic metonymies perform quite differently. For predicate coercions, obtained with the first technique presented in Section 4.1, 36% of the questions were validated, but 72% were selected. The second method produced only 14% valid questions and less than 2% of the questions were selected. This is in part due to the fact that the second method was employed on a much smaller corpus than the one used in (Lapata and Lascarides, 2003). Among the questions originating in the intentional structure, 61% of them were marked as valid, and 75% were selected.

6. CONCLUSIONS

This chapter presented evidence that the recognition of intentions and implicatures in questions can boost the performance of QA. Part of it is due to the fact that complex questions are characterized by complex answer types that display a

semantic structure that can be derived either by a user-centric or a knowledge-centric procedure. The expected answer structure is different from the question topic and the question focus. The former is useful for finding candidate answers, regardless of the question intentions, while the latter participates in the recognition of the intentional structure. The intentional structure and the implications recognized in complex questions enable the generation of predictive questions. Furthermore, implied questions are generated by coercing metonymies. Experiments that employ the intentional structure and the metonymic coercions by giving access to predictive questions have shown how QA results, measured in terms of efficiency, efficacy and user satisfaction, are enhanced.

7. REFERENCES

- ACE, "Entity Detection and Tracking" – Phase 1; EDT and Metonymy Annotation Guidelines, Version 2.5, 2002.
- Collin, F. Baker, Charles, J. Fillmore, and John, B. Lowe, "The Berkeley FrameNet Project" in Proceedings of COLING/ACL '98:86-90, Montreal, Canada, 1998.
- Cosmin, Bejan, Alessandro Moschitti, Paul Morarescu, Gabriel Nicolae and Sanda Harabagiu, "Semantic parsing based on FrameNet". In Proceedings of Senseval-3: The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text, 2004.
- Daniel, M. Bikel, Richard Schwartz and Ralph M. Weischedel, "An Algorithm that Learns What's in a Name", Machine Learning Journal 34, pp. 211-231, 1999.
- Jean Carletta, "Assessing agreement on classification tasks: The kappa statistic". Computational Linguistics, 22(2): 249-254, 1996.
- Sahibsingh, A. Dudani. "The Distance-Weighted k-Nearest-Neighbor Rule". In IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(4): 325-327, 1976.
- Christiane Fellbaum, editor, "WordNet: An Electronic Lexical Database", MIT Press, 1998.
- Donghui Feng and Eduard Hovy, "Handling Biographical Questions with Implicature". In Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-2005), pages 596-603, 2005.
- Elena Filatova and John Prager, "Tell Me What You Do And I'll Tell You What You Are: Learning Occupation-Related Activities for Biographies". In the Proceedings of HLT/EMNLP 2005.
- Daniel Gildea and Daniel Jurafsky, "Automatic Labeling of Semantic Roles" in Computational Linguistics, 28(3): 245-288, 2002.
- Daniel Gildea and Martha Palmer, "The Necessity of Parsing for Predicate Argument Recognition" in Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL 2002): 239-246, Philadelphia, PA, 2002.
- Roxana Girju, Adriana Badulescu and Dan Moldovan, "Learning Semantic Constraints for the Automatic Discovery of Part-Whole Relations". In Proceedings of HLT-NAACL 2003.
- Paul, H. Grice, "Logic and conversation", in Cole P. and J.L. Morgan (ed.) Academic Press N.Y. editors, Syntax and Semantics Vol. 3: Speech Acts, pages 41-58.
- Barbara Grosz and Candace Sidner, "Attention, Intentions and the Structure of Discourse", Computational Linguistics, 1986.
- Sanda Harabagiu, Andrew Hickl, John Lehmann, and Dan Moldovan, "Experiments with Interactive Question-Answering". In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05), 205-214, 2005.
- Sanda Harabagiu, "Incremental Topic Representations", in Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), Geneva, Switzerland, 2004.
- Sanda Harabagiu, Steven Maiorano, Alessandro Moschitti and Cosmin Bejan, "Intentions, Implicatures and Processing of Complex Questions", In Proceedings of HLT-NAACL 2004 Workshop on Pragmatics of Question Answering, Boston, Massachusetts 2004.

- Sanda Harabagiu, Dan Moldovan, Christine Clark, Mitchell Bowden, John Williams and Jeremy Bensley, "Answer Mining by Combining Extraction Techniques with Abductive Reasoning". In Proceedings of the Twelfth Text Retrieval Conference (TREC 2003), 2003.
- Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu, "The Role of Lexico-Semantic Feedback in Open-Domain Textual Question-Answering". In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL '01), 2001.
- Sanda Harabagiu, Marius Pasca and Steven Maiorano, "Experiments with Open-Domain Textual Question Answering" in Proceedings of COLING-2000, Saarbrücken Germany, pages 292-298, August 2000.
- Sanda Harabagiu, George Miller and Dan Moldovan, "WordNet 2 – A Morphologically and Semantically Enhanced Resource," in Proceedings of SIGLEX-99, pages 1-8, 1999.
- Marti Hearst, "TextTiling: Segmenting text into multi-paragraph subtopic passages". In Computational Linguistics, 23: 33-64, 1997.
- Marti Hearst, "Multi-Paragraph Segmentation of Expository Text". In Proceedings of the 32nd Meeting of the Association for Computational Linguistics, pages 9-16, 1994.
- T. Joachims, "Making large-Scale SVM Learning Practical". In B. Scholkopf, C. Burges and A. Smola (ed.), MIT-Press., Advances in Kernel Methods - Support Vector Learning, 1999.
- Megumi Kameyama, "Recognizing Referential Links: An Information Extraction Perspective". In Workshop of Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts, (ACL-97/EACL-97), pages 46-53, 1997.
- Lauri Karttunen, "Implicative Verbs". Language 47 340-58, 1971.
- Boris Katz, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Wesley Hildebrandt, Roni Katzir, Jimmy Lin, Daniel Loreto, Gregory Marton, Federico Mora, Ozlem Uzuner, "Answering multiple questions on a topic from heterogeneous resources". In NIST Special Publication SP 500-261: The Thirteenth Text REtrieval Conference (TREC 2004), 2004.
- Paul Kingsbury, Martha Palmer, and Mitch Marcus, "Adding Semantic Annotation to the Penn TreeBank" in Proceedings of the Human Language Technology Conference (HLT 2002):252-256, San Diego, California, 2002.
- Jon Kleinberg, "Authoritative sources in a hyperlinked environment". In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, pages 668-677, 1998.
- John Lafferty, Fernando Pereira, and Andrew McCallum, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data". In International Conference on Machine Learning (ICML'01), 2001.
- Maria Lapata and Alex Lascarides, "A Probabilistic Account of Logical Metonymy", Computational Linguistics 29:2, 263-317, 2003.
- Beth Levin, "English Verb Classes and Alternations a Preliminary Investigation". University of Chicago Press, 1993.
- Stephen, C. Levinson, "Pragmatics". Cambridge University Press, 1983.
- Chin-Yew Lin and Eduard Hovy, "The Automated Acquisition of Topic Signatures for Text Summarization", in Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000), Saarbrücken, Germany, July 31- August 4, 2000.
- Dekang Lin and Patrick Pantel, "Discovery of Inference Rules for Question Answering", Natural Language Engineering, 7(4): 343-360, 2001.
- S. Lytinen and N. Tomuro. "The Use of Question Types to Match Questions in FAQFinder". In Papers from the 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases, pages 46-53, 2002.
- Chris Manning and Hinrich Schütze, "Foundations of Statistical Natural Language Processing", MIT Press. Cambridge, MA, 1999.
- Gideon S. Mann, David Yarowsky, "Multi-Field Information Extraction and Cross-Document Fusion", In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05), 483-490, 2005.
- Katja Markert and Malvina Nissim, "Metonymy resolution as a classification task". In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 204-213, 2002.
- Andrew McCallum. "Mallet: A machine learning for language toolkit". www.cs.umass.edu/~mccallum/mallet, 2002.

- A. Meyers, R. Reeves, C. Macleod, R. Szekely, V. Zielinska, B. Young and R. Grishman, "The NomBank Project: An Interim Report". In Proceedings of HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation, 2004.
- Dan Moldovan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu, "Performance issues and error analysis in an open-domain question answering system". In ACM Transactions on Information Systems (TOIS), Vol. 21, Nr. 2, 133-154, 2003
- Paul Morarescu and Sanda Harabagiu, "NameNet: a Self-Improving Resource for Name Classification". In Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC-2004), 2004.
- Alessandro Moschitti and Cosmin Adrian Bejan, "A Semantic Kernel for Predicate Argument Classification", In Proceedings of CoNLL-2004, Boston, Massachusetts 2004.
- Srinivas Narayanan and Sanda Harabagiu. "Question Answering Based on Semantic Structures". In Proceedings of the 20th COLING Conference, 2004.
- Marius Pasca and Sanda Harabagiu, "High Performance Question/Answering". In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2001), September 2001.
- Dragomir, R. Radev, John Prager, and Valerie Samm, "Ranking potential answers to natural language questions" in Proceedings of the 6th Conference on Applied Natural Language Processing, pages 150-157, Seattle, WA, May 2000.
- Deepak Ravichandran and Eduard Hovy, "Learning Surface Text Patterns for a Question Answering System", in Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002), pages 41-47, July 2002.
- S. Sekine, K. Sudo, and C. Nobata, "Extended named entity hierarchy". In Proceedings of LREC 2002.
- Rohini Srihari and Wei Li, "A Question Answering System Supported by Information Extraction" in Proceedings of the 6th Conference on Applied Natural Language Processing, pages 166-172, 2000.
- David Stallard, "Two kinds of metonymies" in Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics, ACL-93, pages 87-94, 1993.
- Mihai Surdeanu, Sanda Harabagiu, John Williams, and Paul Aarseth, "Using Predicate-Argument Structures for Information Extraction". In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'03), 2003.
- Bonnie Webber, Arvind Joshi, Mathew Stone and Alistair Knott, "Anaphora and Discourse Structure", Computational Linguistics, 2003.
- Ludwig Wittgenstein, "Philosophical Investigations", Oxford University Press, London, England, 1953.
- Jinxi Xu, Ana Licuanan and Ralph Weischedel, "TREC2003 QA at BBN: Answering Definitional Questions". In NIST Special Publication 500-255: The Twelfth Text Retrieval Conference (TREC 2003), 2003.
- Roman Yangarber, "Counter-Training in Discovery of Semantic Patterns". In Proceedings of the 41th Meeting of the Association for Computational Linguistics, pages 343-350, 2003.
- Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen, "Automatic Acquisition of Domain Knowledge for Information Extraction". In Proceedings of the 18th COLING Conference, pages 940-946, 2000.

TOMEK STRZALKOWSKI, SHARON SMALL, HILDA HARDY,
PAUL KANTOR, WU MIN, SEAN RYAN, NOBUYUKI SHIMIZU,
LIU TING, NINA WACHOLDER AND BORIS YAMROM

QUESTION ANSWERING AS DIALOGUE WITH DATA

Abstract: We describe an interactive approach to question answering where the user and the system first negotiate the scope and shape of information being sought and then cooperate in locating and assembling the answer. The system, which we call HITIQA¹¹, has access to a large repository of unprocessed and unformatted data, and is additionally equipped with limited capabilities to search and navigate this set. The user asks questions in English, for example “What are the effects of pollution on commercial fishing on the Black Sea?” Mediating between the data and the user is a Dialogue Management System that attempts to make some sense of what the user is asking and what can be retrieved from the database, and then reconcile these through a dialogue. The purpose of the Dialogue Manager is to make the communication between the human and the machine possible and as efficient as it can be, but it does not necessarily imply a full understanding thus leaving the initiative fully in user’s hands. The HITIQA system has been designed primarily for information analysts who require answers to complex, analytical questions, not just finding simple facts.

1. INTRODUCTION

Question answering is of course a form of dialogue. In the simplest instance of this process, a question is posed by one party to the conversation and it is immediately answered by the other:

Q: Are you ready for lunch?

A: Sure, let's go!

In another setting, a simple question-and-answer information exchange may look like the following:

Q: What's the first flight out to DC?

A: 6:10 AM.

What makes such exchanges succeed has been explored by various researchers in linguistics and computational linguistics (e.g., Lehnert, 1987; Ferguson, 1998;

¹¹ HITIQA stands for High-Quality Interactive Question Answering.

Traum, 1994) and we are not going to discuss all these right now. Suffice it to say that something like the following must be at play:

1. The first party has some information need or another goal to achieve (change someone's belief state, make them do something, e.g., go for lunch)
2. He/She communicates this through a straightforward utterance, possibly in the form of a question.
3. The other party understands (hopefully) the request and its purpose.
4. He/She performs the requested action, which may involve searching through memory or through some help material (e.g., flight schedule) and communicating the answer.

Of course other things may happen in-between, such as misunderstandings, deceptions, ambiguities, etc., but let's just sweep them under the rug for now. In order to implement the four step process above in an automated system, we need the machine to understand the question well enough to be able to locate and return that bit of information requested by the user, or alternatively to act as expected. In other words, the system responding "6:10 AM" will be taken to mean actually: "the earliest flight on a regular weekday from where you are to DC is at 6:10 AM local time". This is how the answer is understood by the human user, whether or not the machine actually "means" it. Therefore, in order provide effective communication, the system must understand what the expected answer is, preferably *before* any search begins. This, however, requires a substantial amount of knowledge separate from the data repository itself, and that knowledge must be rich enough to make the subsequent search little more than a verification of a fact, in other words, the system needs to know what it is looking for.

The problem with this scenario is that most of the system's world knowledge is actually in that searchable database, and thus is not directly interpretable. To turn this data into usable knowledge requires a great deal of effort, the task that can never be quite completed. Requiring that QA system's performance depends primarily on having access to a rich *a priori* knowledge base would clearly limit the practical utility of this technology. We want a QA system to give us access to data we had no time or resources to process or process fast enough. This means that a very different approach is required. One intriguing possibility is to think of the QA system less as a self-reliant oracle but rather as an extension of its human user, a tool smart enough to exploit what the user knows in order to find what he does not. This is the basis of our approach to dialogue with data. However, before we explore this possibility further, let's digress a little into the area of human-computer dialogue and then work our way back to QA.

2. DIALOGUE WITH INFORMATION AND SERVICES

2.1 *Current Research in Human-Computer Dialogue*

The field of natural language human-computer dialogue is closely linked to the speech (or dialogue) acts theory, which postulates that speakers' utterances carry

embedded communication devices that manipulate the beliefs of the listeners. For example, Bunt (1997) developed theories of dialogue acts within what he called context change theory. These dialogue acts refer to the “functional units” used by a speaker to change the information context of the dialogue. Bunt’s theory makes a general distinction between utterances that accomplish a part of the desired transfer of factual information (known as task-oriented dialogue acts) and those that serve explicitly to control the dialogue (known as dialogue control acts).

Human-computer dialogue has been gaining importance as a technology area in recent years primarily as a result of advances in spoken language research. Being able to voice control robotic devices, navigate a phone directory, or order airline tickets through an automated travel agent gave an impetus to dialogue research within the speech community and beyond. In the U.S., DARPA’s¹² Communicator program has been instrumental in bringing about practical implementations of spoken dialogue systems. In Europe, similar programs include Verbmobil and more recently AMITIÉS, this latter being a cross-Atlantic collaboration between EU’s 5th Framework and DARPA. It is perhaps interesting to note that while the human-computer dialogue research has initially developed independently from spoken language technology (c.f. Allen, 1979; Bunt, 1997; Ferguson and Allen, 1998; Traum, 1994; Kobsa and Wahlster, 1989; etc.), it has come to be dominated by the practical and immediate nature of speech research (Gorin *et al.*, 1997; Ward and Pellom, 1999; Johnson *et al.*, 2002; Litman and Pan, 2002). It is thus partly due to the nature of spoken language applications being developed, that much of the human-computer dialogue work could be fairly characterized as a dialogue with information stored in a computer. Up until now, this information meant predominantly structured information, usually a relational database.

Consider, for example, an automated airline travel reservation system, or a banking transaction call center, or a technical support helpdesk. In all cases, the human operator navigates a database of items that potentially meet the caller criteria: the available flights, the credit card accounts, the spare parts catalogue, and so forth. This may be a little different in a robot navigating system, but even here the user commands must translate into internal instantiations which may be represented as multi-dimensional relations (Aist *et al.*, 2002; Dowding, 2003). The realisation that much of (though arguably not all) the human-computer dialogue can be rendered as a dialogue with structured data has profound consequences on the design and implementation of such systems in practice.

2.2 *The Functional Structure of Dialogue*

Before we proceed any further, let’s take a closer look at Bunt’s Dialogue Act Theory, which has been dominating human-computer dialogue research for some time now. One of the main consequences of this theory has been a formal description of dialogue as a series of functional moves (dialogue acts), which are ascribed to every utterance made by all parties to a conversation. The functional

¹² DARPA is the U.S. Defense Advanced Research Projects Agency.

(ordialogic) aspect of each utterance has to do with its role or purpose in the interchange. Statements, questions, answers, and expressions of thanks are examples of such functions, or dialogue acts. This has been best captured in a dialogue annotation system known as DAMSL¹³ (Allen and Core, 1997), which we will explain here in general terms, using an updated version developed for AMITIÉS project (Hardy *et al.*, 2003). The exposition in this section is intended to give the reader some perspective on what a dialogue is, what it consists of, and what rules of dialogue may exist, but it is not required to understand the remainder of the chapter.

DAMSL has been conceived as a hierarchical system of dialogue act classes laid out into four semi-orthogonal dimensions. These top-level classes include *Information-Level*, *Communicative Status*, *Forward-Looking Functions*, and *Backward-Looking Functions*. This way we can capture broad distinctions between task-oriented core of a dialogue, various unusual occurrences in conversations, as well as ways in which a particular utterance relates to previous or, alternatively, subsequent parts of the dialogue. The organization of dialogue acts within *Information Level* and *Communicative Status* is shown in Figure 1. The category *Information Level* allows us to make an abstract description of the content of each utterance; most utterances in a productive dialogue fall into this category. The *Communicative Status* category is reserved for rare situations where there are some external interruptions of the normal flow of dialogue, such as addressing a third party (“Stop that, kids!”). We will not be concerned with these here.

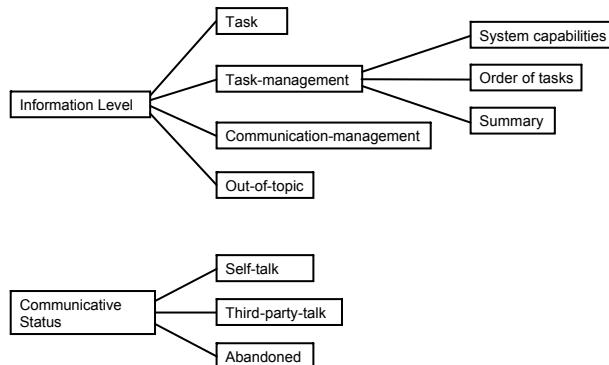


Figure 1. Hierarchy of sub-classes for *Information Level* and *Communicative Status*.

2.2.1 *Information Level Dialogue Acts*

Most of normal human-human dialogue is task oriented, that is, the parties have some goals (possibly implicit) and they attempt to achieve these through conversation. This is where all information exchange happens that affects the semantics of the dialogue. If, on the other hand, a party is not making direct progress

¹³ DAMSL stands for Dialogue Act Markup in Several Layers.

toward completing a task, but instead is talking about the *process* of doing the task, we assign such dialogue acts to *Task-management* category, e.g., statements such as “The task cannot be completed right now”, “Now we can start this task”, “First we’ll do this; then we’ll work toward that goal”, “Ok, that job is done”. Further subdivision is possible as illustrated in Figure 1. The *System capabilities* subcategory of Task-management means that the speaker is addressing problems that the service center can or can’t solve; this is the “competence domain” for customer service.

C: *Erm I'd like to make a payment please on my account*

A: *Certainly Mrs Smith but I'm sorry at the moment we are face, facing a technical problem.*

For *Order of tasks*, the speaker talks about the order in which tasks will be completed, or indicates that a task will be started. Typically these utterances take one of the following forms: “We’ll do this before this”, “Before I can do this task, I need to ask you some questions”, “Now we’ll do this”.

A: *Right, the next thing we're doing now is waiting for your bank to confirm for you*

A third category under Information-level serves to describe utterances that deal with social obligations, such as greetings, introductions, apologies, expressions of gratitude, farewells, as well as marks which maintain the conversation. Conventional phrases such as “hello” and “goodbye”, as well as backchannel words or non-words such as “uh-huh”, “yes”, and “ok” are examples of *Communication-management*. Ordinary sentences and phrases used to signal misunderstanding or to manage delays in the conversation should also be labelled as Communication-management.

Another Information-level category has to do with brief or extensive digressions from the task in the conversation. *Out-of-topic* includes jokes, non-sequiturs, small talk, and any comments or remarks that have no direct bearing on accomplishing the task. While these seemingly inconsequential dialogue acts are often ignored in design of human-computer dialogue systems, they do serve as powerful devices in effecting “naturalness” of conversation. Various Eliza-style¹⁴ “chatterbox” systems exploit this category successfully by creating human-like impression without actually conducting a productive dialogue.

A: *Thank you, you're speaking to Paul, good afternoon.*

C: *Uh hmm*

A: *Right so it's Paul and Pauline*

2.2.2 Communicative Status Dialogue Acts

Whereas most utterance in a dialogue can be assigned to *Information Level* class, *Communicative Status* category is intended to be used only in exceptional circumstances. We use three labels under this category: *Self-talk*, *Third-party-talk*,

¹⁴ ELIZA is a program developed by Weizenbaum (1966) that pretended a psychiatric patient and is said to have come close to passing the Turing Test. An annual Loebner Prize (Loebner, 2003) competition in New York is devoted to building a system that eventually will.

and *Abandoned*. A *Self-talk* utterance indicates that the speaker does not intend the other person to respond to or otherwise use what he is saying. The speaker's apparent intentions are the key, rather than whether or not the other person actually responds to or uses the utterance.

- A: *It's a whole year or more*
- C: *Gosh I must have paid other things by cash then*
- A: *Hmm*
- C: *I can't believe that*
- A: *There you go*

An utterance labelled *Third-party-talk* is one in which the speaker is addressing someone other than the second party to the conversation. A speaker abandons an utterance occasionally, when he makes an error or changes his mind about what to say. An utterance can be marked *Abandoned* whether or not the speaker was interrupted, as long as the speaker actually leaves his thought and does not return to it.

2.2.3 Forward-Looking Function Dialogue Acts

The next two classes of dialogue acts define the internal structure and progression of the dialogue independently from its content. Most dialogue research thus far has concentrated on designing and developing models of dialogue in terms of these functional units. Since these units are content-independent, the resulting structure is often seen as a “syntax” of dialogue, as opposed to its information content, or semantics. In fact DAMSL does not provide any guidelines how to link in semantics, beyond the largely superficial *Information-Level* class discussed above. Since we will address the issue of semantics directly in the next section, here we review only the functional structure of dialogue.

An utterance having a *Forward-Looking Function* anticipates the future in some way, having an effect on what is answered, discussed or done next. The speaker may be making a statement, asking a question, or committing to some course of action. He or she may be suggesting or directing the other party to do something. Forward-looking functions can be distinguished from backward-looking functions in that a forward-looking function typically elicit a response. Some functions, such as the various kinds of statements, as well as the Expression function, may have either a forward or a backward orientation, depending on the context. The hierarchy of Forward-Looking Functions is illustrated in Figure 2.

In general, a sentence or phrase that is a *Statement* can be said to be true or false. A statement makes a claim about the world, and tries to change the beliefs of the listener. We use four sub-classes under the category Statement: *Assert*, *Reassert*, *Explanation*, and *Re-explanation*. *Assertions* and *Reassertions* are ordinary statements, distinguished by whether or not the speaker has already made the claim in an earlier part of the dialogue. Included here are answers to yes/no questions, as well as statements that communicate some specific details, and also recapitulations, reformulations, or summaries.

In contrast to Assertions, which are simple statements, *Explanations* are reasons people give for their answers or for the questions they ask, or elaborations about topics at hand. Explanations and Re-explanations, similar to Assertions and Reassertions, are distinguished by whether the statement has been made previously in the dialogue.

A: *We can't in, no we can't increase the limit if you are already above it.*

Because obviously, well, it's just something we don't offer I'm afraid

C: *Alright okay thanks anyway*

Offers are implicit or explicit questions that, if answered in the affirmative or with some positive information, mean that the speaker will perform some action for the listener.

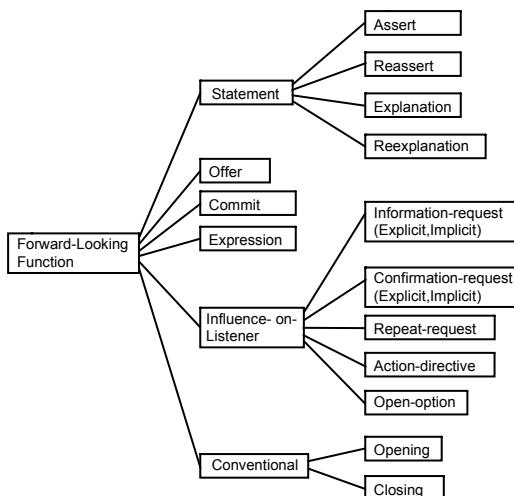


Figure 2. Hierarchy of sub-classes for Forward-Looking Function.

A: *Is there any other accounts that I need to deal with for you?*

C: *Erm actually erm do you also deal with erm accounts for River Island?*

A: *Yes*

The *Commit* category is used for utterances in which the speaker obligates himself or herself to perform a future action, phrased in such a way that the commitment is not contingent on the listener's agreement. A Commit may also be the response to an Action-directive. The *Expression* encompasses conventional phrases such as "Thank you", "I apologize", and "Sorry", exclamations, short words used to hold or grab the turn, such as "Right" or "Okay", and other expressive phrases. These may also be classified as Backchannel, Accept or Non-understanding, depending upon the context.

In the *Influence-on-listener* group of dialogue acts, the speaker is asking the listener a question, directing him or her to do something, or suggesting some course

of action the listener may take. A request for information, whether it is spoken in the interrogative form (Explicit), as in the following two examples, or the imperative or declarative form (Implicit), is tagged *Information-request*. We exclude from this category questions that call for a yes/no answer.

A *Confirmation-request* is an utterance that calls for the listener either to confirm or to deny the request or the question; in other words, it calls for a simple acceptance or rejection: a yes/no answer. In this category we also make a distinction between Explicit (illustrated below) and Implicit.

A: *And this is your Principles card isn't it?*

C: *Yeah*

The category *Repeat-request* is used to mark any request, whether it is for information or confirmation, that has been made earlier in the dialogue.

If the speaker directs the listener to perform some action, we label the utterance *Action-directive*. If the directive is issued to manage some delay in the conversation, for example, “Please wait” or “Bear with me”, then we also classify it as *Communication-management*.

C: *Oh hi I want to change my address please*

A: *Ok, just bear with me a second*

If a speaker suggests a course of action but puts no obligation on the listener, we classify the dialogue act as *Open-option*. The difference between Open-option and Offer has to do with who will perform the action. The Offer means that the speaker proposes to do something for the listener, as in “I can do this for you”, or “What can I do for you?” The Open-option, on the other hand, suggests that the listener or some other person perform the action. This type of utterance takes the form “You can do this”, or “This [option or course of action] can be done.”

A: *Yeah you surely can. You can pay over the phone using a debit card*

The *Opening* act indicates that the speaker is beginning the interaction by using a conventional social phrase to greet the listener, or by replying to such a greeting with a conventional phrase. Often the speaker, if he or she is a customer service representative, will identify the service name and/or the agent name as part of the greeting. These details are marked at the semantic level. The *Closing* act is used for turns in which the speaker utters a conventional social phrase or expression to finish or wrap up the conversation.

2.2.4 Backward-Looking Function Dialogue Acts

Utterances in the *Backward-Looking Function* category respond in some way to one or more previous turns in the dialogue. An answer to a question is one example of a common backward-looking function. If the speaker signals some level of understanding or not understanding what the previous speaker has said, one of the five speech acts in the Understanding sub-category applies. If the speaker signals some level of agreeing or disagreeing with the previous speaker's question (or some

degree of accepting or rejecting the previous speaker's proposal), then we apply a speech act from the Agreement sub-category. We should note that most, if not all, acceptances and rejections are also answers. The dialogue acts within Backward-Looking Function category are shown in Figure 3.

An *Answer* is a response to an Information-request or Confirmation-request. An answer by definition will always be an assertion, as it provides information or confirms a previous supposition, and it makes a claim about the world.

An *Understanding* response to an utterance reveals whether and in what way the speaker heard and understood what the other speaker was saying. This aspect indicates nothing about whether the speaker accepts or rejects what was heard. Because a speaker may be indicating understanding and agreement at the same time, choices in both areas can be appropriate. There are five speech acts in the Understanding category: Backchannel, Repeat-rephrase, Completion, Non-understanding and Correction.

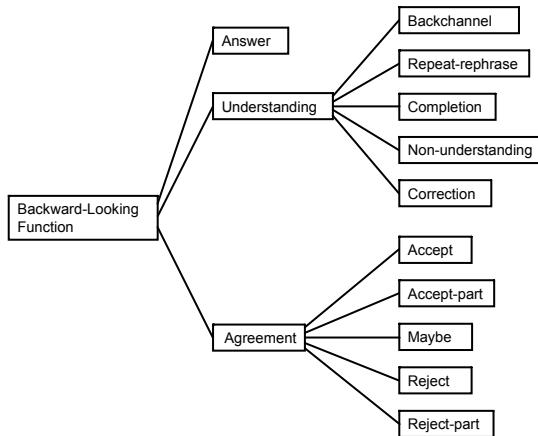


Figure 3. Hierarchy of sub-classes for Backward-Looking Function.

A *Backchannel* response is typically a short phrase such as “okay”, “yes” or “uh-huh”, indicating that the speaker heard and understood the previous utterance, but did not necessarily accept what he heard. A Backchannel utterance may be paraphrased “I heard you”, “I understand what you said”, “I heard that; please go on”, or “That's clear; you can continue”. This type of response may or may not interrupt the previous speaker, or it may occur while the other speaker is still speaking. The *Repeat-rephrase* label is used for utterances that repeat or paraphrase the previous speaker's words, to show that those words were understood but not necessarily accepted. If the speaker repeats or paraphrases some words with uncertainty, or a rising inflection (indicated in the transcription by a question mark), then we label such speech act as Non-understanding.

A: *Have you got erm? Have you got a card? It's got like a picture of a shop on the front*

C: *Picture of a shop, let's see if I can find any*

If the speaker has not understood or has partially understood something he has just heard, the *Non-understanding* speech act may apply. Utterances of this type can usually be paraphrased “What did you say?” “What did you mean?” or “Is this what you said?” Many of these examples can also be labeled Explicit Confirmation-request.

A: *Uh huh the name was Noble*

C: *Sorry?*

A: *N O B L E*

C: *N O B L E? I can't hear properly, on this phone it's on a very very low sound on it thank you very much*

The *Correction* act is used to indicate that the speaker has understood what the other person has said, but wants to correct a perceived error in the other person’s utterance. We reserve the label Correction for cases in which the speaker corrects the other person, not for cases of self-correction.

The set of dialogue acts in the *Agreement* category indicate whether the speaker accepts a proposal, offer or request, or confirms the truth of a statement or confirmation-request. We have five acts in this category: Accept, Accept-part, Maybe, Reject, and Reject-part. We mark an utterance as *Accept* or *Accept-part* if the speaker accepts all or part of the other speaker’s proposal or request; or if the information or claim conveyed in an *Assert* is accepted or confirmed. The *Maybe* act is when the speaker is uncertain of an answer, or says “I’ll have to think about it”, “I’m not sure”. We also use this label when the person cannot answer the question or address the proposal or offer because of lack of knowledge: “I don’t know”. We use *Reject* or *Reject-part* when the speaker disagrees, rejects a proposal or offer, says he will not comply, or says that all or part of the claim or the information conveyed by the other speaker is incorrect.

A: *Morning customer services, could I take the account number?*

C: *I can't, err unfortunately I'm blind*

3. THE AMITIÉS DIALOGUE SYSTEM

We now return to the discussion of the principles for designing a dialogue with structured data using as an example the AMITIÉS project. This project, currently in its fourth year, is co-funded by the European Commission and U.S. DARPA and involves several academic and industrial research organizations on both sides of the Atlantic¹⁵. AMITIÉS project is developing automated human-computer dialogue technology based on a sizable body of conversations between service agents and their customers recorded at large commercial call centers. The subject matter of these dialogues varies from one service to another and includes banking and

¹⁵ AMITIES partners and affiliates include University of Sheffield (UK), University at Albany (US), LIMSI-CNRS (FR), Duke University (US), VECSYS (FR), AGF (FR) and IBM (UK).

brokerage transactions, hardware and software support and insurance transactions. Recorded data comes in three languages, with primary concentration in English and French¹⁶.

The key scientific objective of AMITIÉS is synthesis of dialogue models from human-human dialogues, which are real, purposeful and task oriented (Walker, 2000). These models are subsequently used to develop on-demand robust human-machine dialogue systems capable of handling mixed initiative and spontaneous conversation styles, typical of human conversation. In order to enable this process, one needs to modify and expand the functional dialogue annotation scheme (DAMSL, discussed in the previous section) by adding a semantic layer to it. This is a significant extension because it allows us to tie functional dialogue moves such as “question” or “offer” or “acceptance” to specific information exchange transactions, such as “filling in a missing attribute” or “resolving ambiguous interpretations”, or “supplying value for an attribute” etc. Since these information exchange transactions are manipulations of structured data objects, we can characterize the dialogue as a bridge between the data and the user.

The key concept underlying AMITIÉS dialogue manager is therefore the notion of dialogue with data. This interpretation is further facilitated by the prevalent type of communication in the call center environment, which is information seeking dialogue. In a human-operated call center, an operator mediates between the caller and a variety of data sources: information about customers, products, regulations, etc. Much of this data is in a structured form, usually a relational database (accounts information), while some may remain in semi-structured form (e.g., text memos, flyers, regulations manuals.) The objective of an *automated* call center is to effect a natural and efficient interaction between the caller and the information he or she is seeking, and do so in a way which is as close to human-human conversation as possible.

The automated call center scenario applies to many customer service situations, including the following:

- Financial services (AMITIÉS primary domain)
- Product support (AMITIÉS secondary domain)
- Travel reservations
- Product and service ordering

In all cases, the objective is to locate, insert or update a single (or several) data object in a structured data base. For example, a service technician calling a support line (PARTS-ID) describes the spare part he needs to order, while the system narrows the matching parts down to one (Bagga, Strzalkowski and Wise, 2000):

CALLER: *I need a part check for PCB control.*

PARTS-ID: *Which product is this for?*

CALLER: *COMPAX 40E*

PARTS-ID: *Do you want PCB Display SID or PCB Control?*

CALLER: *Control*

¹⁶ German is the third language in AMITIES.

We may note that the dialogue continues until the unique item has been identified matching the caller's criteria. A similar process is used to determine caller's identity in a banking call center dialogue:

AGENT: *Good afternoon customer services Paul speaking*
CALLER: *Hello I want to inform you of my change of address please*
AGENT: *Certainly what name is it please?*
CALLER: *Name? It's Mrs. Pauline Smith*
AGENT: *Could you confirm your previous postcode and telephone number please?*
CALLER: *Yeah the previous one was AB1 2CD and 01234 565789*

At a more abstract level, these dialogues can be characterized as *Interaction with Structured Data* (ISD). ISD consists of the following components:

1. *Data structure*, which defines the set of basic entities (accounts, spare parts, flights) and their attributes (account number, part size, destination city, etc.) as well as methods for identifying references to these attributes in user statements.
2. *List of basic transactions* supported by the service (account payment, address change, locating a flight) along with methods to detect references to these transactions.
3. *Dialogue models* for handling various conversational situations in human-like fashion (e.g., responding to requests, emotions, indecision) and consistent with the character of the service (polite, helpful, caring).
4. Optional *dialogue meta-strategy* as required to address privacy and security concerns (e.g., positive caller identification must precede exchange of any sensitive information.)

The components 1, 2 and 4 can be built using limited amount of static data about the target service. This information, which is to a large degree domain-independent or at least domain-adaptable, is sufficient to design basic mixed-initiative dialogue capabilities, as explained further. Although the resulting dialogue may not feel very "human-like" it will be quite efficient, giving the user a broad initiative to conduct it as they wish. Dialogue models (component #3) are required to create an illusion of naturalness and these can only be derived from large corpora of actual conversations. Large corpora of real conversations are also needed to develop speech and prosody models.

The prototype AMITIÉS system that can handle basic banking transactions is now available as a demonstrator. Its current capabilities include call triaging (i.e., determining the purpose of the call), customer identification, and execution of several simple transactions such as address change, making a payment, reporting lost credit card, etc. It is built around Galaxy Communicator architecture (Seneff *et al.*, 1998) in a standard configuration shown in Figure 4. The Dialogue Manager can handle dialogues in three European languages, and can additionally switch from one language to another in mid-conversation.

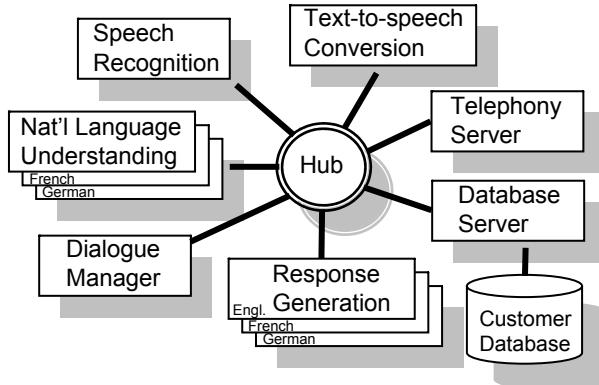


Figure 4. AMITIÉS System Architecture.

We now explain some key principles of designing a (conversational) Interaction with Structured Data (ISD). The overall strategy is to locate an item or items in the database that meet a number of specific user-defined criteria, for example, the most convenient flight, the caller's bank account, etc. For complex tasks, this overall objective is broken down into a set of sub-goals, some of which may need to be satisfied in order to achieve the final objective. The role of ISD dialogue is to chart a path through the sub-goals in such as way that:

1. the objective is achieved
2. any partial constraints on the order or selection of the sub-goals are met, and
3. the most efficient route is chosen.

The dialogue manager identifies a possible goal (or goals) of the conversation, which is usually a transaction of some sort (CHANGE-ADDRESS; LOCATE-FLIGHT, etc.) represented by a typed template with attributes (NAME, TELEPHONE, DESTINATION, etc.). Before such transactions can be executed, the values of their attributes need to be determined, and at least some of them will have to come from the user. The overall mechanism works in such a way that the system interacts with the user while filling attribute values in templates representing goal transactions as well as any sub-goal transactions that may emerge. Spontaneous conversation works in this environment, because values may be filled in any order, or several values may be supplied in one turn. As attribute values in the templates are filled, the need for dialogue decreases.

The system sets key milestones or sub-goals to be reached by gathering sufficient information from the customer, but these milestones may be approached by a variety of different paths. For example, in a banking dialogue, if the customer's last name is misrecognized (no database match), or if multiple database records returned (multiple customers with the same name), the system will ask for a different attribute, such as the address or postal code. The process continues until a unique (e.g., bank account) or best (e.g., a flight) record is identified. Thus the

dialogue system has flexibility to deal with the user input arriving in any order or form and the input that is not completely captured (i.e., allowing for speech recognition errors), without getting stuck on any single attribute. The paths to the key milestones, and even the order of the milestones, may be seen as a series of hidden transitions (Figure 5).

We should note that much of the information seeking dialogue is actually a series of question and answer exchanges, some of which may be more straightforward than others. In the structured data environment, these question-and-answer exchanges are only interesting as part of a larger dialogue, and understandably, research focused on the models of the interaction rather than the mundane information exchange mechanics. In the structured data environment, locating information is trivial by definition: the data structures are designed to represent all the salient semantic attributes of the domain¹⁷.

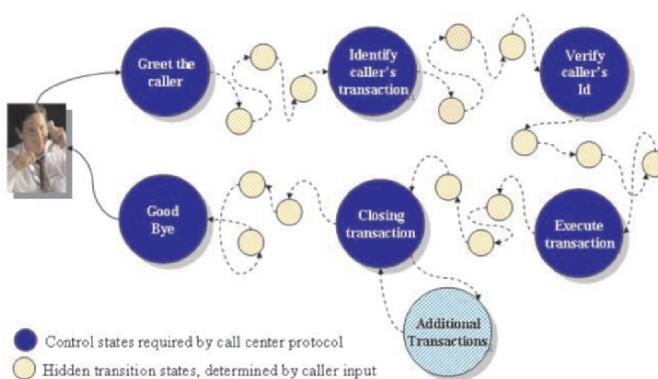


Figure 5. Human-Computer Dialogue as a Network of Hidden Transitions.

This situation changes dramatically when the structured data does not exist, or when the data structure does not correspond to the domain structure. In the latter case we need to find a way to *adapt* the data structure so that it can support the dialogue; in the former case, the structure has to be created from scratch. In the following sections we outline an approach to designing a dialogue system that can work with unstructured data, keeping in mind the ISD strategy outlined above. In the unstructured data environment finding relevant information is no longer a trivial matter, therefore the semantic aspects of dialogue, and consequently its question-answering core, become central issues. This is the focus of the HITIQA project at SUNY Albany and Rutgers University.

¹⁷ This of course does not apply to just any structured database, just those designed specifically to support dialogue systems.

4. DIALOGUE WITH UNSTRUCTURED DATA

HITIQA project (Small *et al.*, 2003; Tang *et al.*, 2003) is part of the ARDA AQUAINT program that aims to make significant advances in the state of the art of automated question answering. In this chapter we discuss two aspects of our work in particular:

1. Handling of question semantics: or how the system “understands” the user.
2. Human-computer dialogue: or how the user and the system negotiate this understanding.

These two open problems form the dual lines of attack into the unstructured data territory from our ISD base.

We now return to the point made earlier that in order to do question answering well from an unstructured data set, one needs a significant amount of knowledge to understand the question and to locate the answer. Indeed, an ideal situation would be to find the answer in a structured knowledge base which has been derived from the available text data. Most of the contemporary “factoid” QA systems (see the next section) follow a variant of this strategy and their effectiveness is clearly related to the amount of structured knowledge available to the system (Harabagiu; Prager *et al.*, this volume). However, as stated before, our objective is to circumvent the costly process of knowledge formation (which may go as high as \$10K per page of text) and instead attempt to bring user’s own knowledge to bear upon the process of finding an answer. This can of course be accomplished through dialogue, and this is what we propose: develop capabilities for human *interaction with unstructured data* (IUD) and use this vehicle as a conduit to locate the answer to the user question. We may note that this scenario transcends a simple question-answer exchange, as it often requires a more complex interaction. For this reason, we concentrate our attention on questions for which such interactions will be justified, in other words, complex, analytical and exploratory questions that professional analysts typically pose.

5. QUESTION ANSWERING: FACTOID VS. ANALYTICAL

The objective in HITIQA is to allow the user to obtain answers to exploratory, analytical questions. There are very significant differences between factoid or fact-finding, and analytical question answering. A factoid question seeks a piece of information that would satisfy the corresponding statement, i.e., it would be accepted as a true statement in specific circumstances. For example: “How many states are in the U.S.” / “There are X = 50 states in the U.S.” In this sense, a factoid question usually has just one correct answer that can generally, be judged for its truthfulness. By contrast, for an analytical question the “truth” is just one of many properties of the answer, which may indeed be a matter of opinion and may depend upon the context in which the question is asked. Answers to analytical questions are rarely unilateral, indeed, a mere “correct” answer may have limited value, and in some cases may not even be determinate (“Which college is the best?”). Instead,

answers to analytical questions are often judged as helpful, or useful, or informative, rather than merely true. Indeed, “technically correct” answers may be considered as irrelevant or unresponsive. In most cases, analytical questions are too complex to be answered simply by a word or a phrase; instead they often require preparation of a report of some sort (“*How was South Africa’s chemical weapons program financed?*”) that pulls together information from multiple sources. Therefore, analytical questions should be considered as problem statements, which may be only partially specified thus leaving the analyst some room to decide how to approach the task. The answers to analytical questions thus must be judged for their completeness, compactness, usefulness, clarity, objectivity and reliability, in addition to “technical correctness” if such can be determined at all. We may note that these criteria make the QA process significantly distinct from document retrieval (Baeza-Yates *et al.*, 1999).

For factoid questions, the process of automated question answering is now fairly well understood. Most factoid questions display a distinctive “answer type”, which is the type of an information piece that needs to be supplied in order to answer the question. Recent automated systems for answering factoid questions deduct this expected answer type from the form of the question and a finite list of possible answer types. For example, “Who was the first man in space” expects a “person” as the answer, while “How long was the Titanic?” expects some length measure as an answer, probably in yards and feet, or meters. This “answer typing” approach is generally a very good strategy, which has been exploited successfully in a number of automated QA systems that appeared in recent years, especially in the context of TREC QA¹⁸ evaluations (Harabagiu *et al.*, 2002; Hovy *et al.*, 2000; Prager *et al.*, 2001).

Unfortunately, the above process is not easily applied to analytical questions. This is because the type of answer for analytical questions cannot always be anticipated due to their inherently exploratory character. In contrast to a factoid question, an analytical question can take a variety of syntactic forms with only a loose connection between their syntax and the expected answer. Given the practically unlimited potential for formation of analytical questions, it would be counter-productive to categorize them into a fixed number of question/answer types. Even finding a non-strictly factual answer to an otherwise simple question about Titanic length (e.g., “two football fields”) would push the limits of the answer-typing approach. Therefore, the formation of an answer should instead be guided by the topics the user is interested in, as recognized in the question and/or through the interactive dialogue, rather than by a single type as inferred from the question in a factoid system.

In the next section, we argue that the semantics of an analytical question is more likely to be deduced from the information that is considered relevant to the question than through a detailed analysis of its particular form. While this may sound circular, it needs not be. Determining “relevant” information is not the same as finding an answer; indeed we can use relatively simple information retrieval methods (keyword

¹⁸ TREC QA is the annual Question Answering evaluation sponsored by the U.S. National Institute of Standards and Technology www.trec.nist.gov. See also Voorhees’ chapter in this volume.

matching, etc.) to obtain perhaps 50, 100, or 200 “relevant” documents from a database. This gives us an initial material to work with in order to scope out the complexity of the answer, but we are nowhere near the answer yet. We now engage the user in a dialogue in order to better pinpoint the answer he or she is seeking. As it turns out, in order to coax a dialogue out of this initial document set we need to structure it somehow. This appears to fly in the face of our earlier claim that we can do without structured knowledge. Well, not quite; however, the structuring process that we propose is very different from an outright knowledge formation effort. The structures we build are automatically derived and lightweight *frames*, and they only capture enough knowledge to enable the system to *start* interacting with the user. The full architecture of HITIQA is given in Figure 6.

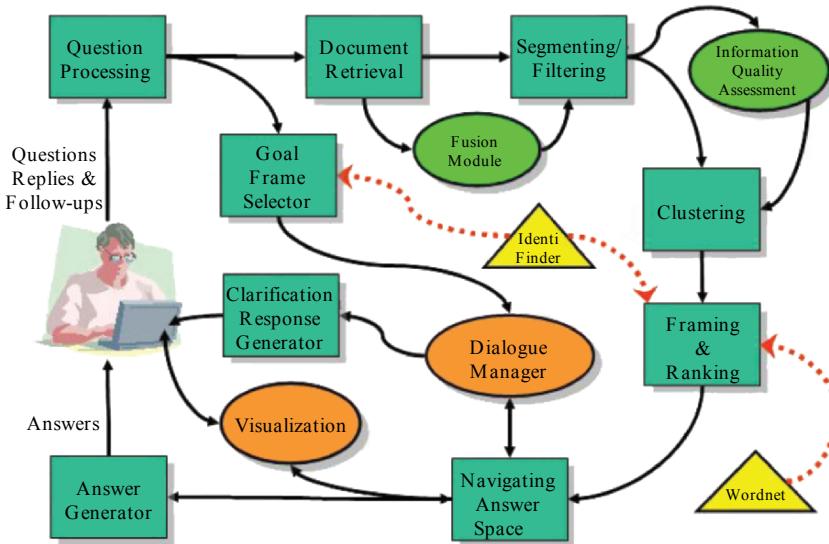


Figure 6. HITIQA System Architecture.

6. DOCUMENT RETRIEVAL

When the user poses a question to a system sitting atop a huge database of unstructured data (text files), we first need to reduce this big pile to perhaps a handful of documents where the answer is likely to be found. The easiest way to do it is to convert the question into a search query (by removing stopwords such as *a*, *the*, *in*, *at*, etc., and stemming and tokenizing other words) and submitting this query to a fast but non-exact document retrieval system, e.g., Smart (Buckley, 1985), or InQuery (Callan *et al.*, 1992), or if you are on the web, to Google, Yahoo, etc. Any number of phrases, names and collocations may be extracted from the question in an attempt to make the search more effective, provided that the search engine is capable

of utilizing these. For example, the question “*What are the effects of pollution on the fishing industry in the Black Sea?*” may yield such basic concepts as *Black Sea*, *pollution*, *fishing industry*, and perhaps related concepts such as *pollution control*, *international cooperation*.

An interesting open issue here is what the accuracy of the initial search should be. Clearly, if no elements of the answer are present in the top retrieved documents, no amount of further processing can produce the desired result, and the user may be led to believe, mistakenly, that there is no answer to be found, or that it does not exist, or worse of all, that the answer is negative (e.g., “*Does North Korea have an active nuclear weapons development program?*”). On the other hand, even a relatively low initial precision may be sufficient to bootstrap the final answer through a dialogue with the user. Our experiments show that more initial precision is better, as would be expected, but that capturing the diversity of the answer space is just as important. This means multiple aspects of relevant and related material need to be present. In other words, better precision matters, up to a point, but too much precision may not be desirable. In this sense, existing information retrieval systems that rank documents and other information objects by a degree of relevance (usually determined through a weighted vocabulary overlap) are quite suitable for the task: one only needs to estimate how far down the ranks have to be examined in order to obtain an optimal mixture of precision and diversity. Some of the rationale for this mix is discussed in the next section.

7. DATA DRIVEN SEMANTICS OF QUESTIONS

The set of documents and text passages returned from the initial search can be considered the first stab at understanding the user’s question by the machine. If no further processing were done, this *Retrieved Set* could indeed be considered to represent the meaning of the question as understood by the system at this point in time. If no further processing were done, the retrieved set could be returned to the user as a ranked list, IR-style. Thus, the better our search capabilities, the closer this set would be to what the user may accept as an answer to the question.

We should point out here that better search does not simply mean better precision or better recall; it means both and it also means utility, novelty and context. In other words, while new information has the highest value to the user, an independent confirmation makes it more reliable. On the other hand, the importance of relevant information is often relative to other, perhaps only marginally related information; such additional information provides a context or a backdrop for the current topic and gives the analyst a fuller picture of relative importance of the key facts. Therefore, the search needs to cast a net wide enough to capture more than just the most directly relevant pieces. Our intention is to obtain the Retrieved Set as a combination of on-target information, highly related contextual information, and possibly, some amount of noise (Figure 7).

The composition of each of these subsets is also important: we may want to know if, for instance, the on-target zone is fairly homogenous (i.e., all texts have very similar content), or whether there are a number of diverse topics represented

there, somehow tied together by a common thread. In the former case, the system may be reasonably confident that it has found the answer; in the latter case, the question appears to be more complex and a negotiation process with the user may be required.

In order to make use of the above scheme, we need to find a way to quantify the degree of fit between the question and the information in each of the three zones in Figure 7. This is accomplished through the framing process described below, the outcome of which is twofold. Firstly, the alternative interpretations of the question are distinguished and ranked. Secondly, salient concepts and attributes for each the resulting groups are extracted into the topic frames. This enables the system to conduct a meaningful dialogue with the user *about the information underlying these frames*.

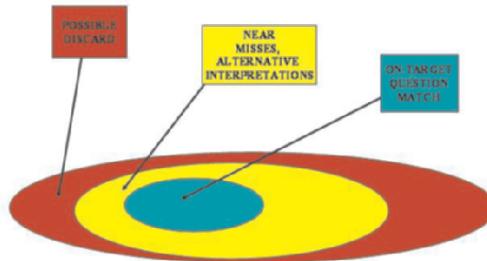


Figure 7. Answer Space Topology: the goal is to align the center ON-TARGET zone with user's expectations of the answer.

8. FRAMING TEXT DATA INTO LIGHTWEIGHT STRUCTURES

8.1 Clustering to detect salient themes

Information contained in the documents which form the Retrieved Set is first clustered to uncover any salient themes running through it. This classification is partially orthogonal to the relevance ranking used in document search because it is reinforced by mutual similarities of text fragments within the set (i.e., by repetition of the same theme or pattern) rather than by similarity to the question. While occasionally these two classifications do converge, more often they form a two-dimensional space of text passages that are on the one hand highly similar to the question and on the other hand are salient within the set because they show up in numbers.

Clustering is executed using a strong similarity metric based on overlap of word n-grams of varying length. Retrieved documents are first broken into passages, which are most often naturally occurring paragraphs, however, care is taken not to severe cohesive links running across paragraphs. The passages are then clustered using a two-step process, which is a combination of hierarchical clustering and n-bin

classification (Hardy *et al.*, 2002). The hierarchical clustering phase forms a small number of tightly connected clusters: typically three to six clusters are generated out of the top 50 documents, which may yield as many as 1000 passages, although only a small subset of these will enter the initial clusters as what we call *cluster seeds*. The goal here is to have each cluster represent a salient theme within the retrieved set: usually an alternative or complimentary interpretation of the user's question. This is of course not always possible, and some clusters may end up merely accidental aggregations of text snippets.

After the clusters are formed, a list of topic labels is assigned to each cluster. These are keywords and concepts derived from the question that are present in the cluster, as well as the n-grams that appear in most of the cluster seed passages. The latter form what we call the *cluster signature*. We also consult Wordnet (Miller, 1995) database for related terms, for example, "rifle" and "machine gun" are kinds of "weapon", which allows an indirect match between *weapon inspectors* and *a discovery of a cache of rifles and machine guns*. In general, rich lexical resources such as Wordnet and Extended Wordnet (Harabagiu *et al.*, 1999) greatly enhance the system's robustness and efficiency by shortcircuiting some "obvious" mismatches (and consequently potential misunderstandings). For example: is *import* a form of *acquisition?* (*Did Iraq attempt to acquire chemical weapon materials from Africa?*)

8.2 Framing imposes partial structures

We use a technique called *text framing* to impose a partial structure on text passages forming the clusters in the Retrieved Set. In very abstract terms, a *frame* represents an event or relationship of a certain kind. For example, a TRANSFER event may subsume all situations where an entity has changed location or ownership, presumably through some external force or process. This may cover events such as sales of controlled substances from one country to another, but also many other events: exports, shipments, acquisitions, etc. A frame contains attributes, which are the entities somehow involved in the event. In the simplest kind of a frame, which we call a *generic frame*, attributes form a loose set and are selected solely based on their own merit, that is, we select entities that are likely to play some role in the frame, although the system may be unable say what that role might be. Obvious candidates for such attributes include people, organizations, locations, times, and so forth. In addition, we attempt to pinpoint a central verb or relation in the passage, which is assigned to the special TOPIC attribute. This is often done with the help of the *cluster signature*, which is, as explained before, a list of terms and n-grams most frequently occurring in the cluster to which the passage belongs. Generic frames are the first layer of structure imposed on text, and they can be constructed using a good information extraction system. Information extraction technology has reached a very high level of performance accuracy following the Message Understanding Conference (MUC) evaluation series¹⁹. Therefore, generic frames can be built on

¹⁹ MUC (Message Understanding Conference) evaluations were sponsored by the U.S. DARPA in connection with various human-language technology programs, including the TIPSTER TEXT Program (1991 – 1998).

demand in most cases. Our experiments indicate that generic frames can support a reasonable level of interaction with the user; nonetheless, their utility is somewhat limited for our purposes. Figure 8 shows a generic frame extracted from a text about weapon trade.

Frame Type: Generic
Topic: imported
Location: Iraq, France, Israel, FRG
Organization: IAEA, Nukem
Person: Leonard Spector
Weapon: uranium, nuclear bomb
Dates: 1981, 30 November 1990, ..
<hr/>
Text: <i>The Bush Administration claimed that Iraq was within one year of producing a nuclear bomb. On 30 November 1990... Leonard Spector said that Iraq possesses 200 tons of natural uranium imported and smuggled from several countries. Iraq possesses a few working centrifuges and the blueprints to build them. Iraq imported centrifuge materials from Nukem of the FRG and from other sources. One decade ago, Iraq imported 27 pounds of weapons-grade uranium from France, for Osirak nuclear research center. In 1981, Israel destroyed the Osirak nuclear reactor. In November 1990, the IAEA inspected Iraq and found all material accounted for....</i>

Figure 8. A generic frame obtained from the text passage (not all attributes shown).

Our goal is to impose enough structure on the text so that it would allow for a systematic comparison among different pieces of text and also against the question. In particular, the framing process must reveal that some topics and themes within the retrieved set are not what the user had explicitly asked for. We refer to these items as the *near-misses*, the middle layer zone in Figure 7. These pieces of information may or may not be relevant, but they are likely highly correlated with the desired answer. The required comparison can be accomplished by aligning frame attributes from different clusters, for example, a cluster frame from the NEAR-MISS zone can be compared to the cluster frame from the ON-TARGET zone. When comparing two generic frames, it is difficult to ask the user about the outcome, unless we know the roles that the mismatched attributes play in the frames: for example, the LOCATION attribute may indicate a start or an end point, or a physical location, or government seat, and so forth. For example, if the user is interested in shipments of uranium from Niger to Iraq, and the system finds a shipment from, say, Libya, then we want to ask if “shipments from Libya” are relevant; however, if we do not know that Libya is a DESTINATION, we can only ask if “information about Libya” is of interest to the user. Such clarification questions are less effective at eliciting helpful cues from the user, and may cause decreased confidence in system’s abilities.

In order to do better, we need to know a little more about the roles that attributes play in the frames. We define a few slightly more specialized *typed frames* by assigning *roles* to selected attributes in the general frame. Some possible typed frames include: (1) a *Transfer frame* with three roles including SOURCE, DESTINATION and OBJECT; (2) a two-role *Relation frame* with AGENT and OBJECT roles; and (3) an one-role *Property frame*. These typed frames represent certain

generic events/relationships, which then map into more specific event types in each domain. Other frame types may be defined if needed, but we do not anticipate there will be more than a handful all together.²⁰ For example, another 3-role frame may be *State-Change frame* with AGENT, OBJECT and INSTRUMENT roles, etc.

Where the generic frame is little more than just a “bag of attributes”, the typed frames capture some internal structure of an event, but only to the extent required to enable an efficient dialogue with the user. Typed frames are “triggered” by appearance of specific words in text, for example the word *export* may trigger a *Transfer* frame. A single text passage may invoke one or more typed frames, or none at all. When no typed frame is invoked, HITIQA will attempt to identify the roles, e.g., SOURCE, DESTINATION, OBJECT, etc. This is done by mapping general frame attributes selected from text onto the typed attributes in the frames. In any given domain, e.g., weapon non-proliferation, both the trigger words and the role identification rules can be specialized from a training corpus of typical documents and questions. For example, the role-id rules rely both on syntactic cues and the expected entity types, which are domain adaptable.

Frame Type:	<i>WMD-Transfer</i>
TRF-Type (Topic):	<i>imported</i>
TRF-Dest (Location):	<i>Iraq</i>
TRF-Source (Location):	<i>France</i>
TRF-Object (Weapon):	<i>uranium</i>

Text: *The Bush Administration claimed that Iraq was within one year of producing a nuclear bomb. On 30 November 1990... Leonard Spector said that Iraq possesses 200 tons of natural uranium imported and smuggled from several countries. Iraq possesses ...*

Figure 9. A typed frame obtained from the same text passage (generic attributes shown in parentheses).

We have further adapted the typed frames to the *weapons of mass destruction non-proliferation domain (WMD)*, which is one of the sources of our data. Within this domain, typed frames represent specific types of events, such as transfer of weapons, or an arms treaty, or a production/development of weapon components, etc. For example the TRANSFER frame (now capturing all transfer events of weapons, technologies, components, etc.) assigns semantic roles to its attributes (e.g., source, destination, weapon type). While this may not capture the entire meaning of a text fragment, it nonetheless allows the system to make more “intelligent” comparisons, for example: one document may report that Egypt obtained missile components from North Korea (TRANSFER *missile-parts* FROM *N. Korea* TO *Egypt*), while another document may report that Egypt transferred weapon technology to Korea (TRANSFER *weapon-technology* FROM *Egypt* TO *Korea*). Figure 9 shows a WMD-domain

²⁰ We are currently working on effective frame acquisition methods. While classifications such as (Levin, 1993) or FrameNet (Fillmore, 2001) are relevant, we are aiming at semantically rather than syntactically based classification.

transfer frame – the reader may compare the generic frame built over the same text (Figure 8). Note that this frame is built on top of the generic frame, thus the generic attributes are still accessible.

Most of the topical frames need to be defined in advance of the QA process, however, dynamic frame expansion is also possible by generalizing salient elements in a cluster for which there is no ready-made attribute. Each of the attributes in a frame is equipped with an extractor function which specializes in locating and extracting instances of this attribute in running text. The extractors are implemented using a variety of information extraction utilities, including components of Sheffield's GATE²¹ system, as well as BBN's Identifinder software, which extracts 24 types of entities and have been used successfully in TREC QA system (Xu, *et al.*, 2002). These basic extraction capabilities are further extended by adding frame-specific extractors such as SOURCE (from where/whom?) and DESTINATION (to where/whom?) in the TRANSFER frame. When moving from one domain to another, or when the original domain is substantially enlarged, these extractors can be bootstrapped over text using an unsupervised learning method developed by Strzalkowski and Wang (1996). We are also investigating a possibility of automatically learning topical frame types for new domains; some preliminary work in this direction has been reported in (Yangarber, 2002; 2003).

At this time, the framing process resembles the template filling task in information extraction (cf. MUC evaluations), but there are also significant differences. The MUC template task was to fill in a single template using information from an entire document (Humphreys *et al.*, 1998). In the framing process, on the other hand, templates are filled by entities extracted from clusters of small size, very similar passages. The added redundancy of information makes this a less error-prone task because repetition strengthens the positive signals while suppressing the noise. In addition, each of the clustered passages may represent the same information in a different way, which again gives the extraction system more chances to get it right. Furthermore, clusters containing adjacent text passages may be framed independently and if their frames are of the same type, they can be merged while combining their attributes. We should note that selecting the right frame type for a passage is an important pre-condition to "understanding". The current version of HITIQA assigns only one frame to a given passage, however, we are considering an extension where multiple frame types may be assigned.

A very similar process is applied to the user's question, resulting in a *Goal Frame*, which can be subsequently compared to the data frames obtained from the clusters in the Retrieved Set. For example, the Goal Frame generated from the question, "*How has pollution in the Black Sea affected the fishing industry, and what are the sources of this pollution?*" is shown in Figure 10. We should note that this is a generic frame and the attributes have no special roles beyond their usual interpretation. Having a question represented as a generic frame is acceptable as

²¹ GATE is Generalized Architecture for Text Engineering, a natural language processing toolkit which includes, among others, an information extraction system. GATE has been developed at the University of Sheffield (Cunningham, 2000).

long as it can be matched against a typed frame in the retrieved set. This in fact allows for a greater flexibility in interpreting user questions.

Frame Type: Generic
Topic: pollution, industry, sources
Location: Black Sea
Industry: fishing
<hr/>
Text: How has pollution in the Black Sea affected the fishing industry, and what are the sources of this pollution?

Figure 10. HITIQA generated generic Goal Frame.

Frame Type: Generic
Topic: pollution
Alt-topic: sources
Location: Black Sea
Industry: fisheries, tourism
<hr/>
Text: In a period of only three decades (1960's-1980's), the Black Sea has suffered the catastrophic degradation of a major part of its natural resources. Particularly acute problems have arisen as a result of pollution (notably from nutrients, fecal material, solid waste and oil), a catastrophic decline in commercial fish stocks, a severe decrease in tourism and an uncoordinated approach towards coastal zone management. Increased loads of nutrients from rivers and coastal sources caused an overproduction of phytoplankton leading to extensive eutrophication and often extremely low dissolved oxygen concentrations. The entire ecosystem began to collapse. This problem, coupled with pollution and irrational exploitation of fish stocks, started a sharp decline in fisheries resources.
Conflicts: 0

Figure 11. A data frame matching the goal frame in Figure 10 with no attribute conflicts. Words in bold were used to fill the attributes in the frame.

The frames obtained from the clusters are called the *Data Frames*, and they form the structure imposed over the Retrieved Set. This is, of course, the structured data we have been trying to build all along. Each frame type can be seen as a “relation” in a database, and each frame as a specific instance of a relation. Multiple frames of the same type may be considered as alternative answer selections, much the same way as multiple flights can be found matching a customer request, or multiple spare parts may be returned given a partial description. If a single answer is desired, or a best fitting answer, or perhaps a subset meeting some additional criteria, further selection has to be made to narrow the size of the answer space. We do this by

comparing data frames to the goal frame, and by comparing data frames against one another, and then asking the user for a hint. This is only a part of the whole story, however. Pinpointing the single best answer may be a good strategy for a factoid-style question, but we must remember that the structure we've just built is nowhere near as complete and reliable as what we had in ISD environment. Therefore, we must use the dialogue to explore the space outside of the frames that completely match the question. In some cases, there may be no such perfectly matching frames at all! Nonetheless, instead of just giving a partial answer or failing to find any, the system now attempts to fill the gaps in its understanding of the topic by asking the user to make some selections or to provide additional cues.

There are many ways to approach this problem, and we are only beginning to understand its full scope. In the remainder of this chapter we will describe just one such approach where a clarification dialogue is used to delineate the scope of the answer space of interest to the user. This appears to work quite well in scenarios involving analytical and exploratory questions of moderate complexity. As our research progresses, additional dialogue strategies will be exploited, for example, the follow-up questions, the digressions, and other exploratory devices that professional analysts use.

9. ENABLING CLARIFICATION DIALOGUE IN HITIQA

The need for clarification dialogue in HITIQA arises from the perceived variation of structures generated from the Retrieved Set clusters. In order to better assess the composition of the available answer space, data frames are first compared to the Goal Frame to note what kind of mismatches may occur, if any. We pay particular attention to matching the TOPIC attribute, before any other attributes are considered. As explained before, the TOPIC attribute is expected to contain the main event or relation expressed in text. If any of the values of the Goal Frame TOPIC attribute are present anywhere in the data cluster, they become candidates for the data frame's TOPIC as well. If more than one candidate values are found, they become alternative topics of the cluster data frame (ALT-TOPIC). On the other hand, if no explicit candidate is found to match against the Goal Frame TOPIC attribute, we attempt to select an item from the list of the Wordnet generated hypernyms for each of the Goal Frame topics. For example, *pollution* is a kind of *environmental condition* according to Wordnet. A more complete example is given in Figure 11 where we show the data frame generated from a text passage retrieved in response to the query about the Black Sea (Figure 10).

After the initial framing is done, a frame merging step is attempted for the frames assigned to the same topic. Frames judged to be related to the same event (same topic, overlap on key attribute values), are merged and values of their attributes are combined. This allows HITIQA to pull together and combine information from multiple sources, which is one of the necessary capabilities for answering analytical questions. Frame merging is still a highly experimental feature of HITIQA and is currently limited to same-type frames generated from the same cluster.

9.1 Judging Frame Relevance

We judge a particular data frame as relevant, and subsequently the corresponding segment of text as relevant, by comparing it to the Goal Frame. A data frame is given a negative point for each value mismatch (called a *conflict*) between the values of its attributes and those of the Goal Frame. This means only frames of the same type can be compared, i.e., a TRANSFER frame against another TRANSFER frame, as well as a generic frame against another generic frame, etc. However, since each topical frame is build upon a generic frame, we can in fact compare any two frames using their generic attributes. If a frame match produces no conflicts, the data frame is given the highest relevance rank and a conflict score of zero. All other data frames are scored with the increasing conflict value, score of 1 for frames with one conflict with the Goal Frame, score of 2 for two conflicts etc. Frames that conflict with all information found in the question are given the score of 99 indicating the lowest relevancy rank. We should note here that such frames may be derived from passages with some keyword overlap with the question, but presumably such overlap is semantically meaningless. Currently, frames with the conflict score of 99 are excluded from further processing. The frame in Figure 9 is scored as fully relevant to the question (0 conflicts).

Framed information represents the system's partial understanding of the user question; specifically, zero-conflicts frames constitute elements of the answer. An answer can now be composed from the underlying text passages, or by generation, using attribute values from the frame. However, a large and diverse population of near-miss frames (1+ conflicts) may indicate that only a part of the answer has been captured; conversely a large number of zero frames may suggest that a narrower interpretation of the question may be warranted. Whether any of these conditions require revisions to the answer space can be determined through a dialogue with the user.

HITIQA automatically evaluates the semantic correlation between frames and initiates a clarification dialogue with the user. The purpose of the dialogue is to help bridge the gap between the user's expectations and the system heretofore "understanding" of the question. Specifically, if the question scope and interpretation could be altered somewhat leading to a significant change of the answer space fit, through expansion, reduction, or shift, it may be reasonable to ask the user if such a change is permissible. In clarification dialogue, the system solicits from the user more details about what information he or she is seeking, based on the composition of the entire answer space, including the near-miss frames. The main principle here is that the dialogue is at the information semantic level, not at the information organization level. Moreover, the dialogue relates directly to the subject matter addressed by the question: it is of course a challenge to make sure that what happens next is indeed a logical follow up to the question. Let us consider the following exchange:

USER: *What is being done to combat AIDS epidemic in Africa?*

HITIQA: *Are you interested in information about International AIDS Conference?*

USER: *yes, but only as related to access to AIDS medication and prevention.*

In this example, the dialogue relates directly to the topic of the user inquiry, and of course it may subsequently proceed in any direction. Here is another example recorded in a recent session with HITIQA:

USER: *How did al-Qaida conduct its efforts to acquire weapons of mass destruction?*

HITIQA: *We have this information about bin Laden, not about al-Qaida; would you like to see this?*

USER: *Yes, sure.*

In both cases, the system is asking the user (indirectly) if a certain piece of information is part of the expected answer by identifying the mismatch that placed it in the *near-miss* middle zone of the answer space (Figure 7). The clarification questions relate directly to the content of the user inquiry, which is what makes these exchanges efficient. On the other hand, the user should never be asked if a particular keyword is useful or not, or if a document is relevant or not. The only exception to this rule is when a misspelling or omission of a key concept is detected in the user input – we occasionally try “Did you mean ...”.

In HITIQA we adopted a three-pronged dialogue strategy, based on the organization of the answer space depicted in Figure 7:

1. *Narrowing dialogue*: ask questions that would allow the system to reduce the size of the answer set. This is to narrow the interpretation of the question and to reduce the number of zero frames. This corresponds most directly to the information seeking strategy in ISD.
2. *Broadening dialogue*: ask questions that would allow the system to decide if the answer set needs to be expanded by information just outside of it (near-misses). The purpose is to move some of the 1-conflict and maybe 2-conflict, etc. frames into the zero-conflict zone. The broadening strategy is central to IUD, which reflects its more exploratory character. It has an equivalent in ISD, although it has been given less attention in automation research. For example, a helpful call center agent may offer to change the address on another of the customer’s accounts, even if she did not explicitly ask for it.
3. *Fact seeking/verification dialogue*: ask the user to supply missing values for certain attributes such as date or location, etc. Alternatively, ask to confirm a piece of information. This information may be required to execute a given transaction and is only relevant if a further action is expected (e.g., issue travel tickets, generate a report in a specified format, etc.). Furthermore, we allow the user to ask questions seeking additional facts and specific examples, or similar situations.

Of the above, we have thus far primarily concentrated on the first two options as part of the preliminary clarification dialogue. The clarification dialogue is when the user and the system negotiate the task that needs to be performed. We call this a “triaging stage” of the interaction, as opposed to the actual problem solving stage

(item 3 above). In practice, these two stages are not necessarily separated and may be overlapping throughout the entire interaction. Nonetheless, these two have decidedly distinct character and require different dialogue strategies on the part of the system.

The third form of dialogue (information seeking/verification) depends more crucially upon the structured frames representing event classes, rather than just the generic frames. As HITIQA supply of frames and their accuracy increases, we began to expand the dialogue scope to introduce fact verification. Specifically, rather than provide a straight away answer, the system may ask if some information captured into a frame slot is indeed of interest. One way to bring up such issues is in a form of definite suggestions:

USER: *Does N. Korea make missiles that can reach the U.S.?*

HITIQA: *Would you be interested in information on Taepodong-2 missile?*

9.2 Clarification Dialogue

Data frames with a conflict score of zero form the initial kernel answer space. Depending upon the size of this set and the presence of other frames outside of it, the system either proceeds to generate the answer or initiates a dialogue with the user. For example, if the answer space appears too large or varied, e.g., consists of many different topics, the system may ask the user how to narrow it. Alternatively, the presence of large groups of texts frames with near-miss frames assigned to them (i.e., frames with 1 or 2 conflicts with the Goal Frame) may indicate that the answer space is actually larger, and the user will be consulted about a possible broadening of the question. Currently, we only initiate a clarification dialogue for 1-conflict frames.

A 1-conflict frame has only a single attribute mismatch with the Goal Frame. This could be a mismatch on any attribute, for example, location, or organization, or time, etc. A special case arises when the conflict occurs on the TOPIC attribute. Since all other attributes match, we may be looking at potentially different events or situations involving the same entities, or occurring at the same location or time. The purpose of the clarification dialogue in this case is to probe which of these topics may be of interest to the user. This is illustrated in the exchange below recorded during an evaluation session with an intelligence analyst:

USER: *Who is Elizardo Sanchez?*

HITIQA: *Are you interested in information on civil rights related to Sanchez?*

In order to understand what happened here, we need to note first that the Goal Frame for the user question does not have any specific value assigned to its TOPIC attribute. This of course is as we would expect it: the question does not give us a hint as to what information we need to look for or may be hoping to find about Sanchez. This also means that all the data frames obtained from the retrieved set for this question will have at least one conflict with the Goal Frame (Figure 12), and thus

will be classified as either near-misses or outliers.²² One such 1-conflict near-miss frame is shown in Figure 13: its topic is *civil rights*, from the list of Wordnet generated hypernyms, and it is about Sanchez. HITIQA thus asks if *civil rights* is a topic of interest to the user. If the user responds positively, this topic will be added to the Goal Frame, and thus to the answer.

Frame Type: Generic
Topic: ? [birth, education, work, positions, ...]
Person: <i>Elizardo Sanchez</i>
Text: Who is Elizardo Sanchez?

Figure 12. The goal frame for Sanchez question.

Frame Type: Generic
Topic: <i>civil-rights</i>
Location: <i>Cuba</i>
People: <i>Sanchez</i>
Document date: [2000, 1, 11]
Source: <i>HAVANA (AP)</i>
Text: <i>I consider that the situation for civil and political rights in Cuba has worsened over the past year... owing to that Cuba continues to be the only closed society in this hemisphere," Sanchez said. "There have been no significant release of prisoners, the number of people sanctioned or processed for political motives increased. Sanchez, who himself spent many years in Cuban prisons, is among the communist island's best known opposition activists. The commission he heads issues a report on civil rights every six months, along with a list of people it considers to be imprisoned for political motives.</i>
Conflicts: 1 (topic)

Figure 13. A 1-conflict frame utilized in generating the Sanchez dialogue.

The above dialogue strategy is applicable to other attribute mismatch cases, and it produces intelligent-sounding responses from the system. During the dialogue, as new information is obtained from the user, the Goal Frame is updated and the scores of all the data frames are re-evaluated. The system may interpret the new information as a positive or negative, moving the corresponding frames to or from their current zone. For example, in the Sanchez dialogue if the user responds “yes” or equivalent to the clarification question about civil rights, *civil-rights* will be added to the TOPIC list in the Goal Frame. Subsequently, all 1-conflict frames with a

²² Another way to interpret such questions is to consider them as biography-template questions where certain topics are assumed relevant by default, and thus placed in the Goal Frame: *birth date, college attended, jobs held, etc.*

civil-rights topic will be re-scored to 0-conflicts. Similarly, 2-conflict frames with *civil-rights* topic will be re-scored to 1-conflict, etc. If the user responds “no”, all frames with *civil-rights* TOPIC will be re-scored to 99-conflict, and effectively removed from further processing.

The clarification dialogue will continue on the topic level until all the significant sets of near-miss frames are either included in the answer space (through user broadening the scope of the question that removes the initial conflicts) or dismissed as not relevant. When HITIQA reaches this point it will re-evaluate the data frames in its answer space. If there are too many answer frames now (more than a pre-determined upper threshold), the dialogue manager will offer to the user to narrow the question using another frame attribute. On the other hand, if the size of the new answer space is still small and there are many unresolved near-miss frames, the dialogue manager will suggest to the user ways of further broadening the answer space, thus making more data frames relevant, or possibly retrieving new documents by adding terms acquired through the clarification dialogue. When the number of frames is within the acceptable range, HITIQA will generate the answer using the text from the frames in the current answer space. The user may end the dialogue at any point and have an answer generated given the current state of the frames.

10. ANSWER GENERATION

In the first version of HITIQA, the answer was simply composed of text passages underlying the zero-conflict frames. The text of these frames is ordered by date and output to the user. Typically, an answer to analytical type questions requires a few pages of information. The example below shows the first portion of the answer generated by HITIQA for the Black Sea query.

USER: *How has pollution in the Black Sea affected the fishing industry, and what are the sources of this pollution?*

HITIQA: *The Black Sea is widely recognized as one of the regional seas most damaged by human activity. Almost one third of the entire land area of continental Europe drains into this sea (...) The management of the Black Sea itself is the shared responsibility of the six coastal countries: Bulgaria, Georgia, Romania, Russia, Turkey, and Ukraine...*

Our current work on answer generation is focused on fusion of more succinct and informative answer summaries. The first step in this direction is to define a process by which frame information can be utilized to automatically generate a “headline” for the underlying text passage. For example:

USER: *Does Syria have any nerve gas manufacturing capabilities?*

HITIQA: **SYRIA REPORTED TO DEVELOP SARIN NERVE GAS,**

WARHEADS AND MISSILES: *Syria is said to possess a large number of warheads filled with VX and sarin nerve gas, which can be mounted on Scud-B and Scud-C missiles. It is reported to have factories near Damascus*

and Homs that are producing hundreds of tons of mustard gas and sarin nerve agent.

The “headline” above is generated by exploiting the semantics of typed frames where attributes are assigned specific roles. In general, the more precise the semantics of a frame, the more satisfactory headline is generated. For generic frames (like those used in Sanchez examples in section 9) it would much harder to automatically come up with the right wording for a headline.

The work on answer fusion is in the initial stages. Eventually, we envision that headlines of multiple text passages will form a fused summary of the answer, something that the user can view first.

11. VISUALIZATION

While the users can effectively interact with HITIQA through language-based dialogue, we have also developed a visual display that allows them to see the entire answer space at once: all frames, their relevancy ranking, as well as their mutual arrangement. The visualization supplies a bird-view of the answer space with several zoom-in/out levels. The users can navigate through this space and rearrange its elements based on broader visual cues. They can also examine content of individual frames, change their relevance assessments, and switch to and from the language-based dialogue at any time. In other words, the visualization extends the dialogue into a multimodal interaction (Hochberg *et al.*, 2002; Johnson *et al.*, 2002). In HITIQA we utilize color coding as a way of making non-binding offers and suggestions to the user. This is an effective method of communicating complex information mismatches as well as the boundaries of useful information that would be too awkward to communicate verbally. A color-coded display, showing information considered relevant (blue) located next to what is currently assumed not relevant (red) may prompt the analyst to examine these regions.

Visualization of an abstract information space faces many practical challenges. It has to supply context information missing from the language-based dialogue, while suppressing details that would make the images hard to comprehend. The golden principle here is that the visualization should necessitate no explanation, which is particularly hard to achieve in abstract displays. Since we cannot rely on explicit visual cues, we need to fall back on widely recognizable symbols, such as the use of color, size and spatial arrangements to convey the meaning. The key danger here is that every element of the display may now carry a meaning for the user, whether or not we have actually planned for it. This means a very careful design is required.

Frames are the key building blocks of HITIQA answer space. They are multidimensional and highly heterogeneous objects. Some attributes are numerical, others are categorical, and the categorical attributes are generally represented by sets of strings. Due to the complexity of the information space, we find it beneficial to use different techniques to visualize different frame attributes, and when appropriate, we combine attributes to apply a specific visualization technique to the

whole group. Multiparametric visualization technique similar to our approach has been discussed before in (Gristein, 1993).

Figure 14 is a snapshot of HITIQA visualization panel showing a high-level view of several frame groups in a constellation-like arrangement. The panel is split into several sub-panels. The top *Frame sub-Panel*, displays groups of frames organized according to the (*conflict-score*, *topic-attribute*, *frame-type*) triad, where the *conflict-score* is the number of mismatches between the highest ranking frame in a group and the user question. To visualize frame groups, we use colored semitransparent circular *hubs* with colored bands around them. Smaller ovals inside each frame group signify individual frames making up the group. The color of each hub corresponds to the conflict score of the group (i.e., the highest ranking frame). A color bar on the left side of the Frame Panel maps conflict scores to colors, with the blue corresponding to 0-conflicts and the orange representing 99-conflicts. The choice of colors is based on a convention accepted for representing elevations on geographical maps: “perfect match” (blue), “near miss” (green), and “off mark” (orange). On the right side of the Frame Panel there is a column of colored dots representing all individual frames and giving the user a quick glance at the population of each of the 3 major score groups. The TOPIC attribute for each frame group is explicitly displayed on the spokes attached to the hubs. In Figure 11 we have frame groups representing topics such as “cooperation,” “assistance,” “development,” “constructed,” and “missile program”.

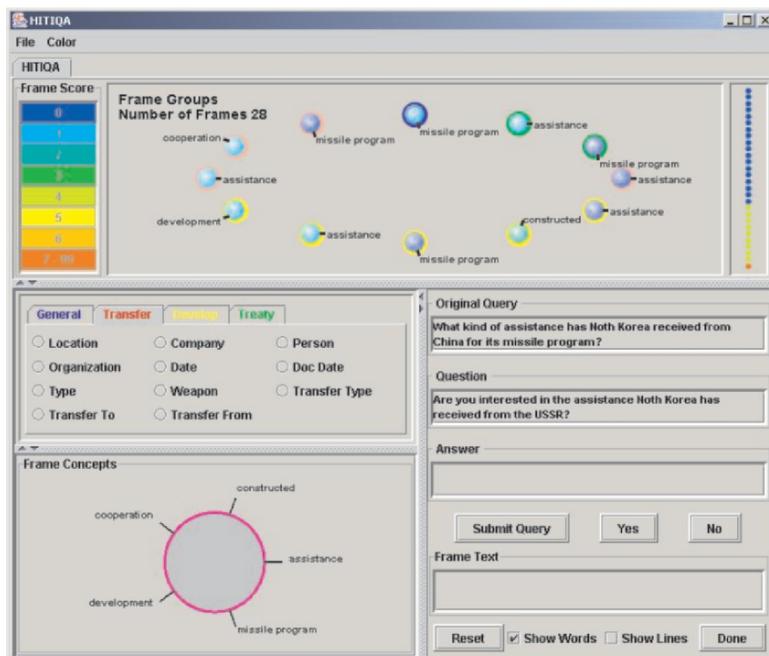


Figure 14. HITIQA visualization panel – frame group view.

The *Frame Concepts* sub-panel in the lower left part of the display aggregates TOPIC attribute values from all frames displayed in the Frame Panel, thus serving as a “legend” for the Frame Panel. The reader should note that the position of each spoke is fixed for easy visual location.

Figure 15 shows the individual frame view, which is obtained by zooming in to a particular frame group (by clicking on it). Now, the Frame Panel contains three individual frames of type “*WMD Develop*” with TOPIC = “*assistance*” and *conflict-score* = 0. In this example, the spokes on frame hubs represent values of the PERSON attribute, but the user can choose any other frame attribute to be displayed instead. The choice of spoke attributes is controlled by the radio buttons located in the middle left panel. Since the set of attributes depends on the frame type, we have a tabbed panel with each tab corresponding to one frame type. The color of the tab labels corresponds to the color bands around the frames and frame groups in the Frame Panel. A small red dot inside one of the frames shows that this frame has been selected for text viewing: the underlying text passage is displayed in the right bottom sub-panel.

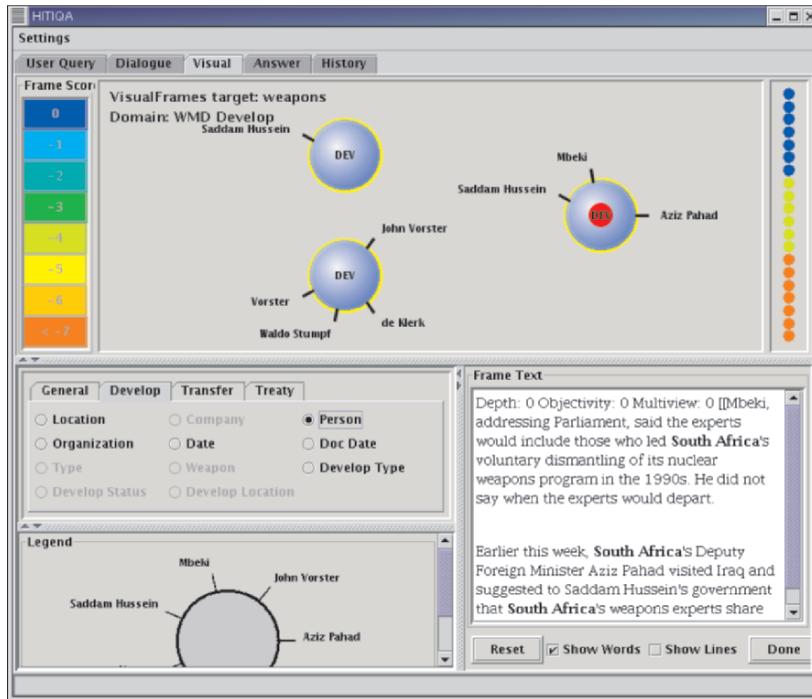


Figure 15. Individual frame panel.

12. EVALUATIONS

We have conducted a series of workshops designed to assess the overall design and usability of HITIQA as an interactive question-answering system. Since the objective of HITIQA is to answer analytical, non-factoid questions such as “*How is the al Qaeda organization funded?*”, the answers to such questions are multi-dimensional, and typically can only be found by bringing together information from multiple sources (see sections 8 and 9). These workshops constituted a formal evaluation of HITIQA as an end-to-end system where our intended users, intelligence analysts employed by the U.S. government, extensively used the system to perform tasks like those that they normally perform at work. Accordingly, the two basic objectives of these workshops were:

1. To perform a realistic assessment of the usefulness and usability of HITIQA as an end-to-end system, from the information seeker’s initial questions to completion of a draft report. In particular, we wanted to determine how long it took users to feel confident of their ability to use HITIQA and whether they were comfortable with the interactive dialogues and visual display panel.
2. To develop metrics to compare the answers obtained by different analysts and evaluate the quality of the support that HITIQA provides.

Scenario #2: South Africa’s WMD Program

You have been given 30 days to develop a comprehensive report on the South African chemical, biological and nuclear warfare program, for your division chief who is to present to the Secretary of Defense. Your report should include several key elements of the South African WMD Program, including what people, organizations, and countries are involved, what chemicals have been purchased and/or used, where the chemicals have been purchased from and from whom, how their WMD program was financed, where these development locations are, any proposed activity (use, distribution, etc.), any money transactions that have been made between these suspects and other organizations, and any other contacts, or travels that have been made by any of the primary figures involved. Supply any further information that can support your documentation.

Figure 16. Example analytic scenario used in evaluating HITIQA.

The analysts’ primary activity at both workshops was preparation of reports in response to complex, multifaceted questions, that we call “analytic scenarios”. Several U.S. Government organizations, NIST, AFRL, CNS and ARDA²³ collaborated in the development of a set of scenarios, and contributed data to this research. An example scenario is shown in Figure 16; the reader may note that it is

²³ NIST is National Institute for Standards and Technology; AFRL is Air Force Research Laboratory; CNS is Center for Non-Proliferation Studies; ARDA is Advance Research and Development Activity.

not so much a question as a problem statement from which possibly multiple questions will be issued to the system. Figure 18 shows an actual interaction log for this scenario recorded by one of the participating analysts.

After each scenario, analysts answered a set of questions about the scenario, their comfort with the system and their level of satisfaction with the results. At the end of the workshop, analysts completed an instrument which assessed their confidence in their ability to use HITIQA successfully, the quality of information it provided, and its readiness to be used in their regular work environment. Figure 17 shows the normalized results of the analysts' assessment of the system, based on a set of 17 criteria, which include categories such as: *speed of finding important information* (#11), *finding enough information* (#13), *ease of formulating effective questions* (#6), *system being suitable for a regular work use* (#16), etc. Each criterion was scored on the scale from 1 to 5, where a higher score indicates a better score. Overall, analysts most frequently used the score of 4 to judge HITIQA on these criteria.

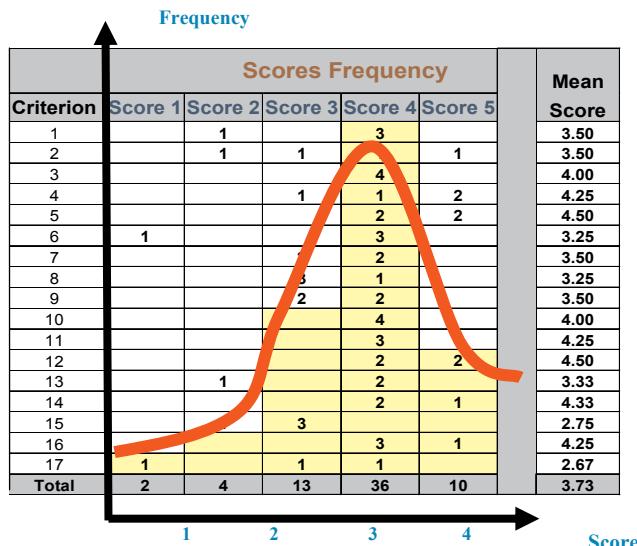


Figure 17. Qualitative assessment of HITIQA by professional analysts (5 is best).

Details of the analyst qualitative evaluations are available in a separate report (Wacholder *et al.*, 2003). Figure 18 presents an abridged transcript of one analytic session on the subject of South Africa in the weapons non-proliferation domain. The transcript shows only the dialogue exchange, and does not include the reading and report building activities. This session took approximately 2 hours to complete.

In addition, two quantitative evaluation pilots have been conducted by NIST during the course of AQUAINT Phase I. The purpose of quantitative evaluations was to measure the accuracy of QA systems to deliver factual and complete answers. Details of these evaluation pilots can be found in (Small *et al.*, 2003).

Starting with AQUAINT Phase II (May 2004), we have evaluated HITIQA in a series of off-site workshops with professional analysts in order to obtain an in-depth assessment of the system usability and performance. Similarly to the preliminary evaluations described above, the primary activity at these workshops involved analysts' preparation of reports in response to complex questions that often encompass multiple sub-questions, aspects and hypotheses. For example, in one scenario, analysts were asked to locate information about the al Qaida terrorist group: its membership, sources of funding and activities. We prepared a database of over 1+GByte of text documents; it included articles from the Center for Non-proliferation Studies (CNS) data collected for the AQUAINT program and similar data retrieved from the web. Over several off-site and on-line drills, a group of fifteen analysts generated multiple reports for 12 realistic intelligence problems, spending anywhere between 1 and 3 hours per report. Each session involved multiple questions posed to the system, as well as clarification dialogue, visual browsing and report construction. The evaluation instruments included questionnaires assessing analysts' opinions about various aspects of the system, as well as a cross-evaluation process where analysts scored each other's reports for completeness and organization. In the largest study in this series, conducted over a three-week period in summer of 2004, analysts worked in parallel sessions using several different analytical tools, with Google document retrieval used as a baseline tool.

While still preliminary, the evaluation suggests two important advantages of HITIQA over other approaches as well as over a document retrieval baseline using Google:

1. the HITIQA interactive approach is significantly more efficient because it requires the analyst to ask fewer questions (nearly 60% fewer than using Google) and consequently spend less time to obtain a report of equal or better content; and
2. HITIQA is more effective because it produces more usable information per user question, evidenced by analysts saving more material for their reports and doing so more often. It makes the collection process twice as effective as searching with Google.

These findings suggest that HITIQA has a potential to provide *cognitive augmentation* for the analysts thus allowing them to produce better reports using fewer resources than is the current practice. Further, larger scale evaluations are currently planned to verify these findings.²⁴

Far more important than the specific evaluation information that we gathered from some of these workshops are the assessment instruments developed for that purpose, and the experience and knowledge gained in conducting the end-to-end evaluation. Semi-structured focus discussions, immediately after the training, and at the end of the workshop, provide valuable insights into how the system is perceived by its users. Similarly, one-on-one debriefings by skilled interviewers elicit a great deal of information about the usability of the system. The analysts represent a very

²⁴ For details of the evaluation methodology and results, see (Morse, 2004; Wacholder *et al.*, forthcoming). The AQUAINT Program-wide evaluations are currently being developed.

precious resource, and their period of interaction with the system must be planned as carefully as any campaign. Design of the tasks and preparation of the materials for the two workshops represents an intensive investment of over six weeks of preparation time. Some of the key considerations when preparing these sessions (whether on-site, off-site, or on-line) include:

1. Selection of tasks: the analytical tasks (scenarios) selected for each session must be realistic both in their content as well as cognitive complexity. Furthermore, the tasks must be interesting and relate to current events; old, tired topics are often boring to the analysts.
2. Preparation of data: even though the topics may be interesting, the real challenge is in finding relevant information for the report. The data sets used in the experiments must be carefully assembled to provide sufficient depth and breadth of topic coverage, as well as enough noise to engage an analyst for 2-3 hours. Analysts are supremely skilled in information retrieval and they can reach the “bottom” of the data quite rapidly.
3. The process of evaluation produces enormous amounts of data, and it is necessary to plan from the beginning, so that formal results are captured at once into well organized database structures and informal data are reliably recorded.

13. FUTURE WORK

This chapter describes a work in progress. We expect that the initial specification of content frame will evolve as we subject the initial system to more demanding evaluations. In order to expand HITIQA topical capabilities beyond the weapons non-proliferation domain, we need to develop a method for rapid acquisition of topical frames of additional kinds. Currently the system relies on the General Frame to handle out-of-domain topics, which is adequate but not sufficiently effective. Our future plans include adaptation of structured data bases as sources of frames, which then can be projected over unstructured data in new domains. This is expected to increase the accuracy and efficiency of the dialogue. Answer generation will involve fusion of information at the level of frame groups, which would allow HITIQA to produce draft analytic reports.

Our plans also involve significant work in adaptation and personalization of the system to individual users. We noted, for example, that individual analysts' work styles differ sometimes dramatically, and thus designing a one-size-fits-all system may not be enough. Furthermore, different analysts judge and select their information according to individual preference and research strategies, and we would like HITIQA to support these to a maximum extent.

We have also extensive plans for conducting qualitative and quantitative evaluations with HITIQA. The system has now reached the stage when it can be considered a viable prototype for future analytic tools. It is therefore important that future development be done in close collaboration with the target users, since introduction of new advanced technology into a workplace is a two-way street, and takes time and adjustment on both sides.

User: *What chemical weapons has the South African government stockpiled?*
 HITIQA: Can we also add material on the transfer of weapons and intelligence?
 User: Yes
 HITIQA: Should we add material on assistance to your answer?
 User: Yes
 HITIQA: Are you also interested in background information on the weapons development program?
 User: Yes
 HITIQA: May we also show you information on possible treaty violations?
 User: Yes
 HITIQA: Please go to the Answer tab.
 [A series of text passages are displayed; the user will now engage in reading and selecting passages for the report. This may last for 10-30 minutes.]
 User: *What countries participated in the Convention on Chemical Weapons?*
 HITIQA: Can we also add material on the transfer of weapons and intelligence to your answer?
 User: No
 HITIQA: Would you be interested in chemical industry?
 User: Yes
 HITIQA: Do you want to see information on danger?
 User: *Only in South Africa*
 HITIQA: Should we add information on danger to your answer?
 User: No
 HITIQA: Please go to the Answer tab.
 [More text passages are displayed, and more reading, navigating, selecting is taking place for another 10-30 minutes]
 User: *What was South Africa's involvement with the Convention on chemical weapons?*
 HITIQA: Can we also add material on the transfer of weapons and intelligence between South Africa and other countries?
 User: Yes
 HITIQA: May we also show you information on possible treaty violations involving South Africa?
 User: No
 HITIQA: Please go to the Answer tab.
 [Still more text passages displayed]
 User: *What materials have been purchased by South Africa to develop their chemical, biological and nuclear weapons?*
 HITIQA: Can we also add material on the transfer of nuclear weapons and intelligence between South Africa and other countries?
 User: No
 HITIQA: Please go to the Answer tab.
 [More answer, and more browsing follows]
 User: *Who supplies South Africa with materials to develop CBW program?*
 HITIQA: Please go to the Answer tab.
 User: *How is the South African WMD program financed?*
 HITIQA: Can we also add material on the transfer of weapons and intelligence?
 User: No
 HITIQA: Do you want to see information on stakeholders?
 User: Yes
 HITIQA: Should we add information on seashore to your answer?
 User: No
 HITIQA: Please go to the Answer tab.

Figure 18. An (abridged) transcript from an analytic session with HITIQA on the South Africa scenario (see Figure 16)

14. ACKNOWLEDGEMENTS

This paper is based on work supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number 2002-H790400-000, by the Defense Advanced Research Project Agency (DARPA) under contract N66001-01-1-8946, and by the European Commission's 5th Framework Programme under project number IST-2000-25033. We acknowledge significant contributions from other members of HITIQA and AMITIÉS projects, including K.B. Ng, Rong Tang, Yorick Wilks, Alan Bierman, Jean-Luc Gauvain, Lori Lamel, Bernard Prouts, Nick Webb, Robert Rittman, Diane Kelly, Liu Ting, Sean Ryan, Robert Salkin, Tracy Janack, Cristi Ursu, Ashley McKenzie, Laurence Devillers, and Sophie Rosset. We would like to thank Ralph Weischedel of BBN for making Identifinder available for this project. We also thank Bruce Croft at the University of Massachusetts for providing the INQUERY system.

15. REFERENCES

- Aist, Gregory, John Dowding, Beth Ann Hockey and Jim Hieronymus. 2002. A Demonstration of a Spoken Dialogue Interface to an Intelligent Procedure Assistant for Astronaut Training and Support Proceedings of the ACL-2002. pp. 83-84
- Allen, J.F. 1979. A Plan Based Approach to Speech Act Recognition. TR University of Toronto.
- Allen, J. and M. Core. 1997. Draft of DAMSL: Dialog Act Markup in Several Layers. <http://www.cs.rochester.edu/research/cisd/resources/damsl/>
- AMITIÉS Project. 2003. <http://www.dcs.shef.ac.uk/nlp/AMITIÉS/>
- Bagga, A., T. Strzalkowski, and G.B. Wise. 2000. PartsID: A Dialog-Based System for Identifying Parts for Medical Systems. *Proc. of the ANLP-NAACL-2*.
- Buckley, Chris. May 1985. Implementation of the Smart information retrieval system. *Technical Report TR85-686*, Department of Computer Science, Cornell University, Ithaca, NY.
- Bunt, H.C. Dynamic Interpretation and Dialogue Theory. In M. Taylor, D. Bouwhuis, F. Neel, eds., *The Structure of Multimodal Dialogue*, volume 2. John Benjamins Publishing Company, Amsterdam, 1997.
- Callan, James, P., W. Bruce Croft, Stephen M. Harding 1992. The INQUERY Retrieval System. *Proc. of DEXA-92, 3rd International Conference on Database and Expert Systems Applications*. 78-83.
- Cunningham, H., D. Maynard, K. Bontcheva, V. Tablan and Y. Wilks. 2000 Experience of using GATE for NLP R&D. In *Coling 2000 Workshop on Using Toolsets and Architectures To Build NLP Systems*.
- DARPA Communicator Program. 2002. <http://www.darpa.mil/iao/communicator>
- Dowding, John and James Hieronymus. 2003. A Spoken Dialogue Interface to a Geologist's Field Assistant. Proceedings of HLT/NAACL 2003. Edmonton, Canada.
- Ferguson, George and James Allen. 1998. "TRIPS: An Intelligent Integrated Problem-Solving Assistant," in *Proc. of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI. 567-573.
- Gorin, A.L., G. Riccardi and J. Wright. 1997. How May I Help You? *Speech Communication*, 23 (1/2): 113-127.
- Grinstein, G.G., H. Levkowitz, R.M. Pickett, and S. Smith. 1993. "Visualization alternatives: non-pixel based images," *Proc. Of IS&T 46th Annual Conf.*, pp. 132-133.
- Harabagiu, Sanda, M., George A. Miller, Dan I. Moldovan. 1999. WordNet 2 - A Morphologically and Semantically Enhanced Resource, SIGLEX 1999.
- Harabagiu, S., M. Pasca and S. Maiorano. 2000. Experiments with Open-Domain Textual Question Answering. In *Proc. of COLING-2000*. 292-298.

- Hardy, H., N. Shimizu, T. Strzalkowski, L. Ting, B. Wise and X. Zhang. 2002. Cross-Document Summarization by Concept Classification. *Proceedings of SIGIR-2002*, Tampere, Finland.
- Hardy, H., K. Baker, L. Devillers, L. Lamel, S. Rosset, T. Strzalkowski, C. Ursu and N. Webb. 2002. Multi-layer Dialogue Annotation for Automated Multilingual Customer Service. *ISLE Workshop*, Edinburgh, Scotland.
- Hochberg, Judith, Nanda Kambhatla and Salim Roukos. 2002. A Flexible Framework for Developing Mixed-Initiative Dialog Systems. *Proc. of 3rd SIGDIAL Workshop on Discourse and Dialogue*, Philadelphia.
- Hovy, E., L. Gerber, U. Hermjakob, M. Junk, C-Y. Lin. 2000. Question Answering in Webclopedia. *Proceedings of Text Retrieval Conference (TREC-9)*.
- Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, Y. Wilks. 1998. Description of the LaSIE-II System as Used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*.
- Johnston, M., Ehlen, P., Bangalore, S., Walker., M., Stent, A., Maloor, P., and Whittaker, S. 2002. MATCH: An Architecture for Multimodal Dialogue Systems. In *Meeting of the Association for Computational Linguistics*.
- Kobsa, A., and Wahlster, W. (eds.). 1989. User Models in Dialog Systems. New York: Springer.
- Litman, Diane J. and Shimei Pan. 2002. Designing and Evaluating an Adaptive Spoken Dialogue System. *User Modeling and User-Adapted Interaction*. 12(2/3): 111-137.
- Loebner Prize web site. 2003. <http://www.loebner.net/Prizef/loebner-prize.html>
- Lehnert, W.G. 1978. The Process of Question Answering. Lawrence Erlbaum Assoc., Hillsdale.
- Miller, G.A. 1995. WordNet: A Lexical Database. *Comm. of the ACM*, 38(11): 39-41.
- Morse, E. 2004. *An Investigation of Evaluation Metrics for Analytic Question Answering*. Proceedings of AQUAINT Phase 2 6-month PI Meeting, Tampa.
- Prager, John, Dragomir R. Radev, and Krzysztof Czuba. 2001. Answering what-is questions by virtual annotation. In Human Language Technology Conference, Demonstrations Section, San Diego, CA.
- Robertson, Stephen, E., Steve Walker, Micheline Hancock-Beaulieu. 2000. Experimentation as a way of life: Okapi at TREC. *Information Processing and Management* 36(1): 95-108.
- Seneff, S. and J. Polifroni. 2000. Dialogue Management in the MERCURY Flight Reservation System. *Proc. ANLP-NAACL 2000, Satellite Workshop*, 1-6, Seattle, WA.
- Small, Sharon, Nobuyuki Shimizu, Tomek Strzalkowski and Liu Ting. 2003. HITIQA: A Data Driven Approach to Interactive Question Answering: A Preliminary Report. Proceedings of AAAI Spring Symposium on New Directions in Question Answering, Stanford University. pp. 94-104.
- Strzalkowski, T. and J. Wang. 1996. The Self-Learning Universal Concept Spotter. *Proceedings of COLING-96*.
- Tang, Rong, K.B. Ng, Tomek Strzalkowski and Paul Kantor. 2003. "Automatic Prediction of Information Quality in News Documents." *Proceedings of HLT-NAACL 2003*, Edmonton, Canada.
- Traum, David. 1994. A Computational Theory of Grounding in Natural Language Conversation, TR 545 and Ph.D. Thesis, Computer Science Dept., U. Rochester.
- Wacholder, N., P. Kantor, S. Small, T. Strzalkowski, D. Kelly, R. Rittman, S. Ryan and R. Salkin. 2003. Evaluation of the HITIQA Analysts' Workshops. Report prepared for Advanced Research and Development Activity (ARDA).
- Wacholder, et al. (forthcoming) *A Model for Realistic Evaluation of an End-to-end Question Answering System*. Accepted for publication in JASIST.
- Walker, Marilyn A. 2000. An Application of Reinforcement Learning to Dialogue Strategy Selection in a Spoken Dialogue System for Email. *Journal of Artificial Intelligence Research*. 12: 387-416.
- Ward, W. and B. Pellom. 1999. The CU Communicator System. <http://communicator.colorado.edu/>
- Weizenbaum, Joseph. 1966. ELIZA – A Computer Program For the Study of Natural Language Communication Between Man and Machine. *Communications of the ACM*, Vol. 9 (1) pp. 6-35.
- Xu, Jinxi, Ana Licuanan, Jonathan May, Scott Miller and Ralph Weischedel. 2002. TREC2002 QA at BBN: Answer Selection and Confidence Estimation. *Proceedings of TREC 2002*. NIST Special Publication.
- Yangarber, Roman. 2002. Acquisition of Domain Knowledge. In M. T. Pazienza (ed.), *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Rome, Italy.
- Yangarber, Roman. 2003. Counter-Training in Discovery of Semantic Patterns. *Proceedings of ACL-2003*, Sapporo, Japan.

B. GRAU, O. FERRET, M. HURAUT-PLANET,
C. JACQUEMIN, L. MONCEAUX, I. ROBBA AND A. VILNAT

COPING WITH ALTERNATE FORMULATIONS OF QUESTIONS AND ANSWERS

Abstract: We present in this chapter the QALC system which has participated in the four TREC QA evaluations. We focus here on the problem of linguistic variation in order to be able to relate questions and answers. We present first, variation at the term level which consists in retrieving questions terms in document sentences even if morphologic, syntactic or semantic variations alter them. Our second subject matter concerns variation at the sentence level that we handle as different partial reformulations of questions. Questions are associated with extraction patterns based on the question syntactic type and the object that is under query. We present the whole system thus allowing situating how QALC deals with variation, and different evaluations.

1. INTRODUCTION

The huge quantity of available electronic information leads to a growing need for users to have tools able to be precise and selective. These kinds of tools have to provide answers to requests quickly without requiring users to explore large amount of texts or documents, or to reformulate their request. From this viewpoint, finding an answer consists not only in finding relevant documents but also in extracting relevant parts from them if the question is a factual one, or to summarize them if the request is thematic. This leads us to express the QA problem in terms of an information retrieval problem that can be solved using natural language processing (NLP) approaches.

Pure NLP solutions were studied at the end of the seventies to answer questions as in QUALM, the well-known system of Lehnert (1977). This system analyzed small stories about specific topics (traveling by bus, going to the restaurant, etc.), transformed them into a conceptual representation and answered questions by choosing a strategy depending on the kind of information sought. It consisted of developing reasoning on the conceptual representation making use of general knowledge. In a restricted domain, some recent work such as Extrains (Berri, Mollá Alliod & Hess, 1998) also made use of an NLP approach: its purpose was to analyze the Unix manual, to represent it in a logical form and to make inferences to answer

questions. Nevertheless, Extras proposed to back off to a weaker mode, exploiting keywords when the NLP resolution fails.

The intensive use of semantic and pragmatic knowledge prevents the application of these approaches to open domain questions. As a matter of fact, the resolution strategy has to be adapted to work in such an environment, relaxing the constraints on the conceptual representation. If sentence representations are closer to the surface form, they involve less knowledge and they can be built automatically on a larger scale. Thus, while knowing that the kind of required information remains the same, one can view searching the answer not as an inference problem, but as a reformulation problem: according to what is asked, find one of the different linguistic expressions of the answer in all candidate sentences. The answer phrasing can be considered as an affirmative reformulation of the question, partially or totally, which entails the definition of models that match with sentences containing the answer. According to the different approaches, the kind of model and the matching criteria greatly differ. Strategies range from finding certain words of the questions in the sentence and selecting a noun phrase of the expected type – a minimal strategy applied by all the Question Answering (QA) systems in TREC – to building a structured representation that makes explicit the relations between the words of the question and which is compared to a similar representation of the sentences (Harabagiu, Pasca & Maiorano, 2000; Hovy, Hermjacob & Lin, 2001b). As realizing a complete parse of sentences remains an unsolved problem, our position is halfway. It consists in a partial reformulation of the question, centered on the question focus and expressed by syntactic constraints.

While the expected answer type is rather precise when the questions ask for a named entity — for example the question *When is Bastille Day?* requires a date as answer and the question *What is the name of the managing director of Apricot Computer?* requires a person name — it remains general for other ones, such as questions asking for a definition as in *What is a nematode?* or for a cause. In the former case, the answer type is such as its recognition in sentences can rely on patterns that are independent from the question terms. Thus, finding an answer mainly requires recognizing an instance of the expected named entity. However, in the latter case, the answer cannot be specified by itself and must be described by a pattern that involves relationships with some question terms and this leads us to talk about linguistic patterns of answers. Nevertheless, whatever criteria are applied, they all require the modeling of linguistic variation at some level.

At the term level, sentences that answer *What is the average salary of a professional baseball player?*, will certainly contain an expression about salary, which might be the *average pay*, and an expression about baseball player, which might be *baseball professional*. The first formulation involves a semantic variation by using a synonym, while the second example relies on a syntactic variation of a noun phrase.

At the sentence level, when looking for a definition, as demanded in *What is epilepsy?*, the answer might be expressed with *epilepsy is a seizure disorder* or *a person has a seizure disorder or epilepsy ...*, corresponding to several formulations of the same information involving syntactic variations. These answer formulations can be described by the following patterns: “*epilepsy is NP*” and “*NP or epilepsy*”

where NP stands for a noun phrase that comprises the answer. The general principle involved in our QA system consists of determining the type of sought information in order to know which patterns better describe an affirmative reformulation. These patterns allow the system to find the answer in a selected sentence.

Before detailing our approach, we will examine in section 2 related work on linguistic variation in order to provide a context. This will be followed in section 3 by a general description of our system, QALC, in order to give a complete vision of our solution and situate within our architecture the role of the different modules we will describe in the further sections. The recognition of term variants, performed by Fastr (Jacquemin, 2001) in our system, help the process that selects relevant passages and the question-sentence pairing process. It will be presented in section 4.

Our criteria for choosing the answering strategy depend on which information is deduced when analyzing the question. It can be one or several of the following features: a) a named entity type that characterizes the answer; b) the syntactic form of the question; c) the question focus, which is a noun that is generally present in the answer formulation; d) the associated answer patterns.

Our question analysis module makes use of a syntactic parser. We will discuss in section 5 why we use such a parser and how it is integrated in our system. We will then discuss how we make use of the different question features. First, the recognition of a noun phrase similar to the question focus in sentences and its impact in the sentence selection process according to other criteria selection will be detailed in section 6. And finally, in section 7, we will present how question categories lead us to associate with each question a set of reformulation patterns.

2. LINGUISTIC VARIATION IN RELATED WORK

2.1 *Paraphrase at the Term Level*

Paraphrase is the natural human capacity to use different wordings for expressing the same conceptual content. Many text processing applications have to deal with paraphrase for covering alternate formulations with a similar semantic content. Generating paraphrases is useful in Natural Language Generation (Robin, 1994; Barzilay and McKeown, 2001) because it offers the possibility to use different formulations depending on the context. In Information Retrieval and, especially, in QA applications, it is necessary to cope with paraphrase at various levels of the process. Paraphrase should be accounted for at the indexing level in order to conflate indexes that correspond to similar concepts. Index conflation is taken into consideration by Fastr, which performs term variant recognition (Jacquemin, 2001). At the querying and pairing level, it is also mandatory to recognize variant phrasings in order to associate different formulation of the same information need with its corresponding indexes (Lin and Pantel, 2001).

Once the need for recognizing paraphrases is established, there are several possibilities for processing text documents and conflating paraphrase text chunks. Early attempts in variant conflation such as (Sparck Jones and Tait, 1984) use

semantically-rich approaches. In these approaches, it is assumed that (1) a full parse tree can be produced for any sentence in the document and (2) the semantic and morphological links required for the detection of any paraphrase exist in a database. Even though there have been important developments in the design of large scale parsers and in the enrichment of thesauri and term banks, it is unrealistic to pretend that the two preceding requirements can be satisfied in large-scale information access applications such as QA.

Recent developments in large scale paraphrase recognition do not require full in-depth analyses. Instead, they rely on shallow parsers such as Minipar (Berwick, 1991), or a combination of part-of-speech patterns and lexical features (Barzilay and McKeown, 2001; Jacquemin 2001). Although exhaustiveness in paraphrase patterns and associated morphological and semantic links is unrealistic, recent approaches to paraphrase recognition combine machine learning techniques, recycling of human-based semantic or morphological databases, and distributional similarities. In (Barzilay and McKeown, 2001), corpus-based paraphrases are extracted from multiple translations of the same text through learning algorithms inspired from machine translation techniques. This technique improves upon classical machine translation by providing associations between single and multiple word expressions. With the same purpose in mind, the classical algorithms for extracting semantic classes through distributional similarities were improved by Lin and Pantel (2001) by using similarities between shallow parse trees. The resulting output contains paraphrases at the lexical or phrase level that are missing from manually-generated variations. The approach to paraphrase pattern discovery relies on progressive corpus-based tuning in (Jacquemin, 2001). This approach separates the different levels of variant construction (structural and syntactic, morphological, and semantic). Through corpus-based tuning the structural similarities are extracted. In a second step, the combination of structure and semantic information can be refined by associating specific structures with specific lexical classes based on shallow semantic features (Fabre and Jacquemin, 2000).

In the QA system, QALC, developed at LIMSI, variation is accounted for at the indexing level and at the question analysis level. At the indexing level, variant indexes are conflated through term variant recognition. Term variation involves structural, morphological, and semantic transformations of single or multi-words terms. The semantic links are extracted from WordNet (Fellbaum, 1998) synonymy relations. The morphological links for inflectional morphology result from lemmatization performed by the *TreeTagger* (Schmid, 1999). As for derivational morphology, two words are morphologically related if they share the same derivational root in the CELEX database (CELEX, 1998). Both morphological and semantic links are combined in the structural associations obtained through corpus-based tuning. Term paraphrase recognition is used for dynamic and query-based document ranking at the output of the search engine. Documents that contain variants of the query terms are paired with the corresponding queries. As a result, linguistic variation is explicitly addressed through the exploitation of word paradigms, contrarily to other approaches like the one taken in COPSY (Schwarz, 1988), where an approximate matching technique between the query and the documents implicitly takes it into account.

2.2 Syntactic Variation at the Sentence Level

Paraphrase at the sentence level is tackled when systems have to provide a diversity of texts as a result, as it is the case in narrative or sentence generation in natural language. It also is an issue for systems dealing with this diversity in input, as in the information extraction (IE) field. In a certain way, information extraction is a specialized case of the general QA problem. We will briefly present how the problem was studied in these domains.

2.2.1 Natural Language Generation

Sentence or text generation systems all support the following subtasks (Zock and Sabah, 2002; Hovy, 1996):

- Macroplanning, at the text level, and microplanning, at the sentence level, to determine messages to generate.
- Surface realization, a linguistic component, that orders words, states their lexical categories and manages syntactic constraints. It converts sentence-sized chunks of representation into grammatically correct sentences.

Linguistic variation is addressed in the surface realization task. One of the simplest approaches consists of modeling the sentences to produce by templates when all the different messages have only slight alterations in their linguistic formulation. A template-based approach is generally used when the text to generate is rather fixed, as form letters for example where only a few fields have to be filled. These approaches do not really deal with flexibility of language. A more flexible approach is developed in phrase-based systems that generalize templates. Phrases resemble phrase structure grammar rules, and a pattern describing a sentence is extended for each of its components by using more specific patterns. Such an approach can be powerful and robust. However, it remains difficult to define patterns beyond a certain size in order to avoid incorrect phrase expansion. This complexity does not really constitute a problem when modeling possible formulations of answers, because, in a QA system, patterns have to match existing sentences. Some work like YAG (McRoy, Channarukul and Ali, 2000) and TG/2 (Busemann, 1996) propose a definition language for patterns in order to cover a small sublanguage at different degrees of sophistication. Their formalism deals with canned text (a direct mapping), templates or grammar rules. YAG also includes a general-purpose, template-based grammar for a core fragment of English.

Nevertheless, few generation systems have to deal with the necessity of a great flexibility. The STORYBOOK system (Callaway and Lester, 2001) was conceived to reproduce either the variety or complexity of naturally occurring stories. It performs surface realization with integrated formatting to produce narrative prose similar to the one found in stories written by humans. However, such a system is far too complex with respect to the problem of modeling local variations in sentences. We will see that information extraction systems have abandoned such a general approach for developing specific patterns.

What can be learned from work in the generation domain, and can be reused in QA systems, is the methodology that consists of deciding which conceptual

primitives have to be generated, and which patterns to associate with them to produce a natural realization. For instance, McKeown (McKeown, 1985) defined five ways for providing definitions, found from dictionary and encyclopedia texts: a) identifying the object as a class member (giving the hypernym); b) presenting the constituents of the object (giving the meronyms); c) giving the object attributes; d) using an analogy; e) giving examples.

2.2.2 Information Extraction Field

Information extraction (Gaizauskas and Wilks, 1997) automatically extracts pre-specified sorts of information from natural language texts, typically newswire articles. Information extraction can be viewed as a template filling task. IE grew very rapidly from the late 1980's when DARPA, the US defense agency, funded the MUC program, the major competition in IE. Systems had to extract information about a specified domain and provided instantiated templates. In Muc-4 (ARPA, 1992) for example, the goal was to extract information in the terrorism domain and in MUC-6 (ARPA 96) in the management succession domain.

IE systems are complex, usually consisting of many components. The generic IE system description provided by Hobbs (Hobbs, 1993) allows grasping their main processing stages. Most IE systems perform the following functions:

- Text Zoner: turns a text into a set of text segments.
- Preparser: tries to identify small-scale structures from a sequence of lexical items.
- Parser: its input is a sequence of lexical items, and small-scaled structures (phrases); its output is either fragments of a parse tree or a complete tree.
- Fragment Combiner: tries to turn results of the parser into a semantic representation of a sentence.
- Semantic Interpreter, Coreference Resolution and Template Generator are the last modules, dealing with semantic disambiguation and anaphora resolution at the discourse level before generating the result in a template.

The processes we are interested in are the preparser and the parser. These two steps aim at preparing the semantic interpretation of sentences. Preparser recognizes small-scaled structures that can be recognized with high reliability, as noun groups or verb groups, appositives that can be attached to a noun, as for genitives and “of” prepositional phrases. It also achieves named entity recognition. The preparser usually identifies small-scaled structure by finite-state pattern matching. The parser tries to produce a parse tree of the entire sentence. However, full-sentence parsing of unrestricted texts remains difficult. Thus many IE systems have abandoned it to the benefit of shallow parsing (we will discuss shallow parsing in section 5) or domain dependant, finite-state pattern matching, trying to locate within the sentence various patterns that are of interest for the application. This break is illustrated by SRI who shifted from the TACITUS system, with its generic text understanding approach, to a dedicated system, FASTUS (Hobbs *et al.*, 1996) that only focuses on elements relevant for the application.

Template filling originated in a project in the mid-60's (Sager, 1981). Templates were defined for extracting information in specialized domains, for instance the medical domain.

A template is defined by the identification of three elements:

- the objects which interacts,
- the relationships representing the interaction between objects,
- the features which are specific to the objects/relationships.

Identification of objects refers to named entity identification. These entities are organizations, person names, currencies, locations, times and dates. Systems in MUC competitions have obtained performances over 90% in this task. QA requires identification of new classes of entities, wider and also more specialized (see (Hovy *et al.*, 2001a) for a detailed hierarchy of answer types).

Interactions between objects can be defined by verb/subject/object relationships in sentences. Features that combine syntactic, semantic and lexical information enable their recognition from sentences. An example of extraction pattern for filling a template would be "< PERSON > was < killed/murdered >".

Extraction patterns were mainly created by experts, even if some work aims at acquiring them automatically, without previous annotations (Riloff, 1996; Yangarber and Grishman, 2000; Poibeau, 2002). However extraction patterns remain very specific to a task, i.e. information sought in a known domain.

The IE task is very close to the answer extraction problem. QA systems have to discover what is searched from questions (equivalent to choosing a template) and specify how to recognize the information (equivalent to the definition of features in terms of lexical, syntactic and semantic criteria for the definition of objects and their relationships), even if the answer to provide (equivalent to a template slot) is more limited.

3. ARCHITECTURE OF QALC

The QALC architecture (see Figure 1) is quite classical. Its presentation aims at giving a global vision of our solution to the QA problem and positioning the different processes relatively to each other. The analysis of questions relies on a shallow parser. It intends to extract the following features: a) the syntactic type of the question, b) the question focus, a noun that is likely to be present in sentences that contain the answer, and its noun phrase, c) the answer type, a named-entity tag.

In order to select the best documents from the results given by the search engine and to locate the answers inside them, we work with terms and their variants, i.e.

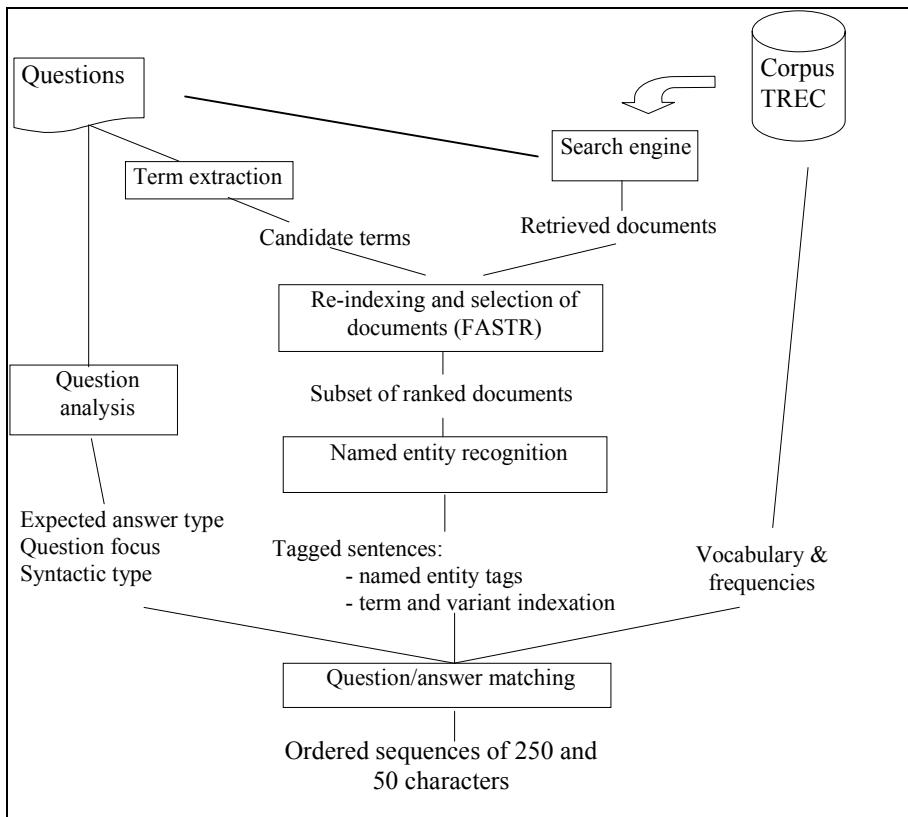


Figure 1. The QALC system.

morphologic, syntactic and semantic equivalent expressions. A term extractor has been developed, based on syntactic patterns that describe complex nominal phrases and their subparts. These terms are used by FASTR (Jacquemin, 1999), a shallow transformational natural language analyzer that recognizes their occurrences and their variants. Each occurrence or variant constitutes an index that is subsequently used in the processes of document ranking and question/document matching.

Documents are ordered according to a weight computed in function of the number and the quality of the terms they contain. This selection finds its justification during application of the last processes which consist of recognizing named-entities and analyzing each sentence to decide whether it is a possible answer or not. As such processes are time consuming, we attempt to limit their application to a minimal number of documents. Named entities are then recognized in the selected documents.

Finally, the question/answer matching module uses the data extracted from the questions and the documents by the preceding modules for ordering document sentences. Then QALC searches for the precise answer in the selected sentences, using different strategies according to the existence and the nature of the answer type.

4. TERMINOLOGICAL VARIATION

In QALC, terminological variation is used in order to select a sub-set of documents from the results of a search engine. Our selection process will prefer documents where multi-word terms (or variants of them) are near one another from documents containing the same words scattered throughout. Candidate terms are extracted from questions, and QALC can deal with single and multi-word terms.

4.1 Term Extraction

For automatic acquisition of terms from questions, we use a simple technique of filtering through patterns of part-of-speech categories. No statistical ranking is possible because of the small size of the questions from which terms are extracted. First, questions are tagged by the *TreeTagger* (Schmid, 1999). Patterns of syntactic categories are then used to extract terms from the tagged questions. They are very close to those described in (Juteson and Katz 1995), but we do not include post-posed prepositional phrases. The pattern used for extracting terms is:

$$((((JJ \mid NN \mid NP \mid VBG)) \ ? \ (JJ \mid NN \mid NP \mid VBG) \ (NP \mid NN))) \mid (VBD) \mid (NN) \mid (NP) \mid (CD))$$

Where NN are common nouns, NP proper nouns, JJ adjectives, VBG gerunds, VBD past participles and CD numeral determiners, and the operators are those of regular expression syntax.

The longest string is acquired first and substrings can only be acquired if they do not begin at the same word position as the superstring. For instance, from the sequence *nameNN ofIN theDT USNP helicopterNN pilotNN shotVBD downRP*, the following four terms are acquired: *US helicopter pilot*, *helicopter pilot*, *pilot*, and *shoot*.

The acquisition mode chosen for terms amounts to consider only the substructures that correspond to an attachment of modifiers to the rightmost constituents (the closest one). For instance, the decomposition of *US helicopter pilot* into *helicopter pilot* and *pilot* is equivalent to extracting the sub-constituents of the structure [*US [helicopter [pilot]]*].

4.2 Variant Recognition through FASTR

The automatic indexing of documents is performed by FASTR (Jacquemin, 1999), a transformational shallow parser for the recognition of term occurrences and variants. Terms are transformed into grammar rules and the single words building these terms are extracted and linked to their morphological and semantic families.

The *morphological family* of a single word w is the set $M(w)$ of terms in the CELEX database (CELEX, 1998) which have the same root morpheme as w . For instance, the morphological family of the noun *maker* is made of the nouns *maker*, *make* and *remake*, and the verbs *to make* and *to remake*.

The *semantic family* of a single word w is the union $S(w)$ of the *synsets* of WordNet 1.6 (Fellbaum, 1998) to which w belongs. A synset is a set of words that are synonymous for at least one of their meanings. Thus, the semantic family of a word w is the set of the words w' such that w' is considered as a synonym of one of the meanings of w . The semantic family of *maker*, obtained from WordNet 1.6, is composed of three nouns: *maker*, *manufacturer*, *shaper* and the semantic family of *car* is *car*, *auto*, *automobile*, *machine*, *motorcar*.

Variant patterns that rely on morphological and semantic families are generated by meta-rules. They are used to extract terms and variants from the document sentences in the TREC corpus. For instance, the following pattern, named NtoSemArg, extracts the occurrence *making many automobiles* as a variant of the term *car maker*:

$$\text{VM}(\text{'maker'}) \text{ RP? PREP? (ART (NN|NP)? PREP?) ART?(JJ | NN | NP} \\ \text{| VBD | VBG)}^{0:3} \text{ NS}(\text{'car'})$$

Where RP are particles, PREP prepositions, ART articles, and VBD, VBG verbs. VM('maker') is any verb in the morphological family of the noun *maker* and NS('car') is any noun in the semantic family of *car*.

Relying on the above morphological and semantic families, *auto maker*, *auto parts maker*, *car manufacturer*, *make autos*, and *making many automobiles* are extracted as correct variants of the original term *car maker* through the set of metarules used for the QA-track experiment. Unfortunately, some incorrect variants are extracted as well, such as *make those cuts in auto* produced by the preceding metarule.

4.2 Document Selection

The output of NLP-based indexing is a list of term occurrences composed of a document identifier d , a term identifier – a pair $t(q,i)$ composed of a question number q and a unique index i – a text sequence, and a variation identifier v (a metarule). For instance, the following index:

LA092690-0038	t(131,1)
making many automobiles	NtoVSemArg

means that the occurrence *making many automobiles* from document $d = \text{LA092690-0038}$ is obtained as a variant of term $i = 1$ in question $q = 131$ (*car maker*) through the variation NtoVSemArg given in Section 0.

Each document d selected for a question q is associated with a weight. The weighting scheme relies on a measure of quality of the different families of variations described by Jacquemin (1999): non-variant occurrences are weighted 3.0, morphological and morpho-syntactic variants are weighted 2.0, and semantic and morpho-syntactico-semantic variants are weighted 1.0.

Since proper names are more reliable clues than common names, each term $t(q,i)$ receives a weight $P(t(q,i))$ between 0 and 1.0 corresponding to its proportion of proper names. For instance, *President Cleveland's wife* is weighted $2/3 = 0.66$. Since another factor of reliability is the length of terms, a factor $|t(q,i)|$ in the weighting formula denotes the number of words in term $t(q,i)$. The weight $W_q(d)$ of a query q in a document d is given by the following formula (1). The products of the weightings of each term extracted by the indexer are summed over the indexes $I(d)$ extracted from document d and normalized according to the number of terms $|T(q)|$ in query q .

$$W_q(d) = \sum_{(t(q,i), v) \in I(d)} \frac{w(v) \times (1 + 2P(t(q,i))) \times |t(q,i)|}{|T(q)|} \quad (1)$$

For each query q , the 200 best ranked documents retrieved by the search engine are processed. Our studies (Ferret *et al.*, 2001) show that 200 is a minimum number allowing that almost all the relevant documents are kept. Mainly two types of weighting curves are observed for the retrieved documents: curves with a plateau and a sharp slope at a given threshold (Figure 2.a) and curves with a slightly decreasing weight (Figure 2.b). Questions in these figures come from the TREC8 data.

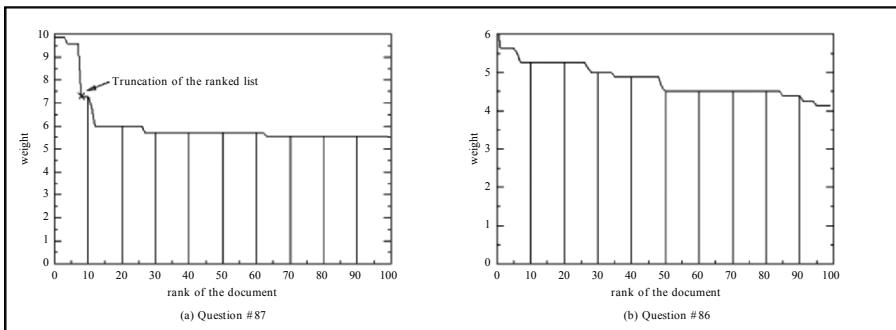


Figure 2. Two types of weighting curves.

The edge of a plateau is detected by examining simultaneously the relative decrease of the slope with respect to the preceding one, and the relative decrease of the value with respect to the preceding one. The following algorithm is used for calculating the cut-off threshold i_0 associated with the weighting scheme W of a given query q :

$$\text{If } \frac{Wq(d2)}{Wq(d1)} \leq 0.5 \text{ then } i_0 = 2$$

$$\text{else } i_0 = \min \left\{ \begin{array}{l} i \in \{3 \dots 100\} : \left(\frac{Wq(di) - Wq(di-1)}{Wq(di-1) - Wq(di-2)} \right) \geq 2 \\ \quad \wedge \frac{Wq(di)}{Wq(di-1)} \leq 0.8 \end{array} \right\} \cup \{100\} \quad (2)$$

Through this method, the threshold i_0 is 8 for question 87 (*Who followed Willy Brandt as chancellor of the Federal Republic of Germany?* Figure 2.a) and 100 for question 86 (*Who won two gold medals in skiing in the Olympic Games in Calgary?* Figure 2.b). As indicated by Figure 2.a, there is an important difference of weight between documents 8 and 9. The weight of document 8 is 9.57 while the weight of document 9 is 7.29, because the term *Federal Republic* only exists in document 8. This term has a higher weight because it is composed of two proper names.

Finally, the system retains the i_0 best ranked documents with a minimum number set to 20 and a maximum number set to 100.

4.3 Results and Evaluation

Document selection relies on a quantitative measure, i.e. the document weight, whose computation is based on syntactic and semantic indexes, i.e. the terms and the terminological variants. Those indexes allow the system to take into account words as well as groups of words and their internal relations within the documents. The following examples, issued from selected documents for the TREC9 QA task, show what kind of indexes are added to the question words.

For the question 252, *When was the first flush toilet invented?*, one multi-word extracted term is *flush toilet*. This term is marked by FASTR when recognized in a document, but it is also marked when a variant is found, as for instance *low-flush toilet* in the following document sentence where *low-flush* is recognized as equivalent to *flush*:

Santa Barbara, Calif., is giving \$80 to anyone who converts to a *low-flush toilet*.
 00252.01 flush toilet[JJ][NN] low-flush[flush][JJ] toilet[toilet][NN] 1.00

In all the given examples, after the identification number of the term, the reference term appears, made of the lemmatized form of the words and their syntactic category, followed by the variant found in the sentence, with each word, its lemmatized form and its category, and finally its weight.

In the example above, the term found in the sentence is equivalent to the reference term and thus, its weight is equal to 1.00.

Two multi-word terms extracted from *Who thought of teaching people to tie their shoe laces?*, question 255, are *teach people* and *shoe lace*. In the first example, the morphological variant *teaches* as VB is found for *teaching* as VBG, and *think* as VB for *thought* as VBD.

You can only say, ‘ I forgot ’ if you have made an effort to notice. Lapp says she *teaches people* to *think* of their mind as a camera.

00255.00 think[VBD] think[think][VB] 0.38

00255.00 teach people[VBG][NN] teaches[teach][VB] people[people][NN] 0.75

One can notice that we favor variants of multi-word terms that we consider more reliable than variants of single-word terms.

In the second example, the morphological variant *laced* as VBD is found for *lace* as NN, and the syntactic variant *past-participle+name* is found for *name+name*: as a result, *laced shoes* is found for the term *shoe lace*.

Hepburn, dressed in black slacks, a black turtle neck and sensible *laced shoes*, was swamped with people seeking to have their picture taken.

00255.02 people[NN]people[people][NN] 0.38

00255.03 shoe lace[NN][NN] laced[lace][VBD] shoes[shoe][NN] 0.50

In this case, two variations entail a term weight lower than the weight resulting from a single such variation as in the first example.

The last example shows a semantic variant. *Salary* is a term extracted from the question 337, *What’s the average salary of a professional baseball player?*. The semantic variant *pay*, retrieved from WordNet, was recognized in the following sentence:

Did the NBA union opt for the courtroom because its members, whose *average pay* tops \$500000 a year, wouldn’t stand still for a strike over free agency?

00337.01 salary[NN] pay[pay][NN] 0.25

00337.00 average [JJ] salary[NN]average[average][JJ] pay[pay][NN] 0.40

In order to evaluate the efficiency of the selection process, we proceeded to several measures. We apply our system on the material given for the TREC8 evaluation, one time with the selection process, and another time without this process. At each time, 200 documents were returned by the search engine for each of the 200 questions. When selection was applied, at most 100 documents were

selected and subsequently processed by the matching module. Else, the 200 documents were processed. The system scored 0.463 in the first case, and 0.452 in the second case. These results show that the score increases when processing less documents above all because many relevant documents are kept, while less irrelevant documents provide less noise.

The benefit from performing such a selection is also illustrated by the data computed on the TREC9 results, given in Table 1. We see that the selection process discards a lot of documents for 50% of the questions (340 questions are processed from less than 100 documents). QALC finds the correct answer more often and in a better position for these 340 questions than for the 342 remaining ones. The average number of documents selected, when there are less than 100, is 37. These results are very interesting when applying such time-consuming processes as named entity recognition and question/sentence matching. Document selection will also enable us to apply syntactic and semantic sentence analysis later on.

Table 1. Evaluation of the ranking process.

Number of documents selected by ranking	100	<<100
Distribution among the questions	342 (50%)	340 (50%)
Number of correct answers	175 (51%)	340 (50%)
Number of correct answer at rank 1	88 (50%)	128 (64%)

5. ROBUST PARSERS AND THEIR APPLICATION TO QUESTION-ANSWERING TASKS

As syntactic parsers currently exist, it is natural to use them since some syntactic features are useful to solve our problem. We will thus present the syntactic features suitable for QA. After a general presentation of some parsers based on different approaches, we will illustrate their behavior on some kinds of questions and how we adapted the parser we choose, IFSP (Aït-Mokhtar and Chanod, 1997). Finally, we will detail the QALC feature extraction process from the syntactic analysis of the question.

5.1 Advantages of Syntactic Analysis to QA Application

Question analysis is performed to infer features from questions in order to use them for sentence selection and extraction of potential answer. Basically, question analysis allows:

- the prediction of the type of the answer (for instance, Person, Number...)
- the determination of the important question words, i.e. those that should be present in the answer.

However, simply determining important question words is not always sufficient for finding the right answer: it is sometimes necessary to take into account the relations between these words. In the example shown in Figure 3, using only the question words leads QALC to find two possible answers. When considering the relations

between the words, the answer has to be a person that must be *an astronaut + Russian + the first to walk in space*. In the second answer, the noun phrase *the first US woman* entails the rejection of this second sentence because of the adjective *US*. Some semantico-pragmatic knowledge might also be available to detect that the two adjectives *Russian* on the one hand and *US* on the other hand, are in contradiction, which would not be the case with *Russian* and *white* or *black*.

Question: *What was the name of the first Russian astronaut to do a spacewalk?*

Answer 1: The broad - shouldered but paunchy Leonov, who in 1965 became **the first man** to walk in space, signed autographs. He drew a dove beside his name for Amanda Clark, 8, of Altadena and pinned on his lapel a Russian-language button from a well-wisher reading . Let us ...

Answer 2: Sullivan was **the first US woman** to perform a spacewalk as she and fellow crewman David Leetsma demonstrated satellite refueling ...

Figure 3. Examples of Candidate Answers.

A syntactic parse produces a segmentation of the sentence, which contains the syntactic structures (also called constituents or chunks) of the sentence: the noun phrases, the verbal phrases, the prepositional phrases, etc... It also produces the dependency relations that exist between words or constituents. Most often these dependency relations are syntactic relations, like subject, adjective, object... The useful relations between the question terms are expressed in these dependency relations (Figure 4).

Question: *What was the name of the first Russian astronaut to do a spacewalk?*

Segmentation: [SC [NP What NP] : v was SC] [NP the name NP] [PP of the first Russian astronaut PP] [IV to do IV] [NP a spacewalk NP]?

Dependency relations: SUBJECT (what, was), ADJECTIVE (first, astronaut), ADJECTIVE (Russian, astronaut), etc.

Figure 4. Example of question syntactic analysis.

Once a candidate sentence has been selected, the dependency relations can also be useful for extracting the answer from this candidate. The dependency links between the question phrases give some clues on the function of the equivalent terms in the answer sentence, as we can see it in the following example (Figure 5) also in (Hovy, Hermjacob & Lin, 2001b). In this example, the question analysis provides the syntactic function of the different noun phrases, the interrogative pronoun *Who* and *Lee Harvey Oswald*, so in the candidate sentences one searches for a person, this person having to be the grammatical subject of the verb *kill* (or a synonym of it), and the object of this verb must be *Lee Harvey Oswald*. So the right answer is *Jack Ruby* and not *J.F Kennedy*.

Question: *Who killed Lee Harvey Oswald?*

Question Syntactic Analysis: Subject (who, kill) Object (kill, Oswald)

→ Answer's Function = Subject

Answer: “*Jack Ruby, who killed J.F Kennedy assassin Lee Harvey Oswald*”

Answer Syntactic Analysis: Subject (Ruby, kill), Object (kill Oswald), Noun-Modifier (assassin, Oswald), Noun-Modifier (Kennedy, assassin),

Figure 5. Utilization of dependency relations for answer extraction.

5.2 What kind of Syntactic Parser?

Over the past years, syntactic parsing has seen an important development. Whereas the first syntactic parsers were developed in order to recognize “linguistic

phenomena” in ideal sentences, today, syntactic parsers are “realist”. They try to parse real-life corpora, which can contain ungrammatical sentences, but also specific phenomena such as dialogs, HTML links, XML tag... The priority of these parsers, called “Robust Parsers”, is robustness: a measure of the ability to return a syntactic parse (which can be minimal) independently of the distance between the effective input material and the type of material for which the parser was developed. A Robust Parser is generally determinist, incremental (gradual parse at different specific levels) and based on a corpus. It can recognize minimal structures, but also more complex structures with their dependency relations. Robust parsing relies on heuristics to extract a likely parse of the sentence; some parsers may even give several possible solutions, as it is often the case for prepositional phrase attachment.

It is *a priori* difficult to know which robust parser to use. We can distinguish two families of parsers: the symbolic/linguistic parsers, based on grammatical formalism and the probabilistic/statistical parsers, based on corpus learning (Collins, 1996). At first, we chose to use a linguistic parser for which we could add or modify rules. A probabilistic parser had the advantage of being corpus-trained but also the drawback that one has to build this corpus, and thus to solve the following problems: how to develop or find an annotated question corpus? how to determine its size? which annotations to use?

At the beginning of their development, the parsers were divided out among three categories, depending on the results they processed:

- constituents-based parsers (SCOL (Abney, 1996) – IPS (Wehrli, 1992)) which return the sentence segmentation. The sentence is first segmented into lexical units and then, into constituents. These parsers are incremental; each module uses the results of preceding module in the processing chain (Figure 6)

Question: *What was the name of the first Russian astronaut to do a spacewalk?*

Segmentation: [S [INT What] [V was] [NP [DET the] [N name]] [PP of [DET the] [ADJ first] [ADJ Russian] [N astronaut]] [IV to [V do]] [NP [DET a] [N spacewalk]] S]?

Figure 6. Example of sentence segmentation.

- dependency relations-based parsers (Link Grammar (Sleator & Temperley, 1991)), which return the word (or phrase) dependencies. The Link Grammar parser uses a specific dictionary in which each word corresponds to a formula that specifies the possible links with the other words. The aim of this parser is to find all the links between the sentence words respecting the properties of planarity (no crossing-links) and connectivity (Figure 7).

Question: *What is the highest dam in U.S?*

Dependency

relations:

What is the highest dam in U.S?

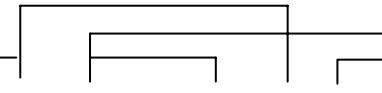


Figure 7. Example of dependency links.

- constituents and dependency relations based parsers (IFSP (Aït-Mokhtar & Chanod, 1997)), which return the sentence segmentation and the phrase dependencies (Figure 8). These parsers develop a non monotonic reasoning, i.e. some premature decisions could be refined or revisited. Diverse strategies can be used, but generally these parsers search for the minimal sentence segmentation, the simple syntactic relations (subject for instance) and then a greater complexity both in segmentation and syntactic relations.

Question: *Who invented a electric guitar?*

Segmentation: [S [NP Who] :v invented] [NP a electric guitar]?

Dependency relations: SUBJ(Who, invent), DOBJ(invent, guitar),
ADJ(electric, guitar)

Figure 8. Example of sentence segmentation and syntactic relations.

Nowadays, all parsers process the sentence segmentation and the dependency relations, even if the result formats and, above all, the way to obtain these results can be very different.

5.3 Which Robust Parser Use?

5.3.1 Question Processing

The parser we used for TREC-10 is IFSP (see section 5.2). Like most robust parsers, IFSP was developed for large-scale text applications such as information retrieval or terminology extraction. Hence, like the other robust parsers, it is not particularly well suited to parse interrogative forms. This is the reason why the

questions processing module we developed had to repair some parsing errors, and this has been done, to a certain extent, by writing specific rules.

In the following examples, we present some forms for which IFSP parsing was erroneous, and we compare the results with the tree other parsers presented in the previous section. It is worth noting that for the sake of time, it was not possible to perform a large-scale comparison on the whole TREC questions corpus²⁵. One should also notice that these tests have been explicitly made on sentences for which IFSP was failing, so it is not surprising that its results compared to those of the other parser were not the best.

- A verb recognized as a noun: in *What year did the Titanic sink?* the four tested parsers (IFSP, FIPS, LGP, SCOL) make the same mistake: *sink* is detected as a noun instead of a verb. In *Why does the moon turn orange?* only the LGP does not make the mistake and recognizes *turn* as the verb.
- The superlative recognized as a noun: in *What metal has the highest melting point?* Only, IFSP, produces an erroneous parse consisting in three noun phrases (one for *highest*, the second for *melting*, and the last for *point*).
- An adjective recognized as a noun: In *Who is the Prime Minister of Canada?* the use of a capital letter for the adjective *Prime* entails an error for IFSP and SCOL, the adjective is not recognized and the noun phrase *Prime Minister* is segmented in two distinct noun phrases (one for *Prime* the other for *Minister*).
- Some of the parsing errors can be repaired by writing ad-hoc rules, based on the results given by the parser. Thus, finally, the use of a shallow parser completed by specific rules allowed us to work out a question type module whose results were satisfying (see section 5.4).

Moreover, one attractive aspect in IFSP is that it returns all necessary information, without any need to modify or add any kind of process or knowledge. IFSP returns, on the one hand the constituent segmentation, on the other hand the dependency relationships between the terms, while FIPS, SCOL or LGP return a unique structure, in which the dependency relations are left anonymous. It is possible to obtain from SCOL or LGP the tagging of the dependency relations, but this would imply modifying the grammar.

Later on, several solutions may be adopted to face parser errors. Errors may sometimes be due to the morpho-syntactic tagging. This is the case with the *TreeTagger* used by SCOL, which does not recognize the verb in the two following questions *Why does the moon turn orange?* *How did Janice Joplin die?*. In SCOL, the morpho-syntactic tagging is not integrated to the parsing, so it is possible to make separate tests and to pinpoint the modules that should be repaired.

Another possibility is to choose a parser for which it is possible and fast to modify or write the syntactic rules necessary to the parsing of interrogative forms. SCOL for instance provides this possibility, so it could be a suitable parser. This is

²⁵ At LIMSI, the new project EASY/EVALDA, on the evaluation of syntactic parsers for French, is under development at the moment. A part of its corpus consists of 500 interrogative forms.

also the case with XIP (Ait-Mokhtar, Chanod and Roux, 2002) developed at Xerox. XIP is a new parser available for French and English, it has better performances than IFSP in processing interrogative forms, and it is possible to modify or to complete its grammar, which was not the case with IFSP.

Conversely some TREC participants chose a parser that could be trained on interrogative forms, those of TREC of course (1300 annotated forms are available) but also some extracted from Internet which represent a large real-life corpus. For Webclopedia, Hovy *et al.* (2001b) have trained their parser, CONTEXT, on approximately 1150 questions, achieving accuracy of approximately 89% for both recall and precision.

We could also adopt a whole new approach and work out a dedicated process that would integrate only the kind of parsing useful for the extraction of the information that is specific to the QA task (the question type, the focus... see section 5.4). However, we prefer keeping a generic approach.

5.3.2 Answer Processing

In the system QALC we implemented for TREC10, we used patterns corresponding to local analysis of the candidate answers, instead of a complete parsing of them. These patterns were written using the characteristics obtained from the question analysis module (see section 7). From now on, one of our priorities is to determine whether a robust parser is worth being used for the answer processing.

In opposition to question processing, the parsers have been prepared and even developed for a task such answer processing, indeed robust parsers process large amount of textual data within an acceptable time. These data can contain very complex sentences as well as ungrammatical ones; they can also contain errors due to acquisition, or transcription. It is however worth noting that if the parsing of constituents is most of the time correct, a full parsing processing the relationships between these constituents is rarely obtained.

Despite these difficulties some of the systems participating in the QA track, with a linguistic approach (by opposition to an IR approach), use a syntactic parsing and even sometimes a semantic one.

Hovy *et al.* (2001b) use CONTEXT a machine-learning based grammar parser whose results are 87.6% precision and 88.4% recall. Their system Webclopedia identifies the candidate answer passages and CONTEXT parses their sentences. Then, a matcher module compares the parse trees of the question and of the candidate sentence; but a second and independent matcher uses a word window on the answer text and seems to be useful when the answer parse tree is not complete.

The FALCON system (Harabagiu, Pasca and Maiorano, 2000) uses a statistical parser (Collins, 1996) that has a large coverage on real-word texts. FALCON then transforms the parse tree into a semantic representation that determines the phrase heads and the relations existing between them, but these relations are left anonymous.

5.4 *Natural Language Question Analysis in QALC*

Question analysis aims at extracting some features from questions that will be used in the answer module. In our system, the question analysis provides several pieces of information to the pairing module:

- a question type with which a list of patterns for extracting the answer will be associated
- an answer type which may be a named entity type or a semantic type to help locating the answer in a sentence
- a question focus to be used as possible criterion for the sentence selection and also to help answer extraction

5.4.1 *Question Type*

The question type corresponds to the question syntactic form. The detection of a question type gives us a clue to determine the different possible expressions of the answer (see section 7) and will also serve to extract the other question features. After studying the questions of TREC8 and TREC9 along with the sentences containing an answer, we found more than 80 syntactic forms of questions, such as “WhatbeNPofNP”; “HowADJ”, “WhatNPbeNP”...²⁶

Other QA systems also determine a question type, at different levels of complexity: for example, the Oracle’s system (Alpha, Dixon, Liao, & Yang, 2001) recognizes the wh-words (who, why, where...) to classify the questions. In comparison, (Soubbotin & Soubbotin, 2001) uses 35 more detailed question types. The number of question types generally depends on their utilization in the other modules.

5.4.2 *Answer Type*

Our question analysis module tries to assign each question an answer type which may be a named entity or a semantic type.

5.4.2.1 *Named Entity*

The module tries to find an answer type that corresponds to one or several named entity tags sorted by their specificity order, from the most specific to the most general type. The named entity tags are hierarchically organized within semantic classes (see Figure 9) such as Person, Organization, Location-City, Weight, etc...

²⁶ NP stands for Noun Phrase in all the following rules

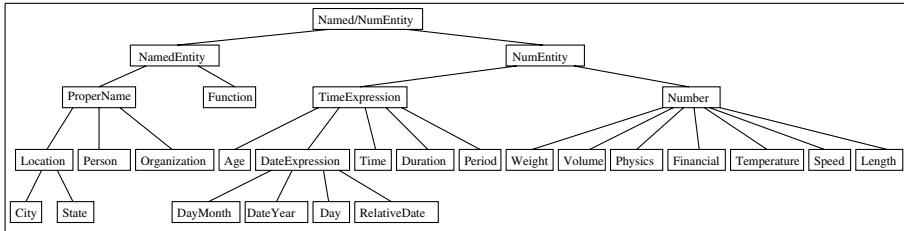


Figure 9. The hierarchy of tags.

To detect a named entity, we built several lexicons: each named entity type is associated with a lexicon that was constructed with the help of WordNet. For example, the LOCATION-CITY lexicon contains, among others, the following terms: capital, town, city, municipality ...

The detection of named entity type is performed by different rules, whose conditions are based on the syntactic structure of the question and the semantic classification, for example:

RULE 1: If Question Syntactic Form = *WhatBeNP1ofNP2*
and *the head of NP1* belongs to one of EN lexicons
then Named Entity = EN

The question *What is the capital of Bahamas?* matches the first condition of RULE 1, with NP1 corresponding to *the capital* and NP2 to *Bahamas*. And the head of NP1, *capital*, belongs to the LOCATION-CITY lexicon, so the answer type of this question is LOCATION-CITY.

As done in QALC, most systems determine the expected answer type by spotting pre-defined patterns in the questions. Prager *et al.* (2000), for instance, had 400 different patterns to identify about 50 types of answers. Using a statistical approach, Ittycheriah *et al.* (2000) based their classification of the answer types on a maximum entropy model. Clarke *et al.* (2000) use question categorization and pattern matching heuristics to determine whether a token is a valid candidate term, and thus to select the best scoring answer “snippets”. On the other hand, the FALCON system (Harabagiu *et al.*, 2000) used a large knowledge data base, holding a taxonomy of answer types extracted from WordNet hierarchies. Incidentally, question types are often drawn from the classification proposed in QUALM (Lehnert, 1977), one of the first QA systems.

When no named entity can be inferred from the question, the system tries to deduce a more general semantic type.

5.4.2.2 Semantic Type

A semantic type has to belong to a reference knowledge database, the WordNet lexical base for QALC. It means that the answer should be a noun phrase whose

head noun is a hyponym of this type. To deduce this type from questions, we wrote rules also based on the question syntactic form, such as:

**RULE 2: If Question Syntactic Form = *WhatNP*
then Answer Type = *NP-HEAD***

For example, the question *What metal has the highest melting point?* matches the question syntactic form of RULE 2 so the answer type of this question is *metal*. It is generally the noun following the interrogative pronoun.

These different rules were written after studying the questions of TREC8 and TREC9 along with the sentences containing an answer.

The Webclopedia system (Hovy *et al.*, 2001a) also calculates an answer type, which is called “qtarget type”, using for this approximately 140 semantic types. However, its hierarchy contains both named entity types and types of answer as definition. As far as we are concerned, we have separated name entity types from definition types since these two kinds of questions are solved in a very different way.

5.4.3 Focus

A question focus corresponds to a noun phrase that is likely to be present in the answer. For each question, we will determine a focus, the focus head (the head noun of the noun phrase) and the modifiers of the focus head (adjective, complement...) using a set of ordered rules that depend on syntactic and semantic knowledge.

For example, the following rules have been written for questions whose syntactic form is **WhatbeNP1ofNP2**:

**RULE 3: If NP1-HEAD belong to ABSTRACT lexicon
then **FOCUS = NP2**
 FOCUS-HEAD = *NP2-HEAD***

**RULE 4: If NP1-HEAD belongs to Person or Organization lexicon
then **FOCUS = NP1ofNP2**
 HEAD-FOCUS = *NP1-HEAD***

**RULE 5: If NP1-HEAD belongs to another entity named lexicon
then **FOCUS = NP2**
 FOCUS-HEAD = *NP2-HEAD***

Thus, if the question is *What is a group of frogs?* the condition of Rule 3 is satisfied, so the question focus is *frog* and the question focus head is *frog*. On the other hand, the question *Who is the president of United States?* does not match the condition of Rule 3 but the condition of Rule 4, so the question focus is *the president of United States* and the question focus head is *president*.

Generally, the principal criteria to determine the focus are the recognition of the question syntactic form and the type of the main NP. For example, if the question is *What city is the capital of France?*, the question focus is *France* (*capital* belongs to Location lexicon) and not *capital of France*; because the answers, *Paris*, *France* or *Paris in France* are more frequently encountered than *Paris*, *capital of France*. Or again in the question *What is the name of the researcher who owned the Calypso?*, the question focus is *researcher*, because *name* is an abstract word and in the answers, *A researcher Cousteau* is more often encountered than *Cousteau, a name of researcher*.

To find if the rule conditions are satisfied, the system uses the results of the robust parser improved with our modifications and also semantic knowledge via the WordNet lexical base. Our system is not the only one to use the notion of focus corresponding to one or several terms of the questions. Soubbotin and Soubbotin (2001), Ittycheriah, Franz and Roukos (2001), Alpha *et al.* (2001) and Hovy *et al.* (2001b), all use also this notion. In particular, Hovy identifies the relevant question terms and expands them using WordNet, and Soubbotin and Soubbotin recognize primary words (the words which are indispensable for sentence comprehension). However, the originality of our approach is in the selection of a noun phrase as focus along with the syntactic relations that exist between the focus head and the other question terms.

5.4.4 Results

The rules to find the question focus, the question type and the answer type were written from syntactic criteria provided by IFSP (Aït-Mokthar and Chanod, 1997) and semantic knowledge extracted from the WordNet lexical base. For the TREC10 questions, our question analysis module found about 85% of correct question focuses, 87% of correct semantic answer types and 90% of correct named entity types.

For easy detection of rules that entail errors, it is more convenient to study the results according to question type. For example, the question focus recognition is more difficult for the question type “WhobeNP” (for TREC10, 84% of correct focus), because the robust parser generates errors that we have to correct. Mistakes can be generated by named entity lexicons²⁷, which may be incomplete, by robust parser mistakes or by incomplete rules for some question types.

²⁷ A locution lexicon must also be available, to allow the system to resolve ambiguities, for instance the term *span* in *What is the average life span for a chicken?* and the term *expectancy* in *What is the life expectancy of a dollar bill?* are ambiguous while the locution *life span* and *life expectancy* are not and should trigger the search for DURATION entity named type.

6. QUESTION FOCUS RECOGNITION IN SENTENCES

6.1 Focus Variant Recognition

The question analysis module produces two types of information concerning the focus it has recognized: the head of the focus and a list of modifiers. QALC then tries to recognize this focus in the pre-selected documents. It first detects the head of the focus, and then identifies the noun phrase in which it is enclosed. To determine the delimiters of this noun phrase, we defined a local grammar for the NP in English. This grammar takes into account the tagging made by the *TreeTagger*.

For example, for the question 827:

Who is the creator of the Muppets?

The focus is *the creator of the Muppets*, with the head: creator.

In a document, we find the following NP: <JJ> late <NN> Muppets <NN> creator <NP> Jim <NP> Henson, which fits the expression: Adjective + Plural Noun + Noun + Proper Noun + Proper Noun.

We also look for NPs with synonyms of the question focus as heads. These synonyms are determined by FASTR. When QALC fails to determine a focus in the question, we decided to consider the proper nouns present in the question as possible focuses since they are also reliable clues for sentence selection. When all these NPs are delimited, we associate them with a score. This score takes into account the origin of the NP and the modifiers found in the question: when the NP contains the modifiers present in the question, its score is increased. The highest score is obtained when all of them are present. In the example of question 827, the score is high since the NP has been obtained directly from the question focus, all the significant words of the focus are present: *creator* and *Muppets*.

When the NP is obtained with a synonym of the head of the focus, the score is slightly decreased, and even more when it is obtained via a proper noun. The scoring algorithm also takes into account the ratio between the number of words present in the question phrase and in the document noun phrase.

For example the score assigned to the NP *their copy of the 13th century Magna Carta* obtained for the question *Which king signed the Magna Carta?*, has a lower score because it has not been obtained from the focus (*king*), but from the proper noun *Carta*, even if it contains all the words of this proper noun phrase, *Magna* and *Carta*.

QALC applies this algorithm for each sentence of the pre-selected documents, thus detecting all the NPs with their associated score. We only keep the NP with the best score in each sentence, which in turn becomes one of the criteria that will be used by the module for sentence selection and the module for answer extraction.

In order to evaluate the relevance of this criterion, we applied this algorithm on all the sentences given as right candidates for the questions of TREC 9. There were 13310 sentences answering to the 693 questions. 57.16% of them contained an NP

similar to the question focus. Overall, at least one focus is found for 89% of the question collection.

6.2 Sentence Selection

The selection of a set of sentences that may contain the answer to a question is based on the following principle: QALC inspects the selected documents for a question, sentence after sentence, and by means of a function that compares sentences according to their similarity with a question, it constantly keeps in a buffer the N^{28} sentences that are the most similar to the question. This comparison relies on the features of the question that have been identified in the sentences of the selected documents by the linguistic modules of QALC. These features are:

- terms;
- focus;
- named entities.

A specific similarity score is computed for each of these features. The score for terms only takes into account single-word terms with no variation. Certainly because of the rough way QALC makes use of terms and the recall level of FASTR, our experiments show that taking multi-word terms or variant terms into account has a negative impact on results. The term score is given by adding the weights of the terms of the question that are in the document sentence. The weight of a term integrates its normalized information with regards to a part of the TREC corpus and whether or not it is a proper noun.

The term score is linearly combined with the focus score (see Section 6.1) and the resulting score constitutes the first criterion for comparing two document sentences $S1$ and $S2$: if $S1$ has a combined score much higher than $S2^{29}$ ten $S1$ is ranked on top of $S2$. Otherwise, the named entity score is used according to the same principle as for terms. It evaluates the extent to which a named entity in a sentence can fit the target of a question when the expected answer is a named entity. This measure relies on the distance between their two types according to the named entity hierarchy of QALC (see section 5.4.2).

When the two preceding criteria are not decisive, the first criterion is used once again but with a smaller threshold for the difference of scores between two sentences. Finally, if there is still an uncertainty, QALC ranks first the sentence that has the shortest matching interval with the question. This interval corresponds to the shortest part of the sentence that gathers all the single-word terms of the question that were recognized in it.

Named Entities play a secondary role in the selection process, as we first select sentences according to the maximum number of terms common with the question,

²⁸ N is at least equal to 5. The selected sentences are ranked according to their similarity to the question.

²⁹ « Much higher » means that the difference of the scores for $S1$ and $S2$ is higher than a fixed threshold.

contrarily to (Prager *et al.*, 2000). Clarke *et al.* (2000) base their QA system on passage retrieval techniques, rather than on IR techniques and they directly select passages from the whole corpus. Their selection only takes into account the terms extracted from the question. We considered that a sentence with too few terms, even with a named-entity, is not really reliable and we prefer then a NIL answer. As we can see in table 2, section 7.6, our selection process was not performing and we lost a lot of correct sentences.

7. ANSWERS EXTRACTION

When the answer corresponds to a named entity, QALC, as all the TREC systems, mainly relies on this knowledge to find the answer in the selected sentence. Otherwise, QALC exploits the question category found by the question analysis module and its associated extraction patterns.

7.1 *Named Entity as Answer*

Named entities receive a type corresponding to one of the following general category: PERSON, ORGANIZATION, LOCATION (city or place), TIME EXPRESSION (age, date, time, etc.), NUMBER (physic or financial expression, etc.). They are recognized through a combination of lexico-syntactic patterns and significantly large lexical data. The three lists used for lexical lookup are CELEX (CELEX, 1998), a lexicon of 160,595 inflected words with associated lemma and syntactic category, a list of 8,070 first names, out of which 6,763 are from the CLR archive (CLR, 1998) and a list of 211,587 family names also from the CLR archive.

Contrarily to some systems (for example, Hovy *et al.*, 2001a), our categories are rather classic and defined in a way similar as for the MUC task. They all correspond to patterns defined independently from the QA task that can be reused in other contexts.

7.2 *Noun or Verb Phrase as Answer*

When the expected answer type is not a named entity, the QALC system locates the exact answer within the candidate sentence through syntactic extraction patterns. These patterns include in particular the noun phrase of the question focus head in the candidate sentence. This focus noun phrase has to be as similar as possible to the noun phrase of the focus in the question. The two noun phrases may be exactly the same but, usually, they are variants of each other. They may contain different words but they share at least the focus head.

7.3 *Determination of Answer Extraction Patterns*

The different extraction patterns were manually determined from corpus analysis. The corpus consisted of the questions and answers provided by the TREC8

and TREC9 evaluation campaigns. From this analysis, we observed that the syntactical structure of a question induces possible syntactical structures for the answer. A typical example of it is the syntactical structure of the direct answer to a question, which is an assertive rewriting of the question. For instance, the answer that our system found for the following TREC10 question is:

Example 1: Question 1008: *What is the Hawaii's state flower?*
 Answer: *Yellow hibiscus is the state flower of Hawaii*

The focus of the question, that has been determined by the rules of the question analysis module (see Section 5.4.3), is *Hawaii's state flower*, and the focus head is *flower*. The noun phrase *the state flower of Hawaii*, in the answer, is a syntactic variant of the focus. The answer of example 1 was extracted from the following candidate sentence: *Yellow hibiscus is the state flower of Hawaii, but Postrzech doesn't recommend them for evening luaus because they close at the end of the day.* The syntactic pattern used to extract this answer is the following:

NPanswer be NPfocus (1)

In this pattern, NPfocus is the noun phrase similar to the question focus within the candidate sentence, and NPanswer is a noun phrase that is supposed to contain the answer. This extraction pattern is very close to a direct answer.

It would be noted that, in this case, we can say *Yellow hibiscus is the state flower of Hawaii* as well as *The state flower of Hawaii is the yellow hibiscus*. Thus, the symmetric pattern,

NPfocus be NPanswer (2)

can also be used to extract an answer to the same type of question. This type of question asks for giving the name of the object that is described in the question. There is a relation “object-description”, between what is asked for and the focus that describes the object in the question. The possible answer strategies may thus consist of attributing a name to the described object, cf. pattern (2), as well as attributing a description to the named object, cf. pattern (1). It should be noted that, usually, the description contains the answer type. In the example 1, the focus head *flower* is a hypernym of the answer *hibiscus*.

As we just showed, some extraction patterns are closely derived from the syntactic structure of the question. Nevertheless, direct answer structures are not often found in documents, but rather variants of them. For instance, another answer to the question 1008 found in the documents was:

Example 2: Question 1008: What is the Hawaii's state flower?

Answer: *the state bird, and the Hibiscus the state flower*

This answer is extracted from the sentence *Utah made the allosaurus the state fossil, and Hawaii's Legislature approved making the nene, a native goose, the state bird, and the Hibiscus the state flower*. In this case, the answer (*Hibiscus*) and the focus (*flower*) are in apposition. The answer is a noun which is the object of the implied verb *making*. It means that the description *state flower* is attributed to the flower named *Hibiscus* in accordance with the previous answer examples. Indeed, the relation object-description between the answer and the focus, inferred from the question, may be expressed by different syntactic variants. As a result, the syntactic category of the question, that reflects what is exactly asked for (in our example, the name of a described object), determines some requirements on the answer-focus syntactic structure in the candidate sentences. Therefore, in the construction of extraction patterns from the answer corpus, we were looking for the syntactic structures that could express the different answer-focus relations underlying the questions.

7.4 *The Question Categories*

We saw in section 5.4 that the questions were parsed in order to get information on the expected answer. A syntactic category is attributed to each question, depending on the syntactic form of the question. However, the same type of question may be expressed by different syntactic forms. For instance, the following questions match a request about a location:

Question 725: *What is the U.S. location of Procter & Gamble corporate offices?*

Question 727: *Where is Procter & Gamble based in the U.S.?*

The phrase *What is the location of* induces a location request, as well as *Where do*. The answer to both questions is a named entity of <LOCATION> type. In both questions, a location relation links an object to the place it is located in, but this relation is expressed through two different syntactic forms, which are two syntactic variants of a same request, and leads to the attribution of the same question category and the same answer type (a named entity type).

Question categorization according to request type is therefore useful for knowing which extraction patterns will be tried. Patterns convey the relation that supports a category, and thus are supposed to be specific to this category. Unfortunately, this is not true for all patterns. Some of them may pertain to more than one category. For instance, the pattern “NPfocus be NPanswer” is relevant for most of the categories which begin by *What be*. This pattern may express various relations such as a description-object relation (and the symmetric object-description relation), or an object-category relation, or a concept-definition relation.

Each request type may be formulated by one or more syntactic forms. Nevertheless, some general syntactic forms correspond to more than one request type, and therefore have to be refined as soon as the answer extraction patterns differ. For instance, the syntactic form “What be NP?” corresponds to two different request types, on the one hand a request for the definition of a concept, and, on the other hand, a request for the name of a described object. Questions from example 1 are all requests for a definition expressed by the syntactic form “What be NP ?”

Example 1: Question 912: What is epilepsy?

Answer: a person has a seizure disorder or epilepsy

Pattern: NPanswer or NPfocus

Question 917: What is bipolar disorder?

Answer: manic-depressive illness (also called bipolar disorder)

Pattern: NPanswer (adverb called Npfocus

Question 980: What is amoxicillin?

Answer: antibiotics such as amoxicillin

Pattern: NPanswer such as NPfocus

Question 354: What is a nematode?

Answer: the nematodes are voracious bug killers

Pattern: NPfocus be NPanswer

Patterns associated to each answer show that there is more than one way to answer to a definition question. Conjunction *or*, within the answer to the question 912, indicates that the NPfocus (*epilepsy*) and the NPanswer (*seizure disorder*) have similar meanings³⁰. The phrase (*also called*, within the answer to question 917, has more clearly the same role as *or*. In both cases, the concept to be defined is defined by a synonymous phrase. On the other hand, the phrase *such as*, within the answer to question 980, shows a link between a concept and its hyponym. The concept is then defined by the category to which it belongs. The answer to question 354 shows a third way of answering to a definition request, by the function that the object has. We retrieve here the different possible answers to definition questions exhibited by Mc Keown (1985) (see section 2.2.1). Thus, different types of patterns, that match different types of links between the object to define (the question focus) and its definition (the answer), will be used for extracting the answer.

Example 2 shows the second type of request expressed by the syntactic form “What be GN?”, a request for the name of a described object.

³⁰ Obviously, it is not always the case. Conjunction *or* could indicate an alternative in another context.

Example 2: Question 1213: What is the brightest star?

Answer: Sirius, *the brightest star visible from Earth.*

Pattern: NPanswer, NPfocus

The superlative *brightest* indicates a definite phrase that points out a precise object. The syntactic form that expresses this type of request has then to be refined in “What be definiteNP?” in order to separate it from “What be NP?” which then corresponds to a definition request. Patterns with *also called* and *or* are not relevant for the “What be definiteNP?” category. It should also be noted that the answer is then an hyponym of the focus head *star*.

In QALC, we only make explicit the answer type when it corresponds to a named entity or to a semantic type that will be found in answer sentences. For other types such as definition or object-description, we based in QALC the selection of extraction patterns on the question syntactic category. These two kinds of information lead to different strategies for finding the answer, even if these strategies are complementary: there are different patterns to name a person, according to the characteristics given in the question that belongs to the focus.

7.5 Extraction patterns

The syntactic patterns for answer extraction that we defined from the corpus, always include the noun phrase of the focus and the noun phrase of the answer, which are usually connected by other elements such as comma, quotation marks, a preposition or even a verb. The only exception occurs when the answer is within the noun phrase of the focus. In this case, there is no connecting element between the noun phrase of the focus head and the noun phrase of the answer in the corresponding syntactical pattern. We distinguished three different pattern structures:

- (1) NPfocus Connecting-elements NPanswer
- (2) NPanswer Connecting-elements NPfocus
- (3) NPanswer-within-NPfocus

The answer of example 1 in section 7.3 (*Yellow hibiscus is the state flower of Hawaii*) corresponds to the pattern (2) with the verb *be* as connecting element. The answer of example 2 in the same section (*the state bird, and the Hibiscus the state flower*) corresponds to the pattern (3), since noun phrases in apposition without punctuation mark are processed by our system as a complex noun phrase.

The following example shows another connecting element between the noun phrase of the focus and the noun phrase of the answer in the syntactic extraction pattern. This example comes from our system run on the TREC10 questions.

Example 3: Question 1345: What is the most popular sport in Japan?

Answer: *baseball as the nation's most popular sport*

In this example, the connecting element is the preposition *as*. The answer has been extracted through the pattern “NPanswer as NPfocus” from the candidate sentence *Now, it is threatening to dislodge Japan’s stodgy baseball as the nation’s most popular sport*. The question focus was *the most popular sport* and the focus head *sport*. We can observe that the focus noun phrase in the answer is a variant of its counterpart in the question. In the candidate sentence, the relation answer-focus is also an object-description relation, the description being attributed to the name of the described sport, *baseball*. Moreover, *sport*, that is the focus head, is a hypernym of the answer *baseball*.

For all the categories we considered 24 extraction patterns. The number of patterns for each question category varies from 2 to 20, with an average of 10 patterns for each category. This approach led us to consider more generic categories than in (Soubbotin and Soubbotin, 2001) and in (Prager *et al.*, 2000), and more generic patterns that are not relative to specific formulations of a kind of request, as all the requests for an acronym, or the requests for a definition for example. Patterns are articulated around a pivot term depending on the question syntactic form, and instantiate some question terms, i.e. the focus terms and eventually the main verb, to produce a partial affirmative form.

The difficulty in finding extraction patterns varies according to the question type. This difficulty is partly due to the small number of some question types within the corpus. For example, there are few *Why* questions (4) and few *How verb* questions (4), such as *Why can’t ostriches fly?* (n° 315) and *How did Socrates die?* (n° 198). Moreover, answers to those questions can hardly be reduced to a pattern. But, for the most part, the difficulty in finding patterns is due to the syntactical diversity of the answers. We hardly found syntactical regularities in the answers to the “What NP be NP? “questions, such as *What format was VHS’s main competition?* (n° 426) or *What nationality was Jackson Pollock?* (n° 402) for instance. Depending on the answers, it is the first NP (*format* or *nationality*) or the second NP (*VHS* or *Jackson Pollock*) which plays the main role in the extraction pattern.

Some other QA systems used extraction patterns. Soubbotin & Soubbotin (2001) make patterns the core of their QA system. The indicative patterns they used are different from ours in two ways. First, indicative patterns do not use lemmas and syntactic categories, but sequence and combination of string elements and lists of words and phrases. Secondly, named entities are introduced in patterns. For instance, the pattern “city name; comma; country name” is associated to a location question such as *Where is Milan?*. The LCC system (Harabagiu *et al.*, 2001) uses the same type of pattern than those we described, but mainly for definition question type, for which too little information comes from syntactical dependencies. Hovy *et al.* (2001a) also defined some patterns that, according to its opinion, are too specific to be really useful.

7.6 Evaluation

We carried out an evaluation of the answers that the QALC system retrieved according to whether the question belongs to a category defined by a named entity

or not. This evaluation used the TREC10 corpus. First, the evaluation was carried out on the candidate sentences, and then on the 50-characters answers. Our goal was to measure the performance of the two modules processing the answer selection within the retrieved documents. Table 2 sums up the evaluation results of QALC.

Table 2. Results of QALC.

Questions	Number of questions	Number of correct answers (sentences)	Number of correct answers(50-char.)	Correct extraction (sentence>50-char.)
Named entity categories	229	96 (42 %)	82 (36 %)	85 %
Other categories	263	103 (39 %)	63 (24 %)	61 %

Table 2 shows that the QALC system brings better performances concerning the named entity categories, as this type of question usually includes a number of words which makes easier the selection of the candidate sentences, compared with the definition questions which often include just one word. Moreover, answer extraction is also facilitated by the named entity tagging of the documents.

8. CONCLUSION

Recent work on QA systems has shown that there is a significant difference between systems that operate in restricted domains but at a conceptual level – such as QUALM for instance – and those that come under the bag of words approach that is widely used in Information Retrieval (IR). Between these two types of system, we find the systems that make use of domain-independent knowledge, such as the WordNet database, and rely on general NLP components achieving tasks such as shallow parsing, named entity recognition, term extraction or question analysis. These systems build and process structured representations but unlike conceptual-level ones, their representations are close to the surface form of texts. As a consequence, tackling linguistic diversity, i.e., the fact that a concept or an idea may be expressed by many linguistic forms, is one of the important challenges for QA systems nowadays.

In this chapter, we have addressed the part of linguistic diversity that concerns terminological and syntactic variations and we have shown that taking into account these two kinds of variations is actually a mean for increasing the accuracy of a QA system. This observation is congruent with a more general trend emerging from TREC evaluations: the best QA systems are generally those that intensively make use of NLP components in order to bypass linguistic variations. This is one of the important results of the TREC evaluations about QA systems, especially because of the following observation: although the interest of NLP seems to be *a priori* evident for information retrieval tasks, it was never confirmed in an unquestionable way for text retrieval systems (Sparck Jones, 1999), even for basic processing such as morphological analysis (Hull, 1996).

Nevertheless, the reasons for the lack of success of NLP in IR systems concern QA systems too. The lack of robustness of NLP components and the fact that this brittleness is not taken into account by IR systems are two causes that are particularly important. When they work with the Web or with large collections of heterogeneous documents, which is a frequent case of use, IR systems cannot impose restrictions about the texts they process. The results of their possible NLP components are necessarily not perfect and their errors are closely linked to the type of the texts they process. But IR systems generally don't take into account the limits of their components and they use their results in the same way in all cases. As a consequence, when their overall results contain NLP components, they may be worse than those equipped with less sophisticated components.

The errors of NLP components can be classified into two classic categories: precision errors and recall errors.

Precision errors occur when an entity or a structure that is extracted from a text is not correct, as when a morpho-syntactic tagger sets a noun tag for a word while it is a verb (see Section 5.3.1). Recall errors take place when an entity or a structure that could be extracted from a text is not identified. This is typically the case when a named entity or a term is missed. This kind of error often occurs because a source of knowledge is too sparse. The fact that syntactic parsers generally do not take into account questions (see Section 5.3.1) is another example of this kind of error.

All errors pose real problems but recall errors are the more critical ones. When an entity or a structure is found but is not correct, auto-evaluation mechanisms can be set in some cases for detecting the error. This is more difficult for recall errors as no entity or structure that could trigger a control mechanism exists.

Several strategies were conceived and tested for reducing the effect of these errors and the resulting brittleness of NLP components and of the systems that use them. We distinguish four main strategies. The first two are local strategies that concern NLP components on their own and the last two are more global:

- A task is achieved by a general NLP component and a set of heuristics corrects its errors in the specific context of its use. This is the approach adopted in QALC for question analysis (see Section 5.3.1): a general syntactic parser allows QALC to identify relations between words and its errors, a part of them coming from a lack of rules for dealing with questions and another part of them resulting from the errors of its tagger, are corrected by a set of specific heuristics that are defined manually. The main drawback of this approach is its lack of generality: heuristics defined for a tool are generally not applicable to another tool achieving the same task. One way to lighten the work that must be achieved for a new tool is to resort to machine learning techniques. Transformation Based Learning (Brill, 1993) is particularly well suitable in that case since it was conceived for learning correction rules.
- A task is performed by a specialized NLP component that is specifically designed for the system in which it is used. This approach is adopted in QALC for the identification of the focus of a question in a document sentence (see Section 6.1). The module that achieves this task in QALC could have relied on a shallow parser for extracting noun phrases from sentences but using a basic set

of patterns turned out to be a more flexible method. More than the previous one, this strategy faces a lack of generality since a specific module must be built for each new context of use.

- A task is achieved in parallel by several modules that rely on different methods. Their results are then combined, generally by a kind of vote. This approach was successfully tested for tasks such as morpho-syntactic tagging (van Halteren, Zavrel and Daelemans, 1998) or speech recognition (Schwenk and Gauvain, 2000). It exploits the fact that several modules based on different methods differ in the errors they make. Thus, minority errors can be discarded by a simple vote. However, it is not clear in which cases this approach can be used. For the moment, it was mainly applied for decision tasks, i.e. tasks where the variety of possible errors is limited. It is not sure that it can be successful in a more general context. Moreover, for some kinds of NLP tasks, it is quite heavy to implement and computationally expensive too, even if it is easy to parallelize.
- Several modules that implement different levels of processing are available for achieving a task and the module that is actually used in a specific situation is dynamically chosen according to the characteristics of this situation. One important aspect of this approach is the ability for a system to determine the type of its inputs and the quality of its results. These two evaluations are complementary: the first one is used for selecting the module that is *a priori* the more suited to the current task and the second one is a way to correct this choice if it turns out that it was not correct. The evaluation of inputs is generally taken into account for questions in an accurate way by QA systems (see Section 5.4 for instance). This is not the case for documents, certainly because work about text typology (Biber, 1993) is not mature enough for being applied to QA systems. The evaluation of the results of a system is an active field but it is mainly done in a supervised way: a system is applied on a restricted set of inputs and its results are compared to a reference. On the other hand, little work has been done in this field about the ability of a system to auto-evaluate its results in order to adapt its processing. The FALCON system (Harabagiu *et al.*, 2000) implements this approach for QA: it first retrieves documents with simple requests, evaluate answers extracted from them and build more complex requests (with lexical or semantic expansion for instance) only when these answers are not satisfactory. FALCON distinguishes in this way three levels of processing for requests. In this approach, the brittleness of NLP components is a less important problem, as these components are used only when more basic solutions have already failed. Hence, they can only improve results.

The most widespread strategy of these four is certainly the second one. The last two are the rarer. Their complexity, especially for the last strategy, and their cost from a computational point of view are probably the causes of this rarity of use. Moreover the second point is particularly important for interactive applications or when a high number of requests must be simultaneously processed, as for search engines on Internet.

Nevertheless, the results of a system such as FALCON show that the most promising approach for QA systems, and perhaps more generally for IR systems,

consists in using NLP components and above all in controlling their use in order to apply them only when they are likely to improve results. This control can be performed only by developing the ability of QA systems to get information about the situation they face to and the quality of their results, which means achieving a kind of auto-evaluation.

9. ACKNOWLEDGEMENTS

The authors would like to thank Sasha Blair-Goldensohn for his attentive rereading and Patrick Paroubek for his remarks and for the fruitful discussions which helped us in making things clearer.

10. REFERENCES

- Abney, S. (1996), Partial Parsing via Finite-State Cascades. *Natural Language Engineering*, 2 (4), 337-344.
- Aït-Mokhtar, S. & Chanod, J-P. (1997). Incremental finite-state parsing. *Proceedings of Applied Natural Language, Washington, DC*.
- Aït-Mokhtar, S., Chanod, J-P. & Roux, C. (2002). Robustness beyond Shallowness: Incremental Deep Parsing. *Natural Language Engineering, special issue on Robust Methods in Analysis of Natural Language Data*, to appear.
- Alpha, S., Dixon, P., Liao, C. & Yang, C. (2001). Oracle at TREC 10: Filtering and Question-Answering. *Proceedings of Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds.
- Appelt, D., Hobbs, J., Bear, J., Israel, D., Kameyama, M., Kehler, A. et al. (1995) SRI International FASTUS system: MUC-6 test results and analysis. *Proceedings of the 6th Message Understanding Conference (MUC-6)*, Morgan Kaufman, pp. 237-248.
- ARPA. (1992). Advanced Research Projects Agency. *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. San Francisco, California, Morgan Kaufman.
- ARPA. (1996). Advanced Research Projects Agency. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. San Francisco, California, Morgan Kaufman.
- Barzilay, R. & McKeown, K. R. (2001). Extracting Paraphrases from a Parallel Corpus. *Proceedings of ACL-EACL '01*, Toulouse.
- Berri, J., Mollá Aliod, D. & Hess M. (1998). Extraction automatique de réponses: implémentations du système ExtrAns. *Proceedings of the fifth conference TALN 1998 (Traitement Automatique des Langues Naturelles)*, Paris, pp. 12-21.
- Berwick, R.C. (1991). Principles of principle-based parsing. In Berwick, R.C., Abney, S.P., & Tenny, C. (Eds.), *Principle-Based Parsing Computation and Psycholinguistics* (pp. 1-38). Kluwer Academic.
- Biber, D. (1993). Using Register-Diversified Corpora for General Language Studies. *Computational Linguistics*, 19 (2), 219-241.
- Brill, E. (1993). *A Corpus Based Approach To Language Learning*, PhD Dissertation, Department of Computer and Information Science, University of Pennsylvania.
- Busemann, S. (1996). Best-First Surface Realization, *Proceedings of the 8th International Workshop on Natural Language Generation*, Herstmonceux, Great Britain.
- Callaway, C.B. & Lester, J.C. (2001). Narrative Prose Generation. *Proceedings of the 8th IJCAI 2001, Seattle*.
- CELEX. (1998). Consortium for Lexical Resources, University of Pennsylvania. From http://www.ldc.upenn.edu/readme_files/celex.readme.html.
- Clarke, C.L.A., Cormack, G.V., Kisman, D.I.E. & Lynam, T.R. , Question answering by passage selection, *Proceedings of the Text retrieval conference, TREC9, Gaithersburg, MD*. NIST Eds.
- CLR (1998) Consortium for Lexical Resources, NMSUs, Eds., New Mexico. From <http://crl.nmsu.edu/cgi-bin/Tools/CLR/clrcat#D3>.
- Collins, M. (1996). A New Statistical Parser Based on Bigram Lexical Dependencies. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, ACL-96*, pp.184-191.

- Fabre, C. & Jacquemin, C. (2000). Boosting variant recognition with light semantics. *Proceedings of COLING 2000, Luxemburg*, pp. 264-270.
- Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press.
- Ferret, O., Grau, B., Hurault-Plantet, M., Illouz, G., Jacquemin, C. (2001). *Document selection refinement based on linguistic features for QALC, a question answering system*, Proceedings of the Euroconference on Recent Advances in Natural language Processing (RANLP), Tsigov Chark, Bulgaria.
- Gaizauskas, R. & Wilks, Y. (1997). *Information Extraction: Beyond Document Retrieval*. Technical report CS-97-10, Department of Computer Science, University of Sheffield, UK.
- van Halteren H., Zavrel J. and Daelemans W. (1998) Improving Data Driven Wordclass Tagging by System Combination, *ACL-COLING'98*, pp. 491-497.
- Harabagiu, S., Pasca, M., Maiorano, J. (2000). Experiments with Open-Domain Textual Question Answering. *Proceedings of Coling'2000, Saarbrucken, Germany*.
- Harabagiu, S., Moldovan, D., Pasca, M., Surdeanu, M., Mihalcea, R., Girju, R., Rus, V., Lactatu, F., Morarescu, P., Bunescu, R. (2001) Answering Complex, List and Context Questions with LCC's Question-Answering Server, *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds., pp. 355-361.
- Hobbs, J.R. (1993). The Generic Information Extraction System. *Proceedings of the Fifth Message Understanding Conference (MUC-5)* (pp. 87-91), Morgan Kaufman.
- Hobbs, Appelt, Bear, Israel, Kameyama, Stickel & Tyson (1996). FASTUS: A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text, in Roche and Schabes, eds., *Finite State Devices for Natural Language Processing*, MIT Press, Cambridge MA.
- Hovy, E. (1996). Language generation, overview. In *Survey of the State of the Art in Human Language Technology* (chap. 4.1). From <http://cslu.cse.ogi.edu/HLTsurvey/HLTsurvey.html>.
- Hovy, E., Hermjacob, U. & Lin C-Y., Ravichandran, D. (2001a). Towards Semantics-Based Answer Pinpointing, *DARPA Human Technology Conference (HLT)*, San Diego.
- Hovy, E., Hermjacob, U. & Lin C-Y. (2001b). The Use of External Knowledge in Factoid QA. *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds. pp. 644-652.
- Hull, D. (1996). Stemming algorithms: A case study for detailed evaluation, *Journal of the American Society for Information Science*, 47 (1), 70-84.
- Ittycheriah, A., Franz, M., Zhu, W-J. & Ratnaparkhi A. (2000), IBM's statistical Question Answering System. *Pre-proceedings of TREC9, Gaithersburg, MD*, NIST Eds, pp. 60-65.
- Ittycheriah, A., Franz, M. & Roukos, S. (2001). IBM's Statistical Question Answering System – TREC-10. *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds.
- Jacquemin, C. (1999). Syntagmatic and paradigmatic representations of term variation. *Proceedings of ACL '99*, pp. 341-348.
- Jacquemin, C. (2001). *Spotting and Discovering Terms through NLP*. Cambridge, MA: MIT Press.
- Justeson, J. & Katz, S. (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1, 9-27.
- Lehnert, W. (1977). Human and computational question answering. *Cognitive Science* 1, 47-63.
- Lin, D. & Pantel, P. (2001). Discovery of Inference Rules for Question-Answering. *Natural Language Engineering* 7 (4).
- McKeown, K. (1985). Discourse strategies for generating natural-language text, *Artificial Intelligence*, 27, 1-41.
- McRoy, S.W., Channarukul, S., & Ali, S.S. (2000). YAG: A Template-Based Generator for Real-Time Systems (System demonstration), *Proceedings of the First International Natural Language Generation Conference (INLG)*, Mitze Ramon, Israel.
- Poibeau, T. (2002). *Extraction d'information à base de connaissances hybrides*. Thèse de l'université Paris-Nord.
- Prager J., Brown, E., Radev, D. & Czuba, K. (2000), One Search Engine or two for Question-Answering, *Proceedings of the Text retrieval conference, TREC9, Gaithersburg, MD*. NIST Eds, pp. 250-254.
- Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text, *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI-96)*, pp. 1044-1049.
- Robin, J. (1994). *Revision-based generation of natural language summaries providing historical background: corpus-based analysis, design, implementation and evaluation*. Ph.D. Thesis. CUCS-034-94, Columbia University, Computer Science Department , New York, USA. 357p.
- Sager, N. (1981). *Natural Language Information Processing*, Addison-Wesley, Reading.

- Schmid, H. (1999). Improvements in Part-of-Speech Tagging with an Application To German. In Armstrong, S., Chuch, K.W., Isabelle, P., Tzoukermann, E. & Yarowsky, D. (Eds.), *Natural Language Processing Using Very Large Corpora*, Dordrecht: Kluwer Academic Publisher.
- Schwarz, C. (1988). The TINA Project: text content analysis at the Corporate Research Laboratories at Siemens. *Proceedings of Intelligent Multimedia Information Retrieval Systems and Management (RIAO'88)*, Cambridge, MA, pp. 361-368.
- Schwenk, H. & Gauvain, J.-L. (2000). *Improved ROVER using Language Model Information*. Proceedings of ISCA ITRW Workshop on Automatic Speech Recognition: Challenges for the new Millennium, pp. 47-52, Paris.
- Sleator, D.D. & Temperley, D. (1991). *Parsing English with a Link Grammar*. Technical report CMU-CS-91-196, Carnegie Mellon University, School of Computer Science.
- Soubbotin, M.M. & Soubbotin, S.M. (2001). Patterns of Potential Answer Expressions as Clues to the Right Answers. *Proceedings of the Text retrieval conference, TREC 10, Gaithersburg, MD*. NIST Eds.
- Spark Jones, K. & Tait, J. (1984). Automatic search term variant generation. *Journal of documentation*, 40 (1), 50-66.
- Spark Jones, K. (1999). The role of NLP in text retrieval. In T. Strzalkowski (Ed.) *Natural Language Information Retrieval* (pp. 1-24). Boston, MA, Kluwer.
- Wehrli, E. (1992). The IPS system. In C. Boitet (Ed.) *Coling-92, GETA*, pp. 870-874.
- Yangarber, R. & Grishman, R. (2000). Extraction Pattern Discovery through Corpus Analysis. Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000), Workshop: Information Extraction meets Corpus Linguistics.
- Zock, M. & Sabah, G. (2002). La génération automatique de textes. In M. Fayol (Ed.) *La production du langage*, Coll. Sciences du Langage, Hermès.

3

QUESTION ANSWERING AS INFORMATION RETRIEVAL

SENTENCE RANKING USING KEYWORDS AND META-KEYWORDS

EXPERIENCE WITH TREC-9 AND TREC-2001 USING PIRCS SYSTEM

Abstract: This paper describes our approach and experience with the question-answering tasks of TREC-9 and TREC-2001. Our approach employed techniques from IR, pattern matching and meta-keyword detection with little linguistic analysis and no natural language understanding. It involved the following four steps: 1) retrieval of top-ranked subdocuments using content keywords of a question as query; 2) weighting of sentences from retrieved subdocuments based on heuristic rules and matching with question keywords; 3) refined weighting and ranking of sentences using agreement with expected answer type suggested by question analysis; and 4) extraction of answer strings from top-ranked sentences based on expected answer type and sentence word pattern rules.

The blind experiments in TREC showed that the approach returned reasonably good results, excluding those questions with NIL answer. It works because the questions are mainly factoid, definitional types. Analysis shows that our system improves with more subdocuments retrieved, and if answer candidates from two different retrieval lists are combined by confirmation. It can identify sentences containing answers quite well, but it often fails when answers need to be extracted correctly within a 50-byte output. These experiments may serve as examples of how far one can attain in open domain question-answering without making use of external resources (e.g. the web) to find answers, and without deeper natural language analysis.

1. INTRODUCTION

Information retrieval (IR) has been an active research topic since the 1960's (Salton, 1968). Its purpose is to help users find needed information in a large collection of free-text documents. Users express their information need also in free text queries. The way IR achieves its objective is to retrieve documents in the order of their closeness to the users' relevant needs. An IR engine does not expressly try to find the answers in the ranked list of documents, and present them to the user. This has been one of the criticisms of IR, since presenting a document (perhaps with some highlighting of the query words) may not be sufficient for the user, who still has to scan through the documents to find the answers. This is especially troublesome with lengthy documents.

Question-answering (QA) has a long history (Licklider, 1965; Winston, 1977). It is a particular paradigm of retrieval where the system is expected to present none other than the correct answer to a question as a result of retrieval. To do so would require sophisticated natural language processing (NLP) techniques to understand what the user wants (i.e. analysis of the question), and to deduce that a certain string in a document sentence, or composition of strings from multiple sentences, is the correct answer to the question (i.e. analysis of document sentences and extraction).

In contrast, the analysis procedures for IR are particularly simple, consisting of extracting certain content-rich words or phrases from the items, stemming them to achieve more uniformity and coverage among word variants. These indices are prepared before retrieval and often are insufficient to satisfy QA needs. Successfully teaching a computer to do natural language understanding in open domain is one of the major unsolved problems in artificial intelligence. Previous attempts on QA have chosen to support only a restricted domain such as (Woods, 1977). However, in the last decade, QA systems supporting open domain becomes in vogue. Examples include commercial systems such as AskJeeves (<http://www.askjeeves.com>), or academic systems such as START (Katz, 1997). The TREC blind experiments organized by NIST also initiated a QA task on open domain collections since 1999, (Voorhees and Harman, 2000), and helped push QA investigation to its current popularity.

Although one could generalize IR by segmenting documents into arbitrarily smaller units such as sub-documents (Kwok, 1992), passages (Callan, 1994) or sentences (O'Connor, 1975) for retrieval and present results in these units, it still would not satisfy the objective of an exact answer for a question-answering system. On the other hand, asking an NLP system to analyze many millions of sentences directly in order to isolate an answer would be quite a massive undertaking even for modern computers. In order to achieve the objective of a QA system in an efficient manner, it is a common and logical practice to use an IR system as a front-end that would employ a user's question to condense a large collection into a more manageable, signal-rich ranked list of documents. This is then followed with more involved techniques that would analyze only the retrieved documents in greater details.

This analysis spans a spectrum of sophistication from deep, AI-oriented to more shallow, IR-oriented approaches, or somewhere in between. Deep analysis may involve many sophisticated techniques such as sentence parsing, semantic processing, discourse analysis, knowledge representation and inference, entity extraction, etc. in order to understand whether and where in the retrieved passages or documents certain answers to a question may be extracted or composed. Shallow analysis on the other extreme may employ only simple procedures such as term weighting, detection of cue words, meta-keywords, pattern matching, proximity analysis, etc. that can provide good plausibility of extracting the right terms as answer to a question.

The organization of this chapter is as follows: Sections 2 and 3 have review of the popular IR models and IR-oriented approaches to QA respectively. Section 4 presents our view of the different question-answer types. Section 5 contains a more detailed analysis of person ('Who') questions. Section 6 describes in detail our approach to the TREC experiments and Section 7 contains the results and their analyses. Section 8 has our conclusions.

2. REVIEW OF SOME POPULAR IR MODELS

Since most QA systems make use of an IR engine as front-end processing, we give a review of the more popular methodologies employed. An IR engine attempts to order a large collection of documents with respect to a query to assist the user find needed information. Optimal ordering of the retrieved documents means all relevant documents are presented to the user earlier than others, resulting in accurate retrieval. Document ranking is one of the most important issues in IR. Several approaches have been proposed in the past and are still being developed for this purpose. A classic method is the Boolean Model of retrieval (Baeza-Yates and Ribeiro-Neto, 1999) where a query is expressed as a Boolean expression of keywords connected by Boolean operators, and documents represented as sets of keywords are evaluated to be TRUE or FALSE according to whether they satisfy the query or not. Ordering then becomes a two-category classification; within the TRUE or FALSE categories, there is no finer differentiation among documents.

Because Boolean retrieval requires query formulation as a Boolean expression (which may not be easy to do), and the output ordering is limited, other models have been proposed that can support unformatted free text queries and provide real ranking of the retrieved documents. The most well-known is the vector or geometric model (Salton & McGill 1983) where queries and documents are considered as normalized vectors in multidimensional space defined by the collection, and both the terms in the query and document are weighted using term usage statistics. The angular closeness of the query with a document vector is used as a retrieval status value (RSV) for ranking documents.

It has been shown (Turtle, 1994) that on average, ranking systems provide better effectiveness than strict classic Boolean evaluation. This disadvantage of the classic Boolean model led to the development of a class of models called Extended Boolean approach. This approach attempts to soften Boolean evaluation and provides real ranking for the retrieved documents. One method is the fuzzy set model (Kraft and Buell, 1983) that evaluates a fuzzy membership value for each document with respect to the concepts of a query. The membership value is the RSV for ranking. Another interesting method is the p-norm model (Salton, Fox, and Wu, 1983) that unifies the vector model with classic Boolean by introducing the concept of a weighted Boolean query and a multidimensional vector space defined by the query terms as its axis. In this model, not only the terms used in the query and documents can be weighted, but also the query Boolean operators as well using a p-parameter. The p-parameter defines the distance metric used in the vector space. By varying p between infinity and 1, one can simulate classic Boolean or vector retrieval respectively, or some model in between.

A different class of models is based on the relevance probability estimates of a document with respect to a given query. The most well known approach is the probabilistic model (Robertson and Sparck-Jones, 1976), where a query is assumed to be associated with a given sample of relevant documents, preferably supplied by the user. These relevant example documents and the collection provide probability estimates of the content terms or features used to represent the documents. The weighted features allow the odds of relevance (i.e. the ratio of probability of

relevance to that of non-relevance) of unseen documents to be estimated and used for ranking. Ordering then becomes proportional to the probability of relevance of a document to the user who issues the query. The probabilistic model involves fewer assumptions than the vector model, and it also incorporates the concept of relevance feedback naturally. If after retrieval the user cooperates and points out some more relevant documents, these can be used for feedback, which just means increasing the size of relevant example documents. On the other hand, this advantage is also a drawback for the probabilistic model: at initial retrieval before feedback, how does one bootstrap the model if the user has no relevant examples?

A number of proposals have been advanced for this issue. The approach our PIRCS system takes is to consider each object (document or query) to be constituted of conceptual components. Components are viewed as mini-documents and are relevant to the object it is derived from – the idea of self-relevance. They can be represented as a single term if each term is considered as a concept. By working in this component universe, every document or query automatically has a relevant component sample for the purpose of indexing itself. Thus, one can provide probability estimates, and rank documents even at the initial retrieval stage without knowing any relevant documents. The PIRCS model further describes retrieval in a network that incorporates relevance feedback in terms of a learning process similar to neural network algorithms, and evaluates document RSVs by spreading of activation in the network (Kwok, 1995).

The INQUERY system (Turtle and Croft, 1991) is another approach using probability estimates as RSV's. It is based on a Bayesian inference network approach, and attempts to directly estimate the probability of relevance of a document with respect to a query rather than using the odds formula, and also provides a framework for combining evidence of relevance.

Since IR deals with textual queries and documents, it is natural to expect linguistic processing to have a major role in improving retrieval. So far, this has not happened. One reason appears to be that IR needs to handle huge volumes of text, and deeper NLP techniques are not yet sufficiently efficient for the problem. A more important reason may be that how humans do conceptual matching in unrestricted domain is unknown, and algorithms proposed to model this process may be either too primitive, or may require unacceptably large amount of external knowledge resources. Over the past five years however, a new way of thinking inspired by computational linguistics has evolved and is generally called a language model for IR such as (Ponte and Croft, 1998; Lafferty and Zhai, 2001), or translation model for IR, (Berger and Lafferty, 1999). The idea is to take each document text as a language description, and to consider the probability of how a document may generate the text of a query as the RSV for ranking. These approaches do not make use of the concept of relevance between a user's needs and the documents. Lately, (Lavenco and Croft, 2001) further extends the language model techniques to include relevance. They consider a query as relevant to a user's information needs similar to the self-relevance concept introduced by our PIRCS system.

3. IR-ORIENTED APPROACHES TO QA

Many systems that participated in the QA track of the past TREC conferences are of the shallow analysis type, perhaps also using some simple NLP procedures. The following describes some representative approaches.

AT&T (Singhal, *et al.*, 2000) scores passages from top-retrieved documents with an algorithm that sums the weight of included question content words and bigrams, and which can also dynamically increase passage size up to a limit. Returned answers can be the top-ranked passages appropriately trimmed to required lengths, or can be obtained by additional entity extraction processing on the passages. Xerox (Hull, 2000) employs weighting to score sentences with respect to a question. The sentence weight is a combination of the normalized document retrieval weight and question content term weights modified by their POS tags. Answer extraction is based on their proper name tagger called ThingFinder and noun phrases in top-ranked sentences. University of Waterloo (Cormack *et al.*, 2000) uses their text-extent retrieval to isolate passages. Text extents are defined as minimal length strings that cover all or a certain number content words of a question, and are weighted by their lengths and the weights of the covered terms. University of Massachusetts (Allan *et al.*, 2001) employs their INQUERY retrieval facility twice: first with query expansion based on local context analysis to retrieve top-ranked documents from the target collection; and second to obtain best passages from these documents using the unexpanded query. Answers are extracted from the passages by trimming to appropriate sizes. The approach by Microsoft (Brill *et al.*, 2002) relies on web searching and redundancy to find candidate answers in the WWW first, and then use these candidates to retrieve documents from the target collection to support these answers. They create multiple simple question rewrite patterns for web searching, and a set of filters to re-score the candidates according to the type of answers wanted by the question. InsightSoft-M (Soubbotin, 2002) makes use of a library of pre-defined, pre-weighted ‘indicative patterns’ to ferret out answers from retrieved passages. ‘Indicative patterns’ are sequences or unordered combinations of string elements such as words, phrases, letters and symbols that may provide a context for an answer to exist. They are created manually and linked to particular question types. No NLP or knowledge-based analyses are employed.

Our system, PIRCS-QA, is also of the IR-oriented type that requires less resource and may be more efficient. Our goal is to investigate how far IR techniques may contribute to this open domain QA problem. We employ our PIRCS IR system to do initial document ranking, followed with question analysis and sentence ranking with keywords and meta-keywords, and finally entity identification within document sentences to extract answer strings. Our PIRCS subdocument weighting has been well tested and quite effective. From the top subdocuments we score sentences like Xerox’s paper, but POS tagging is not employed. This sentence scoring consists of a general weighting that includes subdocument weight, content term weight and other heuristics, and a question-specific weighting involving agreement between a question’s type and that of terms found in a sentence. Question type analysis is based on cue words and meta-keywords as similarly done by others. Some ‘indicative patterns’ are also employed to extract answers from sentences like InsightSoft-M,

especially for definition type questions. External resource like web or encyclopedia searching is not used. Our system may be viewed as forming a baseline of performance without the use of more sophisticated NLP processing.

4. TYPES OF QUESTIONS AND EXPECTED ANSWERS

As discussed in the Introduction, QA is a paradigm of retrieval where an answer string is expected as a response to a question. It is helpful to examine the types of questions (and their answers) that may be posed to a retrieval system in general. (Tague, 1992) has listed four different types of questions that one may face. In general, questions can vary in complexity from simple Fact Retrieval (e.g. 'How old is George Bush?', 'Where is the Empire State Building?'), to Definitional (e.g. 'Who is Picasso?', 'What is a mustang?'), to Yes/No type with explanation ('Is Sara Lee a person?'), to 'How ToDo' type retrieval ('How do you cook lasagna?', 'How do you get a law degree?'), to Topic Retrieval ('Anything about anthrax'), to Decision Making ('Should I buy IBM?'). It can be seen that except for Fact Retrieval and Definitional type where the answer(s) may be a few words or a phrase or short sentence, answers to the other types can be quite long. The difference is not unlike multiple-choice versus essay questions in an examination. Although one would hope to build a system that can compose from a text database a logical step-by-step answer to the 'How ToDo' questions, or to suggest a logical explanation of why one should or should not 'buy IBM', it seems that computers are currently not sufficiently intelligent to do so. IR may still seem to be appropriate for questions requiring 'essay' type answers. Thus, it is reasonable to limit the scope of QA to Fact Retrieval or Definitional types, as was done in TREC (Voorhees, 2000; 2001). An analysis of TREC-2001 QA experiments show that most of these closed-class questions are of the who, when, where, how, and what types that may be roughly classified as follows:

A. Who Type (expecting person name or person_class)

- i) Who 'is' JOB? or What (name | person) 'is' JOB?

Ex. Who is President of U.S.A.?

(what) Name (is) a movie star.

Who is a German philosopher?

- ii) Who 'does' TASK? or What (name | person | person_class) 'does' TASK?

Ex. Who discovered X-ray?

Who delivers mail?

- iii) Who 'is' DESCRIPTION? or What (name | person | person_class) 'is' DESCRIPTION?

Ex. Who was the first woman killed in the Vietnam war?

Who was president in 1913?

B. Where Type (expecting location)

- i) Where ‘is’ ENTITY? or What (LOCATION) ‘is’ ENTITY?
Ex. Where is Carnegie Hall?
 What state is Boston located?
- ii) Where ‘is’ EVENT | ACTION? or What (LOCATION) ‘is’ EVENT | ACTION?
Ex. Where was John Kennedy assassinated?
 In what city does the Rose Parade take place?
 Where do lobsters live?
- iii) Where ‘is’ DESCRIPTION? or What (LOCATION) ‘is’ DESCRIPTION?
Ex. Where was the first golf course in the United States?

C. When Type (expecting date-time)

- i) When ‘is’ ENTITY? or What (TIME) ‘is’ ENTITY?
Ex. When is Veteran’s Day?
 What dates is our car insurance (for)?
- ii) When ‘is’ EVENT | ACTION or What (TIME) ‘is’ EVENT | ACTION?
Ex. When was the first liver transplant?
 When did Howard Hughes die?
- iii) When ‘is’ DESCRIPTION? or What (TIME) ‘is’ DESCRIPTION?
Ex. When is hurricane season in the Caribbean?

D. How Many/Much Type (expecting numeric measure with/without units)

- i) How many people | inhabitant ‘is’ in LOCATION? or What population ‘is’ LOCATION?
Ex. What is the population of China?
 How many people live in metropolitan New York?
- ii) How {much, many, long, wide, high, tall, far, fast, heavy, hot, etc.} ‘is’ DESCRIPTION? or What (amount or price, number, length, width, height, distance, speed, weight, temperature, etc) is DESCRIPTION?
Ex. How fast is alcohol absorbed?
 How many is in one dozen?

E. What Type (expecting definition, entity name)

- i) What ‘is’ PERSON ‘known for’? or Who ‘is’ PERSON?
Ex. Who is Picasso?

- ii) What ENTITY|ACTION|EVENT ‘is’ DESCRIPTION? or What ‘is’ DESCRIPTION (named | done | happened ..)?
 - Ex.* What metal has the highest melting point?
What is Shakespeare’s nickname?
What is done with worn or outdated flags?

- iii) What ‘is’ DESCRIPTION (called)? or Name a DESCRIPTION.
Ex. What is metabolism?

F. Why Type (expecting explanation)

- i) Why ‘is’ DESCRIPTION? or What reason is DESCRIPTION.
Ex. Why is the sky blue?

Because ‘occupying a job’ and ‘accomplishing a task’ are quite unique to humans, (A-i) and (A-ii) are listed separately in the Who type query. These have many variants like ‘Name (who is) a movie star’, and also include passive constructs. (A-iii) is the general format for Who questions that expect a person as the answer. DESCRIPTION is a catch-all that can be arbitrarily complex including many compound conditions. This general format is also true for Where/When (B-iii, C-iii), but listed also are two specific cases: Where/When is ENTITY? (B-i, C-i), and Where/When is EVENT | ACTION? (D-ii) type questions (How Many/Much) expect numeric answers, often with units. Units may or may not occur in the question. For example, in ‘How fast is alcohol absorbed?’, the unit of time is implied by ‘fast’, while ‘How much fiber’ implies another. A special case is the ‘population’ question listed in (D-i). In general, ‘Where’ and ‘When’ question types also have a granularity issue – for example, the location of ‘Carnegie Hall’ can be very general like ‘in New York City’ or can be very specific like the actual street address within New York City.

‘What’ type questions are the most versatile. Using the construct ‘What X’, with ‘X’ denoting reference to person, place, time, etc. is a general way of forming many of the other question types based on the word ‘X’ following ‘What’. However, the ‘X’ in this type of construct often is a meta-keyword that gives a clue as to what the question wants. It can help to identify what one should look for specifically in a document to answer a query.

5. ‘WHO’ QUESTIONS AND THEIR EXPECTATIONS

This section discusses how ‘Who’ questions may arise, and some of the common expected answer types. It seems reasonable to postulate that:

“questions arise from data instances, whether real or imagined”.

By a data instance, we mean an instance of a database relation or predicate. These instances can be real-life, but one may also have imaginary ones like those from fairy tales. Given a 2-place relation such as:

president-of(person, country)

and a data instance: president-of(George Bush, U.S.A.), one can pose questions like:

1. president-of(?, U.S.A.) = Who is President of U.S.A.? (A-i)
2. president-of(George Bush, ?) = What country is Bush president of? (E-ii)
= Bush is president of what? (E-ii)
3. ?(George Bush, U.S.A.) = What job Bush has in U.S.A.?
= What ‘relates’ Bush to U.S.A.?
4. ?(George Bush,?) = Who is George Bush? (E-i)

In the above example questions 1 and 4, one can see that the answer to ‘Who is X?’ depends on X. If X is a job title, then one expects a person-name to be the answer and the question becomes a genuine ‘Who’ question. On the other hand, if X is a person-name, one expects the answer to be ‘definitional’ and should actually be classified as a ‘What’ question. One needs to analyze the property of X in order to answer the question correctly. The example question 2 has two forms. With the format ‘What country’, the meta-keyword provides a clue of what the answer should be and hence is somewhat easier. The second format only gives the clue president-of. The third example question asks for the relationship name, and does not seem to have been used in TREC QA tasks. The form with the word ‘job’ again is easier.

A similar example is from a 1-place relation instance such as:

German-philosopher(Gunther)

Two questions can be formed:

5. ?(Gunther) = Who is Gunther? (E-i)
6. German-philosopher(?) = Name a German philosopher? (A-i)

Additional finer distinctions may also apply to the (A-ii) type of Who-questions. For example, if one assumes tasks done by humans be captured in a relation as follows:

do(person, task)

the cardinality of the expected answer in relation to the task may also give a clue as to whether a name is expected. Consider the following question:

7. do(?, discover X-ray) = Who discovered X-ray?

Because the task is unique or rare, one expects one or a small list of specific person-names. However, the questions:

8. do(?, deliver mail) = Who deliver mail? Answer: mailman | the Post Office
9. do(?, trade stocks) = Who trade stocks? Answer: broker | brokerage house

would not expect person-names as answer, but person-classes instead (or the organization that hired the class).

On the other hand, if the question were: ‘Who deliver your mail?’ (which limits the number of persons), then the answer is expected to be the name of your mailman. Thus, to be able to analyze a question for the cardinality of its expected answer may help a system to determine whether a person-name or a person-class should be returned. This third type of ‘Who’ does not seem to have been used in the TREC QA task.

English text normally makes use of upper and lower case coding, and person-names are expected to be capitalized. Thus, knowing that a question expects a person-name can help a QA system to look for capitalized words and can lead to better accuracy. Similar analysis can be performed for other types such as the ‘When’ and ‘Where’ questions. When questions are presented in logical forms as above, the semantic is clear. In reality one faces tremendous variation of free text expressions in questions and sentences. This sets the limit to how much IR can solve the QA problem.

6. PIRCS-QA: A PATTERN-MATCHING APPROACH TO QUESTION-ANSWERING

PIRCS-QA is a question-answering system constructed using our probabilistic IR system called PIRCS as the front-end retrieval engine, and enhanced with simple morphological analysis and pattern-based heuristics to rank sentences and extract answers to questions. It does not use more sophisticated syntactic and semantic techniques, and contrasts with systems like: (Moldovan *et al.*, 2000; Harabagiu *et al.*, 2002) who made use of advanced inference methodologies, (Srihani *et al.*, this volume) whose Texttract system employs sophisticated information extraction procedures to find answers in text, or (Hovy *et al.*, 2001) who employ a detailed typology to classify queries.

PIRCS-QA has participated in both the TREC-9 and 2001 QA track experiments (Kwok *et al.*, 2001; Kwok *et al.*, 2002) and was reasonably successful. Our mean-reciprocal-rank (MRR) score was among the top performers in the 250-byte-answer task, and in the top 30% for the 50-byte task. At TREC-2001, the 250-byte task was eliminated, leaving only the 50-byte task. This makes getting the right answer much more difficult especially for an IR-oriented approach. We added more heuristics, better pattern recognition and better entity identification to the system, which helped to improve its overall performance. In what follows, we describe the components of the current implementation and some of our experimental results. To clarify the following presentation, we will use the word *question* for the statement of need from the user, and the word *query* for the format derived from the *question* and used for the retrieval part of the QA task.

6.1 Document Processing and Ranked Retrieval

Documents of the collection were prepared for ad-hoc retrieval using the usual PIRCS procedures as described before (Kwok *et al.*, 2000). This includes breaking long documents into shorter subdocuments, employing Porter's algorithm to do word stemming, stop-word removal, indexing with an inverted file that also stores the term collection and document frequencies for later term weighting.

The user *question* is converted into a typical 'bag of words' content-term *query* for retrieval using similar preparation steps as for documents. This is then fed into our PIRCS retrieval system. The words in a query together with the processed collection initialize a 3-layer (query-term-document) network with edge weights defined by the probabilistic formulation of the PIRCS system. The network also supports pseudo-relevance feedback, which means expanding query terms and training the edge weights of the query based on the top-ranked documents from an initial retrieval. Document retrieval becomes a process of spreading activation either from documents to queries via terms (or vice versa). The activation levels received by queries and documents are then combined and employed to rank documents with respect to the query under focus (Kwok, 1995).

This process is an ad-hoc retrieval, and our system is quite effective for this task as has been reported in various TREC proceedings such as (Kwok *et al.*, 2000). The top n highly ranked subdocument ID's are then passed to the QA part of the system. For practical reasons, the QA subsystem accepts two different document formats. One is the subdocument format that is the default in PIRCS: these have all the original SGML tags removed and substituted by a few internal tags. Lengthy documents are segmented into subdocuments varying in size from 360 to 550 words and ending on a paragraph boundary. Consequently, only the most relevant portion of a document is returned. The other format is from the PRISE search engine output made available by NIST (<http://www-nlpir.nist.gov/works/papers/zp2/main.html>). These are whole documents, and all original SGML tags are intact.

We have experimented with returning up to 400 top-ranked subdocuments per query to investigate how the depth of the retrieval may influence getting correct answers to the question.

6.2 Sentence Scoring based on General Question Context

The first step in PIRCS-QA processing after retrieval is sentence scoring with respect to a question. A question contains descriptions about the answer it wants. These descriptive keywords may or may not appear as part of the answer string, but can provide a context in which the answer may occur. Our approach is based on the hypothesis:

"answers have high probability of occurring in sentences having similar topical context as the question"

Thus, we perform ranking of sentences according to its agreement with the context of the given question. The score is based on keyword (content word)

matching and other clues that may exist between a sentence or its source document and the question. Our sentence analysis module mixes IR techniques with simple linguistic considerations, and work on the raw sentences contained in documents returned by the previous retrieval as well as their stemmed version. Only a few stop words are removed. The sentence analysis procedure includes some or all of the following eleven heuristics with manual importance weights assigned and shown in parenthesis:

1. **Coordinate Matching (20):** detecting the presence of content words common to both a document sentence and the question.
2. **Stemming (20):** words are considered matched if they have the same root even if they are not of exactly the same form. We employed Porter's stemmer.
3. **Synonym (20):** question and document words are also considered a match if they match a synonym from a manually created dictionary of common synonyms. This dictionary also contains unusual word forms, which are not handled well by stemming, as well as variant forms for country names, such as US, United States, United States of America. Most of the entries were taken directly from Wordnet. Currently the size of the dictionary is under 500 entries. Some samples are shown in Table 1.
4. **RSV (10):** (Retrieval Status Value): with many short questions containing only a few words, frequently sentences achieve the same score. The fact that every sentence belongs to a document with a returned RSV can be used to break ties. Given two sentences with the same score based on matching terms, preference is given to the sentence that is contained in an earlier-ranked document.
5. **ICTF (15):** (Inverse Collection Term Frequency): this factor ($\log(N_w/F_k)$ where N_w is number of tokens and F_k is number of times term k is used, in the collection) gives more credit to less frequently occurring words. For practical reasons, the collection used to obtain the frequencies is the N top retrieved documents for the query (which may be less accurate).
6. **'Must' Words (40):** extra credit is given for words that are important and which must occur in the answer string. At present, these are the superlatives: 'first', 'last', 'best', 'highest', etc. However, one needs to be careful: when attempting to match the term 'best', one also should exclude those matching 'seventh best', for example.
7. **Proximity (14):** question words that appear in close proximity in a document sentence are likely to refer to the same required concept. This is currently done only if the sentence contains all the question content words, and therefore applies only to short questions.
8. **Heading (10):** newspaper articles often come with a tagged headline. The content of the headline sets the context for the sentences in the document. If a sentence matches some question content words but not others, then a small extra credit is given to the sentence if the document headline contains the unmatched content words of the question.
9. **Phrases (10):** consecutive words in the question receive more credit if they are also consecutive in a sentence.
10. **Caps (5):** Capitalized query words are deemed more important.

11. Quoted (40): Quoted query words are deemed more important.

Many of these eleven features are binary. Others are normalized to between 0 and 1. Based on the above, we score and rank sentences (from the set of retrieved documents) by a weighted sum of the feature values. This *general* scoring step is common to all questions. The next scoring step is done by modules that are tied to more detailed, *specific* analysis of the raw question, and is performed on the top-ranked, nonzero score sentences. These are described in the next section.

Table 1. Samples from our Synonym Table.

Description	Entries
<i>Nationality</i>	roman rome spanish spain portuguese lusitanian portugal sicilian sicily finnish finland swedish sweden danish denmark danes belgian belgium luxembourgian luxembourg
<i>Unusual Verb forms</i>	knew know known knows lend lent lost lose misbecame misbecome misspend misspent mistook mistake misunderstood misunderstand molten melt mown mow mows
<i>Noun synonyms</i>	male men man female women woman
<i>Abbreviations</i>	capt captain united states, us, usa, u.s. united states of america united kingdom, uk, u.k. united nations, un, u.n.

6.3 Sentence Scoring based on Specific Question Analysis

Based on the discussions in Sections 4 and 5, we created a taxonomy for classifying the various questions and their expected answers. This is important since wrongly classified questions would be directed to look for erroneous strings in document sentences. Like others using shallow processing, we employ a suite of

clue words/phrases for this purpose. As examples: ‘What is/are’, ‘Who is (short)’, etc. are considered as definition type, ‘How many/much/far/..’ are numeric type and so forth. Table 8 in Section 7.3 shows our taxonomy.

Once the *specific* expectation of a question is determined, we use specialized modules for the three categories of expected answers that we support, and a fourth to catch all the miscellaneous. Each module returns a weight that is added to the weight of the top-ranked sentences obtained in Section 6.2. The modules make use of a table of weights (for this and the next Section) that are defined heuristically for various pattern rules to detect the presence of a required answer type. Included is consideration of proximity of question words in sentences, and with expected answers. The modules are discussed below.

A) Capitalized-Word Module: this procedure is designed for expected answers requiring capitalized word(s) such as persons, organizations, and places including geographic names, countries, rivers, etc. Questions that would invoke this module include the types: ‘Where’, ‘What X’ (A-i, B-i, C-i, E-i), and the ‘long Who’ (A-iii) – a ‘Who’ followed with several words of description. In particular, for ‘What X’ questions such as ‘What country’, the meta-keyword ‘X’ often serves to describe what the question is looking for. We captured lists of states, countries, oceans from the Internet to assist in confirming answers by giving heavier weight if a ‘Capitalized entity’ in a document sentence exists in the list of ‘X’. For example, the question ‘Where is the tallest roller coaster located?’ directs us to look for a capitalized word. The document sentence ‘The nation’s tallest roller coaster is the 205 foot high Magnum XL 200 at the Cedar Point amusement park in Sandusky, Ohio, but it doesn’t loop.’ has high weight because it has the ‘must’ word ‘tallest’ and matches the question content phrase. It also satisfies the specific requirements of capitalized words after the cue word ‘at’ for places.

B) Numeric Module: this procedure is designed for answers requiring a number and often with units. Meta-keywords like: ‘length, temperature, speed, date-time, zip-code’, etc. or questions like ‘How tall/wide/fast/hot ..’, ‘How many/much’ are clues for invoking this module. Table 2 shows another knowledge source that we compiled that links measure concepts with units and possible question types. If a question is determined to be one of these types, and a document sentence contains a numeric format string, possibly with the expected unit, then the sentence will be given higher weight. Proximity between question words and expected answer in a sentence is also taken into account. For example, the meta-keyword ‘population’ in the question ‘What is the population of Venezuela?’ directs PIRCS-QA to this module. The document sentence ‘Venezuela has 18.43 million people’ has a numeric string adjacent to the unit ‘people’ and gets a high weight. It also gets a bonus because a question word ‘Venezuela’ is in close proximity to ‘people’. The module will assign a lower weight for the document sentence ‘Venezuela has a small population (about 20 million)’ because ‘population’ and numeric string is one word apart.

C) Definition Module: this procedure is designed for answers requiring a string longer than just a word (or two). Questions that would invoke this module include

the types: ‘short What’ and ‘short Who’ such as ‘What is X’, ‘Who is X’, ‘X is also called what?’, ‘What is X famous for?’, where X is one or two words (A-iii, E-iii). These can be likened to dictionary lookup. A sub-type of this is ‘What does Z stands for’, where Z is a fully-capitalized word standing for an acronym. (See examples in Table 3, Section 6.4.2).

Table 2. Measures, Units and Question Types.

Measures	Question Type	Units
Length	HOW LONG, SHORT, TALL, HEIGHT, LOW, WIDE, WIDTH; WHAT DISTANCE	FT, FEET, IN, INCHES, YARD, MILE; MM, MILLIMETER, CM, METER, KM, KILOMETER, LIGHTYEAR.
Time	how long, old, young; what time, what duration.	SEC, SECOND, MIN, MINUTES, HRS, HOURS, DAY, WEEK, MONTH, YEAR, CENTURY
Speed	how fast, slow, what speed/velocity	x/y (where x is from Length, y from Time)
Area	how many ‘units’, what area	square x, sq x, where x is from Length; acre, hectare
Volume	how many ‘units’, what volume	cubic x, where x is from Length
Weight	how heavy, many units, what weight	oz, ounce, lb. pound, gm., gram, kg., kilogram
Currency	how much, what cost, price; what prize, salary.	dollars, cents, pound, yen, yuan, franc, lira, mark, shilling, euro
Temperature	how hot, cold; what temperature	centigrade, Fahrenheit
Date	when; what date, month, year, etc	‘January’ .. ‘December’; ‘Jan’ .. ‘Dec’.
Address	where; what location/address, etc	Street, Avenue
Zip-Code	where; what location/address, etc	None
Population	how many people, population, inhabitants	inhabitant, population, people

Phone Number	what phone, number	None
Percent	what percent, percentage	percent, percentage, %
Numbers	how many, what number, count;	None

D) Unknown Module: this procedure is designed for questions that cannot be answered by the above three modules. Questions that would invoke this module include the types: ‘Why’, ‘What’ and ‘How’ questions with no recognized meta-keywords. This module will just consider word strings in a sentence in close proximity to question words. For example, the question ‘What is the most popular sport in Japan?’ is an unknown type for us. The document sentence ‘.. baseball tournament in Osaka, one of Japan’s most popular sporting events.’ is weighted higher than the following ‘Now, it is threatening to dislodge Japan’s stodgy baseball as the nation’s most popular sport.’ because the question words and stems are in closer proximity to each other.

This taxonomy of course does not work always. Question types such as ‘Who delivers mail?’ or ‘Where do lobsters live?’ need lower case words as answers while PIRCS-QA looks for capitalized words.

6.4 Answer Extraction from Sentences

6.4.1 Entity Identification

It is apparent from the question analysis phase that entity identification is a vital component of any QA system. In fact, one of the stated aims for the QA track in TREC is ‘to provide an opportunity for the information retrieval and information extraction communities to work on a common problem.’ (Voorhees, 2001). Except for questions of definition type, most questions require answers involving a named entity. When question analysis indicates a certain entity type is needed (such as person, country, organization, etc.), and if an entity of the wanted category is recognized in a sentence, one can output the entity and its surrounding context words to form an answer string of the required size. Any sentence that does not meet this requirement is discarded even though sentence analysis shows that it has a high score for context. We have implemented a number of rules and simple pattern recognizers for entity recognition.

1. **Person Entity** – A capitalized word not preceded by ‘the’ is considered having a high probability of being a person name. This evidence would be further strengthened if this word is preceded by a title such as: {Mr. | Mrs. | Ms | Sen. | Gen. | Dr. | Prof. | Capt. | etc.}
2. **Place Entity** – A capitalized word preceded by ‘on’ ‘in’ ‘at’ is probably a place name. They are also recognized by other cue words/phrases such as ‘located’, ‘next to’, ‘east of’, ‘neighboring’, ‘borders’, etc.

3. **Numeric Entity** – These are recognized by both the numerical format and by its English spelling. The semantics of a numerical string can be recognized by its accompanying unit if such exists, as shown in Table 2, Section 6.3. Included in this category are special formats for date strings, telephone numbers, social security numbers, and some addresses. For example, four digit numbers starting with 19xx or 20xx probably denote year.

Certain information contained within SGML tags can also be useful for answering questions. For newspaper genre, terms appearing within <HEADLINE> and <DESCRIPT> tags are assumed to qualify all sentences of the document. <DATE> and <PUBDATE> tags can be used to answer some ‘When’ questions. Certain ‘Where’ questions may have answers within <F>, <CN>, or <DATELINE> tags. Frequently datelines of news reports are also useful for place names. For example:

Question #952:	<i>Where is Milan?</i>
AP Document:	<DATELINE>MILAN, Italy (AP) </DATELINE>
WSJ Document:	MILAN, Italy – The index that measures the aggregate value of orders to Italian industry rose 4.1% in March ...

4. **Acronym/Abbreviation** – A simple acronym recognizer was written which can detect if a group of words can be an acronym as well as a table of state abbreviations. See Example 2 in Section 6.4.4.

6.4.2 Answers Requiring Definitions - Clues from Patterns

Questions of the type ‘What is X?’ can be frequently answered by thesaurus or dictionary consultation, rather than by searching some large text corpora. Results from full text search may be used to supplement or verify the answer. Searching for definitions in large text corpora may be a useful tool for thesaurus or dictionary constructors, who may want to inspect current usage of new terms. Many QA systems make use of external resources such as WordNet, the web, encyclopedia, etc. to enhance their performance. Possible answers were found in these resources first, and the QA system then locates them in the target collection to give support to the answer. Example systems include (Brill *et al.*, 2002; Clarke *et al.*, 2002; Harabagiu, 2002; Prager, 2002) and others. For ad-hoc retrieval, we have previously also introduced a similar technique called collection enrichment (Kwok & Chan, 1998), where an external collection is added to the target collection in order to improve the precision of an initial retrieval. The improved initial retrieval may favorably impact on a second stage retrieval via pseudo-relevance feedback. However, PIRCS-QA currently relies only on internal evidence derived from observed patterns in the target collection. We attempt to detect common usage patterns for uncommon words in texts, where both the term and the definition are included in the same sentence.

Consider Table 3 with some questions and sample answers extracted from document sentences. The bold italicized string patterns, together with the entity to be defined shown in bold face, frequently re-occur and can be used to extract the

definition. Thus, the entity needing definition often occurs following the patterns: ‘called’, ‘known as a’, ‘,_ a’, ‘is a’, ‘equals’. It may also appear just preceding a parenthesis, or inside one, adjacent to ‘and’ or ‘or’. While we created our patterns manually, it would be more interesting to automate discovery of these patterns and associate with each pattern a probability of correctness.

Table 3. String Patterns as Clues for Definitions.

Q#	Question	Example Answer Strings
241	What is a caldera?	<ol style="list-style-type: none"> 1. Large volcanic craters called calderas 2. craters _ called calderas 3. volcanic crater known as a caldera 4. caldera , a volcanic crater
434	What is a nanometer?	<ol style="list-style-type: none"> 1. Nanometer equals one-millionth of a millimeter 2. A nanometer is one-billionth of a meter.
354	What is a nematode?	<ol style="list-style-type: none"> 1. Tiny worm, or nematode 2. nematodes (eelworms) 3. nematodes, _ small roundworms
700	What is another name for nearsightedness?	<ol style="list-style-type: none"> 1. Myopia (nearsightedness)
300	What is leukemia?	<ol style="list-style-type: none"> 1. Leukemia, a cancer of the blood system 2. leukemia, is a cancer 3. Leukemia is a bone marrow cancer
386	What is anorexia nervosa?	<ol style="list-style-type: none"> 1. Anorexia nervosa, an eating disorder 2. anorexia nervosa, the eating disorder 3. the most common eating disorders are anorexia nervosa
635	What is cribbage?	<ol style="list-style-type: none"> 1. Cribbage and other games

6.4.3 Answer String Creation

It is a challenge to select a 50-byte or smaller size answer string from a top-ranked sentence without some understanding of the sentence. For questions needing numeric answers, we include numbers followed by units from the selected sentence. For answers requiring capital letters, we include all capitalized words. This works well for the automatic evaluation. In official manual evaluations however, if two names are included causing ambiguity, the answer is judged wrong. Proximity to question words is the criterion used in all other cases. This misses many answers. As mentioned in Section 6.3 (A), we store a list of countries, states, continents, oceans,

etc. in the system. When a question contains such a meta-keyword, we can return an answer of the correct type.

6.4.4 Some Examples

In what follows, we use a few example questions to illustrate our procedures and show how answers are recognized and returned.

Example 1

TREC QA 461: Who was the author of the book about computer hackers called “The Cuckoo’s Egg: Tracking a Spy Through the Maze of Computer Espionage”?

This question demonstrates how IR methods rank and retrieve the correct sentence for a fairly long question. The system considers the following candidate document sentences with matching words in boldface (a small stopword list is used during question analysis):

1. By far the best account is in Clifford Stoll’s The **Cuckoo’s Egg**, which explains how a German **hacker** hijacked the international telecommunications system and broke into high security **computers** all over the US, including one at the Pentagon.
2. A week ago, West German authorities charged a **computer hacker** with **espionage** and said a total of eight people were suspected of selling sensitive computer data from the US and allied countries to the Soviets from 1986 **through** last year.
3. The story he re enacts, and his equally fascinating **book about** the case, “The **Cuckoo’s Egg**,” are about science, Stoll insists.
4. Alexander Prechtel, spokesman for the Federal Prosecutor’s Office in Karlsruhe, said police in Hanover arrested two **computer hackers** on “suspicion of **espionage** activities for an eastern European intelligence agency.”
5. Stoll, a Berkeley liberal now lionized by many in the **computer** security field, was “a bit of a crank” for **calling** in the FBI and the CIA to investigate the **hackers**, the **authors** say.

Four of the sentences have 4 matching words to the question. Sentence 4 has 3, but it is from the highest-ranking document. Sentence 1 was top-ranked because it has:

- a) matching phrase
- b) two matching upper case query word
- c) matching author quoted pattern: (X’s “ ”)
- d) higher RSV

Example 2

TREC QA 1297: What does the acronym NASA stand for?

This question demonstrates our system's acronym recognizer (Section 6.4.1) at work. Consider the following sentences:

1. The US National Aeronautics and Space Administration (**Nasa**) has been left leaderless after President George Bush fired the former astronaut who had headed it for the last three years.
2. 7. How would you rate the overall effectiveness of the National Aeronautic and Space Administration, known as **NASA** excellent, good, only fair or poor? Excellent: 10 percent.
3. **NASA** administrator James C. Fletcher, who said months ago that he wanted to retire, announced on Monday that he expects President Bush to name a new space agency leader within two weeks.
4. ...they couldn't understand why it (**NASA**) was being torn down and raked over the coals after the accident.

The highest-ranking sentence 1 satisfies the acronym recognizer and the definition pattern by having (Nasa) in parenthesis. The second highest sentence also satisfies both, but scored less because of the greater distance of 'National Aeronautic and Space Agency' from the acronym. Sentence 3 was ranked third after deleting duplicates, since it satisfies the acronym recognizer to lesser degree (**name a new space agency**), but not any patterns. Sentence number 4 satisfies the definition pattern (NASA) but not the acronym recognizer.

Example 3

TREC QA 1367: What is the population of Venezuela?

This question demonstrates how units and pattern help to retrieve a valid answer. Our QA system recognizes that this is a population question, and that the answer must be a number followed by a unit such as: people, inhabitants, population or persons.

1. **Venezuela** has 18.43 million people, and its **population** is growing at a rate of 2.9 percent, one of the highest rates in Latin America.
2. Three quarters of **Venezuela**'s 20m people are under 35, yet the country has just elected a president who first ran for the office in 1947, the year after US President Bill Clinton was born.
3. **Venezuela** has 18 million inhabitants and was for years an island of tranquility in a sea of regional violence, political instability and economic recession.
4. Colombia, Mexico, and **Venezuela** form a market of 145 million people, with \$50 billion in total exports and \$60 billion in imports.

5. The government said 256 people died in the violence, the worst in **Venezuela**'s 30 years of democracy.

The first three sentences are rated highest because they satisfy a population pattern (X has nnn people/inhabitants). Sentence 1 was rated highest, because it was in the highest ranking retrieved document. In Sentence 2, we recognize that $20m$ is a number, and in Sentence 3 we recognize that inhabitants is a valid unit. Sentence 4 has the numeric answer further from the query word. Sentence 5 does not satisfy a population pattern because they do have the right units.

Example 4

TREC QA 1073: How far is Pluto from the sun?

The importance of entity recognition is illustrated in this question. Consider the following high ranked sentences:

1. The discovery of a significant atmosphere could end years of debate over the extent of the atmosphere of **Pluto**, which averages more than 3.7 billion miles from the **sun** in its orbit, Beatty said.
2. The US National Aeronautics and Space Administration (NASA) plans to launch automatic space probes to the planet **Pluto** and the **sun** with Russia's help.
3. Saturn 1980, and now; 2 will do; 2. Neptune usually / Voyager 1 visited Jupiter in 1979 and Saturn in 1980, and now is speeding toward interstellar space just as Voyager 2 will do after it stops studying Neptune on Oct. 2. Neptune usually is the eighth planet from the **sun**, but **Pluto**'s elliptical orbit means it now is closer, making Neptune temporarily the solar system's outmost planet.

PIRCS-QA recognizes that the proper answer requires a distance and therefore must have length unit. It ranks Sentence 1 highest.

7. EXPERIMENTAL STUDIES

In the TREC environment, there are two ways of evaluation for QA tasks. The first way is manual assessment, which means that human assessors did the judging at NIST based on given criteria (Voorhees, 2001). This is the official gold standard by which all sites are evaluated. The second kind is automatic, which means evaluation by Perl scripts. This is for the convenience of experimenters who want to evaluate quickly new algorithms for the QA task without incurring the cost of human assessment. However, automatic evaluation usually over-estimates performance compared to manual assessment. If two or more names are contained in an answer string and only one is correct, the automatic evaluation will find it correct, but a human may judge it ambiguous and wrong. For example:

Question #1040: *Who is a German philosopher?*

Sentence: .. *baroness met with Yuriy Matochkin, head of the Kaliningrad Oblast Administration, visited the Kaliningrad International Business School, toured the city, laid wreaths at the tomb of the German philosopher Kant and the Pushkin Memorial, ...*

Perl pattern: Kant

PIRCS-QA answer: *Yuriy Matochkin,; tomb; Kant; Pushkin Memorial*

PIRCS-QA extraction produces several Capitalized words and nearest non-capitalized words, but only one – Kant – is the correct answer. In what follows, we indicate which kind of evaluation is used for our experimental studies.

7.1 Performance Contribution by Sentence Scoring based on General Question Context

Table 4 shows an early analysis we performed using selected questions from the TREC-8 question set, and returning 250-byte answers. This means that in the large majority of cases, whole sentences are returned as answers and answer extraction is not necessary. It illustrates the contributions of some of the methods of the sentence weighting discussed in Section 6. For example, if only Coordinate Matching is used, the Mean Reciprocal Rank (MRR) score is 0.439. Adding Stemmed Word Matching increases the score to 0.470, a 7% improvement. Cumulatively adding other components lead to (almost) successively better results. Most of the improvements are incremental until a large jump of about 10% in MRR via use of the Name (Capitalized-Word, Section 6.1) module. Adding the Numeric heuristics further improves by about 3%. We have not been able to show that Phrases (9), Caps (10) and Quoted (11) are consistently useful and were not included in Table 4. One problem with having so many components is that their contributions are not independent. Increasing the weight of one may result in unfavorable effects on others.

Table 4. Cumulative Contribution from Sentence Analysis Methods (Automatic Evaluation).

	Sentence Analysis Methods	MRR Score
1	Term matching	0.439
2	Stemming	0.470
3	Synonym	0.478
4	RSV	0.498
5	ICTF Term Weight	0.509
6	‘Must’ Term	0.506
7	Proximity	0.515

	Sentence Analysis Methods	MRR Score
8	Heading	0.515
9	8)+Name heuristics	0.566
10	8)+Numerical heuristics	0.584

7.2 TREC-9 Experimental Results

The QA task in TREC-9 involved 693 questions finding answers in a set of approximately 1 million documents consisting of: the Associated Press Disks 1-3, Wall Street Journal Disks 1-2, San Jose Monthly Notes, Financial Times, Los Angeles Times and the FBIS collections. A system can report five ranked answer strings, and the mean reciprocal rank (MRR) is calculated using the highest ranked answer that is correct. There were two subtasks for this open domain QA track: one allows systems to return answer strings of 250-byte size (long), and the other requires 50-byte size (short). We submitted two runs for each subtask: pir0qal1 that makes use of 50 retrieved documents per question supplied by NIST (using the ATT retrieval engine), and pir0qal2 that makes use of 200 top-ranked subdocuments returned by our PIRCS engine. Table 5 summarizes our results and their comparison with the median of all submissions.

It is not surprising that the 250-byte subtask gives over 60% better performance than the 50-byte subtask based on MRR. This is because the former practically does not need any entity identification and extraction methodology: returning whole sentences would most of the time satisfies the 250-byte requirement. In Table 5, it is shown that QA results using PIRCS retrieval of 200 top-ranked subdocuments is more than 9% better than those using ATT 50 top-ranked documents. Previous TREC ad-hoc experiments have shown that both our PIRCS system and the ATT search engine perform well at quite similar high levels. Thus, we conclude that for our approach to QA, the retrieval depth is important.

Table 5. TREC-9 Results: Comparing PIRCS-QA with Average (Manual Evaluation).

	MRR Score	Comparison with TREC-9 Overall
250-Byte Task		
TREC-9 Overall Avg	0.350	0%
pir0qal1 (NIST 50)	0.433	+23.77%
pir0qal2 (pircs 200)	0.464	+32.73%
50-Byte Task		
TREC-9 Overall Avg	0.218	0%
pir0qas1 (NIST 50)	0.263	+20.82%
pir0qas2 (pircs 200)	0.284	+30.65%

We performed experiments varying the number of top-ranked subdocuments returned by our PIRCS system to observe how this may affect QA results. This is as shown in Table 6. There is a continuing improvement with increasing number of retrieved subdocuments, both for the 50 and 250 byte tasks. It appears that using a larger number of returned subdocuments is beneficial for our system. This may be due to two factors. First, our system returns subdocuments of about 350 to 550 words in length, not whole documents. These units are more precise. For the same retrieval depth value, our system may return only a fraction of the text compared to another system that returns full documents. Secondly, when the pool of sentences is enlarged, the probability of finding sentences with correct answers and satisfying the heuristics of our answer scoring modules is increased, and these sentences will be promoted in rank with high scores assigned. The fact that performance averaging over many questions improves with retrieval depth seems to suggest that our heuristics are working reasonably. Naturally, for certain questions, more text may be detrimental because of additional noisy sentences satisfying our criteria for answers; but the overall average effect is positive.

Table 6. Effect of Number of Top-Ranked Subdocuments on MRR (Automatic Evaluation).

Sub-Docs Retrieved	50-byte Answer MRR Score	% change from 50 subdocs	250-byte Answer MRR Score	% change from 50 subdocs
5	318	-28%	423	-21%
10	374	-15%	470	-12%
20	419	-5%	522	-2%
30	430	-3%	525	-2%
50	442	0%	533	0%
75	452	2%	543	2%
100	465	5%	560	5%
150	471	7%	565	6%
200	477	8%	570	7%
250	478	8%	571	7%
300	483	9%	576	8%

7.3 TREC-2001 Experimental Results

In TREC-2001, the QA task consists of 492 questions targeting on the same collection as in TREC-9. This time only the 50-byte size answer string is allowed. In addition, questions may have no answers in the collection, and a system can return NIL as one of the five ranked answers to show the capability of a system to detect no

satisfactory answer. We submitted three runs named pir1Qqa1 to pir1Qqa3. pir1Qqa1 utilized the 50 top-ranked documents of the PRISE system provided by NIST. pir1Qqa2 used the top 400 subdocuments retrieved by our PIRCS system. pir1Qqa3 combines the two retrievals. Combination strategy is deliberately simple: if the same answer string is pulled from the same document from both retrieval lists, it is promoted up in rank because it is confirmed by two independent retrievals. Combination also leads to better results in our TREC-9 experiments. Table 7 shows our results at TREC-2001. It can be seen that our MRR result is about 39% better than the median when we used the combination strategy. However, we did not do a good job at detecting NIL answers. Overall, the system finished among the top 1/3 of participants.

In Table 8, we analyze the performance of pir1Qqa3 by question type compared to the TREC 2001 overall average. The end of the table also show results aggregated into the four categories (Section 6.3) with and without NIL questions. Except for the short ‘Who’ questions, the definition type works well. The longer ‘what’ questions and the ‘who’ questions, both of which rely on sentence retrieval did well above average. The results are not as good for ‘when’ (date), ‘what unit’ and ‘where’ type of questions.

Table 7. PIRCS-QA MRR Result in TREC-2001: Answer String Size = 50 (Manual Evaluation).

	MRR Over all Questions	Performance compared to TREC 2001 Overall	MRR 443 Questions Without NIL Answer	MRR 49 Questions With NIL Answer
TREC-2001 Overall Avg	0.234	0%	0.239	0.193
pir1Qqa1	0.300	+28%	0.333	0.000
pir1Qqa2	0.314	+34%	0.348	0.000
pir1Qqa3	0.326	+39%	0.362	0.000

Unknown category is assigned to questions not recognized by our system. For example: ‘What is mad cow disease’ may be considered as a Definition question, but we can only recognize Definition question with at most 2 content words. Similarly ‘What is the sales tax rate in New York?’ is not recognized as a Numeric question. Results of our Unknown category questions perform better than the TREC average. They are also better than the Name and Numeric types. It is possible that these questions with less focused answers can be answered better and more robustly by general IR techniques, and we have a strong IR system. For example the query ‘What is the most popular sport in Japan?’ has reasonably good terms for PIRCS to

select the answer sentence. On the other hand questions like ‘How fast is sound?’ requires specialized knowledge to answer correctly.

7.4 Effect of Answer String Size on QA Results

As expected an IR based QA system performs much better if the answer string size is larger. Table 9 shows the MRR result for the pir1Qqa2 run. Using automatic evaluation (instead of manual), the 250-byte answer string task gives an MRR value of 0.482, an improvement of nearly 15% compared to the 50-byte task. Although the TREC QA task favors ‘exact answers’ for evaluation, in real-life QA situations we believe that users may actually favor an ‘answer with context’ – meaning that the surrounding text of an exact answer could also contribute context to help users understand the answer. In some situations, it might in fact assist the user in determining that an answer is acceptable. For example, since none of the current QA systems is perfect, the context of a numeric answer might show clearly whether the answer is correct or not.

Table 8. PIRCS-QA MRR Result by Question Type in TREC-2001 (Manual Evaluation).

Category	Question Type	No	TREC Overall	pir1Qqa3	Pir1Qqa3 vs. TREC-2001 Avg.
Name	what name	21	0.235	0.302	28%
	what name country	2	0.213	0.500	135%
	what name state	5	0.163	0.200	23%
	what name continent	2	0.288	0.500	74%
	where	18	0.228	0.296	30%
	where short	9	0.232	0.139	-40%
	where city	5	0.238	0.400	68%
	who	43	0.227	0.325	44%
Numeric	how many	6	0.117	0.167	43%
	how much	6	0.208	0.417	101%
	what number	22	0.247	0.258	4%
	what num speed	4	0.110	0.133	21%
	what num temp	4	0.247	0.250	1%
	when	38	0.264	0.319	21%
	when born	8	0.209	0.333	59%
Definition	whatis	112	0.256	0.373	45%
	abbreviation	4	0.423	0.750	77%
	who short	3	0.290	0.000	-100%
Unknown	how	3	0.301	0.333	11%
	what	173	0.217	0.325	50%
	why	4	0.247	0.208	-16%

Aggregate Results (with NIL Questions)					
Name		108	0.229	0.295	29%
Numeric		88	0.233	0.290	24%
Definition		116	0.262	0.386	47%
Unknown		180	0.219	0.323	47%
Aggregate Results (without NIL Questions)					
Name		92	0.238	0.347	46%
Numeric		74	0.240	0.345	43%
Definition		107	0.267	0.418	57%
Unknown		170	0.221	0.341	55%

Table 9. TREC-2001 *pir1Qqa2* using 443 Questions Without Null Answers.

Run	Answer length	Evaluation	MRR
Trec-2001 Overall Avg	50 bytes	Manual	0.239
<i>pir1Qqa2</i>	50 bytes	Manual	0.348
<i>pir1Qqa2</i>	50 bytes	Automatic	0.416
<i>pir1Qqa2</i>	250 bytes	Automatic	0.482

7.5 Failure Analysis

The hardness of a question may be ranked by the overall performance from all the participants. Hard questions are difficult for our system without natural language understanding capability such as those requiring composing answers from multiple sentences. It is instructive to look at some easy questions that we missed. One weak point as mentioned earlier is the final string selection. In many cases, we retrieved the correct sentences but did not select the correct string. In others, the correct answer was within the selected answer string, but other extraneous words were added (e.g. returning names of two persons only one is correct, or having extra numbers for numerical questions) making the answer ambiguous for manual judgment. We also assume that words appearing in questions do not appear in answers, and therefore remove them from answer strings. This does not work in some cases.

8. CONCLUSION AND DISCUSSION

We have described our approach to question-answering that is based on:

1. returning n top-ranked subdocuments from PIRCS retrieval using as query the question with stemming and stop-word removal;
2. scoring and returning top-ranked sentences from the subdocuments with respect to the general context of the question keywords using a set of eleven heuristics – both raw and stemmed words were taken into account;
3. analyzing specific properties of the question to obtain its expected answer types, and assigning one of four functional modules that use keywords, meta-keywords and patterns to detect possible answers and add bonus weights to top-ranked sentences for selection purposes;
4. extracting answer strings of required size from top candidate sentences based on the previous question analysis with rules and heuristics for entity definition or identification.

Our approach employs little linguistic analysis and has no language understanding capability. It naturally cannot perform difficult tasks such as those questions requiring answers that need linking of multiple sentences. It did not reach the performance of sophisticated systems that use heavy knowledge inference and NLP techniques such as (Harabagiu, *et al.*, 2002). An analysis of simple questions show that we can identify sentences with answers quite well, but the weak point is in extracting the right string or entity as the answer from these sentences. Our system also failed to confirm that a collection has no answer to a question. Nevertheless, PIRCS-QA has been demonstrated in the TREC open domain blind experiments to return reasonably good results. This is particularly true when longer strings (250-bytes) are allowed as answers. Longer strings as answers may actually be quite acceptable since, for real-life question-answering, users may prefer an answer string with context that may serve as answer explanation. Also, since no current QA systems are perfect, returning several answers in context can facilitate a user to filter out the correct answers from the incorrect.

For our system, it seems beneficial to use more retrieved subdocuments (and sentences) to locate answers. This can provide better chance for the existence of a candidate sentence with the correct answer and the right patterns for our extraction module to work properly. In the same vein, we expect that the use of external resources such as the web to find answers first, and then locate support in the target collection would also be helpful to our system.

Although the current study limits queries to certain closed forms, we envisage that in the future such studies may extend to more general questions like those used in TREC ad-hoc experiments. Finding what a question wants (or does not want) in this setting would be a much greater challenge, but the prospect is that question-answering techniques may be fruitful for ad-hoc retrieval and may bring substantial improvement in retrieval accuracy.

9. ACKNOWLEDGMENTS

This work was partially supported by the Space and Naval Warfare Systems Center San Diego, under Grant No. N66001-00-1-8912. We like to thank the NIST TREC team for providing the data, the evaluation and a wonderful opportunity for us to work on this interesting problem. The comments of the editors and reviewers also lead to improvements in presentation and organization of the chapter.

10. REFERENCES

- Allan, J., Connell, M. E., Croft, W. B., Feng, F-F, Fisher, D. & Li, X. (2001). INQUERY and TREC-9. In E. M. Voorhees & D. K. Harman (Eds.), *The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 551-562). Washington, DC: US GPO.
- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. New York: ACM Press.
- Brill, E., Lin J., Banko, M. Dumais, S. & Ng, Andrew (2002). Data-intensive Question Answering. In E.M. Voorhees & D.K. Harman (Eds.), *The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 393-400). Washington, DC: US GPO.
- Burger, A. & Lafferty, J. (1999). Information retrieval as statistical translation. In *Proc. 22nd Ann. Intl. ACM SIGIR Conf. on R&D in IR*, (pp. 222-229).
- Callan, J. P. (1994). Passage-level evidence in document retrieval. In *Proc. 17th Ann. Intl. ACM SIGIR Conf. on R&D in IR*, (pp. 302-310).
- Clarke, C. L. A. , Cormack, G. V., Lynam, T. R., Li, C. M., & McLearn, G. L. (2002). Web reinforced Q A (MultiText experiments for TREC 2001). In E.M. Voorhees & D.K. Harman (Eds.), *The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 673-679). Washington, DC: US GPO.
- Cormack, G. V., Clarke, C. L. A., Palmer, C. R. and Kisman, D. I. E. (2000). Fast automatic passage ranking (MultiText experiments for TREC-8). In E.M. Voorhees & D.K. Harman (Eds.), *Information Technology: The Eighth Text REtrieval Conference (TREC-8), NIST Special Publication 500-246*, (pp. 735-741). Washington, DC: US GPO.
- Harabagiu, S., Moldovan, D., Pasca, M., Surdeanu, M., Mihalcea, R., Girju, R., et al. (2002). Answering complex, list and context questions with LCC's Q-A server. In E. M. Voorhees & D. K. Harman (Eds.), *The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 355-361). ashington, DC: US GPO.
- Hovy, E., Gerber, L., Hermjakob, U., Junk M., & Lin, C-Y. (2001). Question answering in Webclopedia. In E. M. Voorhees & D. K. Harman (Eds.), *Information Technology: The Ninth Text REtrieval Conference (TREC-9), NIST Special Publication 500-249*, (pp. 655-664). Washington, DC: US GPO.
- Hull, D. A. (2000). Xerox TREC-8 question answering track report. In E. M. Voorhees & D. K. Harman (Eds.), *Information Technology: The Eighth Text REtrieval Conference (TREC-8), NIST Special Publication 500-246*, (pp. 743-752). Washington, DC: US GPO.
- Katz, B. (1997). From sentence processing to information access on the world wide web. *AAAI Spring Symposium on NLP for the WWW*, Stanford University, CA. (available on: <http://www.ai.mit.edu/projects/infolab/start-system.html>)
- Kraft, D. & Buell, D. A. (1983). Fuzzy sets and generalized Boolean retrieval systems. *Intl. J. of Man-Machine Studies*, 19, 45-56.
- Kwok, K. L. (1995). A network approach to probabilistic information retrieval. *ACM Transactions on Office Information System*, 13, 324-353.
- Kwok, K. L. & Chan, M. (1998). Improving two-stage ad-hoc retrieval for short queries. In *Proc. 21st Ann. Intl. ACM SIGIR Conf. on R&D in IR*. (pp. 250-256).
- Kwok, K. L., Grunfeld, L. & Chan, M (2000). TREC-8 Ad-Hoc, Query and Filtering Experiments using PIRCS. In E. M. Voorhees & D. K. Harman (Eds.), *Information Technology: The Eighth Text REtrieval Conference (TREC-8), NIST Special Publication 500-246*, (pp. 217-227). Washington, DC: US GPO.
- Kwok, K. L., Grunfeld, L., Dinstl, N. & Chan, M. (2001). TREC-9 Cross Language, Web and Question-Answering Track experiments using PIRCS. In E. M. Voorhees & D. K. Harman (Eds.), *Information*

- Technology: The Ninth Text REtrieval Conference (TREC-9), NIST Special Publication 500-249*, (pp. 417-426). Washington, DC: US GPO.
- Kwok, K. L., Grunfeld, L., Dinstl, N. & Chan, M. (2002). TREC 2001 Question-Answer, Web and Cross Language Experiments using PIRCS. In E. M. Voorhees & D. K. Harman (Eds.), *The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 452-456). Washington, DC: US GPO.
- Kwok, K. L., Papadopoulos, L. & Kwan, Kathy Y. Y. (1993). Retrieval experiments with a large collection using PIRCS. In D. K. Harman, (Ed.), *The First Text REtrieval Conference (TREC-1), NIST Special Publication 500-207*, (pp. 153-172). Washington, DC: US GPO.
- Lafferty, J., & Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. In *Proc. 24th Ann. Intl. ACM SIGIR Conf. on R&D in IR*, (pp. 111-119).
- Lavenko, V. & Croft, W. B. (2001). Relevance-based language models. In *Proc. 24th Ann. Intl. ACM SIGIR Conf. on R&D in IR*, (pp. 120-127).
- Licklider, J. C. R. (1965). *Libraries of the Future*. Cambridge, MA: MIT Press.
- Moldavan, D., Harabagiu, S., Pasca, M., Mihalcea, R., Goodrum, R., Girju, R., et al. (2000). The structure and performance of an open-domain question answering system. In *Proc. 38th Ann. Mtg. of ACL (ACL-2000)*, (pp. 563-570).
- O'Connor, J (1975). Retrieval of answer-sentences and answer-figures from papers by text searching. *Information Processing & Management*, 11(5/7), 155-164.
- Ponte, J. M., & Croft, B. W. (1998). A language modeling approach to information retrieval. In *Proc. 21st Ann. Intl. ACM SIGIR Conf. on R&D in IR*, (pp. 275-281).
- Prager, J., Chu-Carroll, J. & Czuba, K. (2002). Use of Wordnet hypernyms for answering what-is questions. In E.M. Voorhees & D.K. Harman (Eds.), *The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 250-257). Washington, DC: US GPO.
- Robertson, S. E. & Sparck Jones, K. (1976) Relevance weighting of search terms. *J. of American Soceity of Information Science*, 27, 129-146.
- Salton, G. (1968). *Automatic Information Organization and Retrieval*. New York: McGraw-Hill.
- Salton, G., Fox, E. & Wu, H. (1983). Extended Boolean information retrieval. *Communications of the ACM*, 26(17), 1022-1036.
- Salton, G. & McGill, M. (1983). *Introduction to Modern Information Organization and Retrieval*. New York: McGraw-Hill.
- Srihari, R. K., Li, W. & Li, X. (200x). Question Answering Supported by Multiple Levels of Information Extraction. paper in this volume.
- Soubbotin, M. M. (2002). Patterns of potential answer expressions as clues to the right answers. In E. M. Voorhees & D. K. Harman (Eds.), *Information Technology: The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 293-302). Washington, DC: US GPO.
- Tague-Sutcliffe, J. (1992). Measuring the informativeness of a retrieval process. In *Proc. 15th Ann. Intl. ACM SIGIR Conf. on R&D in IR*, (pp. 23-36).
- Turtle, H. (1994). Natural language vs. Boolean query evaluation: a comparison of retrieval performance. In *Proc. 17th Ann. Intl. ACM SIGIR Conf. on R&D in IR*, (pp. 212-220).
- Turtle, H. & Croft, B. W. (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3), 187-222.
- Voorhees, E. M. (2001). Overview of the TREC-9 Question Answering track. In E. M. Voorhees & D. K. Harman (Eds.), *Information Technology: The Ninth Text REtrieval Conference (TREC-9), NIST Special Publication 500-249*, (pp. 71-79). Washington, DC: US GPO.
- Voorhees, E. M. (2002). Overview of the TREC 2001 Question Answering track.. In E. M. Voorhees & D. K. Harman (Eds.), *The Tenth Text Retrieval Conference, TREC 2001, NIST Special Publication 500-250*, (pp. 42-51). Washington, DC: US GPO.
- Winston, P. H. (1977). *Artificial Intelligence*. Reading, MA: Addison-Wesley.
- Woods, W. A. (1977). Lunar rocks in Natural English: Explorations in NL Q-A. *Linguistic Structures Processing*, 521-569.

CHARLES L. A. CLARKE, GORDON V. CORMACK,
THOMAS R. LYNAM, AND EGIDIO L. TERRA

QUESTION ANSWERING BY PASSAGE SELECTION

Abstract: The MultiText QA System performs question answering using a two step passage selection method. In the first step, an arbitrary passage retrieval algorithm efficiently identifies hotspots in a large target corpus where the answer might be located. In the second step, an answer selection algorithm analyzes these hotspots, considering such factors as answer type and candidate redundancy, to extract short answer snippets. This chapter describes both steps in detail, with the goal of providing sufficient information to allow independent implementation. The method is evaluated using the test collection developed for the TREC 2001 question answering track.

1. INTRODUCTION

In this chapter we treat question answering as a two-step *passage selection* process:

1. *Passage Retrieval:* The extraction of text fragments from a corpus that are related to the topic of the question and therefore may contain the answer. In our approach the extracted fragments are centered on a “hotspot” where the question topic is strongly represented.
2. *Answer Selection:* The identification of short answer snippets within the extracted fragments that are likely to include the answer. Each of these snippets may be a phrase, a sentence, or a fixed-length character string, depending on the requirements of the particular question answering application.

At its simplest, this view of question answering allows issues of question analysis and named entity recognition to be avoided. Our basic version of the technique uses only term and document statistics to drive both retrieval and selection. In practice, the basic version may be augmented with parsing, question classification, pattern matching and other heuristics, but it provides reasonable performance without this augmentation.

The MultiText Question Answering System, developed by our research group at the University of Waterloo, is structured around this two-step process. Using the system, we participated in both the TREC 2000 and TREC 2001 question answering evaluations [6,7]. In both years, the basic technique was augmented only with fairly simple natural language processing and information extraction heuristics. Nonetheless, the system achieved the second-best performance in the TREC 2000 QA experiments and the fifth-best performance in the 2001 experiments. A heavily

modified version of the same system was used in the TREC 2002 evaluation, where exact answers were required and the system again achieved a top-six performance. In this chapter, our focus is on question answering by passage selection, and exact-answer selection will not be discussed. Within this chapter, references to the “MultiText QA system” refer to the system as it existed at the time our TREC 2001 runs were submitted.

Most open-domain question answering systems incorporate an information retrieval component that is used to identify potential answer sources within a corpus [3, 4, 12, 14, 16, 17, 21, 27, 30, 36]. In many systems, entire documents are retrieved by the IR component. In a few systems, predefined components such as paragraphs, sentences or N -word segments are treated as separate documents and document-oriented retrieval techniques are applied to retrieve them. However, it is not obvious that document-oriented IR methods are suited for use in a question answering environment. Text fragments often supply the answer to a question even when the topic of the question is incident to the topic of the document or paragraph containing the fragment.

For question answering, we have developed a passage-retrieval technique that can identify small text fragments that cover as many question concepts as possible. Unlike most other passage-retrieval techniques, our technique does not retrieve predefined document components, but can retrieve any document substring. Usually, the retrieved fragments are considerably smaller than the documents that contain them. The score of a fragment depends on its length, the number of question concepts it contains and the relative weight assigned to each of these concepts. We applied this retrieval technique in all our TREC question answering experiments, including the 1999 experiments when an answer selection component was not used [11] and 50-byte strings centered on the hotspots of the top-scoring fragments were returned as the answer snippets.

In our system, question processing begins by generating a query from the question. The passage-retrieval component uses this query to identify the m highest-scoring fragments from distinct documents. Since these fragments are often only a few words in length, and cover only the question concepts, we retrieve the context surrounding each fragment for use by our answer-selection component. To provide this context, we expand the fragment by n words on each side, and this larger fragment is retrieved from the corpus. The location of the original fragment within each larger fragment is marked as the hotspot for use by the answer-selection component, which selects the final answer snippets from these fragments.

Our answer-selection component identifies potential answer keywords, or “candidate terms”, from the retrieved passages and then ranks these candidate terms using a weighting function based on term statistics. A key aspect of our approach to answer selection is the use of candidate-term *redundancy*.

The use of redundancy allows a combination of evidence from multiple retrieved passages to identify the most likely answer. The appearance of a candidate term in several of the top ranking passages is a strong indication of the importance of the candidate term.

The size of a snippet returned by the answer-selection component depends on the specific requirements of the particular question answering task. For TREC

1999-2001, an answer snippet could be any text string up to 50 bytes in length, but in principle the selection component could be designed to return sentences or short phrases of varying length.

The answer-selection component uses the weights of the candidate terms to select answer snippets from the retrieved passages. Any 50-byte substring of the passages could potentially be returned as an answer snippet. We score each 50-byte substring by combining the weights of the candidates that appear within it and then selecting the highest-scoring snippet. If $M > 1$ snippets are required, the scores of the candidates appearing in the previously selected snippet are set to zero and a second snippet is selected.

We repeat this process, setting candidate weights from selected snippets to zero before selecting each additional snippet, until M snippets have been selected. For TREC 2001 experiments, $M = 5$.

The remainder of the chapter covers the details of passage selection, and provides an experimental evaluation of the approach. Section 2 focuses on the passage-retrieval step. We first discuss our passage scoring method, its rationale, and its efficient implementation. The section then describes our parser, and its role in query generation. Our experience with the TREC QA task indicates that the flexibility of this parser, which is able to make reasonable choices in the face of unexpected question types and formats, makes a substantial contribution to the performance of the system. The section concludes with a review of other passage-retrieval methods. Section 3 begins by presenting a simplified version of our answer selection algorithm, which operates using only basic term statistics. The section then outlines a number of enhancements to the algorithm, including question categorization, pattern matching, and Web data reinforcement. The experimental section (Section 4) examines the performance of the passage-retrieval and answer selection algorithms, including an evaluation of the impact of the various enhancements and a review of our TREC 2001 QA track results.

Our goal in this chapter is twofold: to present a description of our basic question answering method that is complete enough to allow the reader to implement it, and to provide an overview of possible extensions, allowing techniques detailed other chapters to be incorporated into the method. The material presented in this chapter represents a synthesis and substantial expansion of passage retrieval and question answering research previously reported at TREC conferences [6, 7, 11] and elsewhere [9, 10].

2. PASSAGE RETRIEVAL

Our passage-retrieval technique does not depend on predefined passages, but can retrieve any substring of any document in a corpus. Since passages, rather than documents, are the focus of the retrieval technique, our primary view of a corpus is as a single long string, consisting of all the documents in the corpus concatenated together. Under this view, the corpus C can be treated as an ordered sequence of terms:

$$\mathcal{C} = c_1 c_2 \dots c_N$$

A passage from \mathcal{C} is represented by an *extent*, an ordered pair of coordinates (u, v) with $1 \leq u \leq v \leq N$, that corresponds to the subsequence of \mathcal{C} beginning at position u and ending at position v

$$c_u c_{u+1} c_{u+2} \dots c_v$$

A query Q is generated from the original question and takes the form of a term set:

$$Q = \{t_1, t_2, t_3, \dots\}$$

An extent (u, v) *satisfies* a term set $T \subseteq Q$ if the subsequence of \mathcal{C} defined by the extent contains at least one term matching each of the terms from T . The determination that a query term t_i matches a corpus term c_j may take into account stemming, synonym expansion and similar transformations. An extent (u, v) is a *cover* for T if (u, v) satisfies T and the subsequence corresponding to (u, v) contains no subsequence that also satisfies T . That is, there does not exist an extent $(u', v') \neq (u, v)$ with $u \leq u' \leq v' \leq v$ that satisfies T .

An algorithm for efficiently locating covers for subsets of Q is given in Section 2.2. These covers are scored using the equation developed in Section 2.1 and the top-scoring covers become the hotspots required by the answer-selection component. As an example, for TREC question 940 (“How long did Rip Van Winkle sleep?”) our query-generation component produces the query:

“rip” “van” “winkle” “sleep”+“sleeping”+“sleeps”+“slept”

This query consists of four terms, the last of which is a disjunction of morphological variants of the verb “sleep”. A typical fragment retrieved by this query is given in Figure 1. The term cover is shown in **boldface**; two possible answers are shown in *italics*.

While our focus is on passage retrieval, the locations of document boundaries within the corpus are retained for post-retrieval filtering of passages. First, we exclude passages that cross document boundaries. The co-occurrence of question concepts in such a passage may be purely accidental, and a lower scoring passage found at the end of one document or the beginning of the other may be a better choice. Second, we retain only the highest scoring passage from each document. The use of candidate-term redundancy for answer selection depends on the independent appearance of candidates in multiple passages, and the terms appearing in a second passage from the same document are unlikely to exhibit this independence.

...The 20-Year Lost Weekend: Rip Van Winkle Was a Drunk. By DAVID GERMAIN. Associated Press Writer. **Rip Van Winkle did not sleep** the sleep of the enchanted, snoozing 20 years away in the haunted Catskill Mountains. The lovable rogue of Washington Irving's story was a real man who abandoned his wife and children to become an 18th century barfly in New York City, claims literary detective Steven Press...

Figure 1. A typical fragment retrieved for TREC question 940.

2.1 Passage Scoring

Given an extent (u, v) that is a cover for a term set $T = \{t_1, t_2, \dots\}$ we wish to compute a score for the extent with respect to T that reflects the likelihood that an answer to the question is contained in the extent or appears in its close proximity. First, we model a document as a sequence of independently generated terms and assume that there is a fixed probability p_t of a term $t \in T$ matching at any given document location.

Given an extent (u, v) with length $l = v - u + 1$, the probability $P(t, l)$ that the extent contains one or more occurrences of t is

$$\begin{aligned} P(t, l) &= 1 - (1 - p_t)^l \\ &= 1 - (1 - lp_t + O(p_t^2)) \\ &\approx lp_t \end{aligned}$$

The probability $P(T, l)$ that an extent (u, v) contains all the terms from T is

$$\begin{aligned} P(T, l) &= \prod_{t \in T} P(t, l) \\ &\approx \prod_{t \in T} lp_t \\ &\approx l^{|T|} \cdot \prod_{t \in T} p_t \end{aligned}$$

Finally, we estimate p_t from the term frequencies within the corpus:

$$p_t = f_t / N$$

where f_t is the total number of times t appears in the corpus and N is the total length of all documents in the corpus. Our score for an extent of length l containing the terms $T \subseteq Q$ is the self-information of $P(T, l)$:

$$\sum_{t \in T} \log(N/f_t) - |T| \log(l) \quad (1)$$

Equation 1 assigns higher scores to passages whose probability of occurrence is lower. While a higher score does not directly imply a greater likelihood that the answer will appear in close proximity, empirical evidence suggests that this relationship holds. The runs we submitted for the TREC-8 QA experiments in 1999

were based solely on this passage-retrieval technique [11]. At TREC-8 separate experimental tasks allowed both 50- and 250-byte answer snippets to be submitted for evaluation. For TREC-8 we generated queries simply by eliminating stopwords from the questions and treating the result as a term vector. Instead of answer selection, we submitted 250 bytes centered on the five top-scoring extents that covered terms from the query. Despite the simplicity of the technique, answers for 63% of the questions appeared in the submitted snippets, and the system was among the six best-performing systems on the 250-byte task [32].

2.2 Efficient Retrieval

Given a query Q we generate \mathcal{J} , the set of all covers for all subsets of Q , and rank them using equation 1. We discard all but the highest-ranking cover from each document. Of those remaining, the top m are used for analysis by the answer-selection component. Implementation of our passage retrieval and ranking technique depends on a fast algorithm to compute all covers of all subsets of Q . An extent (u, v) is said to *i-satisfy* a query Q if the subsequence of C defined by the extent contains exactly i distinct terms from Q . An extent (p, q) is an *i-cover* for Q if and only if it *i-satisfies* Q and does not contain a shorter extent that also *i-satisfies* Q . Below we present an algorithm to generate \mathcal{J}_i , the set of all *i*-covers of Q , in time $O(|Q| \cdot |\mathcal{J}_i| \cdot \log(N))$. The set of covers for all subsets of Q is simply the union of the *i*-covers

$$\mathcal{J} = \bigcup_{i=1}^{|Q|} \mathcal{J}_i$$

The corpus contains $O(N^2)$ different extents that might legitimately be returned by a passage-retrieval algorithm, and the significance of the extents in \mathcal{J} may not be immediately obvious. If all $O(N^2)$ extents in the corpus were ranked using equation 1, some of the extents not in \mathcal{J} may be ranked before some of the extents in \mathcal{J} . However, from the definition of *i-cover* and from equation 1, any extent not in \mathcal{J} has at least one higher-scoring passage from \mathcal{J} nested within it. Such an extent need not be generated. It would be eliminated from the final ranked list of fragments, since it either occurs in a document with a higher ranking passage (the nested passage) or overlaps a document boundary.

The efficient generation of *i*-covers depends on specific support from the underlying index structures. This support takes the form of two *access functions* $r(t, w)$ and $l(t, w)$ that return positions in the corpus term sequence c_1, \dots, c_N . Both take a term t and a position in the term sequence w as arguments and return results as follows:

$$r(t, w) = \begin{cases} v & \text{if there exists } c_v \text{ matching } t \text{ such that } w \leq v \\ & \quad \text{and does not exist } c_{v'} \text{ matching } t, \text{ such that } w \leq v' < v \\ N + 1 & \text{otherwise} \end{cases}$$

and

$$l(t, w) = \begin{cases} u & \text{if there exists } c_u \text{ matching } t \text{ such that } w \geq u \\ & \quad \text{and does not exist } c_{u'} \text{ matching } t, \text{ such that } w \geq u' > u \\ 0 & \text{otherwise} \end{cases}$$

Informally, the access function $r(t, w)$ returns the position of the first occurrence of the term t located at or after position w in the term sequence. If there is no occurrence of t at or after position w , then $r(t, w)$ returns $N+1$, one position beyond the end of the corpus. Similarly, the access function $l(t, w)$ returns the position of the last occurrence of the term t located at or before position w in the term sequence. If there is no occurrence of t at or before position w , then $l(t, w)$ returns 0 .

The $r(t, w)$ and $l(t, w)$ access functions may be easily implemented using file structures that can resolve a call to either access function in $O(\log N)$ time. The MultiText QA System uses an extended inverted-index file structure originally developed to support retrieval from structured text [8]. The file structures store a sorted list of positions for each term, organized to allow the list to be efficiently addressed by position w , directly supporting the access functions and permitting portions of the list to be skipped when possible. Other file structures that permit portions of the term list to be skipped, such as the self-indexing structures of Moffat and Zobel [23], are also appropriate.

The generation of i -covers is achieved through calls to the r and l access functions. The procedure Cover (Figure 2) takes as its arguments a set of query terms Q , a ranking level i , and a term sequence position w , and generates the first i -cover for Q that starts at or after w .

The loop over lines 2–4 of Figure 2 calls the access function $r(t_j, w)$ for each term $t_j \in Q$, recording the results in \mathcal{R} , an array of positions within the corpus. For each term $t_j \in Q$, $1 \leq j \leq |Q|$, the position of its first occurrence at or after w is assigned to $\mathcal{R}[j]$. At line 5, the variable v is assigned the i th largest element of \mathcal{R} . From the definition of $r(t, w)$, the extent (w, v) i -satisfies Q , and any extent (w, v') with $w \leq v' < v$ will not i -satisfy Q . Therefore, the first i -cover for Q starting at or after position w ends at position v .

Lines 6–11 construct the set Q' consisting of the i terms from Q that appear in the interval of the corpus associated with (w, v) . The loop over lines 13–15 calls the access function $l(t'_j, v)$ for each term $t'_j \in Q'$. For each term $t'_j \in Q'$, $1 \leq j \leq i$, the position of its last occurrence at or before v is assigned to the array element $\mathcal{L}[j]$. At line 16, the integer variable u is assigned the smallest element of \mathcal{L} . From the definition of $l(t, w)$, the extent (u, v) i -satisfies Q , and any extent (u', v) with $u < u' \leq v$ will not i -satisfy Q . Therefore, (u, v) is an i -cover for Q .

At line 17, (u,v) is returned to the caller. If \mathcal{Q} has no i -cover starting after the specified w , the extent $(N+1, N+1)$ is returned. Generating a single i -cover requires $|\mathcal{Q}|+i = O(|\mathcal{Q}|)$ calls to access functions.

By definition, i -covers do not nest. It follows immediately that no two i -covers can share a common starting position. The set \mathcal{J}_i of all i -covers of \mathcal{Q} can therefore be generated by successive calls to the Cover procedure:

$\mathcal{J}_i \leftarrow \emptyset$
$(u,v) \leftarrow \text{Cover}(\mathcal{Q}, i, 0)$
while $u \leq N$ do
$\mathcal{J}_i \leftarrow \mathcal{J}_i \cup \{(u,v)\}$
$(u,v) \leftarrow \text{Cover}(\mathcal{Q}, i, u+1)$
end while

The generation of \mathcal{J}_i requires $O(|\mathcal{Q}| \cdot |\mathcal{J}_i| \cdot \log(N))$ time. Overall, generating \mathcal{J} , the set of all covers for all subsets of \mathcal{Q} requires $O(|\mathcal{Q}| \cdot |\mathcal{J}| \cdot \log(N))$ time.

In practice, all elements of \mathcal{J} may not need to be generated, since only the top m elements of \mathcal{J} from distinct documents are required for answer selection. We begin the ranking process by generating $\mathcal{J}_{|\mathcal{Q}|}$, the set of covers for all terms of \mathcal{Q} , and then generate $\mathcal{J}_{|\mathcal{Q}|-1}$ down to \mathcal{J}_1 in succession. Before the generation of \mathcal{J}_i for a particular i , it may be the case that m or more covers have already been generated, with the lowest scoring cover having score x . Using equation 1 with the length of the covering extent set to zero ($l=0$), \mathcal{Q} can be checked to determine the largest score that can be obtained from i of its terms. If this score does not exceed x , the cover generation process can be halted. Similarly, it may be the case that a particular $t \in \mathcal{Q}$ cannot be combined with any other $i-1$ terms from \mathcal{Q} to produce a cover with score greater than x . In this case, t can be eliminated from further consideration.

$\text{Cover}(\mathcal{Q}, i, w) \equiv$	
1	Let $t_1, \dots, t_{ \mathcal{Q} }$ be the elements of \mathcal{Q}
2	for $j \leftarrow 1$ to $ \mathcal{Q} $ do
3	$\mathcal{R}[j] \leftarrow r(t_j, w)$
4	end for
5	$v \leftarrow$ i th largest element of \mathcal{R}
6	$\mathcal{Q}' \leftarrow \emptyset$
7	for $j \leftarrow 1$ to $ \mathcal{Q} $ do
8	if $\mathcal{R}[j] \leq v$ then
9	$\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{t_j\}$
10	end if
11	end for

12	Let $t'_1, \dots, t'_{ Q' }$ be the elements of Q'
13	for $j \leftarrow 1$ to $ Q' $ do
14	$\mathcal{L}[j] \leftarrow l(t'_j, v)$
15	end for
16	$u \leftarrow$ smallest element of \mathcal{L}
17	return (u, v)

Figure 2. Passage Retrieval — Given a query Q , a ranking level i , and a corpus position w , the cover generation procedure generates the first i -cover for Q starting at or after w . The procedure calls the l and r access functions to generate term positions

The top scoring m elements of \mathcal{J} from distinct documents become the raw material for answer selection. Since an element of \mathcal{J} may cover only query terms, it is expanded by including text at the beginning and end to provide context. Up to n words of text may be added to both the beginning and end. In most cases we use $n = 100$. After expansion, the result is a set \mathcal{F} of m text fragments, which becomes the input to the answer-selection component. Within each text fragment, the location of the original cover is also marked as the hotspot for use during answer selection.

2.3 Question Analysis

The question-analysis component takes a natural language question as input and yields a set of terms amenable for input to the passage retrieval system. Question analysis also yields answer categories used by the answer selection algorithm.

To achieve its objectives, the question analyzer looks for textual elements characteristic of some question form. To this end, we use a context-free grammar generating the forms of interest, and find the most likely derivation of a question using a probabilistic version of Earley’s algorithm. The context-free grammar has been augmented to form an attribute grammar and the attributes are evaluated over the most probable derivation.

The rules of the context-free grammar are augmented by weights. By default, each rule is assumed to have a weight of 1, meaning that it is as likely to be the best rule as any other. A weight of 0.5 means it is half as likely; a weight of 2.0 means it is twice as likely, and so on. The context-free grammar generates a sequence of terminal symbols. The weight of each terminal symbol is the product of the weights of the grammar rules from which it is derived.

The terminal symbols of the context-free grammar comprise parts of speech, numbers, punctuation and a few specific words that often appear in questions (“how”, “which”, “does”, “many”, etc.). Words, numbers, and punctuation are recognized by simple lexical analysis. Parts of speech are first recognized as words, and then looked up in a dictionary (WordNet or WorldBook). The probability that a word is used as a particular part-of-speech is estimated to be proportional to the number of senses it has with that part-of-speech in the dictionary. If the word also

appears literally as a terminal symbol in the grammar, this usage is given high probability. Thus, for each word recognized by lexical analysis a probability distribution is returned to the parser – each element of the distribution indicates the probability that the word represents a particular terminal symbol in the grammar.

Using Earley's algorithm, the best derivation is determined according to the following rules. The sequence of terminals generated by the derivation is matched in 1–1 ordered correspondence with the elements recognized by lexical analysis. The probability that each corresponding pair is correct is estimated using the distribution described above. This probability is multiplied by the weight of the non-terminal to give the weight of the corresponding pair. The product of the corresponding-pair weights gives the weight of the derivation. The best derivation is defined to be one with the highest weight.

The best derivation is represented as a tree, and a recursive procedure computes attribute values for each tree node. Overall attribute values are computed at the root. The most important attribute values computed are: part-of-speech, question-category, quoted-string, and instance-of.

The part-of-speech attribute computed by the parser is used to determine the role of each word as a query term. Nouns, adverbs, and adjectives are used directly as query terms. Verbs are stemmed if they are regular, and expanded if they are irregular. Articles and prepositions are discarded unless otherwise specified in the grammar rules. We treat quoted strings as phrases and use them directly as query terms. In addition, individual words contained in quoted strings are included in the query as separate terms.

2.4 Other Passage Retrieval Techniques

While the use of passage retrieval in question answering dates back two decades to O'Connor's work [26], most research on passage retrieval was undertaken in a general IR context, rather than the specific context of question answering. With a few exceptions [18, 19, 36], most passage retrieval techniques are not designed to return arbitrary document substrings. Generally, passage retrieval techniques require that the division of each document into passages be done *a priori* before an index is created [2, 5, 15, 29, 33, 34]. Typically, documents are divided into (possibly overlapping) segments containing a fixed number of words, sentences or paragraphs, and standard document-oriented IR techniques are used for retrieval.

Woods *et al.* [36] describe a QA system that uses a passage-retrieval technique [37] with properties similar to ours, including the ability to return arbitrary document substrings. Their “relaxation ranking” technique locates passages where multiple query terms occur close together. Preference is given to passages where exact matches to query terms are found and where the terms appear in query order. In addition to the passage-retrieval technique, the system is distinguished by its heavy use of term expansion.

The search engine used in the GuruQA system [27] extracts passages consisting of up to n sentences, where n is a parameter specified at query time. For retrieval, question terms are converted to a canonical form and expansion is used for special

question concepts. For example, the query for a “how tall” question would include a query token that matches any one-dimensional distance measure. Each query term has a weight assigned during question analysis, which is used to score the passages. Shorter passages are preferred and this preference is reflected in the scoring function.

Many question answering systems do not use passage retrieval techniques, but instead use document-oriented IR techniques to extract sets of documents from a corpus for further analysis. After retrieval, these documents are sometimes subject to a second sentence selection phase that essentially simulates passage retrieval at the sentence level [16, 21, 27, 30, 31]. This approach is facilitated by the existence of many public available IR systems. Unfortunately, document-oriented retrieval is not necessarily optimal for question answering, especially when documents have broad topic coverage. As a result, relevant passages in off-topic documents can be ignored. Little effort has been devoted to comparing document retrieval with passage retrieval for question answering.

Hovy *et al.* [16] use the MG search engine [35] to evaluate boolean queries for question answering. For each question, a fixed number of documents is retrieved. If the boolean query does not yield the desired number of documents then it is relaxed and resubmitted. Retrieved documents are divided into sentences that are ranked using a series of heuristics, and the top 300 sentences are retained for processing by their answer-matching component.

Singhal *et al.* [30] use SMART for document retrieval. Retrieved documents are broken into sentences, which are scored on the presence of the query terms within them. If query terms appear next to each other then the score is increased. Passages comprised of multiple sentences are also identified. Adjacent sentences can be attached to a higher scoring one when these sentences contain query terms that do not appear in the higher scoring sentence. Using this technique, they found that the top 50 passages contained at least one answer for 86% of TREC-8 QA test set.

Srihari and Li [31] use AT&T engine for document retrieval. All content words from the question are used as query terms, along with morphological variants of the main verb and generic terms related to the question type. After retrieval, sentences are ranked according to counts of query term occurrences, the order of appearance of the query keywords within sentences, and the nature of the match (exact, synonym or morphological variant).

A different approach to selecting passages from extracted documents is presented by Monz and de Rijke [24]. Question words are stemmed to generate query terms, and an IR system, using the vector space model, retrieves documents. A partial parser identifies a limited set of syntactical structures, including noun phrases, prepositional phrases and verb groups. For noun and prepositional phrases, the head noun is marked, and for each verb group, a dependency structure is generated, having the verb group as head and all the noun and prepositional phrases in the vicinity of the verb as its dependents. This structure represents a passage that is later used to locate the answer for the question.

For each question, the MULDER Web-based QA system [20] issues one or more queries to the Google search engine. To generate these queries, MULDER applies transformations to the question that attempt to anticipate the syntactic form the

answer will take. Possible transformations include verb expansion, quotation of noun phrases, and rewriting of the question into assertions. MULDER downloads the documents referenced in the list of links Google returns for each query. Passages of up to 40 words in length are extracted from these documents and scored with a formula that uses both the idf-weights of the query terms appearing a passage and the distance between the terms. The top scoring passages are then passed to MULDER's answer selection module.

Like MULDER, the AnswerBus system [38] extracts passages from documents identified by a commercial Web search service, but chooses from one of several different search services, depending on the nature of the question. Rather than downloading entire documents, other Web-based QA systems, such as AskMSR system [4] and WebQA [22], use the document summaries provided by the search engine itself.

3. ANSWER SELECTION

The answer selection component analyzes the fragments returned by the passage retrieval component and extracts a ranked list of snippets from these fragments to be returned as the result. Our answer selection algorithm does not attempt to identify exact answers. Instead, the algorithm first extracts and ranks candidate terms from the fragments. Underlying the algorithm is an assumption that a snippet containing high-ranking candidate terms is also likely to contain a complete answer.

The algorithm treats any alphanumeric string surrounded by non-alphanumeric characters appearing in a retrieved fragment as a candidate term. Case normalization is applied to these candidate terms, but stemming and other transformations are not. The algorithm computes a weight for each candidate term and uses these weights to select answer snippets from the retrieved fragments.

Next, we present our approach to candidate-term weighting, followed by a statistical answer selection algorithm using these weights. We then present a number of heuristics that improve the performance of the basic algorithm. One of the most important of these heuristics is the use of question categorization and pattern matching to improve the process of candidate-term identification. Another important heuristic is the use of Web data to support the answer selection process through candidate-term redundancy.

3.1 Candidate-Term Weighting

The weight assigned to a candidate term t combines an idf-like weight with a *redundancy factor* that takes into account the number of fragments containing the candidate term and its location relative to the hotspot within these fragments. In our experience, the redundancy factor is a critical contributor to answer selection [9].

The answer-selection component takes as input an ordered set of m fragments, \mathcal{F} , with each fragment containing a hotspot. We use the notation \mathcal{F}_i ($1 \leq i \leq m$) to refer

to the i th fragment in \mathcal{F} . Each fragment is split into tokens, where tokens are sequences of alphanumeric characters separated by non-alphanumeric characters.

For each fragment F_i containing one or more tokens that match a candidate term t ($t \in F_i$), we determine the distance from the hotspot to the closest token matching t , which we refer to as $loc(F_i, t)$. The distance of a candidate term from the hotspot is measured in token positions, with candidate terms appearing in the hotspot itself treated specially. If t appears in the hotspot then $loc(F_i, t) = 0$. Otherwise, $loc(F_i, t) > 0$. If t appears just outside the hotspot, then $loc(F_i, t) = 1$, with the value of $loc(F_i, t)$ increasing as the closest occurrence of t is found farther away. When comparing tokens with candidates, case is ignored.

The calculation of a weight W_t for candidate term t combines $loc(F_i, t)$ with the term weight from Section 2.1:

$$W_t = \sum_{1 \leq i \leq m, t \in F_i} \log \left(\frac{N}{f_t \cdot (loc(F_i, t) + 1)} \right) \quad (2)$$

where N is sum of the lengths of all documents in the corpus and f_t is the number of occurrences of t in the corpus.

Equation 2 is an extension of the candidate-term weighting formula described in earlier work [6, 7, 9]. In that work, redundancy was measured with a simple count of the number of fragments in which a candidate term appears, in effect defining $loc(F_i, t) = 0$, regardless of the location of the candidate term relative to the hotspot.

3.2 Answer Selection Algorithm

Our basic answer selection algorithm is presented in Figure 3. Along with \mathcal{F} , the ordered set of m fragments, the answer selection algorithm requires as input the target number of answer snippets to be generated (M), and the maximum allowed length of each generated snippet (len).

The selection algorithm views each fragment in two ways, as a character string and as a sequence of tokens. When viewed as a character string, the notation $F_i[j, k]$ refers to the substring of F_i starting at character position j and extending to position k . When viewed as a sequence of tokens, the notation $token(F_i, j)$ refers to the j th token in F_i . The starting character position of $token(F_i, j)$ within F_i is $start(F_i, j)$, and its ending position is $end(F_i, j)$. The total number of tokens in F_i is $tokens(F_i)$.

Line 1 of the algorithm initializes the set $usedTerms$ to the empty set. We use this set to track the candidate terms that have already appeared in generated snippets, so that they do not contribute to the scores of additional snippets.

Each loop over lines 2 – 30 generates a single answer snippet. The variable $bestScore$, which is initialized to zero on line 3, maintains the score of the best snippet seen. The loop over lines 4 – 25 considers each fragment in turn, and the loop over lines 5 – 24 considers each token of the fragment as the possible starting point for a snippet. Lines 6 – 9 expand the snippet under consideration to include as

many tokens as possible, with the test at line 10 ensuring that the first token is not itself longer than the maximum length of an answer snippet.

Lines 11–17 compute the score of the snippet, and lines 18–22 record location of the snippet if its score exceeds the best score seen so far. For a term t to contribute to the score, it must be a valid candidate term and should not have appeared in a previously generated snippet. Line 14 performs this test. If the test succeeds, equation 2 is applied at line 15, and the candidate-term score is combined into a running total for the snippet.

The exponent γ allows us to trade off the desire to include as many candidate terms as possible against the desire to include the highest-scoring candidate terms. Before TREC 2000, we experimented with a range of values $0 < \gamma \leq \infty$. The value $\gamma = 3$ provided consistently good performance and has been used in all our subsequent experiments.

On lines 26–28 the terms appearing in the selected snippet are added to the set *usedTerms*. On line 29 the selected snippet is reported by the algorithm.

The function *validCandidate(t)* (line 14) determines whether a token is a valid candidate term. The question categorization and pattern matching heuristics, which will be considered next, operate by specializing this function according to the category of the question. If these heuristics are not used or if the question category cannot be determined, a generic version of this function is used. In our current implementation of the answer selection algorithm, the generic *validCandidate(t)* function excludes only stopwords and tokens that match query terms.

3.3 Question Categorization and Candidate Pattern Matching

When the question analyzer assigns a specific category to a question, the answer-selection component takes advantage of this information by restricting the set of candidate terms under consideration to those that match patterns corresponding to the category. This restriction can be effected by using a specialized *validCandidate(t)* function for each question category.

The question analyzer uses the question-category and instance-of attributes computed by the parser to assign categories to questions. To assign a category, a table of question-category and instance-of pairs is searched. For example, if question-category is “how much” and instance-of is “money”, we pass the category MONEY to the answer selection process. If question-category is “how many” and instance-of is “dollars”, the same processing would result.

For most categories, patterns are defined using regular expressions. For example, the regular expression for the PROPER category (in POSIX notation) is simply [A-Z][A-Za-z]+, which matches any string of at least two alphabetic characters starting with a capital. The *validCandidate* function for the PROPER category returns true only if a token matches the regular expression and is not a stopword or question keyword.

Select(F, m, M, len)	
1	<i>usedTerms</i> $\leftarrow \emptyset$
2	for $r \leftarrow 1$ to M do
3	<i>bestScore</i> $\leftarrow 0$
4	for $i \leftarrow 1$ to m do
5	for $j \leftarrow 1$ to $\text{tokens}(F_i)$ do
6	$k \leftarrow j$
7	while $k < \text{tokens}(F_i)$ and $\text{end}(F_i, k) - \text{start}(F_i, j) + 1 \leq len$
8	$k \leftarrow k + 1$
9	end while
10	if $j < k$ then
11	<i>score</i> $\leftarrow 0$
12	for $z \leftarrow j$ to $k - 1$ do
13	let t be $\text{token}(F_i, z)$
14	if $t \notin \text{usedTerms}$ and $\text{validCandidate}(t)$ then
15	<i>score</i> $\leftarrow \text{score} + (W_t)^{\gamma}$
16	end if
17	end for
18	if $\text{score} > \text{bestScore}$ then
19	$i' \leftarrow i$
20	$j' \leftarrow j$
21	$k' \leftarrow k$
22	end if
23	end if
24	end for
25	end for
26	for $z \leftarrow j'$ to $k' - 1$ do
27	<i>usedTerms</i> $\leftarrow \text{usedTerms} \cup \text{token}(F_{i'}, z)$
28	end for
29	generate $F_{i'}[j', k' - 1]$ as the r -ranked answer snippet
30	end for

Figure 3. Answer Selection — Given F , a set of m retrieved fragments, select the M best scoring answer snippets with a maximum width of len characters

Candidate-term matching takes place token by token even though many answers will consist of multiple tokens. Thus the regular expression for PROPER does not attempt to match full names, for example, matching the first and last names “Wilbur” and “Wright” separately, rather than the name “Wilbur Wright” as a whole. While this approach might be expected to decrease the precision of pattern matching, it allows the answer “Wilbur and Orville Wright” to be returned in response to the question “Who made the first airplane that could fly?” (TREC question 234).

Even for the DATE category, matching takes place token by token. The *validCandidate(t)* function for dates accepts any numeric value less than 9999, along with a large number of date and time related strings (“October”, “oct”, “Monday”, etc.). While this approach may seem crude, the MultiText QA system usually achieves good results on DATE questions.

Figure 4 informally summarizes that categories and patterns used for the experiments reported in Section 4.2. These are a subset of the categories and patterns used in our TREC 2001 QA system.

```

PROPER=[A-Z][A-Za-z]+
DATE=[0-9][0-9]?[0-9]?[0-9]?+month names and day names
NUMBER=[0-9]+ number words ("one", "two", "twenty",...)
LENGTH = NUMBER + length units
DURATION = NUMBER + time units
HOWLONG = LENGTH + DURATION
MASS = NUMBER + mass and weight units
AREA = NUMBER + area units
VOLUME = NUMBER + volume units
HOWLARGE = MASS + LENGTH + AREA + VOLUME
TEMPERATURE = NUMBER +temperature units
MONEY = NUMBER + list of currencies
COUNTRY = list of terms appearing in country names
STATE = list of terms appearing in state names

```

Figure 4. Summary of Candidate Patterns

3.4 Web Reinforcement

An addition to our system for TREC 2001 was the use of Web data to reinforce the scores of candidate terms by providing additional redundancy. From each question, the system generated queries for two leading commercial search engines, Google and Altavista. The text of the top 200 documents returned by each engine was used to create an auxiliary corpus. The passage-retrieval component then

extracted 20 passages from the TREC corpus and 40 passages from the auxiliary corpus, recording the source of each passage. Since the contents of the auxiliary corpus were biased by the queries, term statistics from the TREC corpus were used during passage retrieval from the auxiliary corpus.

All 60 passages were passed to the answer-selection component. The answer-selection component then proceeded to select answer snippets as usual, except that snippets could not be selected from passages retrieved from the auxiliary corpus. The Web data influenced the answer selection process only through redundancy, by increasing the weight of particular candidate terms.

3.5 *Other Heuristics*

The answer selection heuristics used in our actual TREC 2001 QA system differed in several respects from the presentation given in this chapter. Generally, we have attempted to simplify the details without unduly compromising performance. In a few cases, the simplified heuristics appear to provide superior performance. A few heuristics have been omitted entirely, either because they do not provide sufficient benefit to warrant the complexity of their description or because we do not feel we have identified the most appropriate way to incorporate the information they exploit into our overall approach. Here we provide an outline of some of the main differences between our TREC 2001 system and the approach detailed in this chapter.

The answer selection algorithm given does not use the rank and score of the fragments in which candidate terms and answer snippets appear. Although passage retrieval produces a ranked list of m fragments, this ranking is ignored by the answer selection algorithm, which treats all fragments equally. Our TREC 2001 system adjusts both candidate-term weights and the scores of potential answer snippets in an attempt to account for this information.

Our TREC 2001 system used 22 question categories, arranged hierarchically. If insufficient candidate terms were identified using the question category assigned by the parser, the selection component considered patterns matching categories farther up the hierarchy. Pattern matching also took into account the context of matches when scoring potential answer snippets.

Finally, our TREC 2001 system made extensive use of WordNet for answering definitional questions (“What is an atom?”) and biographical questions (“Who was Galileo?”). If the subject of the question could be found in WordNet, then the scores of candidate terms appearing in the definition or biography were increased.

3.6 Other Answer Selection Techniques

From retrieved documents or passages, many question answering systems extract named entities corresponding to a question's category and treat these named entities as candidate answers. Answer snippets returned by these systems are required to contain one or more occurrences of the extracted entities. Few question answering systems follow our general approach of using statistical answer selection techniques in combination with simple pattern matching to rank text snippets, rather than extracted entities, which allows us to avoid the complexities of named entity recognition. Nonetheless, many of these systems rank extracted entities using techniques that share some of the features of our system.

Abney *et al.* [1] use a form of redundancy to rank extracted entities. Each entity receives a score that is determined by the rank and number of retrieved passages that contain it. An entity receives ten points if it appears in the top-ranked passage and one point each time it appears in any other passage.

The answer selection module of the MULDER system [20] scores each extracted entity with a formula that takes into consideration the entity's distance from query terms. Entities are clustered, and the clusters are ranked by the sum of the scores of the entities contained within them. MULDER chooses the top-scoring entity in each cluster as a representative of the cluster and returns a snippet containing it to the user.

In its approach to answer selection, the AskMSR system [4,13] closely resembles our own. To generate answer candidates, the system mines one- to three-word n-grams from the page summaries returned by a commercial Web search engine, and exploits redundancy to score these n-grams. The n-grams are then filtered to boost the scores of n-grams that most closely fit the question category and tiled to produce answer candidates that may be longer than three words in length.

Several other answer selection techniques exploit redundancy. The Werlect algorithm, described by Prager *et al.* [28], ranks candidate answers using both redundancy and proximity to query terms. The AnswerBus system [38] incorporates a measure of redundancy in the form of sentence-to-sentence similarity. The WebQA system [22] ranks entities by counting the number of retrieved passages in which they occur.

4. EXPERIMENTAL EVALUATION

In this section, the effectiveness of the passage selection process described in this chapter is demonstrated through a presentation of results from TREC 2001 and through a separate comparative evaluation of the individual factors that contribute to system performance. Voorhees chapter provides an overview of the TREC QA track, and it should be consulted for a complete explanation of the terminology and evaluation measures used in this section.

Many of the reported results are generated by automatic evaluation scripts. These scripts were created by Ellen Voorhees for the 1999 and 2000 TREC questions [32] and by Ken Litkowski of CL Research for the 2001 TREC questions. The scripts execute a series of question-specific regular expressions over the answer snippets returned for each question. Whenever a match occurs, the snippet is marked as correct. Generally, automatic evaluation by scripts is not as accurate as manual evaluation by human judges, with scripts being more forgiving. However, automatic evaluation does allow different QA approaches to be compared easily. A comparison between the results of automatic evaluation and manual evaluation may be seen in the TREC 2001 results reported next.

Table 1: TREC 2001 Experiments

Run	MultiText Judgments	Script Judgments	NIST Judgment (strict/lenient)
MultiText baseline (uwmta2)	0.379	0.359	0.346/0.365
+Web re-enforcement (uwmta1)	0.483	0.469	0.434/0.457
Web data only	0.608	0.472	N/A
TREC2000 system	0.317	0.307	N/A

4.1 TREC 2001 Experiments

MultiText submitted three runs for the main task of the TREC 2001 QA track. The first run (labelled uwmta1) was based on the fully enhanced system described above, including the use of Web resources for reinforcement. For the second run (uwmta2) the Web resources were not used for reinforcement, but the system was otherwise identical. A comparison between these runs represented a major goal of our TREC experimental work. The third run (uwmta0) tested a last-minute enhancement that had little or no effect, and it will not be discussed further.

The results are presented in Table 1, which gives mean reciprocal rank (MRR) values for several experimental runs using three different sets of judgments. The fourth column lists the official NIST judgments. The third column lists the results of unofficial judgments made by the automatic evaluation script. As expected the results reported by the evaluation script are somewhat higher than the official results. The second column lists unofficial judgments made by one of the authors (Clarke) immediately after the official runs were submitted to NIST. Although creation of these unofficial judgments required less than two hours of total effort, their relative values appear to correlate well with the official judgments, with slightly higher absolute values. In the discussion below, we use the MultiText judgments and the script judgments to support comments that cannot be supported by the official numbers.

The use of Web reinforcement produced a 25% improvement (uwmta2 vs. uwmta1). Considering that the Web data can influence the answer selection process only through redundancy in candidate-term weighting, the magnitude of the improvement was surprising and provides substantial support for our view that redundancy is a key factor in question answering [9].

To provide an additional comparison, the top five answers were selected from the Web data used to reinforce the MultiText runs (“Web data only”). This run illustrates that open domain question answering over Web data can produce excellent performance. Under the MultiText judgments, this run contains an answer to 72% of the questions within the five snippets returned. The difference between the MRR reported by the judging script and the MultiText judgments largely appears to be an artifact of the script. The patterns used by the script closely match the syntactic forms of the answers as they appear in the TREC corpus, and we believe that the MultiText judgments are closer to those that would be obtained by an independent human judge.

As a final experiment, we executed our TREC 2000 system on questions from TREC 2001. Based on the both the MultiText and script judgments, the total effect of our efforts between 2000 and 2001 was an overall performance improvement of more than 50%. Approximately half of this improvement was due to the use of Web reinforcement.

The remainder was due to improvements in candidate-term weighting, the answer selection algorithm, and other heuristics.

4.2 Comparative Evaluation

We now compare the influence of each of the techniques introduced in Sections 2 and 3 on overall question answering performance. For this comparison we use all the questions from TREC 1999, 2000 and 2001, eliminating only the questions for which no patterns are available in the automatic scripts. These 1313 questions were evaluated over two test corpora, the TREC QA corpus and a Web corpus of roughly one terabyte of HTML and text gathered from the general Web in mid-2001.

The Web corpus was generated by starting with a seed set of 2,392 URLs representing the home pages of universities and other educational organizations. Pages were gathered in breadth-first order with one exception. If a breadth-first ordering would place an excessive load on a single host, defined as more than 0.2% of total crawler activity over a time period of approximately one hour, URLs associated with that host were removed and re-queued until the anticipated load dropped to an acceptable level. Pages at given depth from the seed set were crawled in random order. During the crawl, duplicate pages were detected and eliminated, and do not form part of the final collection. A breadth-first ordering is known to produce high-quality pages [25], and we would expect a breadth-first crawl of a terabyte of Web data to contain the answer to many factual questions.

Table 2: Comparative Evaluation

	TREC Corpus		Terabyte Web Corpus	
	Stopped and Stemmed	Parsed Generated	Stopped and Stemmed	Parsed Generated
Raw Passages	0.158	0.169	0.123	0.140
Statistical Selection	0.328	0.368	0.378	0.429
+ Patterns	0.365	0.412	0.410	0.470
TREC 2001 System	0.383	0.426	0.410	0.464

Our comparative evaluation appears in Table 2. For each corpus the table lists MRR values for two different query generation techniques (“stopped and stemmed” and “parser generated”) and four different answer selection techniques. The “stopped and stemmed” queries were created using the standard IR technique of removing stopwords from the questions and applying stemming to remaining words; the “parser generated” queries were produced by the question analyzer described in Section 2.3. Both sets of queries were evaluated over the test corpora using the passage-retrieval algorithm of Section 2.

The row of the table labeled “raw passages” demonstrates the performance of passage retrieval without answer selection. For each question, the word-aligned snippet of 50 bytes or less closest to the center of each of the top five fragments was selected as an answer snippet. The impact of the basic statistical answer selection algorithm of Section 3.2 is shown in the next row, labeled “statistical selection”. In all cases the MRR values are double, and in some cases triple, the corresponding MRR values for the raw passages. As shown in the third row (“+ patterns”) the use of question categorization and candidate-term pattern matching (Section 3.3) provides a performance improvement of 8% to 12% over the basic statistical selection algorithm. Finally, the last row of the table presents results for the full TREC 2001 system, including all the heuristics described in Section 3.5.

Comparing the stopped-and-stemmed queries with the parser-generated queries, we see that, apart from the “raw passages”, the use of the parser for query generation provides a performance improvement of 11% to 14%. The overall effect of question parsing, through its effects on both query generation and question categorization, can be seen by comparing runs using stopped-and-stemmed queries and no candidate-term pattern matching with the corresponding runs using parser-generated queries and pattern matching. Question parsing improves performance by 26% over the TREC corpus and by 24% over the Web corpus.

Given that the Web corpus is nearly 300 times larger than the TREC corpus, it may be surprising that the MRR values over the Web corpus are at most 17% greater. However, several factors other than size distinguish the TREC corpus from the Web corpus. All questions are known to have answers in the TREC corpus.

Indeed, the existence of a known answer in the corpus was one of the criteria for selecting questions for inclusion in the TREC evaluations. Moreover, it is reasonable to characterize the TREC corpus, which is composed primarily of professionally edited newspaper articles, as a higher quality collection than the hodgepodge of general HTML and text that comprises the Web corpus. Finally, as noted earlier, the automatic evaluation scripts use patterns that reflect the syntactic forms found in the TREC corpus, and an examination of the snippets generated from the Web data reveals many reasonable answers that are not identified by the scripts.

From the comparative evaluation in Table 2 it is clear that good performance cannot be achieved by merely returning a snippet of text centered at the hotspot. However, answers are more likely to appear within or close to the hotspot than farther from it, and this observation is reflected in our candidate-term weighting. *Figure 5* illustrates the relationship between the hotspot and the location of matches to answer patterns within retrieved fragments. The figure is based on the 51,014 fragments (up to 40 for each of the 1313 questions) retrieved from the Web corpus. Each fragment was tokenized and the starting token position of each match to an answer pattern was recorded. Token position 0 corresponds to a match within the hotspot, while positive and negative token positions correspond to matches before and after the hotspot. While less than 5% of the retrieved fragments contain matches within their hotspot, answers clearly cluster around the hotspots.

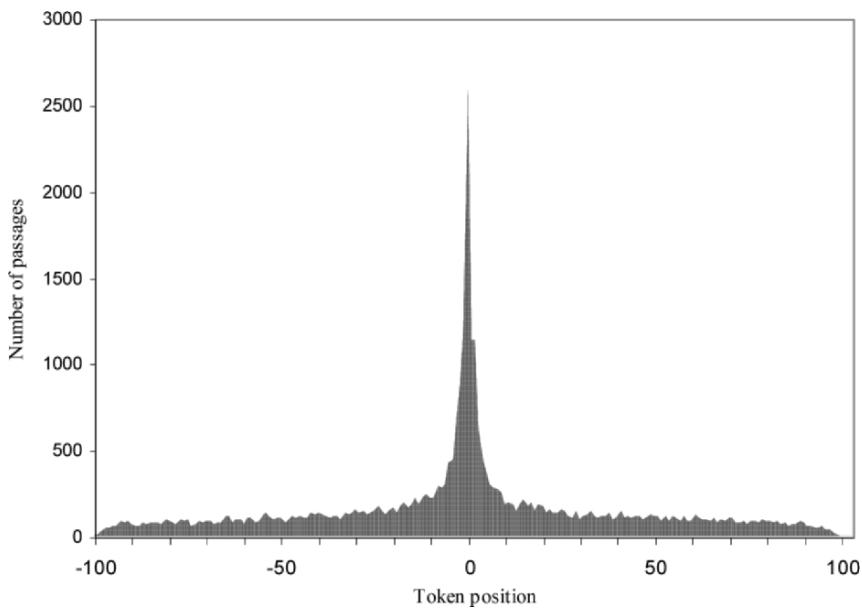


Figure 5. Answer Location Relative to Fragment Hotspot

5. CONCLUSION

This chapter presents a description and evaluation of our statistical passage selection approach to question answering. The approach combines a passage-retrieval component capable of retrieving arbitrary document substrings with a passage-oriented answer-selection component that requires only basic term statistics for operation. With the addition of simple question categorization and pattern matching components, performance is comparable to some of the best-performing systems in the TREC QA evaluations.

We intend the level of detail in this paper to be sufficient to facilitate independent implementation of any or all aspects of the approach, allowing replication of our results. The approach can provide a basis for the exploration and development of additional heuristics. Our TREC 2002 work, where exact answers are required, builds on the approach, adding an improved pattern-matching component that can identify exact answers, along with improved parsing, question categorization and several new components.

6. REFERENCES

- [1] Abney, Steven, Michael Collins, and Amit Singhal. Answer extraction. In Conference on Applied Natural Language Processing, 2000.
- [2] Allan, James Relevance feedback with too much data. In 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 337-343, Seattle, July 1995.
- [3] Breck, Eric, John Burger, David House, Marc Light, and Inderjeet Mani. Question answering from large document collections. In 1999 AAAI Fall Symposium on Question Answering Systems, North Falmouth, MA, 1999.
- [4] Brill, Eric, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-intensive question answering. In 2001 Text REtrieval Conference, Gaithersburg, Maryland, November 2001.
- [5] Callan, James Passage-level evidence in document retrieval. In 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 302-310, Dublin, July 1994.
- [6] Clarke, C. L. A., G. V. Cormack, D. I. E. Kisman, and T. R. Lynam. Question answering by passage selection. In 9th Text REtrieval Conference, Gaithersburg, MD, 2000.
- [7] Clarke, C. L. A., G. V. Cormack, T. R. Lynam, C. M. Li, and G. L. McLarn. Web reinforced question answering. In 2001 Text REtrieval Conference, Gaithersburg, MD, 2000.
- [8] Clarke, Charles L. A., G. V. Cormack, and F. J. Burkowski. Fast inverted indexes with on-line update. Technical Report CS-94-40, University of Waterloo Computer Science Department, November 1994.
- [9] Clarke, Charles L. A., Gordon, V. Cormack, and Thomas, R. Lynam. Exploiting redundancy in question answering. In 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 358-365, New Orleans, September 2001.
- [10] Clarke, Charles L. A., Gordon, V. Cormack, and Elizabeth, A. Tudhope. Relevance ranking for one to three term queries. Information Processing and Management, 36(2): 291-311, 2000.
- [11] Cormack, G. V., C. L. A. Clarke, C. R. Palmer, and D. I. E. Kisman. Fast automatic passage ranking. In 8th Text REtrieval Conference, Gaithersburg, MD, November 1999.

- [12] Diekema, Anne, Xiaoyong Liu, Jiangping Chen, Hudong Wang, Nancy McCracken, Ozgur Yilmazel, and Elizabeth D. Liddy. Question answering: CNLP at the TREC-9 question answering track. In 9th Text REtrieval Conference, Gaithersburg, MD, 2000.
- [13] Dumais, Susan, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. Web question answering: Is more always better? In 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, August 2002.
- [14] Harabagiu, Sanda M. and Steven J. Maiorano. Finding answers in large collections of texts: Paragraph indexing + abductive inference. In 1999 AAAI Fall Symposium on Question Answering Systems, pages 63-71, North Falmouth, MA, 1999.
- [15] Hearst, Marti A. and Christian Plaut. Subtopic structuring for full-length document access. In 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 59-68, Pittsburgh, July 1993.
- [16] Hovy, Eduard, Ulf Hermjakob, and Chin-Yew Lin. The use of external knowledge in factoid QA. In 2001 Text REtrieval Conference, Gaithersburg, MD, 2001.
- [17] Ittycheriah, Abraham, Martin Franz, Wei-Jing Zhu, and Adwait Ratnaparkhi. IBM's statistical question answering system. In 9th Text REtrieval Conference, Gaithersburg, MD, 2000.
- [18] Kaszkiel, Marcin and Justin Zobel. Passage retrieval revisited. In 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 178-185, Philadelphia, July 1997.
- [19] Knaus, Daniel, Elke Mittendorf, Peter Schäuble, and Páraic Sheridan. Highlighting relevant passages for users of the interactive SPIDER retrieval system. In 4th Text REtrieval Conference, Gaithersburg, Maryland, 1995.
- [20] Kwok, Cody C. T., Oren Etzioni, and Daniel S Weld. Scaling question answering to the Web. In Tenth International World Wide Web Conference, pages 150-161, Hong Kong, May 2001.
- [21] Kwok, K. L., L. Grunfeld, N. Dinstl, and M. Chan. TREC 2001 question-answer, Web and cross language experiments using PIRCS. In 2001 Text REtrieval Conference, Gaithersburg, Maryland, November 2001.
- [22] Lam, Sunny K. S. and M. Tamer Ozsu. Querying web data — the WebQA approach. In 3rd International Conference on Web Information Systems Engineering, Singapore, December 2002.
- [23] Moffat, Alistair and Justin Zobel. Self-indexing inverted files for fast text retrieval. ACM Transactions on Information Systems, 14(4):349-379, October 1996.
- [24] Monz, Christof and Maarten de Rijke. Tequeda: The University of Amsterdam's Textual Question Answering System. In 2001 Text REtrieval Conference, pages 519-530, Gaithersburg, Maryland, November 2001.
- [25] Najork, Marc and Janet L Wiener. Breadth-first search crawling yields high-quality pages. In 10th International World Wide Web Conference, pages 114-118, Hong Kong, May 2001.
- [26] O'Connor, John. Answer-passage retrieval by text searching. Journal of the American Society for Information Science, 32(4):227-239, July 1980.
- [27] Prager, John, Eric Brown, Amni Coden, and Dragomir Radev. Question-answering by predictive annotation. In 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 184-191, Athens, August 2000.
- [28] Prager, John, Dragomir Radev, Eric Brown, Anni Cohen, and Valerie Samn. The use of predictive annotation for question answering in TREC8. In 8th Text REtrieval Conference, Gaithersburg, MD, 1999.
- [29] Salton, Gerard, J. Allan, and C. Buckley. Approaches to passage retrieval in full text information systems. In 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 49-58, Pittsburgh, July 1993.

- [30] Singhal, Amit, Steve Abney, Michiel Bacchiani, Michael Collins, Donald Hindle, and Fernando Pereira. AT&T at TREC-8. In 8th Text REtrieval Conference, Gaithersburg, MD, 1999.
- [31] Srihari, Rohini and Wei Li. Information extraction supported question answering. In 8th Text REtrieval Conference, Gaithersburg, MD, 1999.
- [32] Voorhees, Ellen M. and Dawn Tice. Building a question answering test collection. In 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 200-207, Athens, August 2000.
- [33] Wilkinson, Ross Effective retrieval of structured documents. In 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 311-317, Dublin, July 1994.
- [34] Wilkinson, Ross and Justin Zobel. Comparison of fragmentation schemes for document retrieval. In 3rd Text REtrieval Conference, Gaithersburg, Maryland, November 1994.
- [35] Witten, Ian H., Alistair Moffat, and Timothy C. Bell. Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann Publishing, San Francisco, second edition, 1999.
- [36] Woods, W. A., Stephen Green, Paul Martin, and Ann Houston. Halfway to question answering. In 9th Text REtrieval Conference, Gaithersburg, MD, 2000.
- [37] Woods, William A., Lawrence A. Bookman, Ann Houston, Robert J. Kuhns, Paul Martin, and Stephen Green. Linguistic knowledge can improve information retrieval. In Applied Natural Language Processing Conference, Seattle, May 2000.
- [38] Zheng, Zhiping Answerbus question answering system. In Human Language Technology Conference, San Diego, March 2002.

DRAGOMIR R. RADEV, HONG QI, ZHIPING ZHENG, SASHA
BLAIR-GOLDENSOHN, ZHU ZHANG, WEIGUO FAN, AND
JOHN PRAGER

QUERY MODULATION FOR WEB-BASED QUESTION ANSWERING

Abstract: The web is now becoming one of the largest information and knowledge repositories. Many large scale search engines (Google, Fast, Northern Light, etc.) have emerged to help users find information. In this paper, we study how we can effectively use these existing search engines to mine the Web and discover the “correct” answers to factual natural language questions. We propose a probabilistic algorithm called *QASM* (Question Answering using Statistical Models) that learns the best query paraphrase of a natural language question. We validate our approach for both local and web search engines using questions from the TREC evaluation.

1. INTRODUCTION

The web is now becoming one of the largest information and knowledge repositories. Many large scale search engines (Google, Fast, Northern Light, etc.) have emerged to help users find information. An analysis of the Excite corpus (Excite, 1999) of 2,477,283 actual user queries shows that around 8.4% of the queries are in the form of natural language questions. The breakdown of these questions is as follows: 43.9% can be counted as factual questions (e.g., “What is the country code for Belgium”) while the rest are either procedural (“How do I ...”) or “other” (e.g., syntactically incorrect). A significant portion of the 91.6% that are not in the form of natural language questions were still generated by the user with a question in mind.

Traditional information retrieval systems (including modern Web-based search engines as mentioned above) operate as follows: a user types in a query and the IR system returns a set of documents ordered by their expected relevance to the user query and, by extension, to the user’s information need. This framework suffers from two problems: first, users are expected to follow a specific engine-dependent syntax to formulate their information need in the form of a query and second, only a small portion of each document may be relevant to the user query. Moreover, a study of search engine capabilities to return relevant documents (Radev *et al.*, 2002) when the query is in the form of a natural language question shows that search engines provide some limited form of processing of the question, namely removing

stop words such as “who” and “where”. Unfortunately, in factual question answering, such words can indicate the type of the answer sought and therefore simple removal may lead to lower accuracy of the search results.

We address these two problems in the context of factual, natural language question answering. In our scenario, when a user types in factual natural language questions such as “When did the Neanderthal man live?” or “Which Frenchman declined the Nobel Prize for Literature for ideological reasons?” he/she will expect to get back the precise answer to these questions rather than a set of documents that simply contain the same keywords as the questions. We make two assumptions here: one, users should not have to learn idiosyncratic search engine query syntax to ask a factual question; two, a document returned by a search engine may contain only a small portion that is relevant to the user question. It is therefore important to allow users to type questions as they see fit and only get the answer to their questions rather than the full document that contains it.

We introduce a probabilistic algorithm for domain-independent natural language factual question answering, *QASM*, which converts a natural language question into a search engine specific query. In our approach, we view question answering and the related problem of natural language document retrieval as instances of the noisy channel problem (Berger and Lafferty, 1999). We assume that there exists a single best query Q that achieves high precision and recall given a particular information need, a particular search engine, and a particular document collection. The query Q is then transformed into a grammatical natural language question N through a noisy channel by the new proposed *QASM* algorithm. Our goal is, given N , to recover the original query Q from the space U_Q of all possible queries that can be generated from N using a limited sequence of linguistically-justified transformation operators.

QASM is based on expectation maximization (EM) and learns which paraphrase Q' from U_Q achieves the highest score on a standardized benchmark. That paraphrase is then assumed to be the closest approximation to the original query Q . The algorithm makes use of a moderate number of labeled question-answer pairs for bootstrapping. Its generalization ability is based on the use of several classes of linguistic (lexico-semantic and collocational) knowledge.

We have incorporated *QASM* in a system which is used to answer domain-independent natural language questions using both a local corpus and the Web as knowledge sources.

1.1 *Question answering*

The problem of factual question answering in the context of the TREC evaluation is best described in (Voorhees and Tice, 2000). The goal is to return the most likely answers to a given question that can be located in a predefined, locally accessible corpus of news documents. To that end, most systems need to perform the following steps: query analysis (to determine, for example, that a “who” question is looking for a person), document retrieval (to establish which documents are likely to contain the answer to the question), document analysis (to determine which portions of the document are relevant), and answer selection (return a short string containing

the most likely correct answer from all retrieved documents). Note also that the questions used in the TREC evaluation are *guaranteed* to have at least one answer in the corpus, while in general no question is guaranteed to an answer on the Web.

The SMU systems, Lasso (Moldovan *et al.*, 2000) and Falcon (Harabagiu *et al.*, 2001), make significant use of knowledge representation techniques such as semantic unification and abduction to retrieve relevant answers. For example, it is able to answer the question “Who was the first Russian astronaut to walk in space” by combining knowledge from the fact that the first *person* to walk in space was Leonov and from the fact that Leonov is Russian.

The IBM project AnSel (Prager *et al.*, 2000; Radev *et al.*, 2000) requires that the corpus be tagged with semantic tokens called *QA-tokens*. For example, “In the Rocky Mountains” is tagged as PLACE\$, while “The US Post Office” is tagged as ORG\$. When a question is processed by the search engine, the QA tokens are passed along with the words in the query in order to enhance retrieval performance. This process requires that the entire corpus be pre-processed with QA-token information.

We claim that it is unrealistic to rely on semantic tagging or deep semantic understanding of both question and source documents when the QA problem is moved from a pre-defined, locally accessible corpus to the Web. It can be argued that a search engine may indeed annotate semantically all pages in its index, however that process is quite expensive in terms of processing time and storage requirements and it is clearly preferable to not have to rely on it. In other words, we are decoupling the problem of natural language document retrieval from the problem of answer selection once the documents have been found. In that case, only the top-ranked retrieved documents need to be annotated.

We need to note here that the service AskJeeves (www.ask.com) is not a QA system. It does process natural language queries but it returns sites that may be relevant to them without trying to identify the precise answers.

In this chapter we will describe an algorithm for domain-independent factual question answering that does not rely on a locally accessible corpus or to large-scale annotation. A system based on this algorithm thus provides the documents that contain the answers and needs to be connected to a component like AnSel to actually extract these answers. Note that AnSel only needs to annotate a small number of documents retrieved by *QASM* and thus the problem of annotating the whole Web disappears.

Table 1: Sample questions

Q: *When did the Neanderthal man live?*

A: About 125,000 to 30,000 years ago

Q: *Which Frenchman declined the Nobel Prize for Literature for ideological reasons?*

A: Jean-Paul Sartre

Q: *What is the second largest planet in the Solar system?*

A: Saturn

Table 1 shows the types of questions that our system can handle. We should note that we make several assumptions about the questions and their answers. First, we assume that each question may contain several answers (this assumption also appears in the TREC setup), all of which are equally acceptable. For example, both “125,000 to 30,000 years ago” and “the Paleolithic age” can be considered to be correct answers to the question “When did the Neanderthal man live?” Second, for purposes of automatic evaluation, we assume that a document does not need to justify the answer it contains (this is a departure from TREC where all submissions are judged by hand as to whether they contain an instance of the correct answer that is justifiably the answer to the given question). As an example, if a document contains the string “the Paleolithic age” we will consider that it contains the answer to the question even if the document’s topic is not about the Neanderthal man.

Let’s now consider two alternative paraphrases of the same question “**When did the Neanderthal man live?**” When we sent that question as it is to Excite, the first four hits didn’t contain the answer. One of them was the home page of the Neanderthal museum in Germany while the second was about the Neanderthal flute. The third hit was about the discovery of the Neanderthal man but has no mention of the time he lived. The fourth one is an advertisement for a book on the Neanderthal man. Only the fifth match did contain a correct answer. On the other hand, when we used **live OR lived AND “Neanderthal man”** as the query, the first two hits did contain accurate answers.

1.2 Statistical Translation Models

We will now describe a technique used in a large variety of textmining tasks – the statistical translation model. We make extensive use of this paradigm in our work.

A translation model is a probabilistic channel model. Given an observed string t in a *target* language T and a *source* language S , a probability $P(t|s)$ is assigned to all transformations from a string $s \in S$ to t . In statistical machine translation (e.g., French to English), the probability $P(t|s)$ should be high for reasonable translations between the two languages, e.g., $P(\text{small boy} | \text{le petit garçon})$ should be high while $P(\text{the dog ate my homework} | \text{le petit garçon})$ should be low.

The goal of a translation model is to find the t that maximizes the probability $P(s|t)$. In practice, the Bayesian rule is applied:

$$\arg \max_s P(s | t) = \arg \max_s P(s) * P(t | s)$$

and the goal becomes to find the value of s that maximizes $P(t|s)$.

The string s can be also thought as the *original* message that got somehow scrambled in the noisy channel and converted into the *target* string t . Then the goal

of translation modeling becomes that of recovering s . Since the space of all possible strings in S is infinite, different heuristic techniques are normally used. We will discuss some of these techniques below.

Statistical translation models originate in speech processing (see (Jelinek, 1997) for an overview), where they are used to estimate the probability of an utterance given its phonetic representation. They have been also successfully used in part of speech tagging (Church, 1998), machine translation (Brown *et al.*, 1990, Berger *et al.*, 1994), information retrieval (Ponte and Croft, 1998), transliteration (Knight and Graehl, 1998) and text summarization (Knight and Marcu, 2000). The reader can refer to (Manning and Schütze, 1999) for a detailed description of statistical translation models in various applications.

In statistical machine translation (SMT), there exist many techniques to convert a string from the source language to the target language. For example, IBM's model three (Brown *et al.*, 1990) makes use of the translation, fertility, and swap operators. *Translation* probabilistically produces a word in the target language that is among all possible translations of a given word in the source language. For example, the probability distribution for the English word “the” may be as follows: $P(\text{la}|\text{the}) = .45$ while $P(\text{le}|\text{the}) = .41$ and the sum of all other probabilities from “the” is equal to .14. *Fertility* produces a variable number of words, e.g., the English word “play” may produce one word in French (“jouer”) or three words (“pièce de théâtre”). *Swap* produces words in different positions from the positions where they appear in the source language. The goal of SMT is then to find out of the infinite number of possible transformations which one is the most likely to have been generated from the source string. SMT makes use of an intermediate (hidden) structure, the so-called *alignment* between the source and target string. Given that there may be many ways of producing the same target string from a given source string, the alignment specifies the exact transformations that were undertaken to produce the target string. For example, “le petit garçon” may have been produced from “the little boy” by translating “le” into “the” and “petit” into “little” or by translating “le” into “little” and “petit” into “the” and then swapping the two words. The alignment is hidden because the system can only see a pair of French and English sentences that are translations of one another and has no information about the particular alignment. Obviously, all possible alignments of two given strings are not equally likely, however it is practically more accurate to think about the translation process in such a way (Brown *et al.*, 1990; Berger *et al.*, 1994; for more details) and then employ parameter estimation techniques to determine the probabilities of applying given operators on particular input strings.

In question answering, the source string is the query Q that produces the best results while the target string is the natural language question N . Obviously, there is an infinite number of queries that can be generated from a natural language question. These paraphrases are typically produced through a series of transformations. For example, the natural language question “Who wrote King Lear” can be converted into the query **(wrote|author) “king learner”** by applying the following operators: (1) bracket “King” and “Lear” together to form an immutable phrase, (2) insert the word “author” as an alternative to “wrote”, and (3) remove the question word “who” (note that before removing that word, the information that it conveys,

namely the fact that the expected answer to the question is a person, is preserved through the use of the “insert” operator which adds a person word “author” to the query. The problem of question answering using a noisy channel model is reduced essentially to the problem of finding the best query Q' which may have been generated from N . We will discuss in the next section what “best” means.

We have identified the following differences between statistical machine translation and question answering (QA).

1. In QA, the swap operator is not particularly needed as typical search engines give the same hits regardless of the order of the query terms.
2. Since swaps are not allowed, alignments are simpler in QA and the process of parameter estimation is simpler.
3. In QA, the source and target language are essentially the same language. The only difference is the addition of logical operators, parentheses, and double quotes in the queries.
4. The generation of queries in QA is much more robust than translation in the sense that the performance of a query typically degrades gracefully when an operator is applied to it while a correct translation can immediately become a horrible one when two words are swapped.
5. Since queries don't need to be grammatical English sentences, there is no need to use a language model (e.g., a bigram model) to determine the correctness of a query. On the other hand, this places an additional burden on the translation model, given that in SMT, the language model prevents a high-probability but ungrammatical translation from being produced.
6. SMT is trained on a parallel corpus with aligned sentences with hidden alignments. QA needs a parallel corpus of questions and answers while the actual queries that can produce documents containing the answers are hidden.

2. THE QASM ALGORITHM

The *QASM* algorithm (Question Answering using Statistical Models) is based on the premise that it is possible to select the best operator to apply on a particular natural language question (or query). That operator will produce a new query which is “better” than the one from which it was generated using some objective function such as precision or recall. We will later define some empirically justified objective functions to compare query paraphrases. In a sense the problem of finding the best operator given a query paraphrase is a classification problem. The class of a paraphrase is the same as the class of all equivalent paraphrases and is the operator that would most improve the value of the objective function when applied to the paraphrase. For example, the class of the paraphrase “who wrote King Lear” may be “delete-wh-word” if the resulting query “wrote King Lear” is the best of all possible queries generated by applying a Single operator to “who wrote King Lear”. Note that we make a distinction between the *composite* operator that converts a question to a query through a *series* of steps and the *atomic* operators that compose each step. For practical reasons, it is more feasible to deal with atomic operators when training the system.

In other words, we need to build a classifier which decides what operator is best applied on a given question N . In practice, we need to decompose the problem into a series of smaller problems and to produce a sequence of paraphrases Q_i with the following properties:

1. The first paraphrase Q_0 is the same as the natural language question N .
2. Each subsequent paraphrase Q_i is generated from Q_{i-1} using a single atomic operator. Note that operators have to be unambiguous given an arbitrary query. For example, “bracket” is ambiguous, because in a general query, many different subsequences can be bracketed together. On the other hand, “bracket the leftmost noun phrase in the query” is unambiguous.
3. If $F(Q_i)$ is the objective function that determines how good a paraphrase is (we will call this function the fitness function, borrowing a term from the evolutionary computation literature), then we want Q_i to be chosen from among all possible paraphrases $Q_{i,j}$ of Q_{i-1} using a single operator O_j , or that $Q_i \equiv Q_{i,k}$ for $k = \arg \max_j F(Q_{i,j})$.
4. The sequence of operators is interrupted when $k = \arg \max_j F(Q_{i,j})$ is the index of the identity operator I . The identity operator has the following property: $I(Q) = Q$. In other words, when no atomic operator can improve F , the process stops. Note that other stopping conditions (e.g., stability of the probability matrix) are also possible.

Two problems need to be resolved at this stage. First, it is obvious that the sequence of operators depends on the initial query $Q_0 \equiv N$. Since there is an infinite number of natural language questions N , it is necessary to perform some sort of smoothing. In other words, each question (and by extension, each query) has to be converted to a representation that preserves some subset of the properties of the question. The probability of applying a given operator on a question will depend on its representation, not on the question itself. This way, we avoid maintaining an operator probability distribution for each natural language question. In the following section, we will discuss a particular solution to the representation problem.

Second, we need to learn from a set of examples the optimal operator to apply given a particular paraphrase. The decomposition of transformation operators into atomic operators such as “insert” or “bracket” significantly reduces the complexity of finding the right operator to apply on a given question.

Since it is very expensive to produce a large training corpus of pairs of questions and their best paraphrases, we have to recur to an algorithm that is stable with regard to missing data. Such an algorithm is the expectation maximization (EM) algorithm.

2.1 The EM algorithm

The EM algorithm (Dempster *et al.*, 1977) is an iterative algorithm for maximum likelihood estimation. It is used when certain values of the training data are missing. In our case, the missing values are the paraphrases that produce the best answers for

given natural language questions. We only have question-answer pairs but no paraphrases. In other words, the known variables are the scores for each operator; the hidden variables are the probabilities for picking each operator.

The EM algorithm uses all the data available to estimate the values of the missing parameters. Then it uses the estimated values to improve its current model. In other words, the EM algorithm works as follows: first, it seeds the parameters of the model with some reasonable values (e.g., according to the uniform distribution). Then it performs the following two steps repeatedly until a local maximum has been reached.

1. *E*-step: use the best available current classifier to classify some data points
2. *M*-step: modify the classifier based on the classes produced by the *E*-step.

The theory behind EM (Dempster *et al.*, 1977) shows that such an algorithm is guaranteed to produce increasingly better models and eventually reach a local maximum.

2.2 Generic operators

We now need operators that satisfy the following criteria: they must be easy to implement, they must be unambiguous, they must be empirically justified, and they must be implemented by a large number of search engines. A list of such generic operators is shown in Figure. We call them generic because they are not written with any particular search engine in mind. In the following section, we will discuss how some of these operators can be operationalized in a real system.

- INSERT - add a word or phrase to the query (similar to the fertility operator in SMT),
- DELETE - remove a word or phrase (“infertility” operator),
- DISJUNCT - add a set of words or phrases in the form of a disjunction,
- REPLACE - replace a word or phrase with another,
- BRACKET - add double quotes around a phrase,
- REQUIRE - insist that a word or a phrase should appear (most search engines require such operator to explicitly include a word in the query),
- IGNORE - for example the query **cleveland -ohio** will return documents about President Cleveland and not about the city of Cleveland, Ohio.
- SWAP - change the order of words or phrases,
- STOP - this is the identity operator,
- REPEAT - add another copy of the same word or phrase

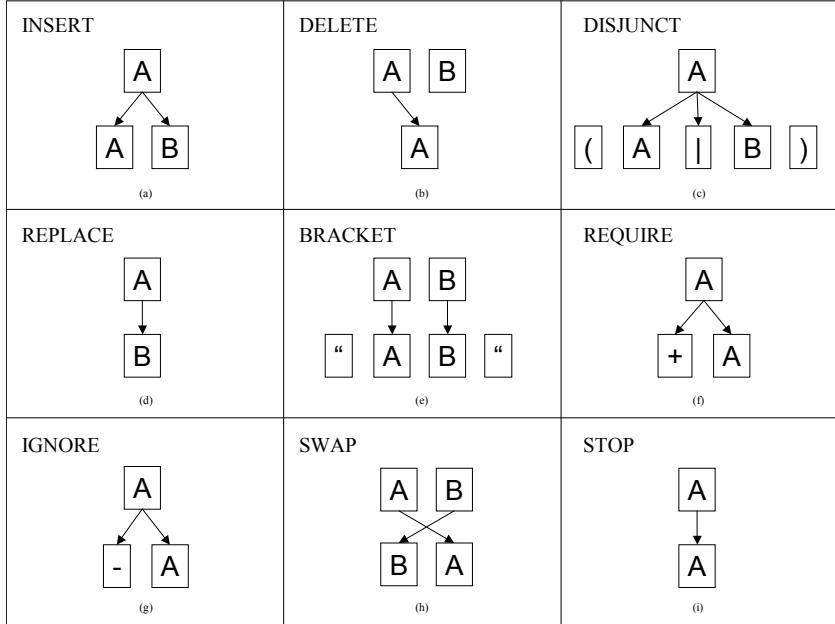


Figure 1: QASM operators

2.3 The probabilistic generation model

We will now turn to the need for question and query representations. The space of questions is infinite and therefore any classification algorithm must use a compact representation of the questions. An empirical analysis (Radev *et al.*, 2002) shows that certain features of questions interact with the scores obtained from the search engines and that questions with the same features tend to be treated similarly. In the next section we will discuss the particular features that we have implemented. At this moment, we only want to specify that all questions with the same feature representation will be treated in the same way by our algorithm. We will call the state of all questions with the same values for all features the *context* C of the query.

The model Θ contains the probabilities $p(O_i|C_j)$ of applying operator O_i given context C_j are represented in a two-dimensional probability matrix. Given a state (context) we can determine the most likely operator. Later in this section we will discuss our algorithm for learning the values of the Θ matrix. Note that Θ has N_s states and N_o operators and the sum of operator probabilities for each row (state) is:

$$\sum_i^{N_o} p(O_i | C_j) = 1 .$$

2.4 Generating query paraphrases

In order to learn the model Θ , we need to be able to produce different paraphrases of a given natural language question N . We will use N as the initial seed query Q_0 : $Q_0 \equiv N$, and then apply *QASM* to produce subsequent paraphrases Q_i from Q_{i-1} .

2.5 Evaluating paraphrase strength

At each iteration, we must determine which operator is the best. We need to define a *fitness* function. In general, such fitness function can be one of a large number of information theoretic measures such as entropy or perplexity. It can also be a metric from information retrieval such as precision (accuracy) or recall (coverage). In the next section, we will introduce *TRDR*, a metric that we found particularly useful.

2.6 The *QASM* training algorithm

We will now introduce the *QASM* algorithm. Some notational clarifications: C_j represents the context (or state) of a question. Θ is the probabilistic model that determines the probability of applying a given operator on question that is represented in a particular state. The initialization step shown in the Figure 2 can be replaced with one that uses an appropriate prior distribution other than the uniform distribution. After each iteration of the EM algorithm, the query is modified using an operator generated probabilistically from the current distribution for the state where the current paraphrase belongs. The global stop criterion is a function of the change in the probability matrix Θ .

```

initialize  $\Theta$  with  $p(O_i|C_j) = 1/N_0$  for all  $i$ 
set  $Q_0 \equiv N$ ; set  $j = 0$ 
repeat
    extract documents that match  $Q_j$ 
    compute paraphrase fitness  $F(O_i|C_j)$  for all  $i$ 
    let  $l = \operatorname{argmax}_k F(O_k|C_j)$ 
    if  $F(STOP|C_j) > F(O_l|C_j)$  then
        next iteration
    else
        recompute context  $C_{j+1}$ 
        rerank operators based on fitness
        readjust/normalize  $\Theta$  based on reranking
        pick an operator  $O_m$  based on  $\Theta$ 
        apply  $O_m$  on  $Q_j$  to produce  $Q_{j+1}$ 
        increment  $j$ 
    until  $\delta(\Theta) < \varepsilon$ 

```

Figure 2: The QASM training algorithm

2.7 Decoding algorithm

Once the probabilistic model Θ has been trained, it can be used to process unseen questions. The decoding algorithm (to borrow another term from SMT) is shown in Figure 3.

```

 $\Theta$  is given from the training stage
set  $Q_0 \equiv N$ ; set  $j = 0$ 
repeat
    pick  $l$  based on  $\Theta$ 
    apply  $O_l$  on  $Q_j$  to produce  $Q_{j+1}$ 
    increment  $j$ 
until  $F(STOP|C_j) > F(O_l|C_j)$ 

```

Figure 3: The QASM decoding algorithm

3. IMPLEMENTATION AND EXPERIMENTS

We now describe the operationalization of the *QASM* algorithm in our system. We will address the following issues: choice of operators, training and test corpora of questions and answers, choice of document corpora and search engine, question preprocessing, etc.

3.1 *Specific operators*

We note here that the operators that we discussed in the previous section are tricky to implement in practice because they are ambiguous. For example, to “insert” a word in a query, we have to know *what* word to insert and to insert it. Another observation is that there exist many “bases” of atomic operators that can be used to generate arbitrary (but reasonable, that is likely to produce superior queries at least occasionally) paraphrases. We should note here, that traditionally, the idea of adding words to a query is called “query expansion” and is done either manually or automatically (in which case it is called “pseudo relevance feedback” (Mitra *et al.*, 1998).

We have chosen a “basis” composed of 15 operators, grouped into four categories: DELETE, REPLACE, DISJUNCT, and OTHER. We use five DELETE operators (delete all prepositions, delete all wh-words, delete all articles, delete all auxiliaries, and delete all *other* stop words based on a list of 163 stop words). We also use four REPLACE operators (replace the first noun phrase with another, replace the second noun phrase, replace the third noun phrase, and replace the first verb phrase in the question). Note that by specifying which exact terms we will replace, we are addressing the “where” question. The four DISJUNCT operators are similar to the REPLACE operators, except that the new words and phrases are disjoined using OR statements. Finally, the IDENTITY operator doesn’t modify the query.

When deciding what word to insert with the REPLACE and DISJUNCT operators, we use two alternative approaches. In the first case, we look at the first three synonyms and the closest hypernym of the first sense of the word (in the correct part of speech: verb or noun) based on the WordNet database (Miller *et al.* 1990). The second case uses distributional clusters of fixed size of the words (Pereira *et al.*, 1993). We don’t have room here to describe the implementation of the distributional algorithm but here is the basic idea: if two words appear in the same context (looking at words in a window) in a large corpus, they are considered to be distributionally similar. As an example, “professor” and “student” are not distributionally similar because they tend to appear together, while “book” and “essay” are since they are in complementary distribution in text. In their right hand columns, Table 2 and Table 3 show up to five words or phrases similar to the word in the left column.

Table 2: Related words produced by WordNet

Word	Similar words
book	publication, product, fact, dramatic composition, record
computer	machine, expert, calculator, reckoner, figurer
fruit	reproductive structure, consequence, product, bear
politician	leader, schemer
newspaper	press, publisher, product, paper, newsprint

Table 3: Related words produced by distributional clustering

Word	Similar words
book	autobiography, essay, biography, memoirs, novels
computer	adobe, computing, computers, developed, hardware
fruit	leafy, canned, fruits, flowers, grapes
politician	activist, campaigner, politicians, intellectuals, journalist
newspaper	daily, globe, newspapers, newsday, paper

Note that we did at this point, our implementation doesn't include the following operators: IGNORE, REPEAT, and SWAP. We implemented BRACKET in the preprocessing stage.

3.2 Fitness functions

Instead of precision and recall, we use total reciprocal document rank (*TRDR*). For each paraphrase, the value of *TRDR* is the sum of the reciprocal values of the rank of all correct documents among the top 40 extracted by the system:

$$TRDR = \frac{1}{n} \sum_{i=1}^n \frac{1}{r_i}, \text{ where } r_i \text{ is the rank of a correct answer.}$$

For example, if the system has retrieved 10 documents, of which three: the second, eighth, and tenth, contain the correct answer, *TRDR* for that given paraphrase is $1/2 + 1/8 + 1/10 = .725$.

The metric used in TREC is slightly different from ours. In TREC's case, only the rank of the best answer counts so if the correct answer is retrieved in first place only, the score will be higher than when the correct answer appears in all places from second to fortieth. Some observations motivated this switch in formulas: (1) Since we are looking at finer distinctions between related paraphrases of the same question, we need to pay more attention to subtle differences in performance, and (2) In a Web environment, some URLs on the hit list may no longer be around when the system tries to retrieve them, so we want to reward systems, which other

things being equal, return more documents with answers and (3) the need to accept different variants of the same answer, as illustrated in the following example.

TREC 9, Q151 Where did Dylan Thomas die? (expected answer: New York)

http://news.bbc.co.uk/hi/english/entertainment/newsid_139000/139337.stm

In 1953 Thomas was on his fourth lecture tour of America and drinking heavily. That *he died in a New York hospital* from alcohol poisoning is beyond dispute.

<http://library.thinkquest.org/3187/thomas.html>

The Welsh poet Dylan Thomas was born in Swansea, Wales, on Oct. 27, 1914... *He died in New York City* on Nov. 9, 1953, and was buried at Laugharne.

<http://www.poetrymagazine.com/archives/1999/august/thomas.htm>

Dylan Thomas was born in Swansea, Wales in 1914... Alcoholism helped usher him to *his death* a few days after his 39th birthday *in New York City* during a lecture tour of the United States in 1953

Figure 4: Sample paraphrases of Q151

Figure 4 gives a feeling for the diversity of the wordings of answers to the same question on the Web and in some way provides an additional justification for our choice to include all correct hits in TRDR. In this example, three competing documents specify the place where Dylan Thomas died.

We should note that the highest score that a given paraphrase can obtain on a given question is 4.279 (the sum of $1/n$ for n from 1 to 40). Qualitatively, a paraphrase is acceptable if it obtains a score of at least .300 which corresponds roughly to one hit among the top five or two hits in the next five.

3.3 Training data and choice of corpora and search engines

To train our system, we used pairs of $\{N, A\}$ collected from a variety of sources: the TREC8 and TREC9 collections (893 data points) plus our own collection of 2,224 data points.

We used zzsearch (a search engine developed by our group) on the local corpus of 240,000 AP news articles from the TREC collection. On the Web, we intended to use Google as the primary search engine, however at some point it stopped processing queries longer than 10 words so we had to switch to the Fast search engine (<http://www.alltheweb.com>). We built a module that converts the same underlying query to the syntax of these two search engines.

3.4 Preprocessing

Before we run any queries on a search engine, we preprocess them in order to identify immutable phrases that would be preserved in the paraphrasing process. We

use the Itchunk utility (Mikheev, 2000) to determine noun phrases and we filter out these that contain an initial determiner followed by a single word such as “the man”.

3.5 Feature representations of queries

We decided to adopt a representation scheme using three features: semantic type of question based on *wh*-word (e.g., PERSON, LOCATION, etc.), number of words in query, number of proper nouns in query.

As an example, **who wrote King Lear** can be represented as (“PERSON”,4,1). The generalizations based on the representation can capture such patterns as “the probability of applying a DELETE operator on a shorter query is smaller than on a longer query”.

3.6 Sample run

We will now run through an example of the system in action. We will show how the *QASM* algorithm readjusts the values of the Θ model using a particular question from the TREC 8 corpus. The fitness scores are computed using the Fast search engine.

In this example, the question N is **which country is the biggest producer of tungsten** (that is question 14 from TREC8). The expected answer is “China”. The initial paraphrase $Q_0 \equiv N$ is relegated to state (“LOCATION”,8,0) since it is a “where” question, it is 8 words in length, and contains no proper nouns. The probability values for the row (“LOCATION”,8,0) are all set to be $1/15 = .667$ under the default assumption that all 15 operators are equally likely.

Table 4 shows the *TRDR* values for the first 8 paraphrases of N . The 8 operators shown are the IDENTITY operator, all five DELETE operators, and the first two REPLACE operators. In this run, we use the variants of the operators based on WordNet and not on distributional clustering.

Table 4: Example of QASM in operation

Operator	Current paraphrase	TRDR
0 IDENTITY	What country is the <i>biggest producer</i> of tungsten	0.837
1 DEL_WH	country is the <i>biggest producer</i> of tungsten	1.755
2 DEL_AUX	What country the <i>biggest producer</i> of tungsten	1.322
3 DEL_ART	What country is <i>biggest producer</i> of tungsten	1.436
4 DEL_PREP	What country is the <i>biggest producer</i> tungsten	1.436
5 DEL_STOP	What country is the <i>biggest producer</i> of tungsten	0.837
6 REPL_1N	What (<i>administrative district</i> OR <i>political unit</i> OR people OR region OR <i>geographical area</i> OR <i>rural area</i>) is the <i>biggest producer</i> of tungsten	1.181
7 REPL_2N	What country is the <i>biggest producer</i> of (<i>metallic element</i> OR wolfram OR w OR atomic number 74)	1.419

Once all fitness scores are computed, the probabilities are readjusted proportionally to the scores. As a result, the probability for operators 1 and 3 increases at the expense of operators 0, 5, and 6. The resulting probabilities for the first 8 operators are as follows (only the first eight are shown for lack of space): 0.0536, 0.1123, 0.0846, 0.0919, 0.0919, 0.0536, 0.0756, and 0.0908.

The next paraphrase is generated according to the new probability distribution. Obviously, at this stage, all operators still have a chance of being selected as no fitness score was zero. In this example, operator 0 (IDENTITY) was selected.

The new state is the same as the previous one: (“LOCATION”, 8,0). The probability distribution has now changed. Since the values of *TRDR* in the second iteration will be the same as in the first, the probabilities will be readjusted once more in the same proportion as after the first iteration. After the second EM iteration, the probabilities for state (“LOCATION”, 8,0) are as follows: 0.0374, 0.1643, 0.0932, 0.1100, 0.1100, 0.0374, 0.0745, and 0.1074. After five iterations in the same state they become 0.0085, 0.3421, 0.0830, 0.1255, 0.1255, 0.0085, 0.0473, and 0.1184. After 23 iterations in the same state, the probability of applying operator 2 is 0.9673 while all 14 other probabilities tend to 0.

If we allow *QASM* to pick each subsequent operator according to Θ we observe that the sequence that achieves the highest score for the state (“LOCATION”, 8,0) is to apply operator 2 followed by operator 4. Note that the state of the paraphrase changes after operator 2 is applied as one word has been removed.

After all probabilities are learned from the training corpus of 2,224 data points, *QASM* can proceed to unseen questions. For lack of space, we omit a complete illustration of the decoding process. We will just take a look at all questions that fall in the same class as the one in the example. There are 18 such questions in our test corpus and for 14 of them (77.8%), as predicted, the sequence of operators 2 and 4 is the one that achieves the highest score. In two other cases, this sequence is second best and in the last two cases it is still within 20% of the performance of the best sequence. For the 18 questions the performance over the baseline (sending the natural language question directly to the search engine) goes from 1.31 to 1.86 (an increase of 42.0%).

4. RELATED WORK

There has been a lot of effort in applying the notion of language modelling and its variations to other problems. For example, Ponte and Croft (1998) adopt a language modeling approach to information retrieval. They argue that much of the difficulty for IR lies in the lack of an adequate indexing model. Instead of making prior parametric assumptions about the similarity of documents, they propose a non-parametric approach to retrieval based probabilistic language modeling. Empirically, their approach significantly outperforms traditional tf*idf weighting on two different collections and query sets.

Berger and Lafferty (1999) suggest a similar probabilistic approach to information retrieval based on the ideas and methods of statistical machine translation. The central ingredient in their approach is a noisy-channel model of how

a user might “translate” a given document into a query. To assess the relevance of a document to a user’s query, they estimate the probability the query would have been generated as a translation of the document, and factor in the user’s general preferences in the form of a prior distribution over documents. They propose a simple, well-motivated model of the document-to-query translation process, and describe the EM algorithm for learning the parameters of this model in an unsupervised manner from a collection of documents.

Brown *et al.* (1990) lay out the mathematical foundation of statistical machine translation, while Berger *et al.* (1994) present an overview of Candide, a system that uses probabilistic methods to automatically translate French text into English.

Text summarization is another field where the language modeling approach seems to work for at least some problems. While most research in this area focuses on sentence extraction, one could argue that when humans produce summaries of documents, they do not simply extract sentences and concatenate them. Rather, they create new sentences that are grammatical, that cohere with one another, and that capture the most salient pieces of information in the original document. Given the large collections of text/abstract pairs now available online, it is possible to envision algorithms that are trained to mimic this process. Knight and Marcu (2000) take the first step in this direction by addressing sentence compression. They devise both noisy-channel and decision-tree approaches to the problem results. Similarly, Banko *et al.* (2000) employ a probabilistic model to generate headlines of news articles.

Glover *et al.* (2001) address the issue of query modification when searching for specific types of Web pages such as personal pages, conference calls for papers, and product announcements. They employ support vector machines to learn engine-specific words and phrases that can be added to a query to locate particular types of pages. Some of the words are quite unobvious, for example adding “w” as a word to a query improves the performance of their system, Inquirus2, when it is set to search for personal pages.

Dempster *et al.* (1977) are the first ones to formalize the EM algorithm as an estimation technique for problems with incomplete data. Neal and Hinton (1993) introduce incremental implementations of the EM algorithm which allow the values of individual unobserved variables to be optimized at each step. McCallum *et al.* (1998) and Nigam *et al.* (2000) describe an EM-based algorithm for document classification. Their approach achieves the same classification accuracy as fully supervised methods but with 30% fewer labeled training documents.

Active learning (Cohn *et al.*, 1996) is a departure from traditional machine learning techniques in that the learner is allowed to request (at a given cost) the correct classification of the most informative unlabeled data points. In *QASM*, the cost paid for the classification is in terms of the time to query the search engine, download the documents locally, and run the evaluation routine on them.

5. CONCLUSION

In this paper we described an algorithm for learning a sequence of transformations that need to be applied on a natural language question so that it can be

run on a search engine and retrieve documents containing the answer to the question. Our algorithm, *QASM*, is based on the EM estimation algorithm and employs linguistically justified transformational operators. *QASM* has been implemented in a natural language question answering system with two different backends – one based on a local corpus and another – using an external search engine which retrieves answers from the World-Wide Web. Our two main contributions are (a) the use of the Web as a knowledge source for domain-independent question answering and (b) the *QASM* algorithm which produces the paraphrase of a natural language question that is most likely to produce a list of hits containing the answer(s) to the question.

5.1 Future work

We plan to link *QASM* with the AnSel system to complete the pipeline: question → documents → set of potential answers. We also intend to study user-specific question answering. For example, the answer to the question “where is the Dynasty restaurant” may be different based on the user’s location, experience, and preferences. We plan also to investigate the problem of cross-lingual question answering where the question is in one language and the answer – in another. Finally, we are pursuing a separate study that will indicate whether the choice of search engine can be parameterized in the probabilistic model. If we are successful in this endeavour, we will be able to build a meta-search engine which will both choose the best paraphrase of a question and the search engine where to send it.

6. ACKNOWLEDGEMENTS

This work was partially supported by the National Science Foundation (NSF)’s Information Technology Research program (ITR) under grant IIS-0082884 and also under NSF’s grant IDM-0329043. All opinions, findings, conclusions and recommendations in any material resulting from this workshop are those of the participants, and do not necessarily reflect the views of the National Science Foundation.

7. REFERENCES

- Banko, M., V. Mittal, and M. Witbrock. Headline Generation Based on Statistical Translation, ACL 2000.
- Berger, A. and J. Lafferty. Information retrieval as statistical translation. In Proceedings, 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, California, August 1999.
- Berger, A., P. Brown, S. Pietra, V. Pietra, J. Lafferty, H. Printz, and L. Ures. The Candide system for machine translation. In Proceedings of the ARPA Conference on Human Language Technology, 1994.
- Brown, P.F., J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. A statistical approach to machine translation. Computational Linguistics, 16(2):79–85, 1990.
- Church, K. A stochastic parts program and a noun phrase parser for unrestricted text. In Proceedings of the Second Conference on Applied Natural Language Processing, Austin, Texas, 1988.

- Cohn, D. and Z. Ghahramani and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research* 4, 1996, pages 129–145.
- Dempster, A.P., N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.
- Excite query corpus. ftp://ftp.excite.com/pub/jack/Excite_Log_12201999.gz, 1999.
- Glover, E., G. Flake, S. Lawrence, W. Birmingham, and A. Kruger. Improving category specific web search by learning query modifications. In *Symposium on Applications and the Internet*, Jan 8–12 2001.
- Glover, E.J., S. Lawrence, M. D. Gordon, W. P. Birmingham, and C. L. Giles. Web search – your way. *Communications of the ACM*, 2001.
- Harabagiu, S., D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, and P. Morarescu. The TREC-9 question answering track evaluation. In *Text Retrieval Conference TREC-9*, Gaithersburg, MD, 2001.
- Jelinek, F. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, Massachusetts, 1997.
- Knight, K. and D. Marcu. Statistics-based summarization – step one: sentence compression. In *Proceedings of Seventeenth Annual Conference of the American Association for Artificial Intelligence*, Austin, Texas, August 2000.
- Knight, K. and J. Graehl. Machine transliteration. *Computational Linguistics*, 24(4), 1998.
- Manning, C. and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.
- McCallum, A. and K. Nigam. Employing EM and pool-based active learning for text classification. *Proceedings on ICML*. Pages 359–367, 1998.
- Mikheev, A. Tagging sentence boundaries. In *Proceedings of SIGIR 2000*.
- Miller, G.A., R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography* (special issue), 3(4):235–312, 1990.
- Mitra, M. and A. Singhal and C. Buckley. Improving Automatic Query Expansion. *SIGIR* 1998.
- Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea, R. Gîrju, R. Goodrum, and V. Rus. The structure and performance of an open-domain question answering system. *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, Hong Kong, October 2000.
- Neal, R. and G. Hinton. A new view of the EM algorithm that justifies incremental and other variant. Technical Report. University of Toronto, 1993.
- Nigam, K., A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning* 39 (2–3), pages 103–134, 2000.
- Pereira, F., N. Tishby, and L. Lee. Distributional clustering of English words. In *30th Annual Meeting of the ACL*, 183–190, 1993.
- Ponte, J. and B. Croft. A language modeling approach to information retrieval. In *Proceedings, 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 275–281, Melbourne, Australia, August 1998.
- Prager, J., E. Brown, A. Coden, and Dragomir R. Radev. Question-answering by predictive annotation. In *Proceedings of 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July 2000.
- Radev, Dragomir R., J. Prager, and V. Samn. Ranking potential answers to natural language questions. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA, May 2000.
- Radev, Dragomir R., Kelsey Libner, and Weigu Fan. Getting Answers to Natural Language Queries on the Web. *Journal of the American Society for Information Science and Technology*, 2002.
- Voorhees, E. and D. Tice. The TREC-8 question answering track evaluation. In *Text Retrieval Conference TREC-8*, Gaithersburg, MD, 2000.

4

ANSWER EXTRACTION

JOHN M. PRAGER, JENNIFER CHU-CARROLL,
ERIC W. BROWN AND KRZYSZTOF CZUBA

QUESTION ANSWERING BY PREDICTIVE ANNOTATION

Abstract: We present in this chapter a description of the major components of a Question-Answering system which has fared well in the three TREC QA evaluations so far, and is currently participating in the ARDA AQUAINT program. Our approach centres around the technique of Predictive Annotation, in which an extended set of named entities is recognized prior to indexing, so that the semantic class labels can be indexed along with text and included in the query string. In addition we present other techniques that are employed for specific question types, such as Virtual Annotation for definition questions. We describe the Answer Selection component, which extracts and ranks answer candidates from the passages returned by the search engine based on linguistic as well as statistical features. We present numerous examples as well as quantitative evaluations.

1. INTRODUCTION

The field of Question Answering (QA) has grown out of dissatisfaction with Information Retrieval (IR) being merely document retrieval (DR). IR has existed ever since documents have been created on-line in any numbers, and has primarily dealt with the location of documents by means of keyword queries. The software components that find the desired documents are search engines, and operate by means of a ranking formula which is used to evaluate each document in the collection. These formulas generally give preference to documents that contain more of the query terms, and at the same time give a greater weight to rarer query terms (Salton, 1989).

The result is that IR systems find documents that are in some sense “about” the terms in the query, but have no way to address the specific information need that motivated the query. In particular, if the query is a natural language question, a typical IR system shoots itself in the foot by eliminating as stop words the very question words (*who*, *where*, *when* etc.) that tell it what the user is looking for. As a consequence, while QA systems currently all employ search engines, considerably more substantial processing is employed both before and after search itself than in traditional IR systems.

Despite early informative work in the field such as (Lehnert, 1978), end-to-end systems did not generally appear until the 1990’s. The basic QA system architecture, present in systems such as MURAX (Kupiec, 1993) and still in use today, is a

pipeline. This pipeline has three principal stages (see Figure 1), which will now be described.

The first stage is Question Analysis. In this stage, two important things happen. First, the question is typically examined to determine the *question focus*, or desired answer type. The question focus is the quantity that the question is looking for and can be used to determine the textual form of an answer. For example, a question beginning with *Name a painter...* indicates that the name of a person is sought, while one starting with *Where...* suggests that a place is requested. The question focus can also be seen as the type of expression that would satisfy the question, and the subsequent system modules (Search, Answer Selection) make use of this information. The second major activity in the question analysis component is the transformation of the question into a search engine query. This process often involves elimination of some question words, morphological expansion and/or stemming of others, and addition of synonyms. Furthermore, term weighting may be specified at this stage to ensure that keywords identified as being more important are preferred over others during the search process.

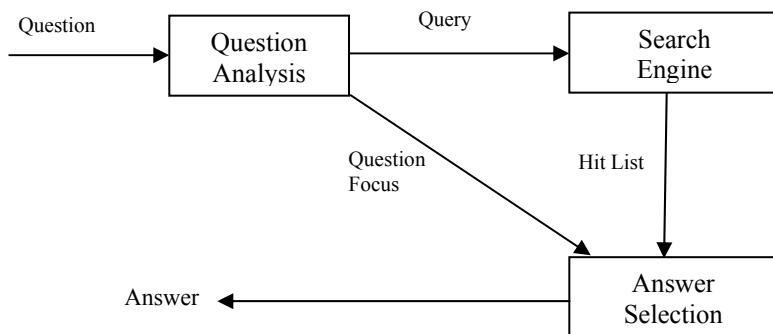


Figure 1: Basic QA Pipeline

The second stage is Search. A search engine is invoked with the query generated in the previous stage, and returns a hit list; the hit list is usually a list of documents, but some systems, including ours, return short passages.

The third stage is Answer Selection. Here, the documents or passages in the hit list are examined for instances of potential answers. Various features are used to rank-order the candidates found, ranging from a weighted average of select features such as search engine score, query term density, and semantic type of candidate answer, to dependency-based analysis or logical inference (Harabagiu *et al.*, 2001).

An enhancement that is getting increasing attention is that of adding feedback loops in the pipeline. Based on the results of decision-making steps at different points in the pipeline, the system can decide that the data that has made it through

thus far is deficient in some fashion and that an earlier stage needs to be re-run with different settings; cf. (Pasca and Harabagiu, 2001). Even in the absence of feedback loops, such decision-making is necessary at some point in the process if the system is to be able to provide “No answer can be found” as one of its answers, since telling a user it doesn’t know is almost always better than giving the user a list of wrong answers.

The system that we describe in this paper follows the general pipeline design showed in Figure 1. The major distinguishing feature of our approach is the use of a technique called Predictive Annotation, which is an effective methodology that we have developed to achieve high precision performance. Predictive Annotation allows us to index answers to future questions by leveraging existing Named Entity recognition and annotation technology and its extensions that we developed for the system. It is based on an off-line document annotation step that creates textual annotations of the semantic types in the documents; these so-called QA-Tokens are indexed by the search engine along with the original document terms. QA-Tokens are also associated with the questions that follow, and they serve a similar purpose as answer types or question foci in other systems. The main function of the QA-Token is to focus search throughout the system on the answer candidates of the right type. Thanks to our use of Predictive Annotation and QA-Tokens, this can be accomplished very early on in the system since we can supply the information on the required answer type to the search engine. This is different from the typical QA system that would apply the answer type information at a much later stage, possibly retrieving and processing passages that do not even contain an expression of the right type. As we discuss in Section 4.4, indexing the QA-Tokens means that very much shorter search engine hit-lists can be used, thus not only eliminating sources of noise but in some cases allowing answers to be found that would otherwise be buried unreasonably deeply in the hit-list.

The development within this chapter generally follows the pipeline we just described. We first describe the technique of Predictive Annotation and how it influences processing in different parts of the system. We next describe Question Analysis. This stage of processing determines the question focus and modifies the question into a query suitable for processing by the search engine. For certain kinds of questions, specialized processing using ontological resources is used. The next section describes Search. We describe how our search engine differs from those engines typically used in document retrieval, and what the syntax and semantics of its query language are. We describe experiments we performed to determine optimal passage length and hit list size. In the following section we describe Answer Selection. We describe the features that are calculated for all of the entities (candidate answers) in the passages returned by the search engine, and an algorithm for ranking and selecting these answers. In the next section we describe specialized processing techniques, specifically Virtual Annotation, which we use for certain identified question types. In the final section we outline our on-going work as part of the ARDA-funded AQUAINT effort. Our new contributions include transforming the topology of the pipeline into a plan-based loop, the use of multiple parallel solution paths utilizing both structured and unstructured information sources, which we expect to manage via heavy use of confidences.

2. PREDICTIVE ANNOTATION

2.1 Why Predictive Annotation?

A key requirement of Question Answering is to determine from the question the entity being sought, and to locate that entity in the text corpus. If a search engine is used to return documents or passages, then the entity must be present in the hit list in order to ultimately be selected. Suppose the question is *How many Grand Slam Titles did Bjorn Borg win?* Many documents or passages returned by the search engine will undoubtedly mention the Grand Slam or Bjorn Borg, and hopefully both, but the QA system must locate a number as the answer to the question. Granted, it must find a number that counts the titles that Borg won, but the first step is to find any candidate numbers. The problem with constructing a query only from the question terms {*Grand, Slam, Bjorn, Borg, win*} is that there is no way to ensure that the returned passages will contain a number.

The approach taken by many systems is to retrieve a sufficiently large number of documents or passages from the search engine that at least, by chance, *some* instances of the entity being looked for will be present. The difficulty here is that one must balance the trade-off between precision and recall – while retrieving more passages increases the likelihood of encountering the answer (better recall), there is the unavoidable companion problem of increasing noise, and hence lowering the system's precision (see Section 4.4 for a description of an analysis of the effects of hit list size on system performance). We have adopted an approach that we call Predictive Annotation, which ameliorates this problem by significantly increasing the likelihood that a returned passage contains at least one entity of the type the question is seeking.

The core of Predictive Annotation (PA) consists of two processes. The first occurs when indexing the corpus for the search engine. In addition to indexing the textual contents of the documents, as do all typical search engines, we also index the semantic classes of the entities in the text. For instance, *Bjorn Borg* is indexed both as the lexical entities *Bjorn* and *Borg*, as well as **PERSON\$**, which is our semantic class for proper names that refer to people. The second part of the Predictive Annotation process happens during Question Analysis. At this stage, one or more semantic class is inserted into the search engine query based on the system's determination of the question focus. This enables the search engine to prefer passages that contain candidate answers of the correct semantic type. In our example, by combining the indexing of all numerical values with the semantic class **NUMBER\$**, and including **NUMBER\$** in our search engine query (in addition to the keywords identified), we are able to retrieve passages about Bjorn Borg and the Grand Slam, along with numerical values that could potentially serve as answers to the question. In other words, our Predictive Annotation mechanism allows us to achieve significantly better recall without sacrificing system precision.

Some questions illustrate dramatically the benefits of using Predictive Annotation. Consider *What is the zip-code for the White House?*. There are in the

TREC corpus about 25,000 documents mentioning the White House. There are 15 that mention the White House and give some zip-code, 4 of which are the actual White House zip-code. However, there are none that mention both *White House* and the term *zip-code* (or *zip* or *zipcode*). Hence a system not using PA must use just *White House* as the query, and must search on average over 6,000 documents before encountering the correct answer. With the query

```
@phr(0 White House) ZIPCODE$,
```

(this uses our phrase operator, defined in Section 4.3) only 15 documents are returned; due to the density contribution in the search engine score (see Section 4.3.3), 4 of the top 5 documents returned all have the correct value.

In the rest of this section, we focus on how semantic classes in our system are selected and how instances of semantic types are recognized in a text corpus. We leave the instantiation of semantic classes in questions to Section 3.1, where we discuss our question analysis process.

2.2 Semantic Class Hierarchy

Almost all existing question answering systems utilize a semantic classification for identifying the question focus. However, not only is there no established “best” such classification, but the number of classes used by various systems varies widely, from approximately 10 to 100; see e.g., (Ittycheriah *et al.* 2001) and (Hovy *et al.* 2001). Our system utilizes a set of about 80 semantic classes, arranged in a (somewhat flat) hierarchy, as shown in part in Table 1. We call these semantic classes *QA-Tokens*, and they are represented by appending the “\$” sign to the name of the semantic class (such as **PERSON\$** and **NUMBER\$**) so as to avoid unintentional clashes with any instances of corresponding English words in text.

Most of our QA-Tokens, such as all of those in Table 1, represent naturally-occurring categories. We also use a small number of *Virtual QA-Tokens*, which as described in Section 6 are only instantiated at run-time, and also a few QA-Tokens, usually used in multiple annotations, representing uncertainty by the Named-Entity Recognizer. As an example of the latter set, **UNL\$**, seen in an example in section 5.3.1, is used for proper names that might be (in other words, can’t be ruled out as) a place.

QA-Tokens are designed to represent classes of objects that people ask questions about. There are obvious choices, based on observation of semantic types of commonly asked questions, such as **PERSON\$**, **PLACES\$**, **DATES\$**, **LENGTH\$** and **NUMBER\$**. But why did we choose to use **PRESIDENT\$**, and why not **ACTORS\$**?

Table 1: Semantic Classes. The 15 most frequent and 5 least frequent QA-Tokens in the annotated TREC corpus. The middle column is total occurrences, the right-hand column is numbers of documents with at least one occurrence. The somewhat surprising prominence of SYMPTOM\$ and BODYPART\$ is due to polysemy of some common terms in those classes.

QA-Token	Occurrences	Documents
THING\$	20,566,320	932,710
ROLE\$	15,132,300	928,864
PERSON\$	7,399,588	778,620
PLACE\$	6,703,793	849,207
ORG\$	5,651,347	855,317
SYMPTOM\$	5,198,001	791,335
BODYPART\$	4,462,547	775,740
DATE\$	3,863,465	839,576
WHOLENO\$	3,302,426	614,568
SMALLNO\$	2,707,944	565,118
COUNTRY\$	2,619,016	482,020
MONEY\$	1,916,151	403,878
CITY\$	1,475,452	494,332
RATE\$	1,417,043	334,543
QUOTE\$	1,331,594	464,838
...
MOUNTAIN\$	5,000	3,045
NICKNAME\$	4,416	3,484
POWER\$	4,129	2,142
MOON\$	3,384	1,573
STAR\$	2,051	1,118

QA-Token definition has essentially been driven by two things: coverage of obvious categories, and the task at hand (e.g., TREC). QA-Token definition is an on-going, iterative process and can be modified according to a particular domain or application.

Furthermore, we have discovered no strict guidelines, but rather that it is a somewhat subjective matter. The technical criteria are those of precision and recall, but these cannot be measured very easily, and are best estimated. Evaluation of precision and recall on semantic type identification over a large corpus is very labour intensive and thus infeasible to carry out each time a new QA-Token is added.

For a QA-Token to be a useful representative of a class of entities, (1) questions about that class need to be recognized, (2) instances of the class in text need to be recognized, and (3) questions about that class should occur with reasonable frequency for it to be useful.

Factors (1) and (2) above explain the dichotomy between **PRESIDENT\$** and **ACTOR\$**. Each is equally easy to recognize as focus of a question (as in *Which president...?* and *What actor...?*), but not so as instances in text. There have been only 43 American presidents, so a list of them, complete with variants, nicknames and abbreviations is easy to generate. A definitive list of actors is much harder to formulate, and would get out of date much more quickly. As for the estimated frequency in questions, one could create a QA-Token **POSTMASTER-GENERAL\$**, since that list is equally bounded, and it would probably have the same performance as **PRESIDENT\$**, but its limited utility would argue against putting in the effort needed to implement it.

2.3 *QA-Token Recognition in Text*

In Information Extraction and Question Answering, it is common practice to employ a named entity recognizer to identify entities such as people, places, dates, numbers, etc. We use and extend for QA our rule-based named entity recognizer known as Textract (Byrd and Ravin 1999). However, in addition to simply recognizing these named entities as candidate answers in relevant passages retrieved by the search engine, we take the extra step to reify the recognized QA-Tokens as text strings themselves during indexing time. This allows us to include appropriate QA-Tokens at search time to increase the system's precision for retrieving relevant passages.

As Textract processes the document collection, it finds part-of-speech and lemma forms for each common word, canonical forms for proper names, and most importantly for us, creates annotations based on the recognized QA-Tokens for the terms in the text. The recognition of QA-Tokens in text is based on a combination of three techniques: finite state machines, rules and vocabulary lists. Figure 2 shows a sample text annotated with QA-Tokens. Since both the lexical items as well as its annotations are indexed, the system can find relevant passages when any one of the annotated strings is the target answer, by constructing queries for the search engine that contain the relevant keywords and the correct QA-Token. The tags in Figure 3 are to be understood as delimiting the textual span of QA-Tokens. For example, the word *institute* is surrounded with *<ORG></ORG>* and the search engine indexer will add to the index a posting for the term *institute* as well as the QA-Token **ORG\$** pointing to the same location in the document collection (see Section 4.2 for details).

THE <ORG>INSTITUTE</ORG> REVISED THE
<PLACE><MOUNTAIN>MATTERHORN</MOUNTAIN></PLACE>'S
HEIGHT TO <LENGTH>14,776 FEET 9 INCHES</LENGTH>.

Figure 2: Sample annotated text

Note that in Figure 2 one lexical item gets multiply annotated. Multiple annotations can occur for two reasons. First, it can happen because of subsumption, where the multiple QA-Tokens subsume one another, as happens here with Matterhorn. Another example would be the annotation of *John F. Kennedy* as both a **PRESIDENT\$** and a **PERSON\$**, since all presidents are people. Second, multiple annotation may occur because of ambiguity, where the alternative semantic meanings of a lexical item get annotated by different QA-Tokens. For instance, *Mercury* can be annotated as a **DEITY\$**, a **PLANETS** or an **ELEMENT\$**, based on its three common word senses.

To address the subsumption relationships between QA-Tokens, we can enable the search engine to find relevant passages in one of three ways.

1. We can do the bulk of the work at indexing time, and annotate each lexical entity with the most general appropriate QA-Token, as well as all QA-Tokens subsumed by it. During question analysis time, a direct instantiation from the question word(s) to an appropriate QA-Token will suffice. For example, with *John F. Kennedy* annotated as both a **PRESIDENT\$** and **PERSON\$** in text, questions such as *Who wrote 'Profiles in Courage'?* suggesting the QA-Token **PERSON\$**,³¹ or *Which president wrote 'Profiles in Courage'?* instantiating the QA-Token **PRESIDENT\$** will both retrieve relevant passages containing *John F. Kennedy* as the correct answer.
2. We can index a lexical item with the most specific QA-Token only, and, when constructing the search engine query, include not only the QA-Token directly inferred from the question, but all classes subsumed by it. In the above example, *John F. Kennedy* would be annotated only as **PRESIDENT\$** in text, and *Who wrote 'Profiles in Courage'?* would instantiate not only the QA-Token **PERSON\$**, but also **PRESIDENT\$** and **ROYALTY\$** and any other subclasses of **PERSON\$**.
3. We can enable the search engine to perform the subsumption, thus having the query include only **PERSON\$** and having the text only annotated with **PRESIDENT\$**.

All three options are functionally equivalent, but the first option, which we selected to implement in our system, has the lowest run-time computational costs. Thus, we annotate items in text with the closest applicable QA-Token, along with all subsuming QA-Tokens, and replace the focus term in the question with the most specific (narrowest subsuming) QA-Token. When the reference of the question

³¹ For details on how QA-Tokens are instantiated during question analysis, see Section 3.1.

focus is ambiguous³² we replace it with all the QA-Tokens that correspond to the different focus senses. This scheme may be pictured as *vertical* for corpus annotations and *horizontal* for QA-Token instantiation in questions.

3. QUESTION ANALYSIS

Before search can take place, the question has to be processed into a suitable query format. This processing consists of eliminating stop-words, instantiating the QA-Tokens, weighting the terms and constructing the appropriate query syntax. The query syntax is discussed in Section 4.3; in this section, we focus on determining the appropriate QA-Tokens from the questions, as well as selecting appropriate keywords to include in the query.

3.1 *QA-Token Instantiation*

The *wh-words* in the question (*who*, *when*, *where*, etc., and also *how*), along with any words they modify, define the *focus* of the question. Other syntactic constructs, such as *name the ...*, are used for this purpose too. The focus is the semantic type of the entity the question is looking for. On the basis of the question focus, the question analysis component instantiates one or more QA-Tokens in the query to represent the sought entity, anticipating that any potential match would have been annotated with the same QA-Token at indexing time.

Our approach to QA-Token instantiation is primarily rule based. The question is matched against a set of rules that correspond to a variety of natural language constructs typically used to express questions about certain semantic classes of objects, or named entities. Each rule is associated with a set of QA-Tokens which are instantiated for the question if it matches the rule. The patterns are generally created when a new QA-Token is added to the system but also when a new, typical way of asking a question is discovered for which an existing QA-Token is appropriate.

Achieving full coverage of all syntactically and semantically correct ways of building questions corresponding to the set of QA-Tokens is very difficult, although our experience shows that with time it is possible to acquire a pattern set with high coverage. There are, however, always likely to be cases that were not included in the rule set. Therefore, to increase system performance, we augment our rule-based QA-Token instantiation mechanism with two additional techniques. The first utilizes an external ontology, in our case WordNet that was partially manually annotated with our QA-Tokens. The second technique adopts a statistical method based on collocation of keywords and QA-Tokens in the TREC9 annotated corpus. As a result, the QA-Token instantiation proceeds in four consecutive steps, with the steps

³²Although it is possible to perform word sense disambiguation to identify the most likely QA-token based on question context, we currently rely on the presence of these context words in search to identify passages with the correct word sense.

later in the sequence invoked only when earlier steps fail to find a match, with the last step assigning default QA-Tokens.

3.1.1 Rule-Based Mechanism

The first, grammar-based step, uses a set of several hundred hand-coded rules that consist of critical text-string fragments, which are matched in turn against the question until the first success. The rule matching mechanism has been designed to help us capture generalizations in question formulation. This is achieved by introducing variables and allowing matching to take place in different parts of the question. As shown in Figure 3, rules contain a combination of fixed text and variables; the variables stand for any of a possibly large set of alternative terms that could complete the fragment with the same sense. Thus, instead of needing rules that separately express *how high*, *how tall*, *how wide*, *how deep* etc., we have one rule

how _size1

where the variable *_size1* is defined to be any of {*high*, *tall*, *wide*, *deep*, *broad*, *long*, ...} This is especially useful when variables can be reused in the various rules that represent paraphrases of a given question type.

How _size1 _aux X	→ LENGTH\$ X
What _aux the _size1a of X	→ LENGTH\$ X
What _aux the area of X	→ AREA\$ X
What _aux the _size3a of X	→ VOLUME\$ X
How much _aux X cost	→ MONEY\$ X
Who X	→ @syn(PERSON\$ ORG\$) X

_size1	= long high tall wide deep broad ...
_size1a	= length height width breadth ...
_size3a	= volume capacity
_aux	= is are was were do does can ...

Figure 3: Sample rules for QA-Token instantiation. The upper box shows the rules that match against the question and transform it by removing wh- and associated words, and inserting QA-Tokens. The lower box shows expansions of the variables used in the upper box

We note that some questions have an ambiguous focus. *How long*, for example, could suggest either LENGTH\$ or DURATION\$³³. *Who* could suggest a PERSON\$ or an ORGANIZATIONS\$. These ambiguities can be handled easily in our system using our @SYN disjunction operator (see Section 4.3).

³³ Although analysis of the rest of the question can often disambiguate the two.

As a side effect of the rule application, we can also identify keywords that are best discarded from the query since they either come from phrases that are used to build questions (such as *name the/an X that* or *tell me X*), or whose semantics is contained in the semantics of the instantiated QA-Token. Examples of the latter include questions beginning with *what year* for which the word *year* can be dropped after **YEAR\$** has been instantiated, or *name the person who* for which the word *person* can be eliminated once **PERSONS\$** became part of the query. We can also add keywords to the query for specific QA-Tokens if such keywords would help limit the search even stronger than the presence of the QA-Token alone. For example, for a question beginning with *how wide* we add with a low weight

```
@SYN(wide width broad breadth across)
```

as it is likely that an answer might contain these terms.

3.1.2 Ontology Lookup

In order to increase our chances of recognizing as many question foci as possible we implemented the second step in the QA-Token instantiation that is based on on-line look-up of an ontology, in this case, WordNet. This mechanism allows the system to consult the WordNet hypernym hierarchy to find the appropriate QA-Token. It is applied only in those cases that did not result in a match in the first, rule-based step (a similar strategy is described in [Pasca *et al.*, 2001]).

First, the question focus is identified using a small set of general syntactic patterns anchored around the wh-word and the copula in the part-of-speech tagged question. The most productive patterns aim at identifying the nominal head of the wh-phrase (as in *what NP is...?*) or the prepositional nominal head following the copula (as in *what is (the) NP that/of...?*). Once the head is available, it can be looked up in WordNet and the list of its hypernyms can be retrieved. The hypernym list is searched for synsets that were previously identified as corresponding to a set of QA-Tokens. If such a synset is found, the QA-Tokens associated with it are instantiated for the question.

The set of synsets that map to a QA-Token set has been created manually by inspecting the top levels of the WordNet nominal hierarchy. Currently, the set includes around 100 synsets.

For some question foci, the two ways of searching for the QA-Token – one based on rules, the other one based on WordNet – are redundant since there is a specific rule in the rule set that matches this focus, and there is a synset that we identified that corresponds to it in the hypernym hierarchy. For example, the wh-phrase *what year* is as easily mapped with a specific rule as it is by finding a synset for *year* that maps to **{YEARS\$, TIME\$, DATE\$}**. However, the rule-based approach has advantages related to both coverage and precision. The rule-based step is very useful in two cases: 1) when the WordNet coverage is not sufficient, 2) when the question focus is ambiguous in WordNet. To illustrate, let's consider the question *What is the life expectancy of an elephant?* Even though our syntactic patterns used to identify the question focus are designed to discover phrases and will produce *life expectancy* as the focus of this question, there is no corresponding phrasal entry in WordNet

(v. 1.6), which means that unless we have a rule in our set (or augment WordNet), the question will not be assigned a meaningful QA-Token.

The second point is illustrated by the following question: *What was the date that...?* It is clear to a human reader that this is a question about a point in time. However, when looked up in WordNet, the noun *date* is ambiguous and in addition to the point-in-time meaning, it also has the meaning that is used to describe a person. This second meaning leads us to the **PERSON\$** QA-Token which is wrong for the given question. However, this reading will never be discovered since we have a high precision rule that will be applied in the first step and will assign the correct QA-Token blocking the application of the second step.

Generally, the WordNet look-up is likely to result in an ambiguity and multiple QA-Tokens are instantiated for the question. The problem of how to restrict the search in the hypernym hierarchy is very difficult and clues from the question can only sometimes be helpful in automatic disambiguation. One such clue is the wh-word that is present in the question. The wh-word *who* usually signals **PERSON\$** (but it might also signal **ORGANIZATIONS\$**, among other things), the wh-word *what* usually does not, especially when a copula separates it from the question focus (as in *what was the date...?*). We exploit this regularity by not allowing the search to enter the hyponym tree hanging off the *{person, someone, somebody, mortal, human, soul}* synset in WordNet when the wh-word is *what*.

3.1.3 Co-occurrence-Based Statistical Method

The third step in the process for QA-Token instantiation is a co-occurrence-based statistical method, described in full in [Prager *et al.*, 2002]. The idea behind this method is based on the observation that focus terms and QA-Tokens representing appropriate semantic types for these focus terms collocate frequently in a large corpus. For example, we found that the term *death toll* collocates frequently with **POPULATION\$** or **NUMBER\$**, while *life expectancy* occurs often with **AGE\$** or **NUMBER\$**, which are examples of QA-Tokens that cannot be instantiated in our QA system without adding large numbers of very specific rules to the first step or augmenting the ontology we use in the second step.

Based on the above observation, we devised a method in which we dynamically calculate how often each QA-Token in our inventory co-occurs with a given question focus in our annotated TREC9 corpus. This co-occurrence count is compared to how often it would co-occur if the pair were independently distributed. The QA-Token(s) with the highest deviation from the default condition would be selected.

We conducted a series of experiments based on different window sizes for determining co-occurrence:

1. On the same or adjacent lines in the source text.
2. In the same sentence.
3. Adjacent or separated by one word in the same sentence.
4. An average of the above methods.

We found that, overall, restricting the question term and QA-Tokens to be adjacent to one another or one word apart yielded the best performance, suggesting that extremely close proximity is key to success. To illustrate the performance of this method, the top 3 suggested QA-Tokens for a selection of question foci are presented in Table 2. The correct QA-Tokens for the corresponding focus term are represented in italics.

To summarize our findings, the process works well, with in the average case the correct classification being between first and second place, but fails in two kinds of cases: (1) the term is ambiguous (e.g. *What plant* gives **POWER\$**, i.e. power plants and power production, since the TREC corpus is more oriented to business and economics than horticulture), and (2) for stylistic reasons, the desired pairing is never or rarely found. For example, *continent* is readily mapped to **CONTINENT\$**, since expressions such as *the African continent* abound, but *country* did not generate **COUNTRY\$**, since the corresponding phrases (e.g. *the French country*) are rare. Further analysis showed that the terms that tended to fail this way (e.g. *country*, *state*, *year*) are all common terms and common in question logs, so it is highly likely that any QA system will already have rules for these, so this statistical method will not need to be applied. The method works particularly well for multi-word terms such as *death toll*, which is especially beneficial since such terms are often absent from WordNet and likely not to occur in a manually-derived rule set, unless previously encountered.

Table 2: Top three suggested QA-Tokens, with scores, for selected question foci, by the average of the three statistical methods. The scores are normalized into the range 0-1 and can be thought of as representing confidence values. Correct classifications are in italics.

Question Focus	1 st choice		2 nd choice		3 rd choice	
	Class	S	Class	S	Class	S
Wingspan	<i>LENGTH</i>	0.67	<i>SPEED</i>	0.12	<i>WEIGHT</i>	0.06
Cancer	<i>DISEASE</i>	0.39	<i>MEDICAL</i>	0.26	<i>CONSTELLATION</i>	0.22
Culture	<i>DURADATE</i>	0.19	<i>NATIONAL</i>	0.16	<i>THING</i>	0.14
Death toll	<i>WHOLENO</i>	0.22	<i>SMALLNO</i>	0.21	<i>DISEASE</i>	0.14
Season	<i>SPORTSTEAM</i>	0.34	<i>TIMEOFYEAR</i>	0.19	<i>WEIGHT</i>	0.12
Bone	<i>MUSICAL</i>	0.38	<i>DISEASE</i>	0.28	<i>MEDICAL</i>	0.12
Plant	<i>POWER</i>	0.52	<i>PROVINCE</i>	0.08	<i>ROLE</i>	0.07
Continent	<i>CONTINENT</i>	0.47	<i>PLACE</i>	0.11	<i>MOUNTAIN</i>	0.10
Country	<i>REGION</i>	0.22	<i>CONTINENT</i>	0.14	<i>PLACE</i>	0.10
State	<i>NICKNAME</i>	0.19	<i>ROLE</i>	0.14	<i>PLACE</i>	0.11
Song	<i>COMPOSITION</i>	0.34	<i>STAR</i>	0.26	<i>MUSICAL</i>	0.12

3.1.4 When all else fails

Finally, the fourth and last step in the QA-Token instantiation process assigns very general tokens such as **THING\$** in most cases when the three preceding steps did not succeed. In addition, the late-binding generic QA-Tokens such as **RELATIVE\$**, described in Section 6.2.1, are used too.

3.2 Query Formulation

In addition to instantiating appropriate QA-Tokens for search, the question analysis component must also select from among the terms in the question those that will enter into the search process. We also perform some non-QA-Token oriented steps, described in section 3.2.2, to generate a final query to be processed by our search engine.

3.2.1 Consideration of question focus terms

Usually, the interrogative pronoun or adjective, along with any word it modifies, is dropped from the question since its function is entirely subsumed by the QA-Token: we saw that occur earlier when *how high* was replaced by **LENGTH\$**. There are occasions, though, when a noun in the focus is retained. We have the very useful QA-Token **PERSON\$** to address most *who* questions, but we have also specially identified certain subclasses of it that we have given their own QA-Token. For example, we have **PRESIDENT\$** (representing US presidents), which is invoked to entirely replace the phrase *what president* (as long as no other nationality is mentioned). However, we do not have a corresponding QA-Token for *actor*, say, or for most of the hundreds of other roles and professions we have extracted from WordNet. Instead, we have the rule that matches:

```
what _profession
```

and instantiates **PERSON\$**, where *_profession* represents a long list of professions. Unlike the case of *president*, we include the word *actor* (or whatever profession is mentioned) in the query. This biases the search engine to finding passages that include the word *actor* as well as a proper name; even though such a collocation is by no means guaranteed to occur, we feel that the loss in recall is a better price to pay than the huge loss in precision if *what actor* were replaced by **PERSON\$** alone.

3.2.2 Other query processing

Having dealt with the desired question focus, we process the rest of the terms in the question in order to maximize the chances of matching an appropriate passage in the corpus. Stop words are removed from the query since they are not indexed, and inflected forms are converted to lemma form. Synonyms and other related terms are not expanded freely, but only under controlled conditions, as follows.

- Common words are expanded by rule to disjunctions including all morphological forms that are not captured by the lemmatisation process. Thus *friend* (or *friendship*) gets expanded to @SYN(friend friendship), *move* (or *motion*) gets expanded to @SYN(move motion).
- A small number of sets of related words have been identified for processing common question types. These sets can contain words that appear to be opposites or at least counterparts of each other. For example, since the answer to *Who is the wife of X?* might say that *X is the husband of Y* etc., we have the set @SYN(wife husband spouse marry), which is instantiated if any of the component words occur in a query.
- Countries and their national adjectives are considered synonyms.
- Numbers in numeric form and spelled out are equated.

Finally, query terms are weighted (see the @WEIGHT operator, described in Section 4.3.1). Rather than the traditional *tf*idf* scheme, we have found that the simple expedient of giving proper names twice the relative weight of common terms sufficient. In order to cause the identified answer-type to be very likely to be present in the passage, we give QA-Tokens a relative weight of 4.

4. SEARCH

4.1 Goals

While at first glance search engines might seem to be the appropriate vehicles for QA, on closer examination we begin to see a number of behavioral differences that translate into design requirements. When the question is fact-seeking, often a single sentence, or a fragment of a sentence, provides the answer, rather than an entire document. Rather than finding a document which contains as many occurrences of the query terms as possible, just one instance of each is often sufficient, as long as they participate in the appropriate semantic relationships with each other. Intuitively, the closer to each other the query terms occur in text, the more likely they participate in a semantic relationship. Thus we are led to the conclusion that we want the search engine to find us not documents but short passages, and that the scoring algorithm should reward the passage for each query term that appears in it, but not necessarily give any more reward for multiple occurrences. Moreover, the scoring algorithm should reward the closeness with which the present query terms congregate with each other.

4.2 Indexing

To satisfy these requirements, we built a custom text search engine. Like any modern text search engine, our custom search engine processes queries using an index built on top of the underlying document collection. The index is created by the

indexer, which takes the output of Textract and the QA-Token annotator and builds an *inverted file*. An inverted file contains an inverted list for each indexable item in the document. The indexable items include lemma forms of the original words in the document (obtained by morphological analysis of the word), Textract-generated canonical form annotations for proper names, and QA-Tokens. Textract annotations and QA-Tokens may span multiple words in the document (see Figure 2), thus we need to keep track of their length. We refer to all indexable items as *t*okens.

The inverted list for a token contains a posting for each occurrence of the token in the document collection. A posting identifies the document that contains the occurrence, the location within the document of the occurrence (specified as word and sentence offsets), and optional attributes associated with the occurrence, such as the length of the occurrence. The inverted list for a given token is located by searching for the token in a dictionary. The dictionary is typically implemented as a hash table or trie. For each token, the dictionary contains global attributes for the token, such as how many times it occurs in the document collection, and a pointer to the token's inverted list. Figure 4 shows portions of the inverted file for the sentence in Figure 2.

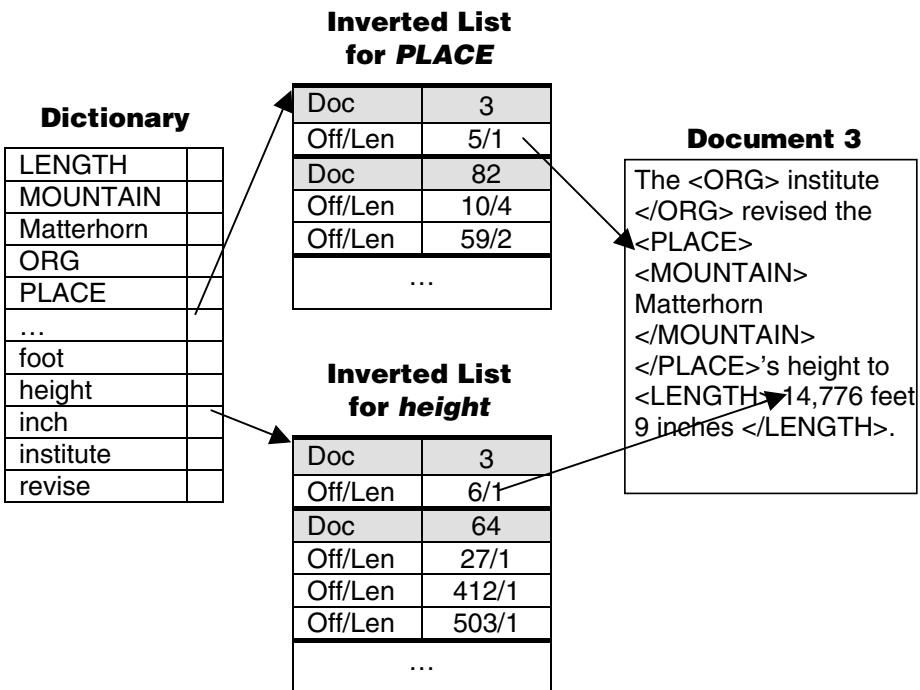


Figure 4: Partial view of index components for the sentence in Figure 2

4.3 Query Processing

The query processor evaluates queries and returns a list of documents sorted in decreasing order of relevance to the query. Queries are expressed using a simple query language. The basic operator is

@WINDOW(n expression)

which looks for **expression** in a window of **n** sentences. The expression is a list of one or more tokens. To evaluate the @WINDOW operator, the processor considers every window of n sentences in the document collection and assigns a score to each window by counting the number of matching tokens in the window (known as the *combination match score*). Multiple occurrences of the same token in the window count as a single match. For example, the query

```
@WINDOW(1 Matterhorn height)
```

applied to the sentence:

The institute revised the Matterhorn's height to 14,776 feet 9 inches

would generate a score of 2 (the possessive form *Matterhorn's* in the sentence would normalize to *Matterhorn* at indexing time).

In addition to simple tokens, **expression** can contain phrases and synonym sets. Phrases are expressed with the **@PHR(n token_list)** operator. The phrase matches wherever the tokens in the token_list occur in the order given with at most **n** other tokens intervening between any two tokens in the phrase. A synonym set is specified with the **@SYN(concept_list)** operator, which considers each of the terms in the concept list as if they were different representations of the same term. The @SYN acts like a single meta-term that occurs anywhere any of the terms in the concept list occur. The concept list may contain simple tokens or phrases.

4.3.1 Weighting

Recognizing that some query terms may be more important than others, the query language provides the **@WEIGHT(f factor)** operator, where **factor** is a simple token, @PHR, or @SYN and **f** is a real number. The @WEIGHT is used in an expression to change the weight of the factor in the scoring function from the default weight of 1 to the new weight **f**. For example, if we wanted *height* to be five times as important as *Matterhorn* in our previous example, we would write

```
@WINDOW(1 Matterhorn @WEIGHT(5 height)).
```

4.3.2 Exclusive vs. non-exclusive

With Predictive Annotation, we have the situation where multiple terms can be indexed in the same location: these terms are the base text terms, the Textract annotations, and one or more QA-Tokens representing their semantic class. The

search engine needs to be careful not to match more than one index item from any text location to a query. For example, suppose the question is

Where is Geneva?

which Question Analysis reduces to the query

PLACE\$ Geneva

We want it to match text that might say *Geneva, Switzerland*, where the query term PLACE\$ will match one of the annotations of Switzerland. However, since Geneva is also a PLACE\$, any text that mentions Geneva will have the annotation PLACE\$, with the undesirable consequence that both query terms will match a single occurrence of *Geneva* in a sentence. This leads us to the notion of an exclusive match, in which at most one of any superimposed index items is permitted to match a query.

The **@EXCWIN(n expression)** operator provides the exclusive match function. It operates similarly to the @WINDOW operator, except that the tokens in the expression are matched exclusively, with the precedence of the tokens determined by their order in the expression. To satisfy the requirements of the previous example, we would write the query as

`@EXCWIN(1 Geneva PLACE$)`

which would prevent PLACE\$ from matching *Geneva* in a sentence since the query term *Geneva* will have already matched exclusively.

4.3.3 Density

Given that two or more different windows of text in the document collection might achieve the same score for a given query, we need to introduce a method for breaking these ties. We accomplish this by preferring the window where the matching query terms occur more “densely” and add a density calculation to the scoring function. This calculation is based on the intuition that the closer the query terms appear to each other, the more likely the window contains the correct answer. Our density function is essentially the inverse of the distance between the first and last query terms in the window, normalized to be in the range 0 to 1. We add the density component to the original window score to achieve the final score for the window.

4.3.4 Dynamic Windows

The @WINDOW and @EXCWIN operators require that we specify a single window size for the query. It is unlikely, however, that a single optimal window size exists for all queries. Although our experience suggests that window sizes in the range 1 to 3 tend to work well, we cannot tell *a priori* what size window to use for a given query. To address this, the query processor provides the **@DYNWIN(n expression)** operator, which is the same as @EXCWIN except that the best scoring window in the size range 1 to *n* is found. The @DYNWIN operator

dynamically finds the optimal window size (within the specified range) for the query. In (Prager *et al.*, 2000) we reported on experiments to find the optimum window size for our system on the TREC8 questions. We supplemented these with experiments on the TREC9 questions and evaluated the system output in terms of Mean Reciprocal Rank (MRR)³⁴; the results are presented in Table 3. From the overall trends we concluded that the local minimum for $n=3$ in TREC8 was an anomaly and we should henceforth use dynamic windows with parameter $n=3$.

Table 3: MRR scores with the @DYNWIN operator and varying values of parameter n. The TREC9 value for n=10 was not measured.

<i>n</i>	1	2	3	4	5	7	10
TREC8	.328	.366	.351	.360	.356	.351	.330
TREC9	.352	.363	.369	.351	.351	.351	-

4.3.5 Execution

The actual execution of the query processor proceeds as follows. The processor parses the input query string and builds a tree representation of the query, with @PHR and @SYN operators at the internal nodes, tokens at the leaves, and @WINDOW, @EXCWIN, or @DYNWIN at the root. The weights from any @WEIGHT operators are saved in the corresponding token, @PHR, or @SYN nodes for use during window scoring. For each token at a leaf node, the processor locates the token in the dictionary and loads the token's inverted list. The processor evaluates @PHR operators by simultaneously traversing the inverted lists for the tokens participating in the phrase, identifying matching phrases using the information in the inverted list postings, and building a new temporary inverted list for the phrase. Similarly, the processor evaluates @SYN operators by “merging” the inverted lists for each token in the synonym set to create a new temporary inverted list for the synonym. The temporary inverted lists are passed up the query tree towards the root.

After the processor has evaluated all @PHR and @SYN operators, we are left with the root node and one or more temporary or token inverted lists. The query processor traverses these lists simultaneously and scores each window by summing the weights for all of the query terms that appear in the window (subject to the exclusive match constraint for the @EXCWIN and @DYNWIN operators) and applying the density calculation. In the case of @DYNWIN, we use a dynamic programming technique to accumulate scores simultaneously for all of the windows in the specified range and then select the best scoring window in the range.

The query score for a document is simply the score of the best scoring window in that document. The query processor sorts all documents by their score and returns

³⁴ Mean Reciprocal Rank is the basic measure used to rank QA systems at the TREC conference and it assumes an evaluation model in which a system returns a ranked list of 5 answer candidates. To compute the MRR for a set of questions, a question score equal to the reciprocal of the rank of the top correct answer candidate in the ranked list is assigned to each question first. The MRR is the arithmetic mean of all question scores in the set (Voorhees, 2000).

the top n scoring documents as the result list. For each document on the result list, the query processor indicates which window was the best scoring window and, in the case of the @DYNWIN operator, the length of the window.

4.4 Hit-list Size

The passages returned by the search engine are passed to the answer selection module, described in the next section. We must decide how many such passages to return. Intuitively, increasing the hit-list size will increase the recall, but due to the extra material that must be analysed may cause the precision to suffer from the extra noise. Thus there may be an optimal hit-list size to use (which will be a function of the criteria adopted by the answer selection component, and possibly the corpus and/or question source).

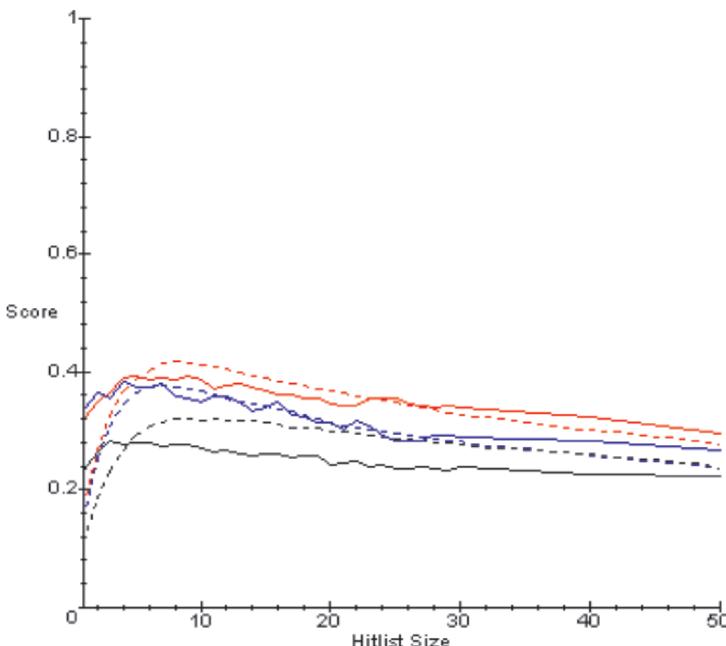


Figure 5: Actual system performance against hit-list size. Viewed at hitlist size=10, the top two curves are for TREC9 questions, the middle two for TREC8 and the bottom two for TREC10. The solid lines are actual scores, the dashed lines are generated by the model. The score is Mean Reciprocal Rank

In (Prager, 2002) a theoretical model of this process is developed, and is shown to agree closely with experimental results. This work shows the expected (and actual) system performance as a function of hit-list size. Some of the results are shown in Figure 5. For all three TREC question sets we see that the experimental

and predicted optimum hit-list size is in the range 6-11. We chose to proceed with a constant hit-list size of 10.

We note that this number is greatly less than that used by most other QA systems, and is directly due to the effects of Predictive Annotation. The *White House* example discussed in Section 2.1 illustrates this phenomenon quite dramatically.

5. ANSWER SELECTION

The task of the answer selection component is to identify likely answers to a question from the passages returned by the search engine given the query constructed as described in Section 3.2. Except for special question types, such as definition questions, for which we construct a larger set of passages, the search engine returns a hit list of 10 passages, each containing 1-3 sentences.

Given the relevant passages, the answer selection component first identifies candidate answers, i.e., text segments that could potentially be answers to fact-seeking questions. These are precisely the text segments spanned by QA-Token annotations in the passages. These candidate answers are, for the most part, proper names and common nouns or noun phrases, but could also include other phrases, such as *during World War II*, *because of his AIDS virus infection*, etc. Once identified, the candidate answers are ranked using a set of features that are selected to evaluate the quality of the candidate answers as well as the passages in which they occur, with respect to the given question and the QA-Tokens identified to represent potential answer types. The top n ranked candidate answers are then returned as the system's best answers to the question.

5.1 Answer Identification

The process for identifying candidate answers, for the most part, utilizes the Predictive Annotation mechanism. The only exception is when the question is identified as a definition question or an abbreviation, thus triggering special processing mechanisms to be described in Sections 6.2 and 6.3.

Figure 6 shows two passages returned by our search engine for the question *Where is the Valley of the Kings?* The candidate answers in these passages are identified and annotated with their semantic types. For example, *Howard Carter* and *Tutankhamun* are both annotated with the QA-Token **PERSON\$**, while *California* is considered both a **STATE\$** and a **PLACE\$**.

Passage 1:

The <THING> **specimens** </THING> were sent in <DATE><YEAR> **1932** </YEAR></DATE> by <PERSON> **Howard Carter** </PERSON>, <ROLE> **the British archaeologist** </ROLE> who discovered <PERSON> **Tutankhamun** </PERSON>'s treasure-laden tomb in the <PLACE> **Valleys of the Kings** </PLACE> at <PLACE><CITY> **Luxor, Egypt** </CITY></PLACE>, in <DATE><YEAR> **1922** </YEAR></DATE>.

Passage 2:

In the <PLACE> **Santa Clara Valley** </PLACE> in <PLACE><STATE> **California** </STATE></PLACE>, <ROLE> **King** </ROLE>'s birthday, indeed, is recognized.

Figure 6: Sample Annotated Passages for Where is the Valley of the Kings?

For certain question types, however, special processing mechanisms are invoked, thus leading to the need for dynamic generation of patterns for semantic annotation. For those questions where WordNet is consulted for potential answers, these answers are annotated with a dynamically generated QA-Token (such as **H1DEFINE\$**, see Section 6.4), in addition to the semantic classes assigned by the static rules and patterns. This ensures that the answers proposed by WordNet will be annotated with semantic types consistent with that sought in the question.

5.2 Answer Aggregation

As will be discussed in Section 5.3, multiple instances of the same candidate answer occurring in different passages provide valuable reinforcement for one another. However, many of our candidate answers differ slightly in surface form while carrying the same semantic content. Therefore, before ranking these candidate answers, our system first performs answer normalization with respect to the following attributes:

- Case: all candidate answers are normalized to lowercased form.
- Punctuation: all punctuation in the candidate answers is removed.
- Article: all articles in the candidate answers are removed.
- Name equivalence: using our named-entity recognizer Textract, we normalize proper name candidate answers which differ in surface form but refer to the same entity. For example, Bill Clinton, Mr. Clinton, President Clinton, and William Jefferson Clinton are all normalized to their canonical form William Jefferson Clinton.

In addition, all candidate answers annotated with the QA-Token **DATE\$** are further normalized to a date canonical form. In particular, relative dates, such as

today and *4 years ago*, are converted to an absolute date based on the ID of the document from which the answer is extracted. A date canonical form is then determined based on the absolute date with respect to the answer type. The canonical form may contain year only (for *What year did the Titanic sink?*), month/day only (for *When is Bastille Day?*), day of week only (for *What day of the week does Mother's Day fall on?*), and full date (for *When was D-Day?*).

5.3 Answer Ranking

The candidate answers identified in Section 5.1 include those that could potentially be an answer to *any* fact-seeking question. Once the candidate answers are identified and normalized, they are evaluated with respect to the given question based on features motivated by a set of *ranking criteria* as follows:

1. Good global context: the global context of a candidate answer evaluates the relevance of the passage from which the candidate answer is extracted to the question. For instance, in the example in Figure 6, the first passage, which discusses the discovery of the tomb at the Valley of the Kings, has better global context than the second passage, which happens to include the query terms generated from the question.
2. Good local context: the local context of a candidate answer assesses the likelihood that the answer fills in the gap in the question. For instance, Luxor, Egypt in the earlier example has good local context since it fills the syntactic role of being the location of the Valley of the Kings, which is sought by the question.
3. Right semantic type: the semantic type of a candidate answer should either be the same as or a subtype of the answer type identified by the question analysis component. For example, if the answer type is PERSON\$, then a candidate answer of the type PERSON\$ or PRESIDENT\$ has the right semantic type, but not an answer candidate of the type NUMBER\$.
4. Redundancy: the degree of redundancy for a candidate answer increases as more instances of the answer occur in retrieved passages. Candidate answers with a higher degree of redundancy are more likely to be correct answers if they occur under different contexts in the passages.

The ranking criteria above are used to motivate selection of scoring features that can be automatically extracted from the passages returned by the search engine, as well as the candidate answers themselves. In the rest of this section, we discuss the scoring features employed by our system, as well as the strategies adopted to utilize these scoring features for candidate answer ranking.

5.3.1 Feature Selection

To evaluate the global context in which a candidate answer occurs, we manually identified four features to assess the relevance of each passage retrieved by the search engine:

- **KeywordsInPassage**: the ratio of keywords present in a passage to the total number of keywords issued to the search engine. This is based on the observation that a passage is often more relevant to the question if more keywords extracted from the question occur in the passage.
- **NPMatch**: the number of words in noun phrases shared by both the question and the passage. Noun phrases in the question and the passage are extracted using Texttract’s shallow parser. This feature helps discriminate between passages where the keywords occur in the same context as in the question (such as *valley* and *king* both occurring as part of the noun phrase *valley of the kings* in passage 1 in Figure 6) and those where keywords occurred together by chance (the same keywords in passage 2).
- **SEScore**: the ratio of the search engine score for a passage to the maximum achievable score. The actual search engine score is computed as described in Section 4.3, while the maximum achievable score is that obtained when all query keywords are present and adjacent to one another in the passage.
- **FirstPassage**: a Boolean value which is true for the highest ranked passage returned by the search engine, and false for all other passages. This feature is motivated by the observation that many correct answers occur in the top-ranked passage in the hit list.

To evaluate the local context of a candidate answer, we utilize the following two features directly related to the candidate answer and its surrounding context:

- **AvgDistance**: the average distance between the candidate answer and keywords that occurred in the passage. This feature is based on the observation that candidate answers occurring in close proximity to the query keywords are more likely to be related to those keywords and are thus considered to have better local context.
- **NotInQuery**: the number of words in the candidate answers that are not query keywords. Those candidate answers that provide more novel information are considered more likely to be correct answers.

The semantic type match is represented by the feature **QATokenMatch**, which is the rank of the QA-Token representing the candidate answer in the ordered list of QA-Tokens that represent the answer type. In the example in Figure 6, *Luxor, Egypt*, which could be either a **PLACE\$** or a **CITY\$**, has a **QATokenMatch** score of 1,

given the candidate answer types **PLACE\$** and **UNL\$**,³⁵ whereas a candidate answer of the type **UNL\$** would have a **QATokenMatch** score of 2. Those candidate answers whose semantic type does not match the answer type has a **QATokenMatch** score of 5, which is greater than the number of QA-Tokens representing the answer type in the vast majority of questions. Finally, the redundancy factor is realized by the **frequency** count of a candidate answer, based on its canonical form.

5.3.2 Ranking Algorithm

When selecting answers to a given question, the feature scores discussed above are combined to determine the likelihood of each candidate being a correct answer. In our system, we adopt a linear combination of feature scores to arrive at an overall score for each candidate answer as follows:

$$\text{Score} = \sum w_i * f_i$$

In order to determine the relative contribution of each feature score, w_i , we generate a set of training data where, for each candidate answer, the feature scores f_i 's are computed as described in the previous section, and the overall score is 1 or 0 depending on whether or not the candidate answer is correct for the given question. Linear regression is then used to obtain the optimal weights for the feature scores. Using the first 500 questions in the TREC9 question set,³⁶ we obtain the following weights:

$$\begin{aligned} w(\text{KeywordsInPassage}) &= 0.022, \\ w(\text{NPMatch}) &= 0.003, \\ w(\text{SEScore}) &= 0.002, \\ w(\text{FirstPassage}) &= 0.012, \\ w(\text{AvgDistance}) &= 0, \\ w(\text{NotInQuery}) &= 0.002, \\ w(\text{QATokenMatch}) &= -0.034, \\ w(\text{Frequency}) &= 0.052. \end{aligned}$$

The weights we obtained through the learning process show that the most important factor in predicting candidate answer correctness is the **redundancy** factor, i.e., high frequency candidates are more likely to be correct than those that occurred only once in the retrieved passages, as observed in [Clarke *et al.*, 2001]. Furthermore, candidates are strongly preferred if they are of the **right semantic type**, i.e., if their semantic type matches the top ranked answer types, and if they have **good global context**, i.e., if a high percentage of query keywords occur in the passage and if the candidate answer occurs in the top-ranked passage. Note that the features that we selected to represent local context (**AvgDistance** and **NotInQuery**)

³⁵ Recall from Section 2.2 that **UNL\$** represents a recognized named entity that could potentially be a **PLACE\$**.

³⁶ This excludes those questions which are paraphrases of earlier questions.

did not turn out to be significant predictors of answer correctness. We are currently exploring other features which more precisely measure local context.

6. SPECIAL PROCESSING

While we have determined that Predictive Annotation is a very good general-purpose mechanism for Question Answering, we have found that we can achieve improved performance by adopting special processing mechanisms geared toward specific question types. Sometimes such processing is a variation of Predictive Annotation, and sometimes not. In this section we examine a variety of common question types for which we deviate from the standard processing path as described up to this point.

6.1 *Locations*

Where questions seek the location of an event or a geographic entity, but unlike *what city* or *what country* questions, for example, they do not specify the granularity of the desired answer. This suggests that the best we can do is to invoke the QA-Token **PLACE\$**. For event questions we have found this to be sufficient, since the remaining question words tend to be specific enough as to select passages that describe the event, and whatever locations are mentioned in those contexts often contain the correct answers.

On the other hand, if the question is *Where is X?*, where X is a geographic entity, then simply generating the query {PLACE\$ X} most often finds places that occur in the same context as X, rather than serve to locate X. For instance, the query often retrieves lists of geographic entities, including X, that satisfy a certain criterion. However, using the heuristic that a good answer to *Where is X?* is the geographic region that immediately contains X, we can provide a more focused search. We first test to see if X is a geographic entity that we know about, and determine its own semantic class, C, say. We then look up C in our geographic class ontology, which specifies the containment relationship between semantic classes represented by our geographic QA-Tokens (Figure 7), and extract all the parent classes. We then use this set of QA-Tokens in the query.

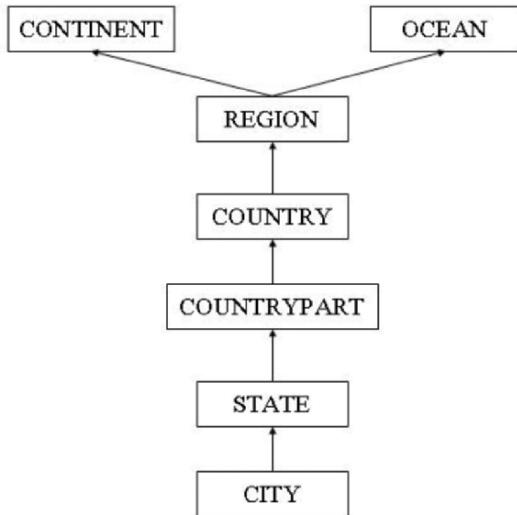


Figure 7: Our geographic classes arranged in a containment hierarchy. COUNTRYPART is a named subregion of a country, such as New England. REGION is a named area that spans countries, such as the Middle East

For example, for the TREC9 question *Where is Belize located?*, we first determine that Belize is a **COUNTRY\$**. Looking this up in the ontology we extract the class set **{CONTINENT\$, REGION\$, OCEAN\$}**. The query we then generate is:

@SYN(Belize Belizean) @SYN(CONTINENT\$ REGION\$ OCEAN\$)

By not simply looking for **PLACE\$**, this approach guarantees that the candidate answer set will not be diluted by geographic entities such as towns and cities that are unlikely to be good answers to such questions, regardless of their physical proximity to Belize in text. To complete the example, when we run our system on this question with and without this filtering function we get the top 5 candidate answers as shown in Table 4.

Table 4: Top 5 answers to Where is Belize located?, using first the generic PLACE\$ QA-Token and then using only those that specify locations larger than countries.

Position	PLACE\$		{CONTINENT, REGION, OCEAN}	
	Score	Answer	Score	Answer
1	5.086	South of Mexico on the Yucatan	4.715	Central America
2	4.619	Guatemala	4.515	Caribbean
3	4.096	Miami	4.096	Yucatan Peninsula
4	4.096	UK	4.096	Latin America
5	4.096	Latin America	1.558	Australia

It is seen from this example that using the filtered QA-Tokens gives a more complete and consistent set of candidate answers, especially if the score is interpreted as being proportional to confidence.

6.2 Virtual Annotation

Predictive Annotation is based on the assumption that the answer type can be derived from the question. However, in some cases, it is difficult, if not impossible, to determine the semantic type of the answer without knowing the answer itself. Examples of this type of questions include definition questions, such as *What is platinum?* and more generally un-typed questions, such as *What is measured in curies?*

We developed a method we call Virtual Annotation (VA) originally just to deal with definition questions, but have found the method general and extensible enough to deal with other questions whose target answer type is unknown or ill-defined. Virtual Annotation gets its name from the fact that answers found by this process get annotated at run time in a copy of the document, in order that our standard answer selection methods can work without modification. We will first explain VA with reference to general un-typed questions, and then describe its use specific to definition questions.

6.2.1 Un-typed Questions

When the semantic type of the answer cannot be readily inferred from the question itself, we perform the search without QA-Tokens, retrieve a larger hit list, and perform more thorough analysis on the passages that are returned. We observe that for questions like *What is X?* and *Who is X?*, where X can be any predicate, or questions like *What does X mean?* or *What does X stand for?* where X is a word or acronym, appropriate answers may well occur in text passages in close proximity to instances of X. For example, an answer passage found to the TREC9 question *What is measured in curies* begins *Radiation is measured in curies* There are a large number of syntactic constructs used to convey semantic equivalence or identity in

text – copulas, parenthetic expressions, setting apart with dashes, relative clauses, nominal pre-modifiers, amongst others.

We take advantage of the fact that the passages returned by the search engine are re-annotated in the answer selection module, in order to introduce new, run-time-only annotations. These annotation patterns, representing the syntactic constructs mentioned above, need an additional feature to those used for Predictive Annotation, since the phrase X must be used as an anchor. Example patterns are shown in Figure 8, where the syntactic construct “!X” is bound at run-time to X in the question. The actual syntax used in our system has been simplified for purposes of exposition.

```

!X , <phrase>,
!X - <phrase> -
!X (<phrase>)
!X, called <phrase>
!X, famous for <phrase>
<nominal phrase> !X
<phrase>, also known as !X

```

Figure 8: Sample patterns used in run-time annotation for Virtual Annotation. The sub-sequences in text that match <phrase> or <nominal phrase> are annotated with the QA-Token RELATIVE\$

We have altogether about a hundred such patterns. These patterns are conceptually similar to those used explicitly in TREC by Insight [Subbotin, 2001] and USC/ISI [Hovy *et al.*, 2001], and effectively similar to the question paraphrases used by Microsoft [Brill *et al.*, 2001].

In operation, the desired tokens in text that match through the contextual patterns in Figure 8 are annotated with **RELATIVE\$**. When the question *Who was Jane Goodall?* was run using this method, the following answers were found:

- ape researcher
- wild chimpanzee researcher
- naturalist
- Chimpanzee researcher
- including famed researcher
- a scientist of unquestionable reputation
- British ethnologist
- used the power
- female professor

One answer is wrong and another includes an extraneous word (since the gerundive looks like an adjective), but otherwise this simple method performed surprisingly well.

6.2.2 Definition Questions

Definition questions are typically of the form *What is X?*, where X is a simple noun phrase, although other forms such as *What does X do?* and *What is X used for?* can also be regarded as definition questions, as can *Who is X?*. The basic method used is to find the answer by means of an ontology such as WordNet, and then locate the known answer in text. The requirement to locate in text an answer already in-hand is an artefact of the current TREC evaluation, but nevertheless is useful in a more general setting as providing textual justification for an answer whose derivation from a structured resource alone may not be sufficiently explained or justified for an end user.

Virtual Annotation as applied to definition questions is fully described in (Prager *et al.*, 2001). When a question that qualifies as a definition question (such as *What is X?*) is identified, the focus term X is looked up in WordNet. If there is no entry, it is treated as an untyped question, and the method described in Section 6.2.1 is used alone. If there is an entry, all of its hypernyms (parents) are read out, under the assumption that a plausible if not good definition of a term is the class to which it belongs – for example *What is platinum?* would be answered by a human with *a metal*, and indeed **{metallic element, metal}** is the grandparent synset of *platinum* in WordNet. The problem to be solved, though, is to choose which level of hypernym to use as the answer. The entire parentage of *platinum* is **{noble metal, {metallic element, metal}, {chemical element, element}, {substance, matter}, {object, physical object}, {entity, something}}**.

To choose an answer, we make two observations:

- A more specific parent class (e.g. noble metal is more likely to be useful than a more general class (e.g. entity).
- A class name that is used more in practice (e.g. metal) is more likely to be useful than a rarer term (e.g. noble metal), especially since these rare terms may sometimes be unknown to an average user of a QA system.

We combine these observations in the following algorithm. For each ancestor **A** we count the number of documents in our corpus in which **A** co-occurs with the question term X in the same or adjacent sentences. We divide this count by the number of hypernym steps in WordNet between X and **A**. Thus, the score selects more common terms (predicted by the fact that it co-occurs with the question term), while penalizing terms that are farther from X in the WordNet hierarchy. The algorithm returns the candidate with the greatest score so computed, with the additional modification that any other parents **A'** with a score within 20% of the highest will be returned too, forming a disjunction. To illustrate, Table 5 shows the results of processing *platinum*; **metal** is the clear winner.

Table 5: Parentage of platinum along with number of hypernym steps in WordNet, the number of co-occurrences of the hypernym with platinum, and the computed scores.

Term	Level Gap	Co-occurrences	Score
Noble metal	1	3	3
Metallic element	2	1	0.5
Metal	2	880	440
Chemical element	3	0	0
Element	3	15	5
Substance	4	2	0.5
Matter	4	15	3.75
Object	5	4	0.8
Physical object	5	0	0
Entity	6	0	0
Something	6	0	0

The entire process for such questions is as follows. After the above-described algorithm is run, the question is converted into a query consisting of the virtual QA-Token **DEFINES\$**, the question focus and the candidate answer – the query in this example becomes {**DEFINES\$** metal platinum}. This is sent to the search engine. Since **DEFINES\$** was not active at indexing time, it matches nothing so the search engine just returns passages scored by the other terms. Patterns are created on the fly to associate **DEFINES\$** with the candidate answer (*metal*), the returned passages are re-annotated and the answer selection component “discovers” all terms there annotated by the question focus, **DEFINES\$**.

6.3 Abbreviation Questions

Just as definition questions are handled by a generic method for untyped questions as well as by a specific method, so too are abbreviation questions, although in this case the specific method employs Predictive Annotation against a special index. To create this index we take advantage of an abbreviation-finding component of our named-entity recognizer Textract (Byrd and Ravin, 1999).

One of the functions of Textract is to identify and output *named relations*, which are tuples of two textual entities and the semantic relation between them, for example:

(International Business Machines, located-in, New York State, <docid>, <para-no>, <sent-no>)

where the last three elements pinpoint the sentence where the relation is detected. Another one of Textract's functions is to identify abbreviations, and it can output them as a kind of named relation, for example,

(IBM, same-as, International Business Machines, <docid>, <para-no>, <sent-no>)

We take advantage of this functionality in QA in the following manner. We process the TREC corpus by Textract and output all of the abbreviation relations. For each relation of the form (<short form>, same-as, <long-form>, ...) we create a pseudo-document with title “<short form> same-as <long form>”, and with the single line of text:

NAME\$ <short form>, <long form>. ABBRCOUNT [n]

where n is the number of times the “same-as” relationship is detected in the corpus, and is used as the frequency count by the answer selection component. These documents are indexed along with the source documents in the TREC collection. When we process questions of the form:

What does XYZ stand for?

(or any of its many paraphrases) we generate the query:

NAME\$ XYZ.

This will match perfectly the pseudo-document generated as described above, and the returned title is simply parsed to extract the long, canonical form of the abbreviation.

6.4 Generating all instances of a class: H1DEFINE and H2DEFINE

An interesting problem is posed by questions such as *What type of bridge is the Golden Gate Bridge?*, *What kind of building was the World Trade Centre?* or *Name a flying mammal*. All these questions can be seen as looking for a hyponym of the question focus: in the TREC QA collection, the first question can be answered with the phrase *suspension bridge*, the second one with the phrase *office building*, the third one with *bat*. All these phrases can be also found in WordNet in the hyponym trees for the foci in the above questions.

Using the QA-Token instantiation algorithm described in Section 3.1, the last question gets easily mapped to a query with the **ANIMAL\$** QA-Token. The other two questions are not as easily handled since there is no QA-Token in the system that would correspond to building or bridge types (see Section 2.2). This is a problem for the basic Predictive Annotation approach since there is no good way to focus the search on the right set of expressions if they are not annotated with a QA-Token. In order to handle the hyponym questions we introduce another extension to

Predictive Annotation to handle dynamically created QA-Tokens. The two QA-Tokens that we introduce for handling hyponym questions are **H1DEFINE\$** and **H2DEFINE\$**.³⁷ These particular dynamic tokens have fixed names and are known to the answer selection module. However, similar to the case with **DEFINE\$**, their extensions, i.e., the phrases that should be annotated with **H1DEFINE\$** and **H2DEFINE\$**, are not known at indexing time and thus will not be searched for as such by the search engine.

After a hyponym question has been identified (typically by the presence of phrases such as *type of* and *kind of*, or of indefinite articles), the extension of the question focus is constructed by looking up all the hyponyms of the focus available in WordNet. This extension is used by the answer selection component to annotate, at run-time, the documents retrieved by the search engine with the **H1DEFINE\$** or **H2DEFINE\$** QA-Tokens, in effect allowing it to focus its search only on the phrases retrieved from WordNet, as in the case when Virtual Annotation is used to answer definition questions (Section 6.2).

The reason for introducing two different QA-Tokens to deal with hyponym questions is illustrated by the following difference between the questions *What type of bridge is the Golden Gate bridge?* and *Name a flying mammal*. The first question tries to establish the type of an existing bridge that is referred to in the question using a proper name. The proper name usually signals a unique entity in the world, and another proper name is unlikely to be an answer.³⁸ When given such a question, the QA-Token instantiation algorithm will assign the **H2DEFINE\$** QA-Token to the question. This means that all the leaves in the hyponym tree of the question focus that are proper names (which we detect by looking at the case of all the words constituting the name) need to be discarded. The second question can be answered by a proper name, e.g., the proper name of a bat species. Thus, if no proper name is present in the question, as is the case for *Name a flying mammal*, the assigned QA-Token is **H1DEFINE\$** and all the hyponyms of the question focus are allowed to enter the QA-Token extension.

To make the reasons clearer, let's consider the question *What type of building was the World Trade Centre?* Both the World Trade Centre and the Empire State Building are WordNet hyponyms of the term *building* and, because of their location, are quite likely to co-occur in documents. However, we do not want the Empire State Building to be a candidate answer for the question. Hence this question is assigned the **H2DEFINE\$** QA-Token whose extension does not contain *Empire State Building*, but does contain *office building*. For the question *Name a flying mammal*, proper names of flying mammals are good answers, and the assigned QA-Token is **H1DEFINE\$**. In making the last assumption we rely on the fact that a general purpose ontology is not likely to have proper names referring to individual

³⁷ The QA-Token names H1DEFINE\$ and H2DEFINE\$ were motivated by the similar processing strategy that is used for definition questions and the two different ways of consulting the WordNet hyponym tree.

³⁸ The exception is when it is the root of an entire hyponym tree, which is a rare case, as it would mean that the proper name would denote a class of object rather than a unique entity.

animals such as *Fido*, but it does have at least some of the well-known buildings from around the world.

6.5 *Towards feedback-driven processing*

One of the characteristics of our current architecture is the sequential, one-pass search strategy for most question types. As has been shown by (Harabagiu et al, 2001; Pasca and Harabagiu, 2001), it might be beneficial to include processing loops in a QA system. Such processing loops allow the system to retry processing steps with different parameters to obtain more or better answer candidates and in the case of (Pasca and Harabagiu, 2001) lead to a significant MRR score boost.

In order to implement such processing loops, the system needs an introspection mechanism that would allow it to assess the quality of the results from an intermediate processing step or of the final system output. A mechanism of this sort is also necessary if the system is to inform the user that no answer could be found in the collection, or more strongly, that there is no answer present in the collection. It could also be used in a system with planning to assess the quality of the results of an operator to decide which operator to apply next.

An evaluation of the TREC9 results from our system showed a significant number of questions for which the document retrieval part of the system did not return a hit list containing a document with a correct answer given the predefined hit list size. There are a number of recall-enhancing techniques that can be applied to try and increase the number of questions for which a good answer candidate is found, including query expansion or increasing the length of the hit list. Since these techniques usually lower the precision, we investigated a method for detecting “bad” hit lists, i.e., hit lists in which there are no passages with correct answer candidates. A full description of the method is presented in (Czuba *et al.*, 2002), this section presents an overview of the most important results.

Because of the use of QA-Tokens, hit lists returned by the search engine are highly focused on Named Entities of the right type. We hypothesize that the score assigned to each passage by the search engine provides important information about the passage quality, and the patterns in which the scores change within the hit list may also indicate the hit list quality. For example, if the score for the top passage is far from the maximum achievable score given the query terms, it is an indication that many important keywords were not present in the passage, and as a result, we may safely conclude that the passage is not a good candidate for finding a correct answer. The correlation of score change patterns with hit list quality may also vary according to QA-Token types, in particular the frequency of QA-Tokens in the corpus. High passage scores indicating that a Name Entity of the right semantic type is present in the passage may be better evidence that the correct answer is present if the candidate QA-Token is relatively rare (e.g., **BODYPART\$**) than if it is very frequent (e.g., **PERSON\$**), since in the latter case, the QA-Token has an increased possibility of being present by chance.

In order to capture such regularities we decided to approach the problem of establishing the hit list quality as a concept learning task. We defined a number of

features based on the scores assigned by the search engine to the documents in the hit list and trained a set of decision tree classifiers with different feature sets to see which feature classes are best predictors of hit list quality; and whether particular question types (those seeking answers of particular QA-Token types) are better candidates for our prediction mechanism than others.

We identified several classes of features which we believe may be correlated with hit list quality. Let N be the hit list length (typically 10), T_{max} and D_{max} be the maximum achievable term and density scores, respectively, and T_i and D_i be the term and density scores of the i -th document in the hit list, respectively (see Section 4.3 for the description of the document scoring formula). We define $T_0 = T_{max}$ and $D_0 = D_{max}$. The classes of features we chose that are based on search engine scores are defined as follows:

1. **F**: N features calculated from the relative change in the term score: $f_i = (T_i - T_{i-1})/T_{i-1}$, $1 \leq i \leq N$.
2. **S**: $N-1$ features calculated from the sign of the second derivative of the score function: $s_i = \text{sign}(f_{i+1} - f_i)$, $1 \leq i < N$;
3. **D**: N features calculated from the relative change in the density score: $d_i = (D_i - D_{i-1})/D_{i-1}$, $1 \leq i \leq N$.

The features in **F** and **S** are based only on the term score, i.e., they reflect the change in the number and kind (QA-Token, proper name, other) of keyword matches. **D** is based solely on the density scores. Additionally, we introduced two alternative feature classes based on the QA-Tokens identified by the question analysis:

1. **T**: 83 binary features, each of which corresponds to a QA-Token type in the system.
2. **P**: 47 binary features, similar to **T**; only the frequently-occurring QA-Tokens are represented, the remaining QA-Tokens are combined to form the type **OTHER**.

These sets make the question type information available to the classifier and are motivated by the intuition that some question types might be more amenable to our classification technique. **P** is introduced in an attempt to reduce the data sparseness problem for rare types.

To test the quality of the above features we defined a number of question subsets based on the TREC9 and TREC10 questions according to the frequency of candidate QA-Tokens assigned to questions in the combined TREC9 and TREC10 question sets and on QA-Token semantics.

Table 6 presents the question subsets (**ABBREV**, **DEFINE**, **ORG**, **PERSON**, **PLACE**, **THING** and **TIME**), their cardinalities, and percentages of questions for which a “good” hit list was found by the system. These question subsets were used to create the training and test data for our decision tree classifiers.

Table 6: Question subset statistics

	TREC9	TREC10	Question Types
ABBREV	83 (66.2%)	136 (56.6%)	Abbreviation questions
DEFINE	84 (65.7%)	144 (52%)	Definition questions
ORG	70 (70.0%)	24 (66.6%)	ORG\$
PERSON	143 (74.8%)	69 (60.8%)	PERSON\$, PRESIDENTS\$ ROYALTY\$, etc
PLACE	114 (77.1%)	63 (57.1%)	PLACE\$, CITY\$, CAPITAL\$, etc
THING	120 (59.1%)	193 (50.2%)	THINGS\$
TIME	76 (76.1%)	53 (77.3%)	YEARS\$, DATE\$, AGOS\$, TIMEOFTIME\$, etc

Table 7: Best performing feature sets, accuracy and relative percentage gains

	TREC9			TREC10		
ABBREV	FSD	73.3	+10.7	FS	55.2	-2.5
DEFINE	FSDT	71.1	+8.2	FP	62.2	+19.6
ORG	F	67.1	-4.1	FSD	58.3	-12.5
PERSON	F	75.4	+0.8	FSDT	73.8	+21.4
PLACE	FD	79.0	+2.5	F	69.0	+20.8
THING	FD	68.3	+15.6	FS	53.4	+6.4
TIME	FSD	71.6	+6.7	FDSP	78.3	+1.3

Table 7 shows the best performing combination of feature sets for a given question class, along with the accuracy of the trained decision tree and the relative improvement over the classifier that classifies all hit lists as “good”. 10-fold stratified cross-validation was used. For most question types we achieve an improvement in classification accuracy when the decision tree with the best feature set is used. The accuracy gain is generally larger for the TREC10 question subsets which we attribute to the fact that the TREC9 questions were used as training material for our system, in particular for the query formulation and expansion strategies. One result worth pointing out is the positive gain in accuracy for the important class of definition questions for both TREC9 and TREC10 question subsets.

7. FUTURE DIRECTIONS – PIQUANT: ADVANCED QA FOR AQUAINT

For our participation in the first phase of the AQUAINT (Advanced QUeStion Answering for INTeLLigence) program sponsored by ARDA, we have developed a new architecture for our QA system. Our new system design is called PIQUANT (Practical Intelligent QUeStion ANswering Technology), and features a plan-based execution engine, use of multiple external knowledge sources and an inference

engine, multiple solution paths and confidence-based answer resolution. Additionally, more sophisticated candidate evaluation in Answer Selection based on modification relationships will, from what we see in our initial experiments, provide another route to better accuracy.

The new design explicitly acknowledges the requirement for different processing techniques, such as those outlined in the previous section, including the option to process a given question in more than one way, and makes such processing the norm rather than the exception. To illustrate, consider the question *What is the largest city in England?* The answer can be found in a variety of ways, using different knowledge sources. For example:

- Perform a text search to find text that explicitly states “London is the largest city in England” (or a paraphrase).
- Find a table of English cities and their populations, and sort it.
- Find a list of the 10 (or 20 or so) largest cities in the world. The first English one encountered is the answer.
- Find the capital of England, since most (but by no means all) largest cities in a nation are the capitals. Use this just as suggestive evidence.
- Develop an inference chain from facts (found in any of the ways listed above) such as Birmingham is England’s second largest city, Paris is larger than Birmingham, London is larger than Paris, London is in England.

Our new QA architecture allows for multiple *answering agents*, each adopting a specialized question answering strategy; with the new architecture we can explore multiple solution paths in parallel, and combine answers proposed by each answering agent based on probabilistic conference processing. One answering agent we are currently focusing on centres around the notion of a **QPLAN**, which is a disjunction of a set of methods to answer a particular kind of question. Via a set of API calls named **QGOALS**, QPLANS query multiple *knowledge sources*, such as our text search plus answer selection, ontologies such as WordNet, databases or knowledge bases; for the last in this list we are exploring the use of Cyc from Cycorp (Lenat,1995). QPLANS can also call other QPLANS, or themselves recursively.

QPLANS are selected after the question is analysed and classified. We have identified the following kinds of questions, each of which has its own QPLAN:

- **Property_of_Object:** What is the GNP of the Philippines?
- **Superlative:** Which country exports the most uranium?
- **Comparative:** Does Kuwait export more oil than Venezuela?
- **Definition:** What is OPEC?
- **Profile/Biography:** Who is Rabbani?
- **What_X:** What precious metal is mined in South Africa?
- **Relationship:** Which countries are allies of Qatar?
- **Chronology:** Was OPEC formed before Nixon became president?
- **Enumeration:** How many oil refineries are in the U.S.?
- **Cause_&_Effect:** Why did Iraq invade Kuwait?
- **Default:** All other questions.

Not only can a given QPLAN cause multiple solution paths to be attempted, but some questions can cause multiple QPLANS to be triggered. For example, *What is the Declaration of Independence?* and *What is the Capital of Britain?* will both trigger the **Definition** and **Property_of_Object** QPLANS, but only one will succeed in each case.

Each QPLAN and each knowledge source will have its own a priori confidence (learned over time) and will associate confidence scores with each candidate answer it produces. These candidate answers will be funnelled through a new answer resolution module (modelled on the current answer selection which deals solely with text-derived candidates), which will eliminate duplicate answers, merge equivalent answers and perform confidence combination. Similarly, answers proposed by different answering agents will be combined to arrive at final scores for the ultimately emitted answer (set).

7.1 Example

This example illustrates the initial results from using a **Property_of_Object** QPLAN, along with a knowledge base (Cyc) as an alternative answering agent to text. The QPLAN is illustrated in Figure 9.

```

QPlan(property, "What is the P of X") ->
OR {
    QPlan(base_property, "What is the P of X")
    If X = Y of Z
        {T = QPlan(property, "what is the Y of Z");
        QPlan(base_property, "what is the P of T")}
    }
}

QPlan(base_property, "What is the P of X") ->
OR {
    QGoal(db_kb, P of X)
    Q$ = find_class(P);
    OR {QGoal(text, "Q$ X");
    QGoal(text, "Q$ P X")}
    QPlan(define, "what is the (P of X)")
}

```

Figure 9: Property_of_Object QPLAN. The QPLAN has a recursive component for directing the plan decomposition, as well as a base component for issuing QGOALS against the different available knowledge sources

The issued question is *What is the population of the capital of Tajikistan?* Without the QPLAN mechanism, a straightforward text-based search finds the passage:

Most shops closed and goods were in short supply in the Tajikistan capital of Dushanbe this week as the ex-Soviet republic began converting to the Russian rouble. With its three-day phasing in of the rouble, to be completed tomorrow, the Central Asian republic of 5.3m people is taking the final step in ceding to Moscow almost complete control of its economy.

and extracts the incorrect answer of *5.3m people*.

The base plan tries issuing knowledge base (Cyc), data-base and definition QGOALS for the object *the population of the capital of Tajikistan*, which fail. The text QGOAL is for the query

POPULATION\$ capital of Tajikistan

which matches the text passage given above, with the same incorrect answer, but this is given a relatively low confidence since the correct modification relationships are missing.

Next, the high-level QPLAN decomposes the question into

T = *What is the capital of Tajikistan?*

and

What is the population of T?

These result in recursive calls to the QPLAN, which attempts to issue base QGOALS for them. Cyc returns T=*Dushanbe*, but does not know what the population of Dushanbe is. Text search, with the QGOAL:

POPULATION\$ Dushanbe?

finds this passage:

Tucker said Dushanbe, with some 460,000 residents, has multi-story buildings, mostly made of concrete slabs loosely hooked together.

and with the high-confidence returns the correct answer of *460,000*.

8. SUMMARY

We have presented here a fairly complete exposition of our text-based Question-Answering system. We have described the method of Predictive Annotation, in which the semantic classes of many types of noun phrases and other entities get indexed along with the original text tokens, so that including these labels in search engine queries can find relevant passages with very short hit-lists. We have described the structure of the index and the syntax and operation of the search engine. We have described how by analysing the hit-lists returned we can estimate whether the question has an answer in the corpus.

We have described the Answer-Selection component which extracts and evaluates candidate answers from the passages in the hit-lists based on a set of statistical and linguistic features. We also described special processing that is incurred for definition-type questions, where the answer may be first located in an external resource such as WordNet, and then found in the appropriate context in text.

Finally, we have described the overall design of and initial results with our new plan-based system PIQUANT. PIQUANT still performs text search, but it puts other resources such as ontologies and knowledge bases on an equal footing. It uses plans specific to the type of question being processed; the plan may call for question decomposition or other restructuring, and will call one or many resources to answer questions and sub-questions. The multi-path approach allows alternative modules to fill in if others fail to return a result, or to provide supporting or alternative evidence if more than one pathway results in an answer set.

9. ACKNOWLEDGMENTS

The authors would like to thank Anni Coden and Dragomir Radev for their contributions to earlier versions of our system, and to Dave Ferrucci, Bob Mack, Alan Marwick and Yael Ravin for their support and helpful comments and suggestions. This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number MDA904-01-C-0988.

10. REFERENCES

- Brill, E., Lin J., Banko M., Dumais S., Ng A. "Data-intensive Question Answering." In *Proceedings of TREC-2001*, Gaithersburg, MD, 2001.
- Byrd, R. and Ravin, Y. "Identifying and Extracting Relations in Text." In *Proceedings of NLDB 99*, Klagenfurt, Austria, 1999.
- Clarke, L.A., Cormack, G.V. and Lynam, T.R. "Exploiting Redundancy in Question Answering", In *Proceedings of SIGIR 2001*, New Orleans, LA, 2001.
- Czuba, K., Prager J., Chu-Carroll, J., "A Machine-Learning Approach to Introspection in a Question Answering System", In *Proceedings of EMNLP-02*, Philadelphia, PA, 2002.
- Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V., and Morarescu, P. "The Role of Lexico-Semantic Feedback in Open-Domain Textual Question-Answering", In *Proceedings of ACL 2001*, Toulouse, France, 2001.
- Hovy, E., Gerber, L., Hermjakob, U., Junk, M. and Lin, C-Y. "Question Answering in Webclopedia. In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, Gaithersburg, MD, 2001.
- Hovy E., Hermjakob U. and Lin C-Y. "The Use of External Knowledge in Factoid QA." In *Proceedings of TREC-2001*, Gaithersburg, MD, 2001.
- Ittycheriah, A., Franz, M., Zhu, W-J., Ratnaparkhi, A. and Mammone, R.J. "Question Answering Using Maximum Entropy Components", *Proceedings NAACL*, Pittsburgh, PA, 2001.
- Kupiec, J. "A robust linguistic approach for question answering using an on-line encyclopedia." In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 181-190, Pittsburgh, 1993.
- Lehnert, W.G. *The Process of Question Answering*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.
- Lenat, D.B. "Cyc: A Large-Scale Investment in Knowledge Infrastructure." *Communications of the ACM* 38, no. 11, Nov. 1995.
- Miller, G. "WordNet: A Lexical Database for English", *Communications of the ACM* 38(11) pp. 39-41, 1995.

- Pasca, M.A. and Harabagiu, S.M. "High Performance Question/Answering", *Proceedings of SIGIR 2001*, New Orleans, LA, 2001.
- Prager, J.M. "In Question-Answering, Hit-List Size Matters", in preparation.
- Prager, J.M., Chu-Carroll, J. and Czuba, K. "Statistical Answer-Type Identification in Open-Domain Question-Answering", In *Proceedings of HLT 2002*, San Diego CA, March 2002.
- Prager, J.M., Radev, D.R. and Czuba, K. "Answering What-Is Questions by Virtual Annotation". In *Proceedings of Human Language Technologies Conference*, San Diego CA, March 2001.
- Prager, J.M., Brown, E.W., Coden, A. and Radev, D. "Question-Answering by Predictive Annotation". In *Proceedings of SIGIR 2000*, pp. 184-191, Athens, Greece.
- Prager, J.M., Radev, R., Brown, E.W., Coden, A. and Samn, V. "The Use of Predictive Annotation for Question-Answering in TREC8". In *Proceedings of TREC-8*, Gaithersburg, MD, 2000.
- Radev, D.R., Prager, J.M. and Samn, V. "Ranking Suspected Answers to Natural Language Questions using Predictive Annotation". In *Proceedings of ANLP 2000*, pp. 150-157, Seattle, WA.
- Salton, Gerald. *Automatic Text Processing: the Transformation, Analysis and Retrieval of Information by Computer*, Addison-Wesley, Reading, MA, 1989.
- Subbotin, M. "Patterns of Potential Answer Expressions as Clues to the Right Answers", in *Proceedings of TREC-2001*, Gaithersburg, MD, 2001.
- Voorhees, Ellen. "The TREC-8 Question Answering Track Report", in *Proceedings of TREC-8*, Gaithersburg, MD, 2000.

QUESTION ANSWERING SUPPORTED BY MULTIPLE LEVELS OF INFORMATION EXTRACTION

Abstract: This chapter discusses the importance of information extraction (IE) in question answering (QA) systems. Most QA systems are focused on sentence-level answer generation. Such systems are based on information retrieval (IR) techniques such as passage retrieval in conjunction with shallow IE techniques such as named entity tagging. Sentence-level answers may be sufficient in many applications focused on reducing information overload: instead of a list of URLs provided by search engines which need further perusal by the user, sentences containing potential answers are extracted and presented to the user. However, if the goal is precise answers consisting of a phrase, or even a single word or number, new techniques for QA must be developed. Specifically, there is a need to use more advanced IE and natural language processing (NLP) in the answer generation process. This chapter presents a system whereby multiple levels of IE are utilized in a QA system that attempts to generate the most precise answer possible, backing off to coarser levels where necessary. In particular, generic grammatical relationships are exploited in the absence of information about specific relationships between entities. An IE engine, InfoXtract, is first described to illustrate the types of IE and NLP output that is needed. Results are presented for QA based on named entity tagging, relationships between entities, and semantic parsing, including more recent work that handles caseless input. This work has implications for broader coverage QA systems since domain-independent IE results can be exploited and noisy input can be “normalized” by restoration techniques.

1. INTRODUCTION

Natural language Question Answering (QA) is recognized as a capability with great potential. The NIST-sponsored Text Retrieval Conference (TREC) has been the driving force for developing this technology through its QA track since TREC-8 (Voorhees, 1999). There has been significant progress and growing interest in QA research in recent years (Voorhees, 2000; Pasca and Harabagiu, 2001).

QA differs from search engines in two aspects: (*i*) instead of a string of keyword search terms, the query is a natural language question, necessitating question parsing, (*ii*) instead of a list of documents or URLs, a list of candidate answers at the phrase level or sentence level is expected to be returned in response to a query. Hence QA requires text processing beyond keyword indexing; typically QA is supported by NLP and IE (Chinchor and Marsh, 1998a; Hovy *et al.*, 2001; Li and Srihari, 2003c). Examples of the use of NLP and IE in Question Answering include shallow parsing (Kupiec, 1993), deep parsing (Litkowski, 1999), Named Entity tagging (Abney *et al.*, 2000; Srihari and Li, 2000b) and high-level IE (Li *et al.*, 2002).

Identifying exact or phrase-level answers is a much more challenging task than sentence-level answer generation. Sentence-level answer generation may be sufficient if the target application is question answering via a graphical user interface. On the other hand, if the goal is eventually to use QA to automatically populate fields in a relational database, more precise answers are required. For example, the field that stores the age of a person in a relational DB should contain a numeric value, not a sentence. Good performance in sentence-level answer generation can be achieved by using sophisticated IR techniques such as passage retrieval in conjunction with some shallow level NLP/IE processing (Kwok *et al.*, 2001; Clarke *et al.*, 2001). Phrase-level answer identification requires sophisticated NLP/IE; IR techniques alone are not sufficient for this task (Prager *et al.*, 1999; Grunfeld and Kwok, 2003). Many systems (e.g., Prager *et al.*, 1999; Clark *et al.*, 2001) take a two-stage approach. The first stage involves retrieving sentences or paragraphs in documents as candidate answer strings. The second stage focuses on extracting phrase-level exact answers from the candidate answer strings.

QA is an ideal test bed for demonstrating the value of IE. This chapter discusses a question answering system supported by increasingly sophisticated levels of information extraction. An integrated NLP and IE engine is used to extract named entities (NE) and their associations and to decode grammatical relationships. The impact of these different types of IE and NLP output on the performance of the QA system will be discussed. The system utilizes a novel approach by attempting to generate the most precise answer, backing-off to coarser levels, wherever necessary. The finest grained answer may be a single word or number, whereas the coarsest level of response is a URL representing an entire document. The focus is on extracting phrase-level exact answers, called answer-points, in situations where candidate answer strings have already been generated. In order to identify the answer-point, question and candidate answer strings are parsed by the same system to generate binary dependency structures for both specific Correlated Entity (CE) relationship links (Li *et al.*, 2003b) and generic grammatical relationships. The system searches for an answer-point by comparing the structural similarity involving the question-phrase and a candidate answer-phrase. Generic grammatical relationships are used as a back-off for specific CEs when the question goes beyond the scope of the specific associations or when the system fails to identify the answer-point which meets the specific CE constraints. These methods are particularly helpful in cases where the question-phrase does not correspond to a known named entity type and in cases where there are multiple candidate answer-points from which to select.

The rest of the paper is structured as follows: Section 2 presents the NLP/IE engine used, section 3 presents InfoXtract QA, a QA system based on the IE engine discussed in section 2, section 4 discusses NE-supported QA, and section 5 discusses QA based on relationships/associations between entities. Section 6 discusses extracting exact answers based on structural links. Finally, section 7 presents more recent work on caseless QA followed by section 8, some concluding remarks.

2. INFOXTRACT – AN INFORMATION EXTRACTION ENGINE

This section discusses *InfoXtract* (Srihari *et al.*, 2005), a domain-independent, modular, hierarchical IE engine that forms the basis of the QA system.

The last decade has seen significant advances in the area of IE. In the US, MUC (Chinchor and Marsh, 1998a) has been the driving force for developing this technology. MUC divides IE into distinct tasks, namely, Named Entity (NE), Template Element (TE: entity subtype, descriptor), Template Relation (TR: employee-of, location-of, product-of), Co-reference (CO), and Scenario Templates (ST) (Chinchor and Marsh, 1998a). NE focuses on the identification and classification of proper names (person, organization, location), time expressions (time, date) and numerical expressions such as money, percentage. TE and TR collect some information about entities and their inter-relationships. ST is designed to represent elaborate events in a specific domain (e.g., *Management Succession*). CO is not a direct IE task but provides fundamental discourse-level support for merging local IE results (e.g. linking *he* with its antecedent *John Smith*).

The most successful IE task thus far has been Named Entity (NE) tagging. The state-of-the-art exemplified by systems such as *NetOwl* (Krupka and Hausman, 1998), *IdentifiFinder* (Miller *et al.*, 1998) and *InfoXtract* (Srihari *et al.*, 2000a) have reached near human performance, with 90% or above F-measure. On the other hand, the deep level MUC IE task Scenario Template (ST) is designed to extract detailed information for predefined event scenarios of interest. It involves filling the slots of complicated templates. It is generally felt that this task is too ambitious for commercial application at present.

InfoXtract reflects an attempt to significantly expand and modify the MUC IE task definitions in order to make them more useful and more practical (Srihari *et al.*, 2003). The goal is to develop a domain independent IE engine with the capabilities for rapid domain portability and customization.

Figure 1 shows the overall system architecture involving all major modules of the *InfoXtract* IE engine. The IE results are stored in the IE Index. The core engine itself consists of two layers, the underlying layer involves a series of linguistic modules which provide NLP support to a corresponding series of kernel IE modules. The parsing and extraction results are stored in IE Index. The core engine has been kept as domain independent as possible; domain specialization or tuning can take place with minimum change to the core engine.

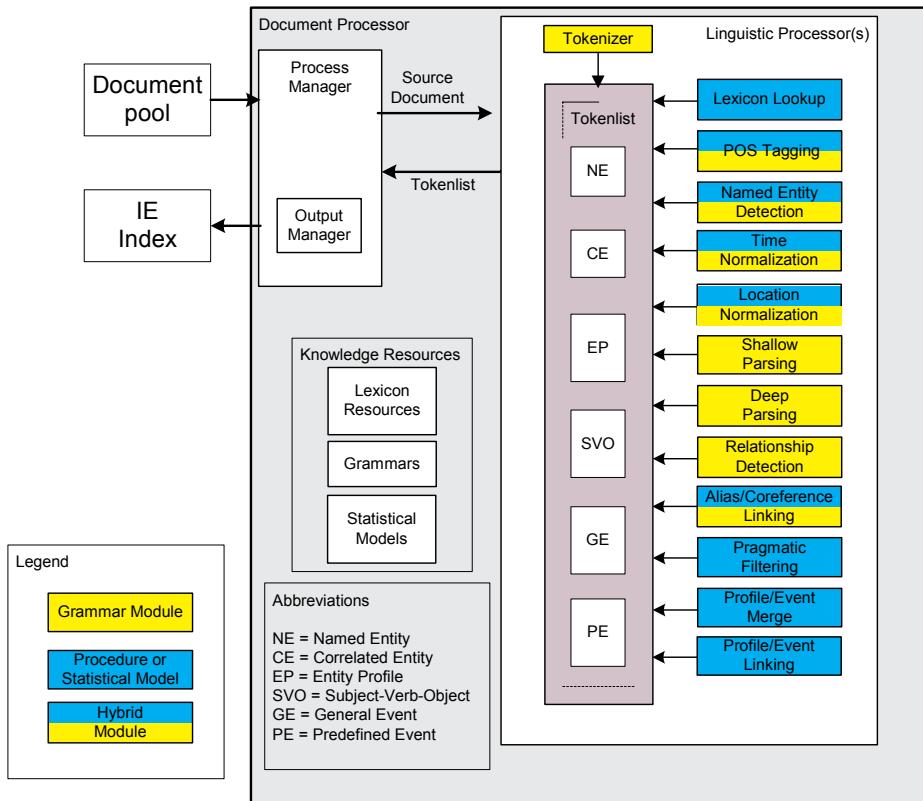


Figure 1: InfoXtract™: Domain independent, modular, hierarchical information extraction engine

2.1 Kernel IE Modules

As shown in Figure 1, the major categories of information objects produced by InfoXtract are defined to be Named Entity (NE), CE relationships, Entity Profile (EP), Subject-Verb-Object (SVO) triples, General Event (GE) and Predefined Event (PE). It is believed that these six types of information objects capture the key content of the processed text. A brief description of these information objects is given below.

- **Named Entity** (Srihari *et al.*, 2000a; Niu *et al.*, 2003a): this component identifies and classifies basic objects such as proper names of *person*, *organization*, *product*, *location*, contact information such as *address*, *email*, *phone number*, *URL*, time and numerical expressions such as *date*, *year* and various measurements such as *weight*, *money*, *percentage*, etc. This is the most

fundamental layer of any information extraction system and provides key support to the QA system. A subset of the NE hierarchy used in our system is illustrated below.

Person: woman, man

Organization: company, government, association, school, army, mass-media

Location: city, province, country, continent, ocean, lake, etc.

Time Expressions: hour, part-of-day, day-of-week, date, month, season, year, decade, century, duration

Numerical Expressions: percentage, money, number, weight, length, area, etc.

Contact expressions: email, address, telephone, etc.

- **Correlated Entity relationships** (Srihari *et al.*, 2003; Niu *et al.*, 2003b): For each entity type, there is a common set of *relationships* involving the entity in question that the system attempts to extract. For example, relationships associated with *person* include *address*, *age*, *education*, *where-employed*, etc. These relationships can be very specific, or they can be more generic such as descriptor phrases used to describe entities, e.g. *the leading software manufacturer*, in reference to Microsoft.
- **Entity Profile** (Li *et al.*, 2003b): An entity profile is an information object that encapsulates key information extracted pertaining to a given entity (*person*, *organization*, *location*, etc.) This includes consolidated local CE relationships as well as links to events (see below) that the entity is involved in.
- **Subject-Verb-Object:** These are sentential parsing results at the logical level: they are the local basis for general events, also called local events or simple events. They are represented as binary dependency links between the basic linguistic units. Both active sentences and passive sentences are parsed into the same underlying links at logical level.
- **General Events:** General events attempt to capture information pertaining to *who did what, when and where?* This is essential in information discovery tasks, where the target of extraction is not well defined. In many cases, it is what is not known beforehand that is most important. General events attempt to extract key event information from running text. Advanced NLP including co-reference, local event merging, and temporal and spatial normalization are also utilized in generating richer output.
- **Predefined Events:** In cases where the domain is well defined, and events of interest can be specified clearly, the engine can be customized to generate specialized event templates referred to as predefined events. Whereas general events represent generic information involving *who did what when and where*, predefined events permit more semantic slots in the event template. For example, an *executive change* event includes the name of the person(s) entering or leaving a role, the company in question, date, and any other relevant information.

2.2 Hybrid Model

InfoXtract reflects a hybrid model of information extraction, combining both grammatical and statistical approaches. Achieving the right balance of these two paradigms has been a major achievement of InfoXtract. We found that IE tasks that primarily call for classification, such as named entity tagging, benefit most from statistical approaches. IE modules that attempt to decode inherent grammatical structures present in surface text benefit most from grammar-based approaches. Unlike other IE systems that tend to favor one or the other approach, InfoXtract effectively combines both.

The grammar formalism used by earlier versions of InfoXtract was finite state transducers (FST). FSTs have traditionally been used in NLP applications for lexical look-up due to speed and efficiency. The FASTUS system developed at SRI (Hobbs, 1993) represented an effort to use cascaded FSTs for grammar development beyond lexical look-up. Each level of the cascade utilizes the results of the previous levels in order to achieve more sophisticated parsing; the *token list* is the data structure that facilitates this pipelined processing. A token list is a hierarchical data structure representing the processed document to the current stage in the cascade; token lists facilitate multi-level processing and propagation of information. The current version of InfoXtract uses an extension of *tree walking automata* instead of FSTs as the underlying automata: grammars are essentially token list transducers. The change has been more or less transparent to grammar writers and was initiated due to the need to support additional features necessary for practical NLP development. A more thorough description of the NLP toolkit is beyond the scope of this chapter. It suffices to say that the NLP toolkit is a rich environment that has been tailored for information extraction, facilitating both shallow and semantic parsing, discourse-level processing, as well as multiple levels of IE.

Statistical techniques are used in several InfoXtract modules, but are primarily used in the named entity tagger. There has been a significant body of work in machine-learning for IE tasks such as relationship and event detection. Most existing efforts in IE are highly domain-specific. In addition, most statistical IE approaches are based on supervised learning. They are not always practical since they require large annotated corpora. InfoXtract takes a different approach: it emphasizes the use of domain-independent grammatical processing to decode underlying linguistic structure. Subsequently, unsupervised machine-learning techniques can be employed to “learn” new categories of entities and relationships on a specific domain or corpus; this eliminates the need for large, annotated corpora (Niu *et al.*, 2003a; 2004b). The output of the InfoXtract engine has been used as input to machine-learning algorithms that are focused on automatic lexical knowledge acquisition for a new domain. These automatically learned, domain specific lexical knowledge resources are subsequently used to achieve enhanced IE performance in the new domain.

2.3 Linguistic Modules

The linguistic modules serve as underlying support system for different levels of IE. This support system involves almost all major linguistic areas: orthography, morphology, syntax, semantics, discourse and pragmatics. A brief description of the linguistic modules is given below.

- **Preprocessing:** This component handles file format converting, text zoning and tokenization. The task of text zoning is performed to identify and distinguish the meta data such as title, author, etc from normal running text. The task of tokenization is to convert the incoming linear string of characters from the normal text part of the given format into a *token list*; this forms the basis for subsequent linguistic processing.
- **Word Analysis:** This component includes word-level **orthographical analysis** (capitalization, symbol combination, etc.) and **morphological analysis** such as stemming. It also includes part-of-speech (POS) tagging which distinguishes, e.g., a noun from a verb based on contextual clues. An optional HMM-based *Case Restoration* module is called when performing case insensitive QA (Li *et al.*, 2003a).
- **Phrase Analysis:** This component, also called *shallow parsing*, undertakes basic **syntactic analysis** and establishes simple, un-embedded linguistic structures such as basic noun phrases (NP), verb groups (VG), and basic prepositional phrases (PP). This is a key linguistic module to provide building blocks to enable various dependency linkage among them in further processing.
- **Sentence Analysis:** This component, also called *semantic parsing*, decodes underlying dependency trees that embody logical relationships such as V-S (verb-subject), V-O (verb-object), H-M (head-modifier). The *InfoXtract* semantic parser transforms various patterns, such as active patterns and passive patterns, into the same *logical form*, with the *argument structure* at its core. This involves a considerable amount of **semantic analysis**. The decoded structures are crucial for supporting structure-based grammar development and/or structure-based machine learning for relationship and event extraction.
- **Discourse Analysis:** This component studies the structure across sentence boundaries. One key task for discourse analysis is to decode the co-reference (CO) links of pronouns (*he, she, it*, etc) and other anaphor (*this company, that lady*) with the antecedent named entities. A special type of CO task is ‘Alias Association’ which will link *International Business Machine* with *IBM* and *Bill Clinton* with *William Clinton*. The results support information merging and consolidation for profiles and events.
- **Pragmatic Analysis:** This component distinguishes *important, relevant* information from *unimportant, irrelevant* information based on lexical resources, structural patterns and contextual clues.

2.4 Semantic Parsing in QA

The Semantic Parsing module is of particular importance in supporting QA. It decodes the grammatical dependency relationships: V-S, V-O, V-C (Verb-Complement), H-M of time, location, reason, manner, purpose, result, etc. This module extends the shallow parsing module through the use of a cascade of handcrafted pattern matching rules. During semantic parsing, not only passive patterns are decoded into the same underlying structures as active patterns, but also structures for verbs such as *acquire* and for de-verbal nouns such as *acquisition* lead to the same dependency links, as shown below.

AOL acquired Netscape in 1998. →

V-S: acquired ← AOL
 V-O: acquired ← Netscape
 H-M: acquired ← in 1998 (time-modifier)

Netscape was acquired by AOL in 1998. →

V-S: was acquired ← by AOL
 V-O: was acquired ← Netscape
 H-M: was acquired ← in 1998 (time-modifier)

the acquisition of Netscape by AOL in 1998... →

V-S: acquisition ← by AOL
 V-O: acquisition ← of Netscape
 H-M: acquired ← in 1998 (time-modifier)

These links can be used as structural support to answer questions like *Who acquired Netscape* or *which company was acquired by AOL?*

InfoXtract semantic parsing is illustrated by the two sample pattern matching rules, as shown below.

V_S_active Rule :=

Pattern:

\$1: NP
 (PP|RB)*
 \$2: VG, active, finite

Action:

LINK: V_S(\$2, \$1)

V6A_transitive_passive Rule :=

Pattern:

\$1: NP
 (PP|RB)*
 \$2: VG, V6A, passive, finite
 (\$3: PP P='by')

Action:

LINK: V_S(\$2, \$3)
 LINK: V_O(\$2, \$1)

Note that a rule in the InfoXtract grammar formalism consists of two parts: a pattern and an action part. Patterns are regular expressions over token lists with parentheses for optionality, kleene star for matching any number of times, a bar for logical OR, a quoted string for checking a word literal and unquoted symbols for checking the token features (e.g. NP for Noun Phrase feature, PP for Prepositional Phrase feature, AP for Adjective Phrase feature, and RB for the part-of-speech feature adverb). The identifier for a token in a pattern is represented by $\$n$ so that dependency link can be established between units. Once a pattern is matched, the *Action* part can perform a list of operations, including establishing binary dependency links using LINK, constructing a new phrase, assigning or deleting the token features, and updating the canonical form for a token.

The *V_S_active Rule* is able to cover cases like NP[*Robert Jackson*] originally/RB PP[from NE[*Washington, D.C.*]] VG[has recently moved]/finite/active/perfect/present here. In the above example, the adverb (with the POS tag RB) *originally* and the prepositional phrase (PP) *from Washington, D.C.* are ignored in pattern matching in order to produce a dependency link of the V_S relationship from *has recently moved* and *Robert Jackson*. Note that the Verb Group (VG) tags for tense, aspect, voice, etc. such as *finite, active* are assigned in the shallow parsing process when the VG units are constructed.

Similarly, the *V6A_transitive_passive Rule* can parse a large portion of the active patterns of the English transitive verb for which we adopted the lexical sub-categorization code *V6A* from Oxford Advanced Learners Dictionary. The optional *by-phrase* is recognized as the logical subject of the VG while the syntactic subject NP is recognized as its logical object. As shown, InfoXtract's semantic parser goes one step further than parsers that simply decode syntactic relationships. It consumes surface structure variations to permit composition of structural similarity at the logical level. However, compared with CE associations such as *Affiliation*, *Position* relationships for a person entity and predefined events such as *Management Succession*, *Product Launch* which are at the deep semantic level, the logical SVO (Subject-Verb-Object) structures still cannot capture semantic relations which are expressed using different head verbs with different structures. An example is the pair: *X borrows Y from Z* versus *Z lends Y to X*.

2.5 Lexical Resources

The InfoXtract engine uses various lexical resources including the following:

- General English dictionaries available in electronic form providing basic syntactic information
- Specialized glossaries for location names, organization names, etc.

- Specialized semantic dictionaries reflecting words that denote *person*, *organization*, etc. For example, *doctor* corresponds to person, *church* corresponds to organization. This is especially useful in QA
- Statistical language models for Named Entity tagging (retrainable for new domains)

2.6 Architecture

The InfoXtract engine has been developed on a Linux/Unix platform and is capable of processing up to 20 MB per hour on a single processor. The system has been tested on very large document collections. Benchmarks for precision and recall for major tasks range from greater than 90% for lower level modules such as NE, POS tagging and Shallow Parsing to 70%–80% for later modules such as entity profile, event, co-reference excluding word sense disambiguation which is still under development.

Finally, the architecture facilitates scalability: a process manager can distribute requests to handle documents over several CPUs. The system has already been deployed with a set of eight processors working simultaneously.

3. INFOXTRACT-QA SYSTEM

Figure 2 illustrates the architecture of the InfoXtract-QA system. The InfoXtract-QA system consists of three components: (i) Question Processing; (ii) Text Processing; and (iii) Answer Ranking. The question processor produces the analysis of *asking point*, a structure encoding the semantics of the question. Both text and answer are sent to the *InfoXtract* engine for processing. This enables the processing results to be compared when matching the question to candidate answers. Besides InfoXtract processing, the question processing component and the text processing component each have their own additional processing: keyword indexing is required in the text processing component and for question processing, a special module for Asking Point Identification is particularly important.

Linking the two processing components is the Answer Ranking component that consists of three modules: Snippet Extraction, Feature Ranking and Answer Point Identification. Answer Ranking relies on access to the information from both the Keyword Index as well as the IE Index generated by InfoXtract to rank all levels of answers. At the simplest level, answer ranking is based on keywords, plus the NE type and their common location within a same sentence; this results in a set of candidate answer snippets. More sophisticated matching calls on the answer point identification module.

Snippet Extraction generates the top n (we chose 200) most relevant sentence-level candidate answer snippets based on the results of Asking Point Identification and the keywords (after removing stop words) in the question. This provides the basis for further Feature Ranking. The purpose of Feature Ranking is to re-rank the candidate snippets based on a list of ranking features. The Answer Point Identification module is intended for extracting phrase-level, “exact” answers based

on sophisticated linguistic analysis: without it, a QA system can still perform snippet-level answers, typically one sentence or a short paragraph, by providing ranked snippets as candidate answers to the user.

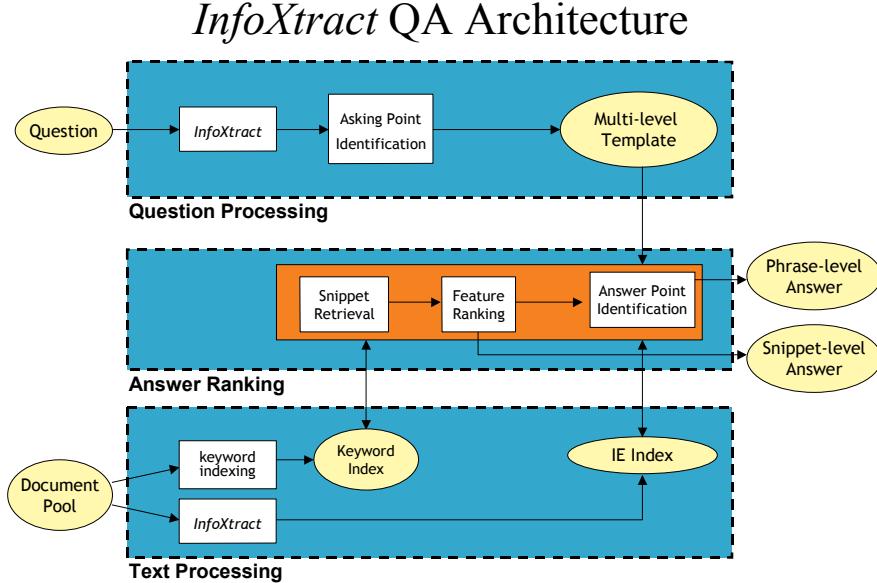


Figure 2: InfoXtract QA System Architecture

3.1 Asking Point Link Identification

Asking point link identification is a crucial step in a QA system. It provides the necessary information decoded from question processing for a system to locate the corresponding answer-points from candidate answer strings. The Asking-point (Link) Identification Module is responsible for parsing *wh*-phrases in context into three categories: NE *Asking-point*, *Asking-point CE Link* and *Asking-point Grammar Link*. *Asking Point* refers to the question phrases along with constraints that a corresponding answer-point should satisfy. *Asking-point Link* is the decoded binary relationship from the asking point to another unit in the question. Identification of the NE asking point is essentially mapping the *wh*-phrase to the NE types or subtypes. For example, *which year* is mapped to [which year]/NeYear, *how old* mapped to [how old]/NeAge, and *how long* mapped to [how long]/NeLength or [how long]/NeDuration, etc. This was the approach used in Cymfony's TREC-8 experiments; results are reported in section 4.

The Asking-point CE Link encodes information regarding predefined associations/relationships that are referred to in the question. For Asking-point CE Link, the module needs to identify the involved entity and the association in question. For example, the Asking-point CE Link for *How old is John Smith* is the AGE relationship of the NePerson *John Smith*, represented as *AGE: John Smith ← [how old]/NeAge*. In general, this type of asking-point leads to the most specific answer. If a correspondence can be made between IE-based associations/relationships and relational tables in a relational database, this type of asking point can facilitate incorporation of structured data into the QA process.

The *wh*-phrases, which may or may not be mapped to NE asking points and whose dependency links are beyond predefined associations lead to Asking-point Grammar Links, e.g. *How did Julian Hill discover nylon?* This asking-point link is represented as *H-M: discover ← [How]/manner-modifier*. An asking-point grammar link only involves generic grammatical constraints: for this example, the constraints a candidate answer-point must satisfy are (i) an H-M link with ‘discover’ as head and (ii) a phrase which must be a modifier of manner.

These three types of asking points and their possible links form a natural hierarchy that can be used to facilitate the back-off strategy for the answer-point extraction module:

Asking-point CE Link → Asking-point Grammar Link → NE Asking Point.

This hierarchy defines the sequence of matching steps that should be followed during the answer-point extraction. The back-off from Asking-point CE Link to Asking-point Grammar Link is necessary since the latter represents more generic, structural constraints than the former. For example, in the sentence *where is IBM located*, the Asking-point CE Link is *LOCATION: IBM ← [where]/NElocation* while the default Grammar Link is *H-M: located ← [where]/location-modifier*. When the specific association constraints cannot be satisfied, the system should attempt to locate an answer-point by searching for a location-modifier of the key verb ‘located’.

The NE asking point constraints are also marked for asking-point CE links and those asking-point grammar links whose *wh*-phrases can be mapped to NE asking points. Backing off to the NE asking point is required in cases where the asking-point association constraints and grammatical structural constraints cannot be satisfied. For *How old is John Smith*, the asking-point grammar link is represented as *H-M: John Smith ← [how old]/NeAge*. If the system cannot find a corresponding AGE association or a modifier of NeAge for the entity *John Smith* to satisfy the structural constraints, it will attempt to locate a candidate answer-point by enforcing the NeAge constraint. When there is only one NeAge in the answer string, the system can extract it as the only possible answer-point even if the structural constraints are not available.

Since QA based on NE asking point forms the basis of much of the QA system, it is discussed in detail here with examples drawn from the TREC-8 data set. For example,

- [1] *Who won the 1998 Nobel Peace Prize?*

contains the following keywords: *won*, *1998*, *Nobel*, *Peace*, *Prize*. The asking point *Who* refers to the NE type *person*. The output before question expansion is a simple 2-feature template as shown below:

- [2] Asking-point: PERSON

Key-words: {won, 1998, Nobel, Peace, Prize}

Basic-units: {NP[Who], VG[won], NP[the 1998 Nobel Peace Prize]/NeAward}

The following is an example where the asking point does not correspond to any type of NE in our definition.

- [3] *Why did David Koresh ask the FBI for a word processor?*

The system then maps it to the following question template:

- [4] Asking-point: REASON

Key-words: {ask, David, Koresh, FBI, word, processor }

Basic-units: {Why, VG1[did], NP[David Koresh]/NePerson, VG2[ask], NP[the FBI]/NeGovernment, PP[for a word processor]}

The question processor scans the question to search for question words (*wh*-words) and maps them into corresponding NE types/sub-types or pre-defined notions like REASON.

We adopt two sets of pattern matching rules for this purpose: (i) structure based pattern matching rules; and (ii) simple key word based pattern matching rules (regarded as default rules). It is fairly easy to exhaust the second set of rules since interrogative question words/phrases from a closed set. In comparison, the first set of rules are continuously being fine-tuned and expanded. This strategy of using two sets of rules leads to the robustness of the question processor.

The first set of rules is based on shallow parsing results of the questions, using the Shallow Parser in InfoXtract. This parser identifies basic syntactic constructions like *BaseNP* (Basic Noun Phrase), *BasePP* (Basic Prepositional Phrase) and *VG* (Verb Group).

The following is a sample of the first set of rules:

- [6] Name NP (city | country | company) → CITY|COUNTRY|COMPANY
- [7] Name NP(person_w) → PERSON
- [8] Name NP(org_w) → ORGANIZATION
- [9] Name NP(NOT person_w, NOT org_w) → NAME

Rule [6] checks the head word of the NP. It includes cases like *VG[Name] NP[a country]* that *VG[is developing] NP[a magnetic levitation railway system]*. Rule [7] works for cases like *VG[Name] NP[the first private citizen]* *VG[to fly] PP[in space]*

as *citizen* belongs to the word class *person_w*. Rule [9] is a catch-all rule: if the NP is not of class person (*person_w*) or organization (*org_w*), then the asking point is a proper name (default NE), often realized in English in a capitalized string of words. Examples include *Name a film that has won the Golden Bear in the Berlin Film Festival*. The word lists *org_w* and *person_w* are currently manually maintained based on inspection of large volumes of text. An effort is underway to automate the learning of such word lists by utilizing machine learning techniques.

The following pattern transformations were used to expand the rule set:

(Please) name NP[X]
 → what/which Aux(be) (the name of) NP[X]
 → NP(what/which...X)

In other words, the four rules are expanded to 12 rules. For example, Rule [10] below corresponds to Rule [6]; Rule [11] is derived from Rule [7].

- [10] what/which Aux(be) NP (city | country | company) → CITY | COUNTRY | COMPANY
- [11] NP(what/which ... person_w) → PERSON

Rule [10] extracts the asking point from cases like *NP[What] Aux[is] NP[the largest country] PP[in the world]*. Rule [11] covers questions such as: *NP[What costume designer] VG[decided] that NP[Michael Jackson] VG[should only wear] NP[one glove], NP[Which former Ku Klux Klan member] VG[won] NP[an elected office] PP[in the U.S.]*.

As illustrated, shallow parsing helps to capture a variety of natural language question expressions. However, there are cases where simple key word based pattern matching is sufficient to capture the asking point. These require a second set of rules, that are used when the first set of rules fail to produce results. The following is a sample of such rules:

- [12] who/whom → PERSON
- [13] when → TIME/DATE
- [14] where/what place → LOCATION
- [15] what day (of the week) → DAY
- [16] how old → AGE
- [17] what → NAME
- [18] how rich → MONEY
- [19] how often → FREQUENCY
- [21] how many → NUMBER
- [22] how long → LENGTH/DURATION
- [23] why/for what → REASON

In the stage of question expansion, the template in [4] would be expanded to the template shown in [24]:

- [24] asking-point: {because | because of | due to | thanks to | since | in order to VB}
 key-words: {ask | asks | asked | asking, David, Koresh, FBI, word, processor}

The last item in the *asking-point* list attempts to find an infinitive by checking the word *to* followed by a verb (with the part-of-speech tag VB). Infinitive verb phrases are often used in English to explain a reason for some action.

3.2 Snippet Retrieval

Snippet retrieval generates the top m relevant sentence-level candidate answer snippets based on the question processing results and the top n relevant documents returned by the search engine. The InfoXtract-QA system uses a search engine based on the vector-space model. We use two types of evidence for snippet retrieval: (i) keyword occurrence statistics at the snippet level (with stop words removed), and (ii) the IE results, including NE Asking Points, Asking Point CE Link, head word of a phrase, etc.

If the Question Processing component detects an Asking Point CE Link, the system first attempts to retrieve snippets that contain the corresponding CE link. If it fails, it backs off to the corresponding NE Asking Point. This serves as a filter in the sense that only the snippets that contain at least one NE that matches the NE Asking Point (in addition to the keywords) are extracted. For questions that do not contain NE Asking Points, the system backs off to the keyword-based snippet retrieval.

In order to retrieve the top m snippets as candidate answers, a weighted-sum ranking model is used. The weight of each keyword in a snippet depends on its lexical features (e.g. NE tags, head-word feature) and keyword occurrence statistics. The distribution of the keywords in the snippets is used in adjusting the weights determined by lexical features (Equation 1).

$$w_t = \log(c * \frac{N}{n} + 1) * w_{lf} \quad (1)$$

where c is a constant, N is the total number of snippets retrieved, n is the number of times the keyword t is matched and w_{lf} is the weight assigned by lexical features.

A synonym lexicon is also constructed for query expansion to help snippet retrieval. This includes irregular verbs (*go/went/gone*, etc.), verb-noun conversion (*develop/development; satisfy/satisfaction*; etc.), and a manually edited conservative synonym list (e.g. *adjust/adapt; adjudicate/judge*; etc.).

Factors that contribute to relevancy weighting in snippet retrieval include giving more weight to the head words of phrases (e.g. ‘disaster’ in the noun phrase ‘the costliest disaster’), more weight to words that are linked with question words (e.g. ‘calories’ in ‘How many calories...’ and ‘American’ in ‘Who was the first American in space’), and discounting the weight for synonym matching. For candidate NE answering points that match the type of the NE Asking Points, weights are calculated depending on the numbers of keywords that appear in the question: the

more terms appear in a question, the more weight is given to the candidate NE answering points. The score of a snippet is assigned by the summation of the weights of all the matched terms and candidate NE answer points. The top m snippets are retrieved for each question based on its score to feed the subsequent module Feature Ranking.

3.3 Feature Ranking

The purpose of Feature Ranking is to re-rank the candidate snippets based on a list of ranking features. Given the question templates as results of question processing and a list of top n candidate answer snippets retrieved in the previous stage, the Feature Ranking module uses a set of re-ranking features to fine-tune relevancy measures of the initial list of snippets in order to generate the final, top five output answer strings.

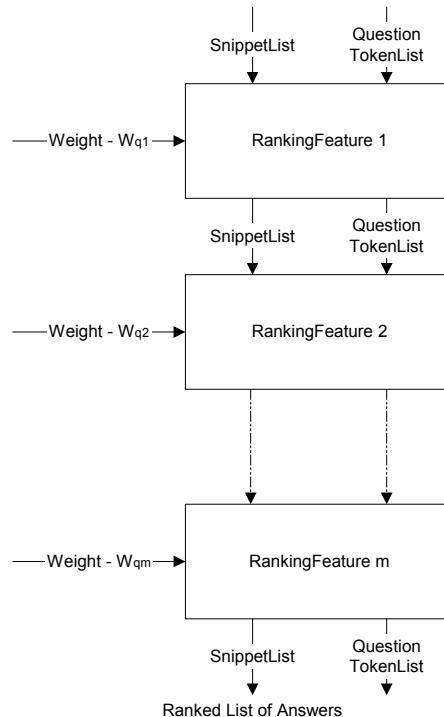


Figure 3: Pipeline for Ranking Features

Figure 3 gives the ranking model designed for the Feature Ranking module. For a given question, Q , let $\{S_1, S_2, \dots, S_n\}$ be the set of candidate answer snippets. Let $\{R_1, R_2, \dots, R_k\}$ be the ranking features. For a snippet S_j , let the ranking feature R_i

assign a relevancy of r_{ij} quantifying the snippet's relevance to the question. The ranking model is given by

$$R(Q, S_j) = \sum_{i=1}^k w_{il} r_{ij} \quad (2)$$

where l represents the question type of Q and w_{il} gives the weight assigned to the ranking feature. The weights w_{il} vary based on question type.

We use both traditional IR ranking features such as Keyword Proximity, Inverse Document Frequency (IDF), Query Expansion (based on a conservative synonym lexicon), as well as the sophisticated ranking features supported by InfoXtract, listed below:

- NE Asking Point supported by NE
- Asking-point CE Link supported by CE
- Asking-point Grammar Link supported by semantic parsing
- Headword Matching for Basic Phrases supported by Shallow Parsing (e.g. NP[an interesting book] vs. NP[an excellent book])
- Alias link (e.g. ‘IBM’ and ‘International Business Machine’) supported by Co-reference³⁹
- NE Hierarchy (e.g. *Company* vs. *Organization*, see Table 1) supported by NE
- Structure based matching (SVO Links, Head-Modifier Link, etc.) supported by semantic parsing

For example, for the feature NE Hierarchy, Table 1 shows the final weights given to different scenarios of matching the NE asking point with the NE in the candidate answer snippets.

Table 1: NE Hierarchy Feature

Asking point	NE in Snippets	Weight
NePerson	NeMan	1.0
NeMan	NePerson	0.8
NeCompany	NeOrganization	0.7
NeShare	NeMoney	0.6
.....		

Based on this table, if a question asks for a person (e.g., *Who is the president of Stanford*), and the system successfully finds an named entity of man, then it receives the full weight 1. On the contrary, if the asking point is a man and the snippet contains a person entity, the system gives discounts the weight from 1 to 0.8.

³⁹ For the more comprehensive use of co-reference in QA, see (Vicedo and Ferrández, this volume).

The results of the Feature Ranking module are further processed by the Answer Point Identification module in order to generate phrase-level, exact answers. If this effort is unsuccessful, the system will back off to snippet level answers decided by the ranking features.

3.4 Answer Point Identification

Answer-point identification is accomplished through matching the asking-point to candidate answer-points using the following back-off algorithm based on the processing results of the question and sentence-level candidate answer string.

1. if there is Asking-point CE Link, call *Match(asking-point CE link, candidate answer-point CE link)* to search for the corresponding association to locate answer-point
2. if step (1) fails and there is an asking-point grammar link, call *Match(asking-point grammar link, candidate answer-point grammar link)* to search for the corresponding grammar link to locate the answer-point
3. if step (2) fails and there is an NE asking point, search for the corresponding NEs: if there is only one corresponding NE, then extract this as the answer-point else mark all corresponding NEs as candidate answer-points

The function *Match(asking-point link, candidate answer-point link)* is defined as (i) exact match or synonym match of the related units (synonym match currently confined to verb vs. de-verbal noun); (ii) match the relation type directly (e.g., V-S matches V-S, AGE matches AGE, etc.); and (iii) match the type of asking point and answer point using a predetermined ontology. Examples of the third category include NePerson asking point matches NePerson and its sub-types NeMan and NeWoman; ‘how’ matches manner-modifier; etc., either through direct link or indirect link based on conjunctive link (ConjLink) or equivalence link (S-P, subject-predicative or appositive relations between two NPs).

Step (1) and Step (2) attempt to leverage structural support from parsing and high-level information extraction beyond NE. In our experiments, the structural support used for answer-point identification checks the binary links involving the asking point and the candidate answer points, instead of full template matching as proposed in (Srihari and Li, 2000b).

4. NE-SUPPORTED QA

This section presents the QA system based on Named Entity tagging. Out of the 200 questions that comprised the TREC-8 QA track competition, over 80% asked for an NE, e.g. *who* (PERSON), *when* (TIME | DATE), *where* (LOCATION), *how far* (LENGTH). Therefore, the NE tagger has proved to be very helpful. The NE of the targeted type is necessary but not complete in answering such questions because NE by nature only extracts isolated individual entities from the text. Nevertheless, in most cases, the QA system was able to extract text snippets which contained correct

answers in the top five list using even crude methods like “the nearest NE to the queried key words” or “the NE and its related key words within the same line or paragraph”.

Although the goal of advanced QA is to extract precise answers, there is a need for coarser granularity techniques as well. In situations where there is an overwhelming amount of information to digest, such a technique can reduce the time overhead involved in reading entire documents. The key to building a successful system based on snippet-level QA is a very intuitive user interface, which permits various types of drill down, as well as traditional search engine results. The other benefit of such an approach is in the limited processing of the document collection: only NE tagging and shallow parsing are required; both are very fast and scalable processes. Finally, by interfacing such a system to a search engine that obtains the initial document set relevant to the question, it is possible to adapt QA to the Internet. Redundancy of information on the Internet can be automatically exploited, whereby the correct answer will appear in multiple documents, thus leading to higher precision QA.

In the TREC-8 QA track, InfoXtract-QA accuracy had a Mean Reciprocal Response (MRR) score of 66.0% (Voorhees, 1999), based on NE and shallow parsing support. After semantic parsing results (SVO, Equivalence Relationships and modifier relationships) are added to the ranking module, proximity ranking feature refined, and support from the extracted CE relationships (e.g. affiliation, position, age, etc) incorporated, the benchmarking on the TREC-8 data was improved to 79%.

5. CE-SUPPORTED QA

The CE module correlates named entities by associating them with other entities or phrases. These are specific, predefined relationships for entities of person and organization. Currently, InfoXtract can extract the following CEs with high precision (over 90%) and modest recall ranging from 50% to 80% depending on the size of grammars written for each specific association.

Person: affiliation, position, age, spouse, birth-place, birth-time, etc.

Organization: location, staff, head, products, found-time, founder, etc.

CEs are semantic structures very useful in supporting QA. For example, from the phrase *Grover Cleveland, who in June 1886 married 21-year-old Frances Folsom*, the IE engine can identify the following associations:

Spouse: Grover Cleveland ← Frances Folsom

Spouse: Frances ← Grover Cleveland

Age: Frances Folsom ← 21-year-old

A question asking about such an association, e.g., Q11: *Who was President Cleveland's wife*, will be parsed into the following CE link between a question-

phrase ‘Who’ and the entity ‘Cleveland’ (see Section 2): *Spouse: Cleveland → Who*. The semantic similarity between this structure and the structure *Spouse: Grover Cleveland → Frances Folsom* can determine the answer point to be ‘Frances Folsom’.

More recently, Ravichandran and Hovy (2002) have used the concept of extraction CE patterns for a web-based question answering system. They exploit the fact that (i) many CE relationships are typically expressed in set patterns, e.g. birth date and year of death expressed as *Person-Name* (1892-1984), and (ii) there is a redundancy of information on the internet, such that the desired CE relationship will be seen at least once in a known extraction pattern. Although it is not possible to determine well-defined extraction patterns for all possible relationships, sufficient numbers can be defined to make this a viable component of a QA system.

6. EXPERIMENTS ON ANSWER POINT IDENTIFICATION

QA based on SVO has been the recent focus of our work on question answering. Full template matching is best exemplified by the following example. If the incoming question is *Who won the Nobel Prize in 1991*, and the candidate answer string is *John Smith won the Nobel Prize in 1991*, the question template and answer template are shown below:

<u>Win</u>	V-S: NePerson [Who]
	V-O: NP [the Nobel Prize]
	H-M: NeYear [1991]
<u>Win</u>	V-S: NePerson [John Smith]
	V-O: NP [the Nobel Prize]
	H-M: NeYear [1991]

The template matching will match the asking point *Who* with the answer point *John Smith* because, for all the dependency links in the trees, the information is compatible and, in this case, an exact match. This is the ideal case of full template matching and guarantees the highest precision of answers.

In practice, full template matching is neither realistic for most cases nor necessary for extracting answer points in a two-stage approach. It is not realistic because natural language semantic parsing is such a challenging problem that a perfect dependency tree (or full template) which links together every linguistic unit is not always easy to generate. For InfoXtract, the majority, but not all, of the decoded binary dependency links are accurate in most cases, as shown in the benchmarks above. In such situations, satisfying every dependency link of a template tree is too strong a condition to meet. On the other hand, it is not necessary to check all the links in the dependency trees for full template matching. With the modular design and work division between the sentence level candidate answer string generation module (Stage One) and answer-point extraction from the candidate answer strings (Stage Two), all candidate answer strings are determined by previous modules as highly relevant. In this situation, a simplified partial

template matching, namely, ‘asking/answer point binary relation matching’, will be sufficient to select the answer-point, if present, from the candidate answer string. In other words, the system only needs to check this one dependency link in extracting the answer-point. For the previous example, only the asking/answer point binary dependency links need to be matched as illustrated below:

V-S win \leftarrow [Who]/NePerson
V-S win \leftarrow [John Smith]/NeMan

Some sample results are given below to illustrate how answer-points are identified based on matching binary relations involving asking/answer points.

In order to conduct the feasibility study on the proposed method, we selected a pool of 200 questions consisting of the first 100 questions from both TREC-8 and TREC-9 QA tracks. This experiment identifies phrase-level answer points from the corresponding *top-ranked* answer sentences. Stage One processing for generating candidate answer sentences was accomplished using the NE and shallow-parsing based ranking module as described in Section 3.3. Stage Two processing for answer-point identification was accomplished using the algorithm described in Section 3.4.

Table 2: Experiment Results for TREC-8 ad TREC-9

	TREC-8				TREC-9			
	CE Links	Grammar Links	NE Points (Baseline)	Overall	CE Links	Grammar Links	NE Points (Baseline)	Overall
Detected	9	44	76	86	7	23	48	64
Correct	8	39	41	71	6	21	25	52
Fail	1	6	35	14	1	2	23	12
Precision	89%	89%	54%	83%	85.7%	91.3%	52.1%	81.3%
Recall	8%	39%	41%	71%	5%	21%	25%	52%

As shown in Table 2, out of the 100 question-answer pairs we selected from TREC-8, nine have detected CE links involving asking/answer points, and 44 are found to have grammar links involving asking/answer points. For NE asking points, 76 TREC-8 questions were identified to require some type of NE as answers. Assuming that a baseline answer-point identification system only uses NE asking points as constraints, of the 76 questions requiring NEs as answers, 41 answer-points were identified successfully because there was only one NE in the answer string which matched the required NE type. The failed cases in matching NE asking point constraints include two situations: (i) no NE exists in the answer string; and (ii) multiple NEs satisfy the type constraints of NE asking points (i.e. more than one candidate answer-points found from the answer string) or there is type conflict during the matching of NE asking/answer points. Therefore, the baseline system would achieve 54% precision and 41% recall based on the standard precision and recall formulas:

Precision = Correct / Detected

Recall = Correct / Relevant.

In comparison, in the answer-point identification system which leverages structural support from both CE links and grammar links as well as NE asking points, both the precision and recall are raised: from the baseline 54% to 83% for precision and from 41% to 71% for recall. The significant improvement in precision and recall is attributed to the performance of structural matching in identifying exact answers. This demonstrates the benefits of using advanced NLP/IE technology, beyond NE and shallow parsing.

The experiment on TREC-9 questions was performed using the same procedure. The improvements were not as good, especially for recall. The performance degradation seems to be related to the difference in the preparation of the question pools between these two tracks. The TREC-8 questions were artificially formed from text in the corpus. This increases the chances of producing identical or similar parsing links from the questions and their corresponding candidate answer snippets. The TREC-9 questions were reported to be collected from real life query logs. So it is a more challenging task. Nevertheless, due to the multi-level answer point identification strategy and the use of fuzzy matching techniques such as synonym matching, NE ontology matching, and indirect link matching, our experiment shows that this approach can lead to performance among the best in TREC-9.

Using grammar links alone, exact answers were identified for 39 out of the 44 candidate answer-points satisfying the types of grammar links in 100 TREC-8 cases. During matching, six cases failed either due to the parsing error or due to the type conflict between the asking/answer points (e.g. violating the type constraints such as *manner-modifier* on the answer-point for ‘how’ question). The high precision and modest recall in using the grammar constraints are understandable since the grammar links impose very strong constraints on both the nodes and the structural type. The high precision performance indicates that grammar links not only have the discriminating ability to identify exact answers in the presence of multiple NE options, but also recognize answers in the absence of asking point types. Even stronger structural support comes from the semantic relations decoded by the CE extraction module. In this case, the performance is naturally high-precision (89%), low-recall (8%) as predefined CE links are by nature more sparse than generic grammatical relations.

We illustrate below, some examples from the TREC-8 QA task on how the match function identified in Section 3 applies to different question types.

Q4: *How much did Mercury spend on advertising in 1993?* →

asking-point grammar link: V-O spend ← [How much]/NeMoney

A: *Last year the company spent Pounds 12m on advertising.* →

candidate answer-point grammar link: V-O spent ← [Pounds 12m]/NeMoney

Answer-point Output: Pounds 12m

This case requires (i) an exact match in its original verb form between *spend* and *spent*; (ii) a V-O type match; and (iii) an asking/answer point type NeMoney match through direct link.

- Q63: *What nuclear-powered Russian submarine sank in the Norwegian Sea on April 7, 1989?* →
 asking-point grammar link: H-M submarine ← [What]
 A: *NEZAVISIMAYA GAZETA* on the Komsomolets nuclear-powered submarine which sank in the Norwegian Sea five years ago: →
 candidate answer-point grammar link: H-M submarine ← Komsomolets
 Answer-point Output: Komsomolets

This case requires (i) an exact match of *submarine*; (ii) an H-M type match; and (iii) an asking/answer point match through a direct link: there are no asking point type constraints because the asking point goes beyond existing NE. This case highlights the power of semantic parsing in answer-point extraction. Since there are no type constraints on answer point,⁴⁰ candidate answer points cannot be extracted without bringing in structural context by checking the NE type. Most *what*-related asking points such as those in the patterns '*what/which...N*', '*what type/kind of ...N*' go beyond NE and require this type of structural relation checking to locate the exact answer. The case below is another example.

- Q79: *What did Shostakovich write for Rostropovich?* →
 asking-point grammar link: V-O write ← [What]
 A: *The Polonaise from Tchaikovsky's opera Eugene was a brief but cracking opener and its brilliant bluster was no sooner in our ears than forcibly contradicted by the bleak depression of Shostakovich's second cello concerto, Op. 126, a late work written for Rostropovich in 1966 between the thirteenth and fourteenth symphonies.* →
 candidate answer-point grammar link: V-O written ← [a late work]/NP
 S-P [Op. 126]/NP ← [a late work]/NP
 Answer-point Output: Op. 126

This case requires (i) an exact match in its original verb form between ‘written’ and ‘write’; (ii) a V-O type match; and (iii) an asking/answer point match through indirect link based on equivalence link S-P. When there are no NE constraints on the answer point, a proper name or an initial-capitalized NP is preferred over an ordinary, lower-case NP as an answer point. This heuristic is built-in so that ‘Op. 126’ is outputted as the answer-point in this case instead of ‘a late work’.

⁴⁰ Strictly speaking, there are some type constraints on the answer point. The type constraints are something to the effect of ‘a name for a kind of ship’ which goes beyond the existing NE types defined.

7. CASELESS QA

The corpus from which current QA systems attempt to retrieve answers is usually mixed case text. However, there are numerous corpora that consist of case insensitive documents, e.g. speech recognition results. This section presents an effective restoration approach to case insensitive QA, whereby a preprocessing module is designed to restore the case-sensitive form. The case-restored question and corpus are then input to the QA system, which remains unchanged. The case restoration preprocessing is implemented as a Hidden Markov Model trained on a large raw corpus of case sensitive documents. Experiments show that this approach leads to fairly limited performance degradation compared to case sensitive QA, mainly due to the limited degradation in the underlying information extraction support. Compared with the baseline case insensitive QA where the case restoration technique is not used, a significant performance enhancement (32%) is observed.

In real-life QA applications, a system should be robust enough to handle diverse textual media degraded to different degrees. One of the challenges from degraded text is the treatment of case insensitive documents such as speech recognition results, broadcast transcripts, and the Foreign Broadcast Information Service (FBIS) sources. In the intelligence domain, several document sources are in all uppercase. A speech QA application requires that the QA system interface to a speech recognition interface through which a spoken question is transcribed into case insensitive text.

Orthographic case information for written text is an important information source. In particular, the basic IE support for QA, namely NE tagging, relies heavily on case information for recognizing proper names. Almost all NE systems utilize case-related features. When this information is not available, if the system is not re-trained or adapted, serious performance degradation will occur. The key issue here is how to minimize the performance degradation by adopting some strategy for the system adaptation. For search engines, case information is often ignored in keyword indexing and retrieval for the sake of efficiency and robustness/recall. However, QA requires fine-grained text processing beyond keyword indexing since, instead of a list of documents or URLs, a list of candidate answers at phrase level or sentence level is required in response to a query.

For a system based on language models, a feature exclusion approach is used to re-train the models, excluding features related to the case information (Kubala *et al.*, 1998; Miller *et al.*, 2000; Palmer *et al.*, 2000). In particular, the DARPA HUB-4 program evaluates NE systems on speech recognizer output in Standard Normalized Orthographic Representation (SNOR) that is case insensitive and has no punctuations (Chinchor *et al.*, 1998b). Research on case insensitive text has mainly been restricted to NE and the feature exclusion approach (Chieu and Ng, 2002; Kubala *et al.*, 1998; Palmer *et al.*, 2000; Robinson *et al.*, 1999). For systems going beyond the NE level, the traditional feature exclusion approach may not be feasible. A sophisticated QA system usually involves multiple modules, reflecting NLP/IE processing at various levels. Each processing module may involve case information as constraints. It is too costly and sometimes impossible to maintain two versions of

a multi-module QA system for the purpose of handling two types of documents, with or without case.

This section presents the use of a case restoration approach to this problem, as applied to case insensitive QA. The focus is to study the feasibility of case insensitive QA using the case restoration approach. A preprocessing module is designed to restore the case-sensitive form to feed to the QA system. The case restoration module is based on a Hidden Markov Model (HMM) trained on a large raw corpus of case sensitive documents, which are drawn from the target domain with no need for human annotation. With the incorporation of this preprocessing module, the entire QA system with its underlying NLP/IE components needs no change or adaptation in handling a case insensitive corpus and questions. Using the TREC corpus with case information artificially removed, this approach has been benchmarked with only limited degradation in QA performance. In the literature, this is the first time a QA system is applied to case insensitive corpora and case insensitive questions.

Although the artificially generated case insensitive corpus is an easier case than real life case insensitive corpora, e.g. speech recognition results, the insight and techniques gained in this case insensitive QA research are helpful in further exploring solutions to spoken language QA. In addition, by using the TREC corpus and the TREC benchmarking standards, the QA degradation benchmarking is easy to interpret and to compare with other QA systems in the community.

The case restoration approach has the following advantages: (*i*) the training corpus is almost limitless, resulting in a high performance model, with no knowledge bottleneck as faced by many supervised learning scenarios, (*ii*) the case restoration approach is applicable no matter whether the core system is statistical model, a hand-crafted rule system or a hybrid, (*iii*) when the core system consists of multiple modules, as is the case for the QA system presented in this chapter that is based on multi-level NLP/IE, the case restoration approach relieves the burden of having to re-train or adapt each module in respect of case insensitive input, and (*iv*) the restoration approach reduces the system complexity: the burden of handling degraded text (*case* in this case) is reduced to a preprocessing module while all other components need no changes. In terms of benchmarking, we first benchmarked the underlying IE engine on both case sensitive and case insensitive text to measure the degradation in IE performance, followed by the case insensitive QA benchmarking. The benchmarks include both the performance degradation from case sensitive QA to case insensitive QA as well as the performance enhancement of the case restored QA to a baseline case insensitive QA.

Case Restoration: In the literature, Niu *et al.*, (2003c; 2004) and Lita *et al.*, (2003) reported their respective case restoration approaches as an alternative to the traditional retraining approach. Before we proceed with the presenting our experiments on using this approach to support caseless QA, we summarize the restoration approach used in our work, and leave the details by referring to (Niu *et al.*, 2004). A simple bi-gram Hidden Markov Model (Bikel *et al.*, 1999) is selected as the choice of language model for case restoration. The restoration engine is based on a bi-gram model trained on a large case sensitive raw corpus: a corpus of 7.6 million words drawn from the general news domain is used in training case

restoration. Three orthographic tags are defined in this model: (*i*) initial uppercase followed by at least one lowercase, (*ii*) all lowercase, and (*iii*) all uppercase. Benchmarking shows the overall F-measure is 97% for case restoration. Degradation tests show that when this restoration algorithm is applied to NE tagging which is the cornerstone for supporting QA, the degradation is only 2.1%. This result is significantly better than the traditional NE re-training approach with 6.3% degradation. Beyond NE, the major InfoXtract support for QA comes from the CE relationships and the SVO parsing results. Degradation tests show that there is almost no statistically significant difference in the SVO performance. The degradation due to the case restoration was only 0.07%. The degradation for CE is also fairly limited, about 6% (Niu *et al.*, 2004).

The caseless QA experiments were conducted following the TREC-8 QA standards in the category of 250-byte answer strings. In addition to the TREC-8 benchmarking standards Mean Reciprocal Rank (MRR), we also benchmarked precision for the top answer string.

QA on Caseless Corpus: Table 3 shows the results of our first QA experiment where questions remain in normal mixed case but the TREC corpus is used in three modes for benchmarking. The three modes of the TREC corpus are: (*i*) original case sensitive corpus, (*ii*) case insensitive corpus, i.e. simply applying the QA system to the same corpus which is artificially made into all upper case, and (*iii*) case restored corpus via the case restoration preprocessor. The results on the case restored corpus show very limited performance degradation from the QA on the normal case sensitive corpus: only 2.8% for MRR and 3.1% for the top answer string. This performance is close to the state-of-the-art case sensitive QA despite the loss of case information in the underlying corpus. Compared with the baseline performance when the QA is directly applied to the case insensitive corpus, the QA using the case restored corpus shows an enhancement of about 26%. These degradation and enhancement benchmarks demonstrate the power of the case restoration approach in QA applications on case insensitive corpora.

Table 3: QA Benchmarking Using Case Sensitive Questions

Type	Top 1 Precision	MRR
QA on Normal Case Sensitive Corpus	65.7%	73.9%
Baseline QA on Case Insensitive Corpus	36.9%	45.3%
QA on Case-restored Corpus	62.6%	71.1%
Degradation from QA on Normal Case Sensitive corpus	3.1%	2.8%
Enhancement to Baseline QA on Case Insensitive Corpus	25.7%	25.8%

Comparing QA benchmarks with benchmarks for the underlying IE engine shows that the limited QA degradation is in proportion with the limited degradation in NE, CE and SVO. The following examples illustrate the chain effect: case restoration errors → NE/CE/SVO errors → QA errors.

Q137: 'Who is the mayor of Marbella?'

This is a CE question, the decoded CE asking relationship is *CeHead* for the location entity ‘Marbella’. In QA on the original case sensitive corpus, the top answer string has a corresponding CeHead relationship extracted as shown below.

Input:

Some may want to view the results of the much-publicised activities of the mayor of Marbella, Jesus Gil y Gil, in cleaning up the town
 → [NE tagging]

Some may want to view the results of the much-publicised activities of the mayor of <NeCity>Marbella</NeCity>, <NeMan>Jesus Gil y Gil</NeMan>, in cleaning up the town

→ [CE extraction]

CeHead (Marbella, Jesus Gil y Gil)

In contrast, the case insensitive processing is shown below:

Input:

SOME MAY WANT TO VIEW THE RESULTS OF THE MUCH-PUBLICISED ACTIVITIES OF THE MAYOR OF MARBELLA, JESUS GIL Y GIL, IN CLEANING UP THE TOWN

→ [case restoration]

some may want to view the results of the much-publicised activities of the mayor of marbella, Jesus Gil y Gil, in cleaning up the town

→ [NE tagging]

some may want to view the results of the much-publicised activities of the mayor of marbella, <NeMan>Jesus Gil y Gil</NeMan>, in cleaning up the town

The CE module failed to extract the relationship for MARBELLA because this relationship is defined for the entity type NeOrganization or NeLocation which is absent due to the failed case restoration for ‘MARBELLA’. The next example shows an NE error leading to a problem in QA.

Q119: ‘What Nobel laureate was expelled from the Philippines before the conference on East Timor?’

In question processing, the NE Asking Point is identified as NePerson. Because *Mairead Maguire* was successfully tagged as NeWoman, the QA system got the correct answer string in the following snippet: *Immigration officials at the Manila airport on Saturday expelled Irish Nobel peace prize winner Mairead Maguire*. However, the case insensitive processing fails to tag any NePerson in this snippet. As a result the system misses this answer string. The process is illustrated below.

Input:

IMMIGRATION OFFICIALS AT THE MANILA AIRPORT ON
SATURDAY EXPELLED IRISH NOBEL PEACE PRIZE WINNER
MAIREAD MAGUIRE

→ [case restoration]

immigration officials at the Manila airport on Saturday expelled Irish Nobel Peace Prize Winner Mairead Maguire

→ [NE tagging]

immigration officials at the <NeCity>Manila</NeCity> airport on
<NeDay>Saturday</NeDay> expelled <NeProduct>Irish Nobel Peace Prize
Winner Mairead Maguire </NeProduct>

As shown, errors in case restoration cause mistakes in the NE grouping and tagging: *Irish Nobel Peace Prize Winner Mairead Maguire* is wrongly tagged as NeProduct.

We also found one interesting case where case restoration actually leads to QA performance enhancement over the original case sensitive processing. A correct answer snippet is promoted from the 3rd candidate to the top in answering *Q191 ‘Where was Harry Truman born?’*. This process is shown below.

Input:

HARRY TRUMAN (33RD PRESIDENT): BORN MAY 8, 1884, IN
LAMAR, MO.

→ [case restoration]

Harry Truman (33rd President) : born May 8, 1884, in Lamar, MO.

→ [NE tagging]

<NeMan>Harry Truman</NeMan> (<NeOrdinal>33rd</NeOrdinal>
President) : born <NeDay>May 8, 1884</NeDay>, in <NeCity>Lamar,
MO</NeCity>.

As shown, LAMAR, MO gets correctly tagged as NeCity after case restoration. But LAMAR is mis-tagged as NeOrg in the original case sensitive processing. The original case sensitive snippet is Harry Truman (33rd President): Born May 8, 1884, in Lamar, Mo. In our NE system, there is such a learned pattern as follows: *X, TwoLetterUpperCase → NeCity*.

This rule fails to apply to the original text because the US state abbreviation appears in a less frequently seen format *Mo* instead of *MO*. However, the restoration HMM assigns all uppercase to ‘MO’ since this is the most frequently seen orthography for this token. This difference of the restored case from the original case enables the NE tagger to tag *Lamar, MO* as ‘NeCity’ which meets the NE Asking Point constraint ‘NeLocation’.

QA on Caseless Corpus and Caseless Questions: this *entirely* caseless QA research is meaningful because, when interfacing a speech recognizer to a QA system to accept spoken questions, the case information is not available in the incoming question. We want to know how well the same case restoration technique

applies to question processing and gauge the degradation effect on the QA performance and the ways for further enhancement.

Table 4 shows the results of our second QA experiment comparing case insensitive QA with case sensitive QA. For the case insensitive QA, the case restored QA performance is contrasted with the baseline performance. Compared with the first experiment where questions remain in normal mixed case, there is a greater degree of degradation (close to 10%, in contrast to 3%) from the case sensitive QA to the case restored QA while the enhancement to the baseline is also greater (about 32%, in contrast to 26%). Further examination shows that the current case restoration model is still effective but not optimal for question processing, compared with the corpus processing, most probably because the model is not trained on mixed case question pool. If the question case restoration reaches the same performance of the corpus case restoration, the case restored QA system is expected to show further improvement in performance.

Table 4: Benchmarking for Entirely Caseless QA

Type	Top 1 Precision	MRR
Normal Case QA	65.7%	73.9%
Baseline Case Insensitive QA (on caseless corpus, with caseless questions)	23.2%	32.4%
Case Restored QA (on case-restored corpus, with case-restored question)	56.1%	64.4%
Degradation from Normal Case QA	9.6%	9.5%
Enhancement to Baseline Case Insensitive QA	32.9%	32.0%

We notice that the question processor missed two originally detected NE Asking Points and one Asking Point CE Link. There are a number of other errors due to incorrectly restored case, including non-asking-point NEs in the question and grouping errors in shallow parsing as shown below for Q26: *What is the name of the “female” counterpart to El Nino, which results in cooling temperatures and very dry weather?*

Input:

WHAT IS THE NAME OF THE “FEMALE” COUNTERPART TO EL
NINO ... ?

→ [case restoration]

What is the name of the “Female” counterpart to El Nino, ... ?

→ [question shallow parsing]

NP[What] VG[is] NP[the name] PP[of the] “ AP[Female]” NP[counterpart]
PP[to El Nino] , ... ?

In the original mixed-case question, after parsing, we get the following basic phrase grouping:

NP[What] VG[is] NP[the name] PP[of the “female” counterpart] PP[to El Nino] , ... ?

There is only one difference between the case-restored question and the original mixed-case question, i.e. *Female* vs. *female*. This difference causes the shallow parsing grouping error for the PP of the “female” counterpart. This error affects the weights of the ranking features Headword Matching and Phrase-internal Word Order. As a result, the following originally correctly identified answer snippet was dropped: *the greenhouse effect and El Nino – as well as its “female” counterpart, La Nina – have had a profound effect on weather nationwide.*

As question processing results are the starting point and basis for snippet retrieval and feature ranking, an error in question processing seems to lead to greater degradation, as seen in almost 10% drop compared with about 3% drop in the case when only the corpus requires case restoration.

A related explanation for this degradation contrast is as follows. Due to the information redundancy in a large corpus, processing errors in some potential answer strings in the corpus can be compensated for by correctly processed equivalent answer strings. This is due to the fact that the same answer may be expressed in numerous ways in the corpus. Some of those ways may be less subject to the case effect than others. Question processing errors are fatal in the sense that there is no information redundancy for its compensation. Once it is wrong, it directs the search for answer strings in the wrong direction. Since questions constitute a subset of the natural language phenomena with their own characteristics, case restoration needs to adapt to this subset for optimal performance, e.g. by including more questions in the case restoration training corpus.

8. CONCLUSIONS

This chapter has presented a QA system based on exploiting multiple levels of information extraction: the goal is to produce the most precise answer possible, backing off in a novel approach, to coarser-grained answers where necessary. Three levels of IE have been exploited in the QA system: (i) NE tagging in conjunction with shallow parsing, (ii) CE links, and (iii) semantic parsing links. A single QA framework incorporates all three levels. Experimental results have been shown for all three levels of QA. In addition, experiments using the same QA system plus a case restoration preprocessor demonstrate limited performance degradation in caseless QA from normal case QA (2.8%). This approach uses a high performance case restoration module based on HMM. There is no need for any changes on the QA system and the underlying IE engine which were originally designed for handling normal, case sensitive corpora. It is observed that the limited QA degradation is due to the limited IE degradation.

The highlight of this work is an approach to exact answer determination using binary structural links involving the question-phrases. Based on the experiments conducted, some preliminary conclusions can be drawn.

- CE extraction helps in pinpointing exact answers precisely
- Grammar dependency links enable the system to not only identify exact answers but also answer questions not covered by the predefined set of available NEs/Associations
- Binary dependency links, instead of full structural templates provide sufficient and effective structural leverage for extracting exact answers

Some cases remain beyond the current level of NLP/IE. For example,

Q92: *Who released the Internet worm in the late 1980s?* →

asking point link: V-S (released, NePerson[Who])

A: *Morris, suspended from graduate studies at Cornell University at Syracuse, N.Y., is accused of designing and disseminating in November, 1988, a rogue program or “worm” that immobilized some 6,000 computers linked to a research network, including some used by NASA and the Air Force.* →

answer point link: V-S(disseminating, NePerson[Morris])

In order for this case to be handled, the following steps are required: (i) the semantic parser should be able to ignore the past participle post-modifier phrase headed by ‘suspended’; (ii) the V-O dependency should be decoded between ‘is accused’ and ‘Morris’; (iii) the V-S dependency should be decoded between ‘designing and disseminating’ and ‘Morris’ based on the pattern rule ‘accuse NP of Ving’ → V-S(Ving, NP); (iv) the conjunctive structure should map the V-S (‘designing and disseminating’, ‘Morris’) into two V-S links; and (v) ‘disseminate’ and ‘release’ should be linked somehow for synonym expansion. Expecting an NLP/IE system to accomplish the above challenges indicates some directions for further research in this topic.

We plan to extend the experiments to a larger set of questions to further investigate the effectiveness of structural support in extracting exact answers. The TREC-9 and TREC 2001 QA pool and the candidate answer sentences generated by both NLP-based or IR-based QA systems would be ideal for further testing this method. An extension from the current use of CE relationships to the use of the Predefined Event and Entity Profile results will also enhance the QA capability. This capability will enable the system to answer challenging questions: questions about key events such as “Who was appointed as CEO of ABC Inc. in July 2002?” that are expressed in different key verbs and/or different structures and definition questions such as “Who was Julian Hill?”, “What is ABC Inc.? ” or “How is ABC Inc. covered in the media? ” For the latter type of questions, Entity Profiles, which collect various key information and relationships about a specific entity from different parts of text provide the richest information to satisfy users.

We have presented a framework for QA based on exploiting multiple levels of IE. As IE technology improves, QA systems will also improve. InfoXtract is currently being enhanced to support broader relationship extraction both within and across documents, as well as event detection for a wider set of events in both the

business and intelligence communities. Rapid, semi-automated domain porting is also an important focus: certain domains may require different types of entities/relationships/events to be detected. Finally, there is also a major effort to normalize extracted information, in particular temporal and location normalization. QA can only be used successfully in arbitrary document collections if time stamping of information is reliable. An observation from the research of handling caseless questions is that question processing degradation has more serious consequence affecting the QA performance. As a future effort, we plan to focus on enhancing the case restoration performance by including as many mixed-case questions as possible into the training corpus for case restoration. Although there are many directions in which QA can be enhanced, including the incorporation of sophisticated inference systems, it is felt that high-quality IE will have the biggest impact in making robust, deployable QA systems a reality.

9. ACKNOWLEDGEMENTS

The authors wish to thank Carrie Pine and Walter Gadz of AFRL for supporting this work. Thanks also go to Cheng Niu, M. Srikanth, Lars Nilsson and Xiuhong Zhang for their efforts in this endeavor. Finally, the authors were affiliated with Cymfony Inc. during much of this effort; Janya was spun off from Cymfony. Several past and present members of Cymfony participated in the QA experimentation.

10. REFERENCES

- Abney, S., Collins, M. and Singhal, A. 2000. Answer Extraction. In *Proceedings of ANLP-2000*, pages 296-301, Seattle, Washington.
- Bikel, D.M., R. Schwartz, and R.M. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning*, Vol. 1,3: pages 211-231.
- Chinchor, N. and Marsh, E. 1998a. MUC-7 Information Extraction Task Definition (version 5.1), In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/ie_task.html
- Chinchor, N., P. Robinson and E. Brown. 1998b. HUB-4 Named Entity Task Definition Version 4.8. www.nist.gov/speech/tests/bnr/hub4_98/hub4_98.htm
- Chieu, H.L. and H.T. Ng. 2002. Teaching a Weaker Classifier: Named Entity Recognition on Upper Case Text. In *Proceedings of ACL-2002*, Philadelphia, PA, pages 481-488.
- Clarke, C.L. A., Cormack, G.V. and Lynam, T.R. 2001. Exploiting Redundancy in Question Answering. In *Proceedings of SIGIR'01*, pages 358-365, New Orleans, LA.
- Grunfeld, L. and K.L. Kwok 2005. Sentence Ranking Using Keywords and Meta-keywords: Experience with TREC-9 and TREC-2001 Using PIRCS System, in T. Strzalkowski & S. Harabagiu (eds.) *Advances in Open-Domain Question Answering*, Kluwer Academic Publishers, 2005 (in this volume).
- Hobbs, J.R. 1993. FASTUS: A System for Extracting Information from Text. In *Proceedings of the DARPA Workshop on Human Language Technology*, pages 133-137, Princeton, NJ.
- Hovy, E.H., U. Hermjakob, and Chin-Yew Lin. 2001. The Use of External Knowledge of Factoid QA. In *Proceedings of the 10th Text Retrieval Conference (TREC 2001)*, page 644, Gaithersburg, MD, U.S.A., November 13-16, 2001.
- Krupka, G.R. and Hausman, K. 1998. IsoQuest Inc.: Description of the NetOwl (TM) Extractor System as Used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html

- Kubala, F., R. Schwartz, R. Stone and R. Weischedel. 1998. Named Entity Extraction from Speech. In *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, VA.
- Kupiec, J. 1993. MURAX: A Robust Linguistic Approach For Question Answering Using An On-Line Encyclopaedia, In *Proceedings of SIGIR-93*, pages181-190, Pittsburgh, PA.
- Kwok, K.L., Grunfeld, L., Dinstl, N. and Chan, M. 2001. TREC2001 Question-Answer, Web and Cross Language Experiments using PIRCS. In *proceedings of the Tenth Text REtrieval Conference (TREC-2001)*, page 452, Gaithersburg, MD.
- Li, W., R. Srihari, C. Niu and X. Li. 2003a. Question Answering on a Case Insensitive Corpus. In *Proceedings of Multilingual Summarization and Question Answering – Machine Learning and Beyond* (ACL-2003 Workshop), Sapporo, Japan.
- Li, W., R. Srihari, C. Niu, and X. Li 2003b. Entity Profile Extraction from Large Corpora. In *Proceedings of Pacific Association for Computational Linguistics 2003* (PAACL'03). Halifax, Nova Scotia, Canada.
- Li, W. and R. Srihari, 2003c. *Flexible Information Extraction Learning Algorithm, Phase 2 Final Technical Report*, Air Force Research Laboratory, Rome Research Site, NY.
- Li, W., R. Srihari, X. Li, M. Srikanth, X. Zhang and C. Niu. 2002. Extracting Exact Answers to Questions Based on Structural Links. In *Proceedings of Multilingual Summarization and Question Answering (COLING-2002 Workshop)*, Taipei, Taiwan.
- Lita, L.V., A. Ittycheriah, S. Toukos and N. Kambhatla. 2003. tRuEcaSing. In *Proceedings of ACL-2003*. Sapporo, Japan.
- Litkowski, K.C. 1999. Question-Answering Using Semantic Relation Triples. In *proceedings of the eighth Text Retrieval Conference (TREC-8)*, page 349, Gaithersburg, MD.
- Miller, Scott; Crystal, Michael; Fox, Heidi; Ramshaw, Lance; Schwartz, Richard; Stone, Rebecca; Weischedel, Ralph; and Annotation Group (BBN Technologies). 1998. BBN: Description of the SIFT System as Used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference*, www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html
- Miller, D., S. Boisen, R. Schwartz, R. Stone, and R. Weischedel. 2000. Named Entity Extraction from Noisy Input: Speech and OCR. In *Proceedings of ANLP 2000*, Seattle.
- Niu, C., W. Li, J. Ding, and R. Rohini. 2004a. Orthographic Case Restoration Using Supervised Learning Without Manual Annotation. *International Journal of Artificial Intelligence Tools*, Vol. 13, No. 1, 2004.
- Niu, C., W. Li and R. Srihari, 2004b. A Bootstrapping Approach to Information Extraction Domain Porting. *AAAI-2004 Workshop on Adaptive Text Extraction and Mining* (ATEM), California.
- Niu, C., W. Li, J. Ding, and R. Srihari 2003a. A Bootstrapping Approach to Named Entity Classification Using Successive Learners. In *Proceedings of 41st Annual Meeting of ACL*. Sapporo, Japan. pp. 335-342.
- Niu, C., W. Li, R. Srihari, and L. Crist, 2003b. Bootstrapping a Hidden Markov Model for Relationship Extraction Using Multi-level Contexts. In *Proceedings of Pacific Association for Computational Linguistics 2003* (PAACL'03). Halifax, Nova Scotia, Canada.
- Niu, C., W.i, J. Ding and R. Srihari. 2003c. Orthographic Case Restoration Using Supervised learning Without Manual Annotation. In *Proceedings of the 16th International FLAIRS Conference 2003*, Florida
- Palmer, D., M. Ostendorf and J.D. Burger. 2000. Robust Information Extraction from Automatically Generated Speech Transcriptions. *Speech Communications*, Vol. 32: 95-109.
- Pasca, M. and Harabagiu, S. M. 2000. High Performance Question/Answering. In *Proceedings of SIGIR 2000*. pages 366-374.
- Prager, J., Radev, D., Brown, E., Coden, A. and Samn, V. 1999. The use of predictive annotation for question answering in TREC8. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, Gaithersburg, MD.
- Ravichandran, D., and Hovy, E. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 41-47, Philadelphia, PA.
- Robinson, P., E. Brown, J. Burger, N. Chinchor, A. Douthat, L. Ferro, and L. Hirschman. 1999. Overview: Information Extraction from Broadcast News. In *Proceedings of The DARPA Broadcast News Workshop*. Herndon, Virginia: pages 27-30.

- Srihari, R., W. Li, C. Niu and T. Cornell. 2005. InfoXtract: A Customizable Intermediate Level Information Extraction Engine. *Journal of Natural Language Engineering (forthcoming)*.
- Srihari, R., C. Niu and Li, W. 2000a. A Hybrid Approach for Named Entity and Sub-Type Tagging. In *Proceedings of ANLP 2000*, pages 247-254. Seattle, Washington.
- Srihari, R. and Li, W. 2000b. A Question Answering System Supported by Information Extraction. In *Proceedings of ANLP 2000*, pages 166-172, Seattle, Washington.
- Srihari, R. and Li, W. 1999. Information Extraction supported Question Answering. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, pages 185-196. Gaithersburg, MD.
- Vicedo, J.L. and A. Ferrández, 2006. Co-reference in Q&A, in T. Strzalkowski & S. Harabagiu (eds.) *Advances in Open-Domain Question Answering*. Kluwer Academic Publishers. (in this volume).
- Voorhees, E. 1999. The TREC-8 Question Answering Track Report. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, page 77, Gaithersburg, MD.
- Voorhees, E. 2000. Overview of the TREC-9 Question Answering Track. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 77-82. Gaithersburg, MD.

ABDESSAMAD ECHIHABI, ULF HERMJAKOB,
EDUARD HOVY, DANIEL MARCU, ERIC MELZ,
DEEPAK RAVICHANDRAN

HOW TO SELECT AN ANSWER STRING?

Abstract: Given a question Q and a sentence/paragraph SP that is likely to contain the answer to Q, an answer selection module is supposed to select the “exact” answer sub-string A ⊂ SP. We study three distinct approaches to solving this problem: one approach uses algorithms that rely on rich knowledge bases and sophisticated syntactic/semantic processing; one approach uses patterns that are learned in an unsupervised manner from the web, using computational biology-inspired alignment algorithms; and one approach uses statistical noisy-channel algorithms similar to those used in machine translation. We assess the strengths and weaknesses of these three approaches and show how they can be combined using a maximum entropy-based framework.

1. INTRODUCTION

The recent activity in research on automated question answering concentrating on factoids—brief answers—has highlighted the two basic stages of the process—information retrieval, to obtain a list of candidate passages likely to contain the answer, and answer selection, to identify and pinpoint the exact answer among and within the candidates. In this paper we focus on the second stage. We situate this work in the context of the TREC question answering evaluation competitions, organized annually since 1999 by NIST (Voorhees, 1999; 2000; 2001; 2002), and use both the TREC question and answer collections, the TREC text corpus of some 1 million newspaper and similar articles, and the TREC scoring method of Mean Reciprocal Rank (MRR), in order to make our results comparable to other research.

What constitutes a correct, exact answer to a natural language question is easiest described by means of examples. The TREC guidelines (trec.nist.gov/pubs.html) specify, for instance, that given the question *What river in the US is known as the Big Muddy?*, strings such as “Mississippi”, “the Mississippi”, “the Mississippi River”, and “Mississippi River” should be judged as exact answers, while strings such as “2,348 miles; Mississippi”, “Mississip”, and “known as the Big Muddy, the Mississippi is the longest river in the US” should be considered inexact.

Automatically finding in a document collection the correct, exact factoid answer to a natural language question is by no means a trivial problem, since it involves several processes that are each fairly sophisticated, including the ability to understand the question, derive the expected answer type, generate information retrieval queries to select documents, paragraphs, and sentences that may contain the

answer, and pinpoint in these paragraphs and sentences the correct, exact, answer sub-string. The best question answering systems built to date are complex artefacts that use a large number of components such as syntactic/semantic parsers, named-entity taggers, and information retrieval engines. Unfortunately, such complexity masks the contribution of each module, making it difficult to assess why the system fails to find accurate answers. For this reason, we are constantly designing experiments that will enable us to understand better the strengths and weaknesses of the components we are using and to prioritize our work, in order to increase the overall performance of our system. One such experiment, for example, showed clearly that answer selection is far from being a solved problem. To diagnose the impact of the answer selection component, we did the following:

- We used the 413 TREC-2002 questions that were paired by human assessors with correct, exact answers in the TREC collection.
- For each question, we selected all sentences that were judged as containing a correct, exact answer to it.
- We presented the questions and just these answer sentences to the best answer selection module we had available at that time; in other words, we created perfect experimental conditions, consistent to those that one would achieve if one had perfect document, paragraph, and sentence retrieval components.

To our surprise, we found that our answer selection module was capable of selecting the correct, exact answer in only 68.2% of the cases. That is, even when we gave our system *only* sentences that contained correct, exact answers, it failed to identify more than 30% of them! Two other answer selection modules, which we were developing at the same time, produced even worse results: 63.4% and 56.7% correct. Somewhat more encouraging, we determined that an oracle that could select the best answer produced by any of the three answer selection modules would have produced 78.9% correct, exact answers. Still, that left over 20% correct answers not being recognized.

The results of this experiment suggested two clear ways for improving the performance of our overall QA system:

- Increase the performance of any (or all) answer selection module(s).
- Develop methods for combining the strengths of the different modules.

In this chapter, we show that the Maximum Entropy (ME) framework can be used to address both of these problems. By using a relatively small corpus of question-answer pairs annotated by humans with correctness judgments as training data, and by tuning a relatively small number of log-linear features on the training corpus, we show that we can substantially increase the performance of each of our individual answer selection modules. Pooling the answers produced by all systems leads to an additional increase in performance. This suggests that our answer selection modules have complementary strengths and that the ME framework enables one to learn and exploit well the individualities of each system.

Ultimately, real users do not care about the ability of an answer selection module to find exact, correct answers in hand-picked sentences. Because of this, to

substantiate the claims made in this chapter, we carried out all our evaluations in the context of an end-to-end QA system, TextMap, in which we varied only the answer selection component(s). The TextMap system implements the following pipeline (see (Hermjakob *et al.*, 2002) for details):

- A question analyser identifies the expected answer type for the question given as input (see Section 2.1 for details).
- A query generator produces Web- and TREC-specific queries. The query generator exploits a database of paraphrases (see Section 2.3).
- Web queries are submitted to Google and TREC queries are submitted to the IR engine Inquiry (Callan *et al.*, 1995), to retrieve respectively 100 Web and 100 TREC documents.
- A sentence retrieval module selects 100 sentences each from the retrieved Web and TREC documents that are most likely to contain a correct answer.
- Each of the answer selection modules described in this paper pinpoints the correct answers and in the resulting 200 sentences and assigns them a score.
- The highest ranked answer is presented to the user.

For the contrastive analysis of the answer selection modules we present in this chapter, we chose to use in all of our experiments the 413 factoid questions made available by NIST as part of the TREC-2003 QA evaluation. In all our experiments, we run our end-to-end QA system against documents available on either the Web or the TREC collection. We pinpoint exact answers in Web- or TREC-retrieved sentences using different answer selection or combinations of answer selection modules.

To measure the performance of our answer selection modules in the context of the end-to-end QA system, we created by hand an exhaustive set of correct and exact answer patterns. If the answer returned by a system matched perfectly one of the answer patterns for the corresponding question, the answer was considered correct and exact. If the answer did not match the answer pattern, it was considered incorrect. Naturally, this evaluation is not 100% bullet proof. One can still have correct, exact answers that are not covered by the patterns we created; or one can return answers that are correct and exact but unsupported. We took, however, great care in creating the answer patterns. Qualitative evaluations of the correctness of the results reported in our experiments suggest that our methodology is highly reliable. Most importantly though, even if the evaluation results are off by 1 or 2% in absolute terms due to incomplete coverage of the patterns, the methodology is perfectly sound for measuring the relative performance of the different systems because all systems are evaluated against a common set of answer patterns.

In this chapter, we present three different approaches to answer selection. One approach uses algorithms that rely on rich knowledge bases and sophisticated syntactic/semantic processing (Section 2); one approach uses patterns that are learned in an unsupervised manner from the web, using computational biology-inspired alignment algorithms (Section 3); and one approach uses statistical, noisy-channel algorithms similar to those used in machine translation (Section 4). We assess the performance of each individual system in terms of:

- number of correct, exact answers ranked in the top position;
- number of correct, exact answers ranked in the top 5 positions;
- MRR score⁴¹ based on the top five answers.

We show that maximum entropy working with a relative small number of features has a significant impact on the performance of each system (Section 5). We also show that the same ME-based approach can be used to combine the outputs of the three systems. When we do so, the performance of the end-to-end QA system increases further (Section 5).

2. KNOWLEDGE-BASED ANSWER SELECTION

This section describes a strongly knowledge-based approach to question answering. As described in the following subsections, this approach relies on several types of knowledge. Among them, answer typing (“Qtargets”), semantic relationship matching, paraphrasing, and several additional heuristics all heavily rely on parsing, of both the question and all answer sentence candidates.

We use the CONTEX parser (Hermjakob, 1997; 2001), a decision tree based deterministic parser, which has been enhanced for question answering by an additional treebank of 1,200 questions, named entity tagging that among other components uses BBN’s IdentifiFinder (Bikel *et al.*, 1999), and a semantically motivated parse tree structure that facilitates matching for paraphrasing and of question/answer pairs.

2.1 Qtargets

After parsing a question, TextMap determines its answer type, or “Qtarget”, such as *PROPER-PERSON*, *PHONE-NUMBER*, or *NP*. We have built a typology of currently 185 different types, organized into several classes (Abstract, Semantic, Relational, Syntactic, etc.). An older version of the typology can be found at http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html.

As the following example shows, Qtargets can significantly narrow down the search space (neither “exactly” nor “2.8% taller than K2” conform to a DISTANCE-QUANTITY answer type):

Question: How tall is Mt. Everest?

Qtarget: DISTANCE-QUANTITY

Answer candidates:

- Jack knows exactly how tall Mt. Everest is.
- Jack climbed the 29,028-foot Mt. Everest in 1984 and the 7,130-foot Mt. Kosciusko in Australia in 1985.
- Mt. Everest is 2.8% taller than K2.

⁴¹ TREC’s Mean Reciprocal Rank assigns a score of 1 if the correct answer is in first place (of five guesses), 1/2 if it is in second place, 1/3 if in third place, etc.

2.2 Semantic Relations

As words and answer types are often not enough to identify the proper answer, the knowledge-base answer selection modules also boosts the scores of answer candidates whose constituents have the same semantic relationships to one another as those in the question:

Question: Who killed Lee Harvey Oswald?

Text: Jack Ruby, who killed John F. Kennedy assassin Lee Harvey Oswald

While “John F. Kennedy” is textually closer to the question terms “killed” and “Lee Harvey Oswald”, our QA system will prefer “Jack Ruby”, because its logical subject relation to the verb matches that of the interrogative in the question. Semantic relations hold over all roles and phrase types, and are independent of word order.

2.3 Paraphrases

Sentences with a good answer often don’t match the wording of the question; sometimes simply matching surface words can result in an incorrect answer:

Question: Who is the leader of France?

Candidate answers:

- Henri Hadjenberg, who is the leader of France’s Jewish community, endorsed confronting the specter of the Vichy past.
- (100% word overlap, but sentence does not contain answer.)
- Bush later met with French President Jacques Chirac.
- (0% word overlap, but sentence does contain the correct answer.)

Neither word reordering nor simple word synonym expansion will help us to identify Jacques Chirac as the correct answer.

To bridge the gap between question and answer sentence wordings, TextMap uses paraphrasing. For any given question, TextMap generates a set of high-precision meaning-preserving reformulations to increase the likelihood of finding correct answers in texts:

Question: How did Mahatma Gandhi die?

Reformulation patterns:

- Mahatma Gandhi died <how>?
- Mahatma Gandhi died of <what>?
- Mahatma Gandhi lost his life in <what>?
- Mahatma Gandhi was assassinated?
- Mahatma Gandhi committed suicide?
- ... plus 40 other reformulations ...

The fourth reformulation will easily match “Mahatma Gandhi was assassinated by a young Hindu extremist,” preferring it over alternatives such as “Mahatma Gandhi died in 1948.”

Paraphrases can span a wide range, from simple syntactic reformulations (When did the Titanic sink? => The Titanic sank when?) to rudimentary forms of inference (Where is Thimphu? => Thimphu is the capital of <which place>?); for example:

Question: How deep is Crater Lake?

Reformulation patterns:

- Crater Lake is <what distance> deep?
- depth of Crater Lake is <what distance>?
- Crater Lake has a depth of <what distance>?
- <what distance> deep Crater Lake?
- and more

Question: Who invented the cotton gin?

Reformulation patterns:

- <who> was the inventor of the cotton gin?
- <who>‘s invention of the cotton gin?
- <who> was the father of the cotton gin?
- <who> received a patent for the cotton gin?

2.3.1 Paraphrase Patterns: A Resource

Rather than creating and storing thousands of paraphrases, we acquire paraphrase patterns, which are used at run-time to generate instantiated patterns against which candidate answers are matched. Paraphrase patterns are acquired either by manual entry or by automated learning (see Section 3) and subsequent manual refinement and generalization. The paraphrase collection is pre-parsed, and then, at run-time, pattern matching of questions and paraphrases is performed at the parse tree level.

TextMap paraphrase patterns are expressed in an extended natural language format for high user friendliness:

```
:anchor-pattern "SOMETHING_1 costs MONETARY_QUANTITY_2."
:is-equivalent-to "the price of SOMETHING_1 is MONETARY_QUANTITY_2."
:is-equivalent-to "SOMETHING_1 is on sale for MONETARY_QUANTITY_2."
:can-be-inferred-from "to buy SOMETHING_1 for MONETARY_QUANTITY_2."

:anchor-pattern "SOMEBODY_1 sells SOMETHING_3 to SOMEBODY_2."
:is-equivalent-to "SOMEBODY_2 buys SOMETHING_3 from SOMEBODY_1."
```

Expressing phrasal synonyms in extended natural language makes them easy to write, or, when they are automatically generated, easy to check and filter. The relatively generic declarative format also facilitates reuse in other applications and systems. The expressiveness and focus of the patterns is greatly enhanced when variables carry syntactic or semantic restrictions that can transcend parts of speech. Compared to automatically generated patterns such as (Ravichandran and Hovy,

2002) and (Lin and Pantel, 2001), there are also no limits on the number of variables per reformulation and, since all patterns are checked by hand, only very few misreformulations. The reformulation collection currently contains 550 assertions grouped into about 105 equivalence blocks.

At run-time, the number of reformulations produced by our current system varies from one reformulation (which might just rephrase a question into a declarative form) to more than 40, with an average of currently 5.03 reformulations per question for the TREC-2003 questions.

2.3.2 Advanced Forms of Reformulation

As seen in earlier examples, the paraphrase paradigm can implement a form of inference. Other advanced forms of reformulation in our system are reformulation chains, answer generation, and cross-part-of-speech placeholders. Based on

```
:anchor-pattern "SOMEBODY_1 is a student at COLLEGE_2."
:answers "Where does SOMEBODY_1 go to college?" :answer COLLEGE_2
:anchor-pattern "SOMEBODY_1 was a student at COLLEGE_2."
:can-be-inferred-from "SOMEBODY_1 dropped out of COLLEGE_2."
:anchor-pattern "SOMEBODY_1 dropped out of COLLEGE_2."
:is-equivalent-to "SOMEBODY_1 is a COLLEGE_2 dropout."
```

TextMap can produce the following reformulation chain:

Text corpus: Bill Gates is a Harvard dropout.

Original question: Where did Bill Gates go to college?

Reformulations:

- Bill Gates was a student at <which college>
- Bill Gates dropped out of <which college>
- Bill Gates is a <which college> dropout.

Answer: Harvard

Allowing placeholders to cross syntactic categories makes reformulations even more powerful. To support this type of reformulation, we draw on a number of cross-part-of-speech lists, which include entries such as [France/French] and [invent/invention/inventor]:

```
:anchor-pattern "NATIONALITY_1 OCCUPATION_2 PERSON_3"
:is-equivalent-to "PERSON_3 is the OCCUPATION_2 of COUNTRY_1"
```

which enables:

Text corpus: ... French President Jacques Chirac ...

Question: Who is the president of France?

Reformulation: French president <who>

Answer: Jacques Chirac

Paraphrases not only improve answer pinpointing, but can also support document retrieval and passage selection by proving alternate and/or multi-word search expressions, and increase confidence in many answers.

2.4 Additional Heuristics

Answer selection is further guided by several additional heuristics that penalize answers for a variety of reasons, including:

- Qtarget match factor: Q: How long did the Manson trial last? Semantic mismatch: 20 miles
- Vagueness penalty: Q: Where is Luxor? Too vague: on the other side
- Negation penalty: Q: Who invented the electric guitar? Negation: Fender did **not** invent the electric guitar.
- Bad mod penalty: Q: What is the largest city in Iraq? Relevant modifier: Basra is the **second** largest city in Iraq.
- Novelty factor: Q: Where in Austin is Barton Springs? Nothing novel: in Austin
- Reporting penalty: Q: Who won the war? Reported: Saeed al-Sahhaf **claimed** that Iraq had crushed the invading American forces
- Surface distance factor: favors answers near question terms
- Structural distance factor: favors answers near question terms in parse tree constituent distance
- Bad gender penalty: Q: Which actress played ...? Bad gender: John Wayne

2.5 External Knowledge

The knowledge-based answer selection module uses a limited amount of external knowledge to enhance performance.

- For some questions, WordNet glosses provide an answer, either directly, such as for definition questions, or indirectly; e.g., “What is the capital of Kentucky?” can be answered using the WordNet gloss for Frankfort, “the capital of Kentucky; located in northern Kentucky”, from which the relevant fact has been extracted and stored in a database.
- Internal quantity and calendar conversion routines can answer questions such as “How much is 86F in Celsius?”
- Abbreviation routines score how well an abbreviation conforms with its expansion, and for example strongly prefer NAFTA as an abbreviation for “North American Free Trade Agreement” rather than as an abbreviation for “rice market”.

2.6 Evaluation

When evaluated against the answer patterns we created for the 413 factoid questions in the TREC-2003 collection, the knowledge-based answer selection module described in this section produced 35.83% correct, exact answers. There were 57.38% correct, exact answers in the top 5 candidates returned by a system, with a corresponding MRR score of 43.88%. More details are provided in Section 5.

3. PATTERN-BASED ANSWER SELECTION

At the TREC 2001 conference, several systems emphasized the value of matching surface-oriented patterns, even without any reformulation, to pinpoint answers. The top-scoring Insight system from Moscow (Soubbotin and Soubbotin, 2001) used some hundreds of surface-level patterns to identify answer strings without (apparently) applying Qtargets or similar reasoning. Several other systems also defined word-level patterns indicating specific Qtargets; e.g., (Oh *et al.*, 2001). The Microsoft system (Brill *et al.*, 2001) extended the idea of a pattern to its limit, by reformulating the input question as a declarative sentence and then retrieving the sentence verbatim, with its answer as a completion, from the web, using normal search engines. For example, “Who was Chester F. Carlson?” was transformed to, among others, “Chester was F. Carlson”, “Chester F. was Carlson”, and “Chester F. Carlson was”, and submitted as web queries. Although this approach yielded many wrong answers (including “Chester F. Carlson was born February 8, 1906, in Seattle”), the sheer number of correct answers returned often won the day.

Our estimate is that a large enough collection of word-level patterns, used even without reformulation, can provide at least 25% MRR score, although some systems claimed considerably higher results; see (Soubbotin and Soubbotin, 2001).

3.1 Automated Learning of Patterns

The principal obstacle to using the surface pattern technique is acquiring patterns in large enough variety and number. For any given Qtarget, one can develop some patterns by hand, but there is no guarantee that one thinks of all of them, and one have no idea how accurate or even useful each pattern is.

We therefore developed an automated procedure to learn such patterns from the web (Ravichandran and Hovy, 2002). Using a regular search engine, we collected all the patterns associated with many frequently occurring Qtargets (some Qtargets, such as *Planets* and *Oceans*, are known closed sets that require no patterns). Some of the more precise patterns, associated with their Qtarget in the QA Typology, can be found at http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html.

In addition to using the learned patterns as starting points to define reformulation pattern sets (Section 2), we used them to construct an independent answer selection module. The purpose of this work was to empirically determine the limits of a QA system whose pinpointing knowledge is derived almost fully automatically.

The pattern learning procedure can be phrased as follows. Given a Q_{target} (a relation such as YEAR-OF-BIRTH), instantiated by a specific QA pair such as (NAME_OF_PERSON, BIRTHYEAR), extract from the web all the different lexicalized patterns (TEMPLATES) that contain this QA pair, and also determine the precision of each pattern. The procedure contains two principal steps:

- Extracting the patterns
- Calculating the precision of each pattern

Algorithm 1: Extracting patterns

We wish to learn the surface-level patterns that express a given QA relation such as YEAR-OF-BIRTH.

1. An instance of the relation for which the pattern is to be extracted is passed to a search engine as a QA pair. For example, to learn the patterns for the pair (NAME_OF_PERSON BIRTHYEAR), we submit a pair of anchor terms such as “Gandhi 1869” as a query to Altavista.
2. The top 1000 documents returned by the search engine are retrieved.
3. These documents are broken into sentences by a simple sentence breaker.
4. Only sentences that contain both the Question and the Answer terms are retained. (BBN’s named entity tagger Identifinder (Bikel *et al.*, 1999) was used to remove variations of names or dates.)
5. Each of these sentences is converted into a Suffix Tree using an algorithm from Computational Biology (Gusfield, 1997), to collect counts on all phrases and sub-phrases present in the document.
6. The phrases obtained from the Suffix Tree are filtered so that only those containing both the Question and Answer terms are retained. This yields the set of patterns for the given QA pair.

Algorithm 2: Calculating the precision of each pattern

1. The Question term alone (without its Answer term) is given as query to Altavista.
2. As before, the top 1000 documents returned by the search engine for this query are retrieved.
3. Again, the documents are broken into sentences.
4. Only those sentences that contain the Question terms are saved. (Again, Identifinder is used to standardize names and dates.)
5. For each pattern obtained in step 6 of Algorithm 1, a pattern-matching check is done against each sentence obtained from step 4 here, and only the sentences containing the Answer are retained. This data is used to calculate the precision of each pattern according to the formula:

$$\text{Precision} = \frac{\text{\# patterns matching the Answer (step 5 in Alg 2)}}{\text{total \# patterns (step 4 in Alg 1)}}$$

6. Furthermore, only those patterns are retained for which sufficient examples are obtained in step 5 of this algorithm.

To increase the size of the data, we apply the algorithms with several different anchor terms of the same Qtarget. For example, in Algorithm 1 for YEAR-OF-BIRTH we used Mozart, Gauss, Gandhi, Nelson Mandela, Michelangelo, Christopher Columbus, and Sean Connery, each with birth year. We then applied Algorithm 2 with just these names, counting the yields of the patterns using only the exact birth years (no additional words or reformulations, which would increase the yield score).

The results were quite good in some cases. For the rather straightforward YEAR-OF-BIRTH, some patterns are:

Prec.	#Correct	#Found	Pattern
1.00	122	122	<NAME> (<BD>- <DD>
1.00	15	15	<NAME> (<BD> - <DD>),
1.00	13	13	<NAME> (<BD> - <DD>)
0.9166	11	12	<NAME> was born on <BD> in
0.9090	10	11	<NAME> : <BD> - <TIME>
0.6944	25	36	<NAME> was born on <BD>

Note the overlaps among patterns. By not compressing them further we can record different precision levels. Due to the many forms of expression possible, the Qtarget DEFINITION posed greater problems.

For example, the anchor term *disease* paired with *jaundice*, *measles*, *cancer*, and *tuberculosis* (but not also paired with *illness*, *ailment* etc., which would have increased the counts), yields:

Precision	#Correct	#Found	Pattern
1	46	46	heart <TERM>, <NAME>
1	35	35	<NAME> & tropical <TERM> weekly
1	30	30	venereal <TERM>, <NAME>
1	26	26	<NAME>, a <TERM> that
1	24	24	lyme <TERM>, <NAME>
1	22	22	heart <TERM>, <NAME>
1	21	21	's <TERM>, <NAME>
0.9565	22	23	lyme <TERM> <NAME>
0.9	9	10	's <TERM>, <NAME> and
0.8815	67	76	<NAME>, a <TERM>
0.8666	13	15	<TERM>, especially <NAME>

The anchor term *metal*, paired with *gold*, *silver*, *platinum*, and *bronze*, yields:

Precision	#Correct	#Found	Pattern
1.00	13	13	of <NAME> <TERM>.
1.00	12	12	the <TERM> <NAME>.
0.90	9	10	<TERM> : <NAME>.
0.875	14	16	the <NAME> <TERM>.
0.8181	27	33	<TERM> (<NAME>)
0.80	8	10	's <TERM> (<NAME>
0.7575	25	33	<TERM>, <NAME>, a
0.75	177	236	the <NAME> <TERM>
0.75	12	16	<TERM> : <NAME>,
0.7427	179	241	<NAME> <TERM>,
0.7391	17	23	<TERM> (<NAME>,
0.70	7	10	<TERM> - <NAME>,

The similar patterns for Disease and Metal definitions indicate that one should not create specialized Qtargets *Definition-Disease* and *Definition-Metal*.

3.2 Integrating Patterns into a QA System

We have learned patterns for numerous Qtargets. However, for questions about a Qtarget without patterns, the system would fail. There are two possible approaches: either develop a method to learn patterns dynamically, on the fly, for immediate use, or integrate these patterns with other pinpointing methods. The former approach is problematic, since we do not (yet) have automated ways to produce the seed anchor terms to learn new patterns. Therefore we take the second, using Maximum Entropy to integrate the patterns along with a set of other features and thereby create a standalone answer selection module. We follow Ittycheriah (2002), who was the first to use a completely trainable statistical (Maximum Entropy) QA system.

3.2.1 Model

Assuming we have several methods for producing answer candidates (including, sometimes, patterns), we wish to select the best from all candidates in some integrated way. We model the problem of answer selection as a re-ranking problem (Ravichandran *et al.*, 2003):

$$P(a | \{a_1 a_2 \dots a_A\}, q) = \frac{\exp[\sum_{m=1}^M \lambda_m f_m(a, \{a_1 a_2 \dots a_A\}, q)]}{\sum_{a'} \exp[\sum_{m=1}^M \lambda_m f_m(a', \{a_1 a_2 \dots a_A\}, q)]}$$

where

a is the answer under consideration, one of the candidates $a \in \{a_1 a_2 \dots a_A\}$

q is the question

$f_m(a, \{a_1 a_2 \dots a_A\}, q), m=1, \dots, M$ are the M feature functions providing answers

$\{a_1 a_2 \dots a_A\}$ is the answer candidate set considered for the question.

and the decision rule for the re-ranker is given by:

$$\begin{aligned}\hat{a} &= \arg \max_a [P(a | \{a_1 a_2 \dots a_A\}, q)] \\ &= \arg \max_a [\sum_{m=1}^M \lambda_m f_m(a, \{a_1 a_2 \dots a_A\}, q)]\end{aligned}$$

Maximum Entropy is used to determine optimal values for the coefficients λ that weight the various feature functions relative to each other.

3.2.2 Feature Functions

For the base pattern-based answer selection module described in this section, we use only five basic feature functions.

1. Pattern: The pattern that fired for the given answer and Qtarget. This is a binary feature function: 0 if no pattern fired, 1 if some pattern(s) did.
2. Frequency: Magnini *et al.* (2002) have observed that the correct answer usually has a higher frequency in the collection of answer chunks. Hence we count the number of time a potential answer occurs in the IR output and use its logarithm as a feature. This is a positive continuous valued feature.
3. Expected Answer Class (Qttarget): If the answer's class matches the Qttarget, as derived from the question by Contex, this feature fires (i.e., it has a value of 1). Details of this module are explained in (Hovy *et al.*, 2002). This is a binary-valued feature.
4. Question Word Absent: Usually a correct answer sentence contains a few of the question words. This feature fires if the candidate answer does not contain any of the question words. This is also a binary valued feature.

Word Match: This function is the sum of ITF⁴² values for the words in the question that match identically with words in the answer sentence. This is a positive continuous valued feature.

⁴² ITF = Inverse Term Frequency. We take a large independent corpus & estimate ITF(W) = 1/(count(W)), where W = Word.

3.2.3 Training the system

We use the 1192 questions in the TREC 9 and TREC 10 data sets for training and the 500 questions in the TREC 11 data set for cross-validation. We extract 5000 candidate answers for each question using CONTEX. For each such answer we use the pattern file supplied by NIST to tag answer chunks as either correct (1) or incorrect (0). This is a very noisy way of tagging data: in some cases, even though the answer chunk may be tagged as correct, it may not be supported by the accompanying sentence, while in other cases, a correct chunk may be graded as incorrect, since the pattern file list did not represent an exhaustive list of answers.

3.3 Results

When evaluated against the answer patterns we created for the 413 factoid questions in the TREC-2003 collection, the pattern-based answer selection module described in this section produced 25.18% correct, exact answers. There were 35.59% correct, exact answers in the top 5 candidates returned by a system, with a corresponding MRR score of 28.57%. More details appear in Section 5.

4. STATISTICS-BASED ANSWER SELECTION

Being inspired by the success of noisy-channel-based approaches in applications as diverse as speech recognition (Jelinek, 1997), part of speech tagging (Church, 1988), machine translation (Brown *et al.*, 1993), information retrieval (Berger and Lafferty, 1999), and text summarization (Knight and Marcu, 2002), we have developed a noisy channel model for question answering. This model explains how a given sentence S_A that contains an answer sub-string A to a question Q can be rewritten into Q through a sequence of stochastic operations. Given a corpus of question-answer pairs (Q, S_A), we can train a probabilistic model for estimating the conditional probability $P(Q | S_A)$. Once the parameters of this model are learned, given a question Q and the set of sentences Σ returned by an IR engine, one can find the sentence $S_i \in \Sigma$ and an answer in it $A_{i,j}$ by searching for the $S_{i,A_{i,j}}$ that maximizes the conditional probability $P(Q | S_{i,A_{i,j}})$.

4.1 A Noisy-Channel Model for Question Answering

Assume that we want to explain, for instance, why “1977” in sentence S_A in Figure 1 is a good answer for the question “When did Elvis Presley die?” To do this, we build a noisy channel model that makes explicit how answer sentence parse trees are mapped into questions. Consider, for example, the automatically derived answer sentence parse tree in Figure 1, which associates to nodes both syntactic and shallow semantic, named-entity-specific tags. In order to rewrite this tree into a question, we assume the following generative story:

1. In general, answer sentences are much longer than typical factoid questions. To reduce the length gap between questions and answers and to increase the likelihood that our models can be adequately trained, we first make a “cut” in the answer parse tree and select a sequence of words, syntactic, and semantic tags. The “cut” is made so that every word in the answer sentence or one of its ancestors belongs to the “cut” and no two nodes on a path from a word to the root of the tree are in the “cut”. Figure 1 depicts such a cut graphically.
2. Once the “cut” has been identified, we mark one of its elements as the answer string. In Figure 1, we decide to mark DATE as the answer string (A_DATE).
3. There is no guarantee that the number of words in the cut and the number of words in the question match. To account for this, we stochastically assign to every element s_i in a cut a fertility according to table $n(\phi \mid s_i)$. We delete elements of fertility 0 and duplicate elements of fertility 2, etc. With probability p_1 we also increment the fertility of an invisible word NULL. NULL and fertile words, i.e. words with fertility strictly greater than 1, enable us to align long questions with short answers. Zero fertility words enable us to align short questions with long answers.
4. Next, we replace answer words (including the NULL word) with question words according to the table $t(q_i \mid s_j)$.
5. In the last step, we permute the question words according to a distortion table d , in order to obtain a well-formed, grammatical question.

The probability $P(Q \mid S_A)$ is computed by multiplying the probabilities in all the steps of our generative story (Figure 1 lists some of the factors specific to this computation.) The readers familiar with the statistical machine translation (SMT) literature should recognize that steps 3 to 5 are nothing but a one-to-one reproduction of the generative story proposed in the SMT context by Brown *et al.* (see (Brown *et al.*, 1993) for a detailed mathematical description of the model and the formula for computing the probability of an alignment and target string given a source string).⁴³

To simplify our work and to enable us exploit existing off-the-shelf software, in the experiments we carried out, we assumed a flat distribution for the first two steps in our generative story. That is, we assumed that it is equally likely to take any cut in the tree and equally likely to choose as answer any syntactic/semantic element in an answer sentence.

⁴³ The distortion probabilities depicted in Figure 1 are a simplification of the distortions used in the IBM Model 4 model by Brown *et al.* (1993). We chose this watered down representation only for illustrative purposes. Our QA system implements the full-blown Model 4 statistical model described by Brown *et al.*

Q: When did Elvis Presley die?

S_A: Presley died of heart disease at Graceland in 1977, and the faithful return by the hundreds each year to mark the anniversary.

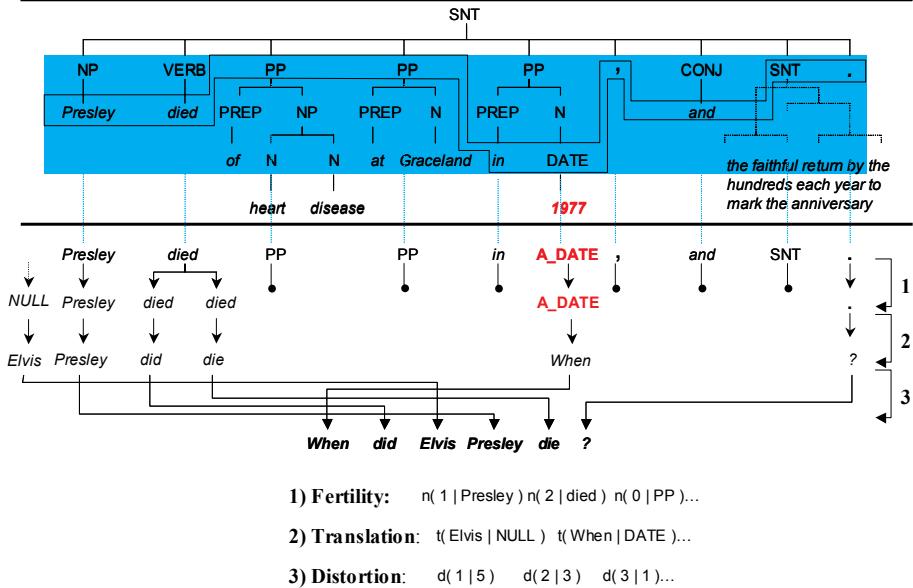


Figure 1. A generative model for question answering.

4.2 Training the Model

Assume that the question-answer pair in Figure 1 appears in our training corpus. When this happens, we know that 1977 is the correct answer. To generate a training example from this pair, we tokenize the question, we parse the answer sentence, we identify the question terms and answer in the parse tree, and then we make a “cut” in the tree that satisfies the following conditions:

- Terms overlapping with the question are preserved as surface text
- The answer is reduced to its semantic or syntactic class prefixed with the symbol “A_”
- Non-leaves, which don’t have any question term or answer offspring, are reduced to their semantic or syntactic class.
- All remaining nodes (leaves) are preserved as surface text.

Condition (a) ensures that the question terms will be identified in the sentence. Condition (b) helps learn answer types. Condition (c) brings the sentence closer to the question by compacting portions that are syntactically far from question terms and answer. And finally the importance of lexical cues around question terms and

answer motivates condition (d). For the question-answer pair in Figure 1, the algorithm above generates the following training example:

Q: When did Elvis Presley die ?

S_A: *Presley died PP PP in A_DATE, and SNT.*

Figure 2 represents graphically the conditions that led to this training example being generated.

Our algorithm for generating training pairs implements deterministically the first two steps in our generative story. The algorithm is constructed so as to be consistent with our intuition that a generative process that makes the question and answer as similar-looking as possible is most likely to enable us learn a useful model. Each question-answer pair results in one training example. It is the examples generated through this procedure that we use to estimate the parameters of our model.

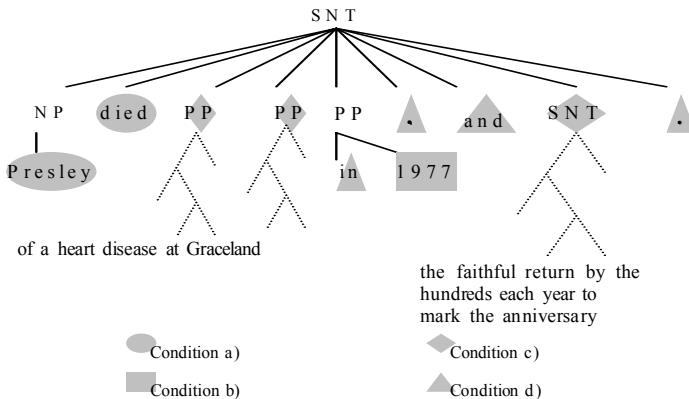


Figure 2. Generation of QA examples for training.

For training, we use TREC 9 and TREC 10 questions (1091) with answer sentences (18618) automatically generated from the corresponding judgment sets. We also use questions (2000) from <http://www.quiz-zone.co.uk> with answer sentences (6516) semi-automatically collected from the web and annotated for correctness by a linguist⁴⁴. To estimate the parameters of our model, we use GIZA, a publicly available statistical machine translation package available at www.clsp.jhu.edu/ws99/projects/mt/.

4.3 Using the Model to Select an Answer String

Assume now that the sentence in Figure 1 is returned by an IR engine as a potential candidate for finding the answer to the question “When did Elvis Presley die?” In this case, we don’t know what the answer is, so we assume that any

⁴⁴ We are grateful to Miruna Ticrea for annotating the question-answer pairs.

semantic/syntactic node in the answer sentence can be the answer, with the exception of the nodes that subsume question terms and stop words. In this case, given a question and a potential answer sentence, we generate an exhaustive set of question-answer sentence pairs, each pair labeling as answer ($A_{_}$) a different syntactic/semantic node. Here are some of the question-answer sentence pairs we consider for the example in Figure 1:

- Q:** When did Elvis Presley die?
S_{A1}: Presley died A_PP PP PP, and SNT.
Q: When did Elvis Presley die?
S_{Ai}: Presley died PP PP in A_DATE , and SNT.
Q: When did Elvis Presley die?
S_{Aj}: Presley died PP PP PP, and NP return by A_NP NP.

If we learned a good model, we would expect it to assign a higher probability to $P(Q | S_{A1})$ than to $P(Q | S_{Ai})$ and $P(Q | S_{Aj})$.

Hence, to select the answer string for a question Q, we exhaustively generate all $Q-S_{i,A_{ij}}$ pairs for each answer sentence S_i , use GIZA to assess $P(Q | S_{i,A_{ij}})$ and pick the answer A_{ij} that maximizes $P(Q | S_{i,A_{ij}})$. Figure 3 depicts graphically the noisy-channel based answer selection.

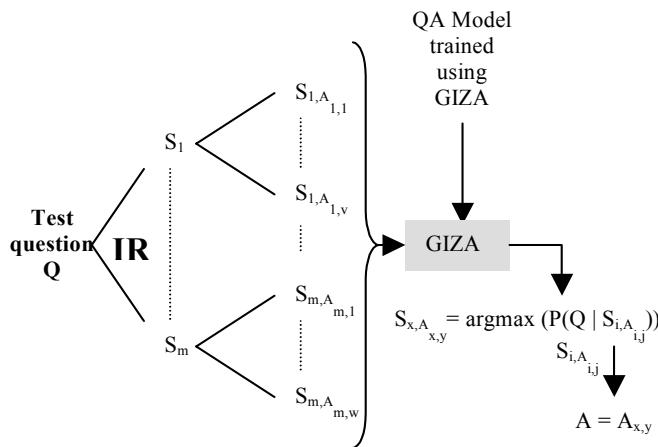


Figure 3. The noisy-channel-based answer selection.

4.4 Results

When evaluated against the answer patterns we created for the 413 factoid questions in the TREC-2003 collection, the statistical-based answer selection module described in this section produced 21.30% correct, exact answers. There

were 31.23% correct, exact answers in the top 5 candidates returned by a system, with a corresponding MRR score of 24.83%. For more details, see Section 5.

5. MAXIMUM ENTROPY TO IMPROVE INDIVIDUAL ANSWER SELECTION MODULES AND COMBINE THEIR OUTPUTS

5.1 *Motivation*

Simple inspection of the outputs produced by our individual components on a development corpus of questions from the TREC-2002 collection revealed several consistent patterns of error:

- The pattern-based answer selection module performs well on questions whose Qtargs are recognizable named entities (NAMES, ORGANIZATIONS, LOCATIONS). However, this module performs less well on questions with more general QTargs (NPs, for example).
- The statistics-based answer selection module does not restrict the types of the answers it expects for a question Qtargs: it assumes that any semantic constituent can be an answer. As a consequence, many of the answers produced by this module may not be exact.
- Overall, all modules made some blatant mistakes. The pattern-based and statistics-based modules in particular sometimes select as top answers strings like “he”, “she”, and “it”, which are unlikely to be good answers for any factoid question one may imagine.

To address these problems, we decided to use the maximum entropy framework to re-rank the answers produced by the answer selection modules and root out the blatant errors. In addition to fixing these blatant errors, we also hoped to also create the means for capitalizing on the strengths of the individual modules. A post-analysis of the TREC 2003 questions showed that if we used an oracle to select the best answer from the pooled top 50 answers returned by each of the three modules, we could produce 77.23% correct, exact answers. This result shows that there is a big opportunity to increase the performance of our end-to-end system by combining in a suitable manner the outputs of the various answer selection modules.

5.2 *Maximum Entropy Framework*

There are several ways of combining the output of various systems. One simple way would be to add or multiply the answer score produced by various systems. However, this method would not exploit several useful dependencies (e.g., answer redundancy or Qtarget matching) that are not explicitly modeled by all the individual answer selection modules. A good approach to overcoming this problem is to use machine learning re-ranking wherein these dependencies and the individual answer scores are modeled as feature functions. We implemented this re-ranking

approach using maximum entropy, which is a linear classifier. We chose to work with a linear classifier because we had very little data for training; in addition, any non-linear classifier is more prone to over-fitting on the training data (principle of bias-variance trade-off).

As in Section 3, we model the problem as a re-ranker as follows (Ravichandran *et al.*, 2003):

$$P(a | \{a_1 a_2 \dots a_A\}, q) = \frac{\exp[\sum_{m=1}^M \lambda_m f_m(a, \{a_1 a_2 \dots a_A\}, q)]}{\sum_{a'} \exp[\sum_{m=1}^M \lambda_m f_m(a', \{a_1 a_2 \dots a_A\}, q)]}$$

where the decision rule for the re-ranker is given by:

$$\begin{aligned} \hat{a} &= \arg \max_a [P(a | \{a_1 a_2 \dots a_A\}, q)] \\ &= \arg \max_a [\sum_{m=1}^M \lambda_m f_m(a, \{a_1 a_2 \dots a_A\}, q)] \end{aligned}$$

5.3 Feature Functions

We use 48 different types of feature functions to adjust the parameters of the Maximum Entropy model. These features can be broadly classified into the following categories.

1. Component-specific: Scores from the individual answer selection module and associated features like the rank of the answer and the presence/absence of the answers produced by the individual answer selection module. We also add features based on the scores produced by the IR module and word overlap between question and answers.
2. Redundancy-specific: Count of candidate answers in the collection. We also add additional features for the logarithm and the square root of the counts.
3. Qtarget-specific: It was observed that some of the answer selection modules answer certain Qtargets better than others. We therefore model a set of features wherein we combine certain classes of Qtarget with the score of the individual answer selection module. This enables the maximum entropy model to assign different model parameters to different types of questions and answer selection module.
4. Blatant-error-specific: We model a set of features based on the development set in an iterative process. These include (negative) features such as “answers usually do not have personal pronouns” and “WHEN questions usually do not contain day of the week as answer”, among others.

5.4 Experiments

We perform the following three sets of experiments.

1. Maximum Entropy re-ranking performed on individual answer selection modules: In this experiment, we turn off all the features in any answer selection module that depends on another answer selection module and we use for re-ranking only the top 50 answers supplied by one answer selection component. Thus this experiment enables us to assess the impact separately on each individual answer selection module of redundancy-, Q_{target} -, and blatant-error-specific features, as well as improved weighting of the features specific to each module.
2. Maximum Entropy re-ranking on all answer selection modules: In this experiment we add all the features described in Section 5.3 and test the performance of the combined system after re-ranking. Re-ranking is performed on the answer set that results from combining the top 50 answers returned by each individual answer selection module. This experiment enables us to assess whether the ME framework enables us to better exploit strengths specific to each answer selection module.
3. Feature Selection: Since we have only 200 training examples and almost 50 features, the training is likely to be highly prone to over-fitting. To avoid this problem we apply feature selection as described in (Della Pietra *et al.*, 1996). This reduces the number of features from 48 to 31.

Table 1 summarizes the results: it shows the percentage of correct, exact answers returned by each answer selection module with and without ME-based re-ranking, as well as the percentage of correct, exact answers returned by an end-to-end QA system that uses all three answer selection modules together. Table 1 also shows the performance of these systems in terms of percentage of correct answers ranked in the top 5 answers and the corresponding MRR scores.

The results in Table 1 show that appropriate weighting of the features used by each answer selection module as well as the ability to capitalize on global features, such as the counts associated with each answer, are extremely important means for increasing the overall performance of a QA system. ME re-ranking led to significant increases in performance for each answer selection module individually. When measured with respect to the ability of our system to find correct, exact answers when returning only the top answer, it accounted for 14.33% reduction in the error rate of the knowledge-based answer selection module; 7.1% reduction in the error rate of the pattern-based answer selection module; and 13.85% reduction in the error rate of the statistical-based answer selection module. Combining the outputs of all systems yields an additional increase in performance; when over-fitting is avoided through feature selection, the overall reduction in error rate is 3.96%. This corresponds to an increase in performance from 45.03% to 47.21% on the top-answer accuracy metric.

Table 1. Performance of various answer selection modules in TextMap, an end-to-end QA system.

Metric	Knowledge-Based		Pattern-Based		Statistical-Based		Base from all syst's followed by ME re-ranking (no feature selection)	Base from all syst's followed by ME re-ranking (with feature selection)
	Base	Base + ME re-ranking	Base	Base + ME re-ranking	Base	Base + ME re-ranking		
Top answer	35.83%	45.03%	25.18%	30.50%	21.30%	32.20%	46.37%	47.21%
Top 5	57.38%	56.41%	35.59%	43.09%	31.23%	40.92%	57.62%	57.62%
MRR	43.88	49.36	28.57	35.37	24.83	35.51	51.07	51.27

The inspection of the weights computed by the ME-based feature selection algorithm shows that our log-linear model was able to lock and exploit strengths and weaknesses of the various modules. For example, the weight learned for a feature that assesses the likelihood of the pattern-based module to find correct answers for questions that have QUANTITY as a Qtarget is negative, which suggests that the pattern-based module is not good at answering QUANTITY-type questions. The weight learned for a feature that assesses the likelihood of the statistics-based module to find correct answers for questions that have an UNKNOWN or NP Qtarget is large, which suggests that the statistical module is better suited for answering these types of questions. The knowledge-based module is better than the others in answering QUANTITY and DEFINITION questions.

These results are quite intuitive: The pattern-based module did not have enough QUANTITY-type questions in the training corpus to learn any useful patterns. The Statistics-based module implemented here does not explicitly use the Qtarget in answer selection and does not model the source probability of a given string being a correct answer. As a consequence, it explores a much larger space of choices when determining an answer compared to the other two modules.

At the moment, the knowledge-based module yields the best individual results. This is not surprising given that it is a module that was engineered carefully, over a period of three years, to accommodate various types of Qtargs and knowledge resources. Many of the resources used by the knowledge-based module are not currently exploited by the other modules: the semantic relations identified by CONTEX; the ability to exploit the paraphrase patterns and advanced forms of reformulation for answer pinpointing; the external sources of knowledge (WordNet; abbreviation lists; etc.); and a significant set of heuristics (see Section 2.4).

Our results suggest that in order to build good answer selection modules, one needs to both exploit as many sources of knowledge as possible and have good methods for integrating them. The sources of knowledge used only by the knowledge-based answer selection module proved to have a stronger impact on the overall performance of our answer selection systems than the ability to automatically train parameters in the pattern- and statistics-based systems, which use poorer representations. Yet, the ability to properly weight the contribution of various knowledge resources was equally important. For example, Maximum Entropy naturally integrated additional features into the knowledge-based answer selection module; a significant part of the 9.2% increase in correct answers reported in Table 1 can be attributed to the addition of redundancy features, a source of knowledge that was unexploited by the base system.

6. CONCLUSIONS

Given their large conceptual similarity, factoid question answering may seem an extension of Information Retrieval, with substrings and sentences replacing documents and document collections. However, the problem has so far stubbornly resisted attempts to find a single, uniform solution that provides as good results as systems employing multiple parallel strategies to perform answer selection.

In this paper, we show how three rather different modules provide somewhat complementary results, and in the paragraphs above, we suggest some reasons for the complementarity. It is of course possible that a uniform solution will be found one day, but whether that solution will rest upon as simple a representation as vector spaces, or be as easy to train up as surface patterns, remains to be seen. After all, as TextMap shows, factoid QA seems to benefit both from relatively deeper notions such as Qtypes and (semantic) parsing as much as from relatively shallower operations such as reformulations of surface patterns (whether created manually or by learning, and whether the starting point is anchor terms or input questions). A uniform representation/approach that encompasses all this has yet to be conceived.

Nonetheless, we remain optimistic that by investigating alternative approaches, we will be able to determine which types of modules work best for which types of problem variations (including parameters such as domain, amount of training data, ease of acquisition of knowledge, types of question asked, complexity of answers required, etc.).

7. REFERENCES

- Berger, A. and J. Lafferty. 1999. Information Retrieval as Statistical Translation. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Information Retrieval*. Berkeley, CA, 222–229.
- Bikel, D., R. Schwartz, and R. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning—Special Issue on NL Learning*, 34, 1–3.
- Brill, E., J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-Intensive Question Answering. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD, 183–189.

- Brown, P.F., J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics* 16(2):79–85.
- Brown, P.F., S. Della Pietra, V. Della Pietra and R. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics* 19(2):263–311.
- Callan, J.P., W.B. Croft, and J. Broglio. 1995. “TREC and Tipster Experiments with Inquiry”. *Information Processing and Management* 31(3), 327–343.
- Church, K.W. 1988. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. *Proceedings of the Second Conference in Applied Natural Language Processing*. 136–143.
- Della Pietra, S., V. Della Pietra, and J. Lafferty. 1995. Inducing Features of Random Fields. *Technical Report* Department of Computer Science, Carnegie-Mellon University, CMU-CS, 95–144.
- Gusfield, D. 1997. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Chapter 6: Linear Time Construction of Suffix Trees, 94–121.
- Hermjakob, U. 1997. *Learning Parse and Translation Decisions from Examples with Rich Context*. Ph.D. dissertation, University of Texas Austin. file: ftp.cs.utexas.edu/pub/mooney/papers/hermjakob-dissertation-97.ps.gz.
- Hermjakob, U. 2001. Parsing and Question Classification for Question Answering. *Proceedings of the Workshop on Question Answering at ACL-2001*. Toulouse, France.
- Hermjakob, U., A. Echihabi, and D. Marcu. 2002. Natural Language Based Reformulation Resource and Web Exploitation for Question Answering. *Proceedings of the TREC-11*. NIST, Gaithersburg, MD.
- Hovy, E.H., U. Hermjakob, C.-Y. Lin, and D. Ravichandran. 2002. Using Knowledge to Facilitate Pinpointing of Factoid Answers. *Proceedings of the COLING-2002 Conference*. Taipei, Taiwan.
- Ittycheriah, A. 2001. *Trainable Question Answering System*. Ph.D. Dissertation, Rutgers, The State University of New Jersey, New Brunswick, NJ.
- Jelinek, F. 1997. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA.
- Knight, K. and D. Marcu. 2002. Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression. *Artificial Intelligence*, 139(1).
- Lin, D. and P. Pantel. 2001. Discovery of Interface Rules for Question Answering. *Journal for Natural Language Engineering* 7(4), 343–360.
- Magnini, B., M. Negri, R. Prevete, and H. Tanev. 2002. Is it the Right Answer? Exploiting Web Redundancy for Answer Validation. *Proceedings of the 40th Meeting of the Association of Computational Linguistics (ACL)*. Philadelphia, PA, 425–432.
- Oh, J.H., K.S. Lee, D.S. Chang, C.W. Seo, and K.S. Choi. 2001. TREC-10 Experiments at KAIST: Batch Filtering and Question Answering. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD, 354–361.
- Ravichandran, D. and E.H. Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. Philadelphia, PA 41–47.
- Ravichandran, D., E.H. Hovy and F.J. Och. 2003. Statistical QA – Classifier vs Re-ranker: What's the difference? *Proceedings of the Association for Computational Linguistics (ACL) Workshop on Multilingual Summarization and Question Answering–Machine Learning and Beyond*. Sapporo, Japan, 69–75.
- Soubbotin, M.M. and S.M. Soubbotin. 2001. Patterns of Potential Answer Expressions as Clues to the Right Answer. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD, 175–182.
- Voorhees, E. 1999. Overview of the Question Answering Track. *Proceedings of the TREC-8 Conference*. NIST, Gaithersburg, MD, 71–81.
- Voorhees, E. 2000. Overview of the Question Answering Track. *Proceedings of the TREC-9 Conference*. NIST, Gaithersburg, MD, 71–80.
- Voorhees, E. 2001. Overview of the Question Answering Track. *Proceedings of the TREC-10 Conference*. NIST, Gaithersburg, MD, 157–165.
- Voorhees, E. 2002. Overview of the Question Answering Track. *Proceedings of the TREC-11 Conference*. NIST, Gaithersburg, MD, 115–123.

5

EVALUATING QUESTION ANSWERING SYSTEMS

ELLEN M. VOORHEES

EVALUATING QUESTION ANSWERING SYSTEM PERFORMANCE

Abstract: The TREC question answering (QA) track was the first large-scale evaluation of open-domain question answering systems. In addition to successfully fostering research on the QA task, the track has also been used to investigate appropriate evaluation methodologies for question answering systems. This chapter reviews the TREC QA track, emphasizing the issues associated with evaluating question answering systems.

1. INTRODUCTION

Evaluation is a fundamental component of the scientific method: researchers form a hypothesis, construct an experiment that tests the hypothesis, and then assess the extent to which the experimental results support the hypothesis. A very common type of experiment is a comparative experiment in which the hypothesis asserts that Method 1 is a more effective solution than Method 2, and the experiment compares the performance of the two methods on a common set of problems. The set of sample problems together with the evaluation measures used to assess the quality of the methods' output form what Tichy (1998) calls a benchmark task.

Benchmarking has a well-established tradition in natural language processing research (Sparck Jones & Galliers, 1996; White, 1999). Information retrieval researchers have used test collections, a form of benchmark task, for forty years (Sparck Jones & Willett, 1997). Other large-scale benchmark evaluations have occurred for speech processing (Pallett, Garofolo, & Fiscus, 2000), information extraction (Chinchor, Hirschman, & Lewis, 1993), parsing (Harrison *et al.*, 1991), machine translation (International Standards for Language Engineering (ISLE), 2000), and summarization (Mani *et al.*, 1998; Harman & Over, 2002). While the details of these evaluations vary greatly, each evolved a common evaluation methodology for its area and created suitable infrastructure such as training materials for further research. The Text REtrieval Conference (TREC) question answering track, the first large-scale evaluation of question answering (QA) systems, was developed to bring these benefits of large-scale evaluation to bear on the QA task.

This chapter examines the issues involved with the evaluation of question answering systems, with the focus on question answering as it is defined within

TREC. Since the TREC track was originally conceived as a place where information retrieval and information extraction researchers could work on a common problem, the next section summarizes the evaluation methodologies in use in those areas. Section 3 motivates the particular definition of question answering used in TREC and describes in detail how the QA evaluations have been implemented. The following section examines the problem of building true equivalents of information retrieval's test collections for the QA task. Section 5 summarizes extensions to the basic QA task that have already been implemented in TREC, while the final section concludes with a list of future extensions that will broaden the evaluations with the goal of increasing the kinds and difficulty of questions that systems can answer.

2. EVALUATION TRADITIONS

The most important factor in the success of an evaluation is the definition of the task to be performed. As Tichy (1998) notes:

The most subjective and therefore weakest part of a benchmark test is the benchmark's composition. . . Hence, benchmark composition is always hotly debated.

To be amenable to evaluation, the task is normally an abstraction of the real-world tasks of the target users of the system so variables affecting performance can be adequately controlled. Nonetheless, the benchmark task must capture salient features of the user task. Sparck Jones (2001) calls this abstracted task the "core competence" required to perform the user task. The metrics used in the benchmark must accurately reflect the real task such that better scores actually reflect better behavior with respect to the user task. The benchmark task must also have an appropriate level of difficulty for the current state of the art. If the task is too difficult, all systems fail and nothing is learned. Similarly, if the task is too easy, all systems do well and again nothing is learned.

Many areas within natural language processing (NLP) have defined and built benchmarks through community evaluations. Examples include TREC for information retrieval (Voorhees, 2000a), MUC for information extraction (Chinchor *et al.*, 1993), SUMMAC (Mani *et al.*, 1998) and DUC (Harman & Over, 2002) for summarization, PARSEVAL for parsing (Harrison *et al.*, 1991), SENSEVAL (Kilgarriff & Palmer, 2000) for word sense resolution, various DARPA-sponsored speech processing evaluations (Pallett *et al.*, 2000), and machine translation evaluations (International Standards for Language Engineering (ISLE), 2000). The number and variety of the different evaluations attest to the benefits they provide. An evaluation draws together a community, providing a forum where research can be discussed and where good research methodology can be established. By demonstrating which are effective techniques, an evaluation facilitates technology transfer both among researchers and between researchers and developers. Broad-based participation in the evaluations amortizes the costs of building evaluation infrastructure (such as training materials and scoring procedures) as well as enabling comparisons among a variety of different approaches.

Despite the benefits that community evaluations afford, they also have negative aspects that must be minimized to make the endeavor worthwhile. The evaluation itself takes resources that could be used in other ways such as supporting additional research. These resources include not only the money required to fund the evaluation, but also researcher time. This effect can be minimized by making the evaluation infrastructure reusable, to leverage the costs of constructing it, and by ensuring that tasks that relate only to the evaluation (such as results reporting) are not onerous. The other major drawback of community evaluations is the danger of an entire community overfitting to the peculiarities of the evaluation task at the expense of real progress on the user task. A benchmark task is by definition a sample from the problem domain; a skewed sample will waste research effort. Efforts to improve effectiveness scores in the nth decimal place are a warning sign of overfitting. The danger of overfitting can be minimized by evolving the task as the field matures.

The particular evaluation paradigms that have evolved from the community evaluations for different NLP tasks are quite different (Sparck Jones & Galliers, 1996; Sparck Jones, 2001; Hirschman, 1998). These differences arise for a variety of reasons: the user tasks are different, the amount of abstraction differs (i.e., the relative importance given to realism vs. control of the variables that affect system performance), and the goals of the evaluations differ. Since the TREC question answering track was established as a place for the information retrieval and information extraction communities to work on a common problem, the evaluation paradigms used in these areas were the biggest influences in developing the task definition and evaluation methodology for question answering. This section reviews the evaluation paradigms in use in these areas as illustrated through TREC and MUC.

2.1 *Information Retrieval Evaluation*

TREC is a workshop series sponsored by the National Institute of Standards and Technology (NIST) and the U.S. Department of Defense. The series is designed to encourage research on text retrieval for realistic applications by providing large test collections, uniform scoring procedures, and a forum for organizations interested in comparing results. The conference has focused primarily on the traditional IR problem of retrieving a ranked list of documents in response to a statement of information need, but also includes other tasks, called tracks, that focus on new areas or particularly difficult aspects of information retrieval. For example, the TREC interactive track, which focuses on human use of retrieval systems, is one of the subjects covered by Hersh's chapter "Evaluating Interactive Question Answering" in this volume.

Text retrieval had a well-established evaluation tradition known as the Cranfield paradigm before TREC began, and TREC has helped to extend and refine this paradigm. The Cranfield paradigm is based on an abstraction of the retrieval process known as a test collection. A test collection consists of three components, a document set, a set of information need statements (called "topics" in TREC), and a set of relevance judgments. The relevance judgments are a list of which documents should

be retrieved for each topic. In TREC, the document sets have consisted mostly of newspaper and newswire articles, though some government documents (the *Federal Register*, patent applications) have also been included. Figure 1 shows three examples of TREC topics.

To use a test collection, a retrieval system produces a ranked list of documents for each topic, where the document lists are ordered by decreasing likelihood that the document should be retrieved for that topic. The effectiveness of the system for a single topic is measured as a function of the ranks at which the relevant documents are retrieved. The effectiveness of a run is the average score of the effectiveness measure computed over the set of topics in the collection. Different retrieval mechanisms are compared by computing the average score for each mechanism using the same collection, and concluding that the mechanism with the better average score is the more effective system.

The Cranfield paradigm makes a number of assumptions: that all relevant documents are equally desirable; that the relevance of one document is independent of the relevance of any other document; that a user's information need is static; that a single set of relevance judgments is representative of the user population; and that all relevant documents are known (i.e., the relevance judgments are complete). In general, none of these assumptions is strictly true, which makes the evaluation of retrieval systems a noisy process. There are two main consequences of this noise. First, evaluation scores are relative, not absolute. Second, there is an (unknown) amount of experimental error associated with each comparison.

Evaluation scores are not absolute because of the Cranfield paradigm's emphasis on the user as the sole judge of a satisfactory response. Since different people have different opinions as to which documents should be retrieved for a topic (Schamber, 1994), there can be no single universally correct response included in a test collection. Instead, test collections use the set of documents produced by one person (the relevance assessor) as an example of a correct response, and systems are evaluated using this sample. The absolute scores of effectiveness measures change when different assessors are used, but relative scores remain stable (Voorhees, 2000b). The stability of the relative scores supports the validity of using test collections to compare runs on the same test collection, though this is the only valid use of such scores.

The experimental error associated with comparing two retrieval runs arises from the fact that the topics in a test collection are a sample of the universe of possible topics. Better performance of one system compared to another on a single test collection might be the result of a lucky sample of topics. An important way of increasing the confidence in a decision that one method is better than another is to compare the two methods using many different test collections. Within a single test, the specifics of the experimental design can be manipulated to decrease experimental error. Using more topics, inherently stable effectiveness measures, and larger differences in scores before concluding that two systems are different all increase the confidence an experimenter can place in the experiment's conclusion (Buckley and Voorhees, 2000; 2002).

<p><num> Number: 051</p> <p><dom> Domain: International Economics</p> <p><title> Topic: Airbus Subsidies</p> <p><desc> Description:</p> <p>Document will discuss government assistance to Airbus Industrie, or mention a trade dispute between Airbus and a U.S. aircraft producer over the issue of subsidies.</p> <p><narr> Narrative:</p> <p>A relevant document will cite or discuss assistance to Airbus Industrie by the French, German, British or Spanish government(s), or will discuss a trade dispute between Airbus or the European governments and a U.S. aircraft producer, most likely Boeing Co. or McDonnell Douglas Corp., or the U.S. government, over federal subsidies to Airbus.</p> <p><con> Concept(s):</p> <ol style="list-style-type: none"> 1. Airbus Industrie 2. European aircraft consortium, Messerschmitt-Boelkow-Blohm GmbH, British Aerospace PLC, Aerospatiale, Construcciones Aeronauticas S.A. 3. federal subsidies, government assistance, aid, loan, financing 4. trade dispute, trade controversy, trade tension 5. General Agreement on Tariffs and Trade (GATT) aircraft code 6. Trade Policy Review Group (TPRG) 7. complaint, objection 8. retaliation, anti-dumping duty petition, countervailing duty petition, sanctions
<p><num> Number: 168</p> <p><title> Topic: Financing AMTRAK</p> <p><desc> Description:</p> <p>A document will address the role of the Federal Government in financing the operation of the National Railroad Transportation Corporation (AMTRAK).</p> <p><narr> Narrative:</p> <p>A relevant document must provide information on the government's responsibility to make AMTRAK an economically viable entity. It could also discuss the privatization of AMTRAK as an alternative to continuing government subsidies. Documents comparing government subsidies given to air and bus transportation with those provided to AMTRAK would also be relevant.</p>
<p><num> Number: 207</p> <p><desc> What are the prospects of the Quebec separatists achieving independence from the rest of Canada?</p>

Figure 1. Example information needs expressed as TREC topics.

2.2 *Information Extraction Evaluation*

In contrast to TREC, the MUC(K) conferences had no previously established evaluation methodology to adopt, and thus evolved the evaluation strategy, as well as the tasks, over the course of the evaluations. The description of the MUC evaluations given here is taken largely from Chinchor *et al.* (1993) and Sparck Jones and Galliers (1996). For more detailed information about MUC, see the MUC web site (SAIC, 2001).

The MUC conferences were concerned with “message understanding”, that is, the ability of systems to extract desired information from running text. The first MUCK conference was small and informal, with no specific task definition and no quantitative scoring. MUCK-II formalized the task as template filling from tactical naval operations messages, where templates had slots for the primary features of the incidents reported in the messages. Participants scored their own runs by hand following a set of scoring guidelines. MUC-3 significantly increased the difficulty of the task and the rigor of the evaluation. The task was still a template-filling task, using foreign news reports of Latin American terrorist activity rather than naval messages. The change to foreign news reports rather than naval messages required systems to handle a much larger vocabulary (especially proper nouns, which were essentially unlimited) and a broader range of linguistic expressions. Figure 2 shows the skeleton of an ARSON event type template.

Participants received 1300 reports to use as a training and development set, and then received a second set of 300 reports from the same time period to use as a test set. Systems were required to decide which of the 300 reports in the test set actually reported on terrorist activities and generate filled templates describing those that did. Runs were scored using an automated scoring program that computed scores based on the number of matching slots between the system-generated templates and manually produced answer key templates. MUCs 4 and 5 were similar to MUC-3, though the specific effectiveness metrics used continued to evolve and MUC-5 used a different domain (joint ventures) and news reports in both English and Japanese. In the final two MUCs, additional tasks were added and the templates for the basic task were streamlined. These changes were inspired by the research community’s desire to have more diagnostic evaluation included within MUC and to develop more general-purpose natural language processing components. The desire for more general NLP components was addressed by the two new new tasks, named entity recognition and coreference resolution. The interest generated by these tasks within MUC fostered the development of fast, largely domain-independent recognizers for these linguistic phenomena.

Superficially, MUC’s answer key templates and TREC’s relevance judgments are similar in that they both are produced by humans for input to a scoring procedure. However, the assumptions as to what each set of answer data actually represents are very different, and this has a profound effect on how evaluation results can be interpreted. An answer key template is assumed to enumerate all possible correct responses.

0. MESSAGE: ID
1. MESSAGE: TEMPLATE
2. INCIDENT: DATE
3. INCIDENT: LOCATION
4. INCIDENT: TYPE
5. INCIDENT: STAGE OF EXECUTION
6. INCIDENT: INSTRUMENT ID *
7. INCIDENT: INSTRUMENT TYPE *
8. PERP: INCIDENT CATEGORY
9. PERP: INDIVIDUAL ID
10. PERP: ORGANIZATION ID
11. PERP: ORGANIZATION CONFIDENCE
12. PHYS TGT: ID
13. PHYS TGT: TYPE
14. PHYS TGT: NUMBER
15. PHYS TGT: FOREIGN NATION
16. PHYS TGT: EFFECT OF INCIDENT
17. PHYS TGT: TOTAL NUMBER
18. HUM TGT: NAME
19. HUM TGT: DESCRIPTION
20. HUM TGT: TYPE
21. HUM TGT: NUMBER
22. HUM TGT: FOREIGN NATION
23. HUM TGT: EFFECT OF INCIDENT
24. HUM TGT: TOTAL NUMBER

Figure 2. Format of an ARSON event template from MUC-4. Starred entries indicate slots that are not applicable for an ARSON incident.

MUC's answer key templates were complex, containing optional slots and lists of alternative valid fillers for many slots, but, by definition, a filled template was correct if and only if it could be generated from the answer key template. This assumption allows the scores computed from the answer key to be absolute scores, though creating good answer keys is very expensive and some realism is sacrificed since humans don't always agree on correct fills. As described above, differences in human opinions as to which documents should be retrieved in response to a query are factored into the Cranfield paradigm where the relevance judgments are assumed to represent a sample of a correct response. Scores computed from relevance judgments are relative since different scores are computed for different samples (i.e., when a different relevance assessor is used).

3. THE TREC QA TRACK

The QA track was started in 1999 (TREC-8) to support research on methods that would allow systems to move away from document retrieval toward information retrieval. Document retrieval systems' ability to work in any domain was considered an important feature to maintain. At the same time, the technology that had been developed by the information extraction community appeared ready to exploit. Thus one goal of the track was to define a task that would appeal to both the information retrieval and information extraction communities. A second goal of the track was to explore appropriate evaluation methodologies for the QA task in light of the different methodologies used by the two communities. To this end, the first running of the track investigated whether different people have different opinions as to what constitutes an acceptable answer, and, if so, how those differences affect QA evaluation.

This section describes the implementation of the TREC QA tracks. The basic task is defined first, followed by a description of how the test questions were collected. Section 3.3 describes the procedure used by the human assessors to judge the quality of a response and analyzes the effect of using different assessors. The section concludes with a brief summary of the evaluation results.

3.1 *The Task*

The basic task used within the QA track is defined similarly to the task the MURAX system performed (Kupiec, 1993). The MURAX system used an on-line encyclopedia as a source of answers for closed-class questions, which Kupiec defined as "a question stated in natural language, which assumes some definite answer typified by a noun phrase rather than a procedural answer." The TREC task used a large corpus of documents rather than an encyclopedia as a source of answers, but was otherwise the same. Since the documents consisted mostly of newswire and newspaper articles, the domain was essentially unconstrained. However, only closed-class questions were used, so answers were generally entities familiar to information extraction systems.

Participants were given a document collection and a test set of questions. The test was a blind test: participants were not allowed to change their systems once they received the questions, and the answers were required to be produced completely automatically. The questions were generally fact-based, short-answer questions as shown in Figure 3. For the first two years, each question was guaranteed to have at least one document in the collection that explicitly answered it. There was no such guarantee in the TREC 2001 track. Systems could return 'NIL' as a response to indicate that the corpus did not contain an answer.

Participating systems returned a ranked list of five [*document-id, answer-string*] pairs per question such that each answer string was believed to contain an answer to the question. Answer strings were limited to either 50 or 250 bytes depending on the run type (TREC 2001 allowed only the 50 byte option), and could either be extracted from the corresponding document or automatically generated from information

contained in the document. Human assessors read each string and decided whether the string actually did contain an answer to the question in the context provided by the document. Taking document context into account allowed a system that correctly derived a response from a document that was in error (for example, a typographical error or a misreported fact such as an incorrect number of casualties) to be given full credit for its response.

Given a set of judgments for the strings, the score computed for a submission was the mean reciprocal rank (MRR). An individual question received a score equal to the reciprocal of the rank at which the first correct response was returned, or zero if none of the five responses contained a correct answer. The score for a submission was then the mean of the individual questions' reciprocal ranks. The reciprocal rank has several advantages as a scoring metric. It is closely related to the average precision measure used extensively in document retrieval. It is bounded between zero and one, inclusive, and averages well. A run is penalized for not retrieving any correct answer for a question, but not unduly so. However, the measure also has some drawbacks. The score for an individual question can take on only six values (0, .2, .25, .33, .5, 1). Question answering systems are given no credit for retrieving multiple (different) correct answers. Also, since systems were required to return at least one response for each question, systems could receive no credit for realizing they did not know the answer.

- How many calories are there in a Big Mac?
- What two US biochemists won the Nobel Prize in medicine in 1992?
- Who is the voice of Miss Piggy?
- Where is the Taj Mahal?
- What costume designer decided that Michael Jackson should only wear one glove?
- In what year did Joe DiMaggio compile his 56-game hitting streak?
- What language is commonly used in Bombay?
- Where is Rider College located?
- Name a film in which Jude Law acted.

Figure 3. Example questions from the TREC question answering track.

3.2 Building Test Materials

Table 1 summarizes the test materials used in the TREC QA tracks. The document corpus used in TREC-8 consisted of the set of documents on TREC disks 4 and 5. TREC-9 and TREC 2001 used a larger corpus consisting of all the news documents on all five TREC disks.

Table 1. Data used in the TREC question answering track.

COLLECTION	TREC-8	TREC-9	TREC 2001
Number of documents	528,000	979,000	979,000
MB of document text	1904	3033	3033
Document sources	TREC disks 4-5	news from TREC disks 1-5	news from TREC disks 1-5
Number of questions	200	693	500
Question sources	FAQ Finder log, assessors, participants	Encarta log, Excite log	filtered MSNSearch and AskJeeves logs

A more substantive difference between the tracks was the source of the test questions; each running of the track used more realistic questions than the preceding version. In TREC-8, the majority of the questions were created expressly for the track, and thus tended to be back-formulations of a statement in a test document. In TREC-9, the questions were selected from an Encarta log that contained actual questions, and a raw Excite log. Since the raw Excite log did not contain many grammatically well-formed questions, question authors used the Excite log as a source of ideas for actual questions, and created questions without referring to test documents. The resulting test set of questions was much more difficult than the TREC-8 set, mainly because the TREC-9 set contained many more high-level questions such as *Who is Colin Powell?*.

In TREC 2001, the source of questions was a set of filtered MSNSearch logs and AskJeeves logs. Raw logs were automatically filtered (at Microsoft and AskJeeves) to select queries that contained a question word (e.g., what, when, where, which, etc.) anywhere in the query; that began with modals or the verb to be (e.g., are, can, could, define, describe, does, do, etc.); or that ended with a question mark. Additional human filtering of the logs then removed queries that were not in fact questions; questions that asked for a list of items; procedural questions; questions that asked for the location of something on the web (e.g., pictures of someone); yes/no questions; and questions that were obviously too current for the document collection (e.g., questions about Britney Spears, etc.). The spelling, punctuation, and occasionally the grammar of the queries selected to be in the final question set were manually corrected, but except for a very few questions, the content of the question was precisely as it appeared in the log.

Once a set of candidate questions was selected, assessors searched the document corpus looking for an answer to the question. For TRECs 8 and 9, a candidate was discarded if no answer could be found. For TREC 2001, 49 of the test questions have no known correct answer in the document collection.

NIST has made no attempt to control the relative number of different types of questions in the test set from year to year. Instead, the distribution of question types in the final test set has reflected the distribution in the source of questions. The number of questions that asked for a definition was dramatically greater in the TREC 2001 track than in previous years. While a large fraction of definition questions is “real” in that the filtered MSNSearch and AskJeeves logs contain many definition questions, definition questions are not a good fit for way the TREC task is defined (e.g., a deployed QA system could just return a dictionary definition rather than search a large corpus of documents for a concise definition). NIST will need to exert more control over the distribution of question types in future tracks, though this runs the risk of skewing the test set in other directions.

3.3 Assessing System Responses

The rationale for using human assessors to evaluate a task is to incorporate the perceptions of the end-users of the technology into the evaluation to the greatest extent possible. To investigate the effect of using human judges as the basis of a QA evaluation, each TREC-8 question was independently judged by three different assessors. The separate judgments were combined into a single judgment set through adjudication for the official track evaluation, but the individual judgments were later used to measure the effect of differences in judgments on systems’ scores.

Assessors were trained for the QA task before they did any judging. The purpose of the training was to motivate the assessors’ task and provide general guidance on the issues that would arise during assessing rather than to drill the assessors on a specific set of assessment rules. To begin, each assessor was given the following instructions.

Assume there is a user who trusts the answering system completely, and therefore does not require that the system provide justification in its answer strings. Your job is to take each answer string in turn and judge if this answer string alone were returned to the trustful user, would the user be able to get the correct answer to the question from the string.

Assessors then judged four sample questions whose response strings were concocted by NIST staff to illustrate various fundamentals of QA judging:

- that answer strings contain fragments of text that are not necessarily grammatically correct;
- that answer strings do not need to contain justification;
- that the assessor is to judge the string, not the document from which the string was drawn;
- that document context must be taken into account; and
- that the string must be responsive to the question.

Document context is vital for questions whose answers change over time. For example, responses to questions phrased in the present tense (*Who is the prime minister of Japan?*) are judged as correct or incorrect based on the date of the document associated with the response. Requiring that the answer string be

responsive to the question addresses a variety of issues. Answer strings that contain multiple entities of the same semantic category as the correct answer but do not indicate which of those entities is the actual answer (e.g., a list of names in response to a *who* question) are judged as incorrect. Certain punctuation and units are also required. Thus “5.5 billion” is not an acceptable substitute for “5.5 billion”, nor is “500” acceptable when the correct answer is “\$500”. Finally, unless the question specifically states otherwise, correct responses for questions about a famous entity must refer to the famous entity and not to imitations, copies, etc. For example, two separate TREC-8 questions asked for the height of the Matterhorn (i.e., the Alp) and the replica of the Matterhorn at Disneyland. Correct responses for one of these questions are incorrect for the other.

The results of the TREC-8 assessment process demonstrated that assessors do have legitimate differences of opinion as to what constitutes an acceptable answer even for the deliberately constrained question types used in the track (Voorhees & Tice, 2002b). Two prime examples of where such differences arise are the completeness of names and the granularity of dates and locations. For example, for the question *When did French revolutionaries storm the Bastille?*, some assessors accepted “July 14”, others accepted “1789”, and everyone accepted “July 14, 1789”. Assumed context also mattered, and differed among assessors. For the question *Where is the Taj Mahal?* one of the three assessors accepted Atlantic City, NJ (home of the Taj Mahal casino) as an acceptable response even in light of the guidelines that stated replicas and imitations should not be used as answers for questions about a famous entity. For this assessor, the casino was sufficiently well-known to be an entity in its own right.

Fortunately, as with document retrieval evaluation, the relative scores between QA systems remain stable despite differences in the judgments used to evaluate them. Indeed, for the purpose of a comparative evaluation, the judgment sets that consisted of a single judge’s opinion for each question were not only equivalent to one another, but were also equivalent to the adjudicated judgment set (Voorhees & Tice, 2000b).

One change to the assessment procedure was made after TREC-8, the addition of an “unsupported” judgment. There were a number of instances during the TREC-8 judging when an answer string contained the correct answer, but that answer could not possibly have been determined from the document returned. For example, the correct answer for *Who is the 16th President of the United States?* is Abraham Lincoln. One of the answer strings returned contained Abraham Lincoln, but the associated document discussed Lincoln’s Gettysburg Address. The document does not even mention that Lincoln was president, let alone that he was the sixteenth president. Since the TREC-8 task guidelines did not specifically require that the document returned with the answer string support the string as the answer, these cases were judged as correct in TREC-8, even though the assessors were uncomfortable doing so. In later tracks, the guidelines required that the document returned with the answer string actually support the answer contained in the string. If the answer string did not contain a correct answer, the response was judged incorrect. If the string did contain a correct answer, but the document did not support that answer (such as the Lincoln/Gettysburg Address example above), the response was judged unsupported. Otherwise, the

response was judged correct. Two MRR scores can thus be computed for each QA run, a strict score in which unsupported answers are considered incorrect, and a lenient score in which unsupported answers are considered correct.

3.4 Evaluation Results

Table 2 gives the number groups that participated in each running of the QA track and the number of runs of various types NIST received. While a detailed analysis of the results from these runs is beyond the scope of this chapter, a recap of the highlights from each track is given below. Full details of all QA track results can be found in the TREC proceedings available online at trec.nist.gov/pubs.html.

Table 2. Number of participants and runs in the TREC QA tracks.

DATA COLLECTION	TREC-8	TREC-9	TREC 2001
Number of participants	20	28	36
Number of 50-byte runs	20	34	67
Number of 250-byte runs	25	44	—

The most accurate of the TREC-8 systems were able to answer more than 2/3 of the questions. When an answer was found at all, it was likely to be highly ranked. Not surprisingly, allowing 250 bytes in a response is an easier task than limiting responses to 50 bytes. Indeed, traditional passage retrieval techniques are effective when a response as long as 250 bytes is acceptable (Singhal *et al.*, 2000).

Most TREC-8 participants used a version of the following general approach to the question answering problem. The system first attempted to classify a question according to the type of its answer as suggested by its question word. For example, a question that begins with “who” implies a person or an organization is being sought, and a question beginning with “when” implies a time designation is needed. Next, the system retrieved a small portion of the document collection using standard text retrieval technology and the question as the query. The system performed a shallow parse of the returned documents to detect entities of the same type as the answer. If an entity of the required type was found sufficiently close to the question’s words, the system returned that entity as the response. If no appropriate answer type was found, the system fell back to best-matching-passage techniques.

The absolute value of the scores for TREC-9 systems was lower than for TREC-8, but in fact the TREC-9 systems were significantly improved (the TREC-9 task was much more difficult because of the change in the source of questions). The improvement in QA systems came from refinements to the individual steps of the general strategy described above rather than an entirely new approach. TREC-9 systems were better at classifying questions as to the expected answer type, and used a wider variety of methods for finding the entailed answer types in retrieved passages. Many systems used WordNet (Fellbaum, 1998) as a source of related words for the

initial query and as a means of determining whether an entity extracted from a passage matched the required answer type.

Many systems continued to refine this approach in the TREC 2001 track. However, the TREC 2001 track also saw a resurgence of approaches that relied on simpler pattern matching methods using very large corpora (generally the web) rather than sophisticated language processing. The idea exploited in the massive data approach is the fact that in a large enough data source a correct answer will usually be repeated often enough to distinguish it from the noise that happens to occasionally match simple patterns; for an example of this approach see (Dumais *et al.*, 2002).

4. BUILDING QA TEST COLLECTIONS

The primary way TREC has been successful in improving document retrieval performance is by creating appropriate test collections for researchers to use when developing their systems. Obtaining an adequate set of relevance judgments for a large collection can be time-consuming and expensive, but once a test collection is created researchers can automatically evaluate the effectiveness of a retrieval run. Such reusable test collections facilitate research by individual research teams, accelerating the pace of improvement and amortizing the cost of building the collection over a much larger group. One of the key goals of the QA track was to build a reusable QA test collection that is, to devise a means to evaluate a QA run that uses the same document and question sets but was not among the runs judged by the assessors.

Unfortunately, the judgment sets produced by the assessors for the TREC QA tracks do not constitute a reusable test collection because the unit that is judged is the entire answer string. Different QA runs very seldom return exactly the same answer strings, and it is quite difficult to determine automatically whether the difference between a new string and a judged string is significant with respect to the correctness of the answer. Document retrieval test collections do not have this problem because the unique identifiers assigned to the documents makes it trivial to decide whether or not a document retrieved in a new run has been judged.

The problems caused by not having a reusable test collection are illustrated by the plight of researchers at the University of Ottawa. The pair of runs they submitted to the TREC-8 QA track were mis-numbered, and the mistake was not discovered until judging was complete. They were unable to get an official score for their correctly numbered run because they could not map their answer strings to the judged answer strings. As an approximate mapping, they marked any string that completely contained a string that was judged correct in the official adjudicated judgment set as being correct, and all other strings were marked as incorrect (Martin and Lankester, 2000). This is a conservative mapping that will almost never mark a string correct that would have been judged incorrect at the expense of marking as incorrect many strings that would have been judged correct. As such it provides a lower bound on the score of the unjudged run, but is sufficiently biased against unjudged runs to make the scores between judged and unjudged runs incomparable.

MITRE also developed an approximate mapping technique based on their work with reading comprehension tests (Breck *et al.*, 2000). A human created an answer key for the question set, and strings are then marked based on word recall. If an answer string matches a sufficiently high proportion of the words in an answer key entry, the string is marked correct and otherwise it is marked incorrect. Their analysis demonstrated a high correlation between low word recall scores and judgments of incorrect, and high word recall scores and judgments of correct across the set of runs submitted to the QA track.

NIST created an approximate mapping algorithm similar in spirit to word recall (Voorhees & Tice, 2000a). A set of Perl string-matching patterns is created (by a human) from the set of strings that the assessors judged correct. An answer string that matches any pattern for its question is marked correct, and is marked incorrect otherwise. The patterns have been created such that almost all strings that were judged correct would be marked correct, sometimes at the expense of marking as correct strings that were judged incorrect. Patterns are constrained to match at word boundaries and case is ignored.

NIST created the pattern sets for the TREC-8 and TREC-9 tracks. An average of 1.7 patterns per question was created for the TREC-8 test set, with sixty-five per cent of the questions having a single pattern. The TREC-9 set averaged 3.5 patterns per question with only forty-five per cent of the questions having a single pattern. The TREC 2001 pattern set was created as a volunteer service to the community by Ken Litkowski of CL Research. They averaged 2.83 patterns per question with forty-nine per cent of the questions having a single pattern.

Using the patterns to evaluate the TREC-8 runs produced differences in the relative scores of different systems that were comparable to the differences caused by using different human assessors. However, unlike the different judgments among assessors, the patterns misjudge broad classes of responses classes that are usually precisely the cases that are difficult for the original QA systems. For example, an answer string containing multiple entities of the same semantic category as the answer will always be judged correct if the correct answer is mentioned. Document context is also not taken into account. A more sophisticated pattern matching technique could eliminate some of this problem by conditioning patterns by the document ids that are in the judged set, so, for example, questions such as *Who is the prime minister of Japan?* only accept certain names for certain time ranges. But this does not solve the problem for completely wrong context that happens to contain a correct string.

Using patterns to evaluate the TREC-9 runs produced larger differences in the relative scores, especially for 50-byte runs. One way to quantify differences in relative scores is to use a measure of association between system rankings produced when MRR scores are first computed using the human judgments and then computed using the patterns. Kendall's is one such measure of association that has been used in earlier evaluations (Voorhees, 2000b; Voorhees and Tice 2000a). Kendall's computes the distance between two rankings as the minimum number of pairwise adjacent swaps to turn one ranking into the other. The distance is normalized by the number of items being ranked such that two identical rankings produce a correlation of one, the correlation between a ranking and its perfect

inverse is negative one, and the expected correlation of two rankings chosen at random is zero. For TREC-8, the computed between the rankings using all runs was .96. For TREC-9, the computed between the rankings of 250-byte runs was .94, while it was only .89 for 50-byte runs. This smaller correlation is probably the result of a several factors. The TREC-8 human judgment scores were produced using an adjudicated judgment set that was a combination of three different assessors' judgments, and therefore was of particularly high quality. The TREC-9 judgment set is known to contain more errors (i.e., mistaken judgments caused by the assessor hitting the wrong button or similar error) than the TREC-8 judgment set, and these errors could reduce the correlation with the pattern judgments. The more ambiguous questions in the TREC-9 test set were also harder to create patterns for. Nonetheless, the decrease in correlation suggests that concerns about pattern-based judgments are well-founded. Researchers need a fast, reliable method to receive feedback as to the relative quality of alternative question answering techniques. The full benefits of test collection evaluation will not be fully realized for the QA task until more satisfactory techniques for evaluating new runs are devised.

5. QA TASK EXTENSIONS

Each running of the TREC QA track has focused on the basic task described in Section 3. Later tracks extended the basic task by either adding additional tasks or including additional aspects in the main task. In particular, the TREC-9 question set included different expressions of the same underlying question to explore the effect of syntactic variation on systems' question processing, and the TREC 2001 track added an additional list task. Table 3 lists the various tasks that were included in the three QA tracks.

Table 3. Tasks within the TREC QA tracks.

TREC-8	50 byte snippets 250 byte snippets
TREC-9	50 byte snippets 250 byte snippets test set contained multiple syntactic variants of some questions
TREC 2001	50 byte snippets list task: return a set of instances drawn from multiple documents context task: answer a series of related questions

The context task that was part of the TREC 2001 track was an attempt to test systems' abilities to maintain state information to answer a series of questions. However, the task was not successful in that the results were completely dominated by the systems' abilities to answer the particular question type regardless of previous context. Thus the context task is not discussed further in this chapter.

5.1 Question Variants

The TREC-9 question set contained 500 questions drawn from search engine logs, plus an additional 193 questions that were syntactic variants of an original question. The purpose of the the syntactic variants was to investigate whether QA systems are robust to the variety of different ways a question can be phrased. Once the first 500 questions were selected, NIST assessors were given a subset of the questions and asked to create “natural” variants of the question. The intent was that the variant should have the same semantic meaning of the original, as well as be phrased in a way that a native English speaker might ask the question. For example, the test set contained four variants for the question *What is the tallest mountain?*: *What is the world's highest peak?*, *What is the highest mountain in the world?*, *Name the highest mountain.*, and *What is the name of the tallest mountain in the world?*. The 193 variants included variants for fifty-four different original questions, with a range of one to seven new questions per original.

While the intent had been for each variant to have identical semantics, assessment demonstrated that this was not always the case. Sometimes rewording a question caused the focus of the question to change slightly, so that some answer strings were acceptable for some variants but not others. For example, the assessor accepted “November 29” as a correct response for *What is Dick Clark's birthday?*, but required the year as well for *When was Dick Clark born?*. Similarly, the question *Where is the location of the Orange Bowl?* had many more acceptable responses than did *What city is the Orange Bowl in?*. A few variants had no answer in the document collection and had to be discarded from the evaluation. For example, the original question *Who invented silly putty?* Had “General Electric scientist” as an answer, but that answer was not acceptable for the variant *What is the name of the inventor of silly putty?*.

Figure 4 shows a plot of the average score for each question in a variant set. The average score for a question is the mean of the reciprocal rank scores averaged over the thirty-three runs that used the 50-byte limit on responses and using strict evaluation. The y-axis in the plot is the average score and the x-axis represents the different variant sets. The variant sets are identified by the question number of the original question that was used to generate the variants.

Many variant sets show little variability in the average score. Generally, the average score for each of these variants is low, indicating that the underlying information being sought was difficult to obtain no matter how the question was phrased. A few variant sets did have a wide range of average scores. Frequently the difference was caused by different word choices in the variants. For example, the variant set generated from question 413 asked for the location of the U.S. headquarters of Proctor & Gamble. The variant with the lowest average score was question 725 which used “corporate offices” instead of “headquarters”. For the variant set generated from question 440, the original question was *Where was Poe born?*, which had a much higher score than any of the variants that all asked for Poe's birthplace. The unintentional change in focus of some variants also made differences in average scores. “New Jersey” was an acceptable (and common)

answer to Question 448, *Where is Rider College located?*, but it was not acceptable for the variant *Rider College is located in what city?*.

5.2 The List Task

One of the goals for the TREC 2001 QA track was to require systems to assemble an answer from information located in multiple documents. Such questions are harder to answer than the questions used in the basic task since information duplicated in the documents must be detected and reported only once. TREC 2001 introduced a list task to accomplish this goal.

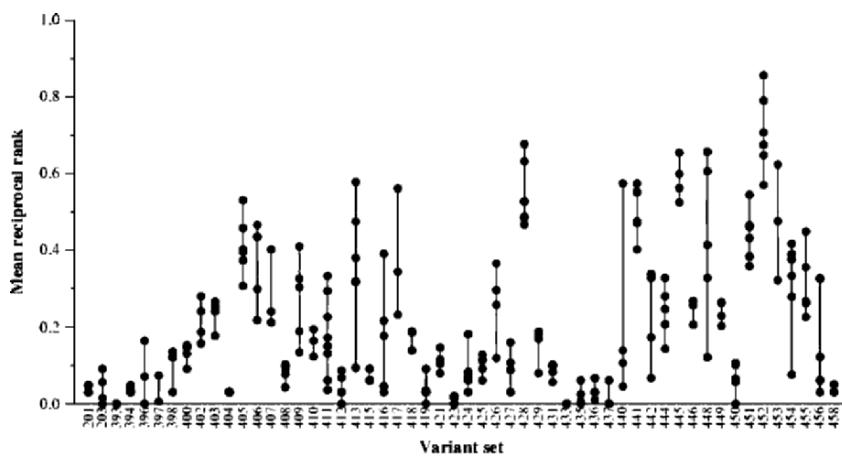


Figure 4. Average reciprocal rank for each question variant computed over thirty-three 50-byte runs using strict evaluation. The x-axis represents different variant sets, identified by the question number of the original question from which the variants were generated.

Each question in the list task specified the number of instances of a particular kind of information to be retrieved, such as in the example questions shown in Figure 5. Each instance was guaranteed to obey the same constraints as an individual answer in the main task and was judged in the same manner as a response in the main task: each true instance was no more than 50 characters long; some document was explicit that it was an instance of the desired type; each answer string had to have an associated document that supported the answer; answer strings had to be responsive; etc. The document collection was guaranteed to contain at least the target number of instances. Systems returned an unordered list of [*document-id*, *answer-string*] pairs where each pair represented a single instance. The list could contain no more than the target number of instances.

- Name 4 U.S. cities that have a “Shubert” theater.
- Name 30 individuals who served as a cabinet officer under Ronald Reagan.
- Who are 6 actors who have played Tevye in “Fiddler on the Roof”?
- Name 4 countries that can produce synthetic diamonds.
- What are 9 novels written by John Updike?

Figure 5. Example list task questions.

The 25 questions used as the list task test set were constructed by NIST assessors and NIST staff since there were not enough appropriate questions in search engine logs. The assessors were instructed to construct questions whose answers would be a list of entities (people, places, dates, numbers) such that the list would not likely be found in a reference work such as a gazetteer or almanac. Each assessor was asked to create one small question (five or fewer expected answers), one large question (between twenty and forty expected answers), and two medium questions (between five and twenty expected answers). The assessors searched the document collection to find as complete a list of instances as possible. The target number of instances to retrieve was then selected such that the document collection contained more than the requested number of instances, but more than one document was required to meet the target. A single document could contain multiple instances, and the same instance might be repeated in multiple documents.

Judgments of correct, incorrect, or not supported were made individually for each *[document-id, answer-string]* pair. The assessor was given one list at a time, and while judging for correctness he also marked a set of responses as distinct. The assessor arbitrarily chose any one of a set of equivalent responses to mark as the distinct one, and marked the remainder as not distinct. Incorrect responses were always marked as not distinct (the assessment software enforced this), but unsupported responses could be marked distinct.

Since answer strings could be up to fifty bytes long, a single string might contain more than one instance. The track guidelines specified that the left-most instance in a string was always counted as the instance of record for that string. For example for the question *Name 9 countries that import Cuban sugar.*, the string “China and Russia imported Cuban sugar” was counted as an instance of China only. If another answer string in the list was “China imports”, one of the two responses would be marked as distinct for China, and Russia still would not be counted as a retrieved instance.

List results were evaluated using accuracy, the number of distinct responses divided by the target number of instances. Since unsupported responses could be marked distinct, the reported accuracy is a lenient evaluation.

Ten groups submitted eighteen runs for the list task. The best performing system had an average accuracy of 76%, suggesting that the list task as defined is feasible with current technology. Note, however, that the list task questions were intrinsically easier than the questions in the main task since the questions were specifically created to ask for a list of concrete entities. Most systems found at least one instance for most questions. Each system returned some duplicate responses, but duplication was not a major source of error for any of the runs. (Each run contained

many more wrong responses than duplicate responses.) With just 18 runs, there is not enough data to know if the lack of duplication is because the systems are good at recognizing and eliminating duplicate responses, or if there simply wasn't all that much duplication in the document set.

6. FUTURE QA EVALUATIONS

Evaluating competing technologies on a common problem set is a powerful way to improve the state of the art and hasten technology transfer. As the first large-scale evaluation of domain-independent question answering systems, the TREC question answering track brings the benefits of large-scale evaluation to bear on the question answering task. The track has stimulated research on open-domain question answering and has created a foundation on which future evaluations can build. The data used in the TREC tracks, including questions, answer patterns, sentences containing answers, and evaluation scripts are available on the TREC web site (National Institute of Standards and Technology, 2002).

To date, the TREC QA track has focused on a narrow piece of the general question answering problem: answering factoid questions. To continue to support the QA community, the evaluation must broaden with the goal of increasing the kinds and difficulty of the questions that systems can answer. The original TREC task focused on factoid questions because systems at the time could not handle other question types for unrestricted domains, but also because responses to such questions can be judged as being simply right or wrong, greatly facilitating the evaluation of the systems. More advanced question types do not appear to be amenable to binary judgments, however. A question whose answer requires the synthesis of multiple facts, or a summary of events, or the explication of the relationship between two entities cannot usefully be judged simply right or wrong. The next step, therefore, is to investigate evaluation paradigms that incorporate finer-grained assessment of answer quality.

As of this writing, the U.S. Department of Defense Advanced Research and Development Activity (ARDA) AQUAINT program (ARDA, 2002) is initiating a series of pilot studies to investigate evaluation techniques for several different types of questions whose answers are more complex than closed-class questions. The first pilots will focus on definition questions (including generic *Who is X?* questions), relationship questions (*How are X and Y related?* with no predefined set of relationship types), and opinion questions (recognition of explicit and implicit opinions in text). Each pilot study will first define a workable evaluation plan for its area and then have a few (approximately 5) sites produce results that will be evaluated according to the plan. A workable evaluation plan includes:

1. A description of the characteristics of the type of question under consideration in enough detail that question authors can create (or recognize) questions of the appropriate type;
2. A characterization of the expected response from the system when it receives such a question; and

3. A specific description of how system responses are scored what input (if any) is needed from human assessors, what metric is used once that input is obtained, etc.

The expected result from the pilots is an evaluation of the evaluation plan: Was the evaluation meaningful in that scores correlate with the quality of a systems' response (both good and bad)? Was the scoring procedure consistent across scorers? Was the scoring procedure reasonable to apply? Tasks with successful pilots will be incorporated into future TREC tracks where the task can draw a larger number of participants.

7. REFERENCES

- ARDA. (2002). *Advanced question and answering for intelligence (AQUAINT)*. <http://www.icarda.org/InfoExploit/aquaint/index.html>.
- Breck, E., Burger, J., Ferro, L., Hirschman, L., House, D., Light, M. & Mani, I. (2000). How to evaluate your question answering system every day ... and still get real work done. In *Proceedings of the second international conference on language resources and evaluation (LREC-2000)* (Vol. 3, pp. 1495-1500).
- Buckley, C. & Voorhees, E. M. (2000). Evaluating evaluation measure stability. In N. Belkin, P. Ingwersen & M. Leong (Eds.), *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 33-40).
- Chinchor, N., Hirschman, L. & Lewis, D. D. (1993). Evaluating message understanding systems: An analysis of the third Message Understanding Conference (MUC-3). *Computational Linguistics*, 19(3), 409-449.
- Dumais, S., Banko, M., Brill, E., Lin, J. & Ng, A. (2002). Web questions answering: Is more always better? In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 291-298).
- Fellbaum, C. (Ed.) (1998). *Wordnet: An electronic lexical database*. The MIT Press.
- Harman, D. & Over, P. (2002). The DUC summarization evaluations. In *Proceedings of the international conference on human language technology*. (In press.)
- Harrison, P., Abney, S., Black, E., Flickenger, D., Gdanee, C., Grishman, R., Hindle, D., Ingria, R., Marcus, M., Santorini, B. & Strzalkowski, T. (1991). Evaluating syntax performance of parser/grammars of English. In *Proceedings of the workshop on evaluating natural language processing systems*. Association for Computational Linguistics.
- Hirschman, L. (1998). Language understanding evaluations: Lessons learned from MUC and ATIS. In *Proceedings of the first international conference on language resources and evaluation (LREC)* (pp. 117-122). Granada, Spain.
- International Standards for Language Engineering (ISLE). (2000). *The ISLE classification of machine translation evaluations*. <http://www.isi.edu/natural-language/mteval/cover.html>.
- Kilgarriff, A. & Palmer, M. (2000). Introduction to the special issue on SENSEVAL. *Computers and the Humanities*, 34(1-2), 1-13.
- Kupiec, J. (1993). MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In R. Korfage, E. Rasmussen & P. Willett (Eds.), *Proceedings of the sixteenth annual international acm sigir conference on research and development in information retrieval* (pp. 181-190). (Special issue of the SIGIR FORUM.)
- Mani, I., House, D., Klein, G., Hirschman, L., Obrst, L., Firmin, T., Chrzanowski, M. & Sundheim, B. (1998). *The TIPSTER SUMMAC text summarization evaluation* (Tech. Rep. No. MTR 98W0000138). McLean, Virginia: MITRE, Washington C3 Center.
- Martin, J. & Lankester, C. (2000). Ask Me Tomorrow: The NRC and University of Ottawa question answering system. In *Proceedings of the eighth Text REtrieval Conference (TREC-8)* (pp. 675-683). (NIST Special Publication 500-246.)
- National Institute of Standards and Technology. (2002). *The Text Retrieval Conference web site*. (<http://trec.nist.gov>)

- Pallett, D. S., Garofolo, J. S. & Fiscus, J. G. (2000). Measurements in support of research accomplishments. *Communications of the ACM*, 43(2), 75-79.
- SAIC. (2001). *Introduction to information extraction*. www.itl.nist.gov/iaui/894.02/related_projects/muc.
- Schamber, L. (1994). Relevance and information behavior. *Annual Review of Information Science and Technology*, 29, 3-48.
- Singhal, A., Abney, S., Bacciani, M., Collins, M., Hindle, D. & Pereira, F. (2000). AT&T at TREC-8. In E. Voorhees & D. Harman (Eds.), *Proceedings of the eighth Text REtrieval Conference (TREC-8)*. (NIST Special Publication 500-246. Electronic version available at <http://trec.nist.gov/pubs.html>)
- Sparck Jones, K. (2001). Automatic language and information processing: Rethinking evaluation. *Natural Language Engineering*, 7(1), 29-46.
- Sparck Jones, K. & Galliers, J. R. (1996). *Evaluating natural language processing systems*. Springer.
- Sparck Jones K., & Willett P. (1997) Evaluation. In K. Sparck Jones & P. Willett Eds.), *Readings in information retrieval* (167-174). Morgan Kaufmann.
- Tichy, W. F. (1998). Should computer scientists experiment more? *Computer*, 31(5), 32-40.
- Voorhees, E. & Harman, D. (Eds.). (2000). *Proceedings of the eighth Text REtrieval Conference (TREC-8)*. (NIST Special Publication 500-246. Electronic version available at <http://trec.nist.gov/pubs.html>)
- Voorhees, E. M. (2000a). Special issue: The sixth Text REtrieval Conference (TREC-6). *Information Processing and Management*, 36(1).
- Voorhees, E. M. (2000b). Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing and Management*, 36, 697-716.
- Voorhees, E. M. & Buckley, C. (2002). The effect of topic set size on retrieval experiment error. In *Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 316-323).
- Voorhees, E. M. & Tice, D. M. (2000a). Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 200-207).
- Voorhees, E. M. & Tice, D. M. (2000b). The TREC-8 question answering track evaluation. In E. Voorhees & D. Harman (Eds.), *Proceedings of the eighth Text REtrieval Conference (TREC-8)* (pp. 83-105). (NIST Special Publication 500-246. Electronic version available at <http://trec.nist.gov/pubs.html>)
- White, J. (1999, April). *Evaluation and assessment techniques*. www-2.cs.cmu.edu/~ref/mlim/.

WILLIAM HERSH

EVALUATING INTERACTIVE QUESTION ANSWERING

Abstract: This volume is filled with a variety of innovative approaches to helping users answer questions. In much of the research, however, one part of the solution is missing, namely the user. This chapter describes evaluation of interactive question answering with a focus on two initiatives: the Text REtrieval Conference (TREC) Interactive Track and studies in the medical domain. As is seen, there is considerable overlap between the two in terms of the model underlying the research and the methods used.

1. INTRODUCTION

The history of information retrieval (IR) is known for a dichotomy of evaluative research between those who perform predominantly systems-oriented evaluation and those who carry out user-centered evaluation. This dichotomy is also evidenced in the area of question answering, as shown by the techniques and metrics used in the TREC Question Answering Track (see Voorhees' chapter). The involvement of users in IR evaluation is not new, and much early research focused on how well users could answer questions with IR systems. Egan *et al.* (Egan, Remde *et al.*, 1989) piloted this approach with a statistics textbook, finding significant performance differences with changes in the user interface. Mynatt *et al.* (Mynatt, Leventhal *et al.*, 1992) used a similar approach to assess the ability of college students to find answers to questions in an online encyclopedia.

At this time, all interactive question answering evaluation has been carried out using traditional IR systems, i.e., document retrieval systems. Little research has been done looking at how real users might fare with the types of advanced question-answering systems described elsewhere in this volume. This chapter describes the interactive evaluation research that has been done, its results and limitations, and the challenges ahead.

2. FROM DOCUMENT RETRIEVAL TO QUESTION ANSWERING: THE TREC INTERACTIVE TRACK

The TREC Interactive Track has been studying users and their interaction with documents and queries commonly used in TREC almost since its inception (Over, 2000). The Interactive Track was originally formed for TREC-3. In the first few

years, the track used the queries and documents from the main TREC tasks, ad hoc retrieval and routing.

With TREC-6, the Interactive Track began to focus on measures of user performance more independent from ad hoc and routing. In particular, an *instance recall* task was employed (Hersh, 2001). Recognizing that the number of relevant documents retrieved was not a good indicator of a user's performance with an IR system, the track moved toward a question answering approach. The specific approach chosen was to measure how well users could define instances (called "aspects" at TREC-6 but "instances" subsequently) of a topic. For example, How many stars have been discovered by the Hubble Telescope? or How many countries import Cuban sugar? The database chosen for searching was the 1991-1994 *Financial Times* database of the overall TREC collection, consisting of 210,158 documents (articles). In the searching experiments, users were asked to save the documents that contained instances and record the specific instances identified. As with the development of document relevance judgments, assessors at NIST devised a set of "correct" instances and the documents they occurred in based on the results submitted from participating sites. This allowed the primary dependent variable, instance recall, to be measured:

$$\text{Instance recall} = \frac{\text{number of relevant instances in saved documents}}{\text{number of relevant instances for topic}} \quad (1)$$

One of the challenges for the Interactive Track has been to define the degree of standardization of experiments across sites. Certainly one of the values of TREC in general has been the standardization of tasks, queries, and test collections. Counter to this, however, are not only the varying research interests of different participating groups, but also the diversity of users, settings, and available systems. The design of the TREC-6 Interactive Track attempted to measure the effect of the system, minimizing differences between sites and searchers. Site-specific effects were minimized by use of a common control system, the NIST zPRISE system, which was installed by every group. Searcher-specific effects were minimized by a Latin squares design for topics and the order in which they were searched by each subject. Unfortunately, the use of a common control system introduced problems of its own. Not only were most participating groups interested in comparing their own systems against controls of their own choosing, but a statistical analysis showed that the approach used in TREC-6 was incapable of filtering out site-specific interactions (Lagergren and Over, 1998). In the TREC-7 Interactive Track, the common control system was abandoned and each site was free to use its own control system.

Beginning with TREC-6, a number of other standardized procedures were undertaken. In particular, common data were collected about each user and each of their searches. The data elements, which were collected from common questionnaires used by all sites or extracted from search logs (Table 1). Searchers were also given an explicit amount of time for searching: 20 minutes in TREC-6 and TREC-8 and 15 minutes in TREC-7 and TREC-9. Sites were also asked to provide a narrative of one user's search on one topic each year.

Table 1. Common Variables Collected by All Research Groups for the TREC Interactive Track.

Variable	Definition
<u>Demographic</u>	
Gender	Male vs. female
Age	In years
<u>Experience</u>	
Years	Years experience of online searching (1-least; 5-most)
Point	Experience with point-and-click interface (1-least; 5-most)
Catalogs	Experience using online library catalogs (1-least; 5-most)
CDROM	Experience using CD-ROMS (1-least; 5-most)
Online	Experience searching commercial online systems (1-least; 5-most)
WWW	Experience searching Web (1-least; 5-most)
Frequency	How often searching done (1-least; 5-most)
Enjoy	How enjoyable searching is (1-least; 5-most)
<u>Cognitive traits</u>	
FA1	Controlled associations test to assess associational fluency (TREC-7 only)
<u>Satisfaction post-topic</u>	
Familiar	User familiar with topic (1-least; 5-most)
EasyStart	Search was easy to get started (1-least; 5-most)
EasyUse	Search was easy to do (1-least; 5-most)
Satisfied	User was satisfied with results (1-least; 5-most)
Confident	User had confidence that all instances were identified (1-least; 5-most)
TimeAdequate	Search time was adequate (1-least; 5-most)
<u>Satisfaction post-system</u>	
SysEasyLearn	System was easy to learn to use (1-least; 5-most)
SysEasyUse	System was easy to use (1-least; 5-most)
SysUnderstand	User understand how to use system (1-least; 5-most)
<u>Satisfaction post-experiment</u>	
Understand	User understand nature of experimental task (1-least; 5-most)
TaskSim	Task had similarity to other searching tasks (1-least; 5-most)
TaskDiff	Systems were different from each other (1-least; 5-most)

<u>Search mechanics</u>	
Saved	Documents saved by user
DocRec	Document recall (relevance defined as having one or more instance)
Time	Time in seconds for search
Terms	Number of unique terms used for topic
Viewed	Number of documents viewed for topic
Seen	Number of documents seen for topic
Cycles	Number of search cycles for topic

The instance recall approach was used for three successive years of the Interactive Track. With TREC-9, a more explicit question answering approach was adopted (Hersh and Over, 2000). While questions from the non-interactive TREC-8 Question Answering track were considered for use, preliminary assessment found them too easy for the interactive task. Instead, a number of candidate questions were developed by the participating research groups inspired more by the data at hand than by any systematic considerations. Four types were considered and tested to gauge their suitability:

- Find any n X's (e.g., Name three US Senators on committees regulating the nuclear industry.)
- Comparison of two specific X's (e.g., Do more people graduate with an MBA from Harvard Business School or MIT Sloan?)
- Find the largest/latest/... n Xs (e.g., What is the largest expenditure on a defense item by South Korea?)
- Find the first or last X (e.g., Who was the last Republican to pull out of the nomination race to be the candidate of his/her party for US president in 1992?)

In the end, eight questions were chosen, four of each of the first two types. Examples included Name three US national parks where one can find redwoods and Is Denmark larger or smaller in population than Norway? Questions of the last two types were deemed too difficult to find or create and, given their "superlative" nature, seemed less likely for users to be able to do successfully.

The TREC-9 Interactive Track continued many of the other standard approaches adopted for TREC-6 through TREC-8. One change was an expansion of the collection used to include more collections from the Associated Press, *Wall Street Journal*, *San Jose Mercury News*, *Los Angeles Times*, and the Foreign Broadcast Information Service. However, the same user questionnaires were employed and the searching logs were again processed to obtain most of the data listed in Table 1. The correct answers to the questions were determined by NIST assessors, with searcher results judged as either completely correct, partially correct, or incorrect. Most sites required the answer to be completely correct in their data analysis.

Most participating research groups have assessed a small number of research questions in their experiments over the years in the Interactive Track. These can be

grouped into three categories: assessing the similarity of results obtain in batch and user evaluations, evaluating relevance feedback, and determining the effect of presentation of search results.

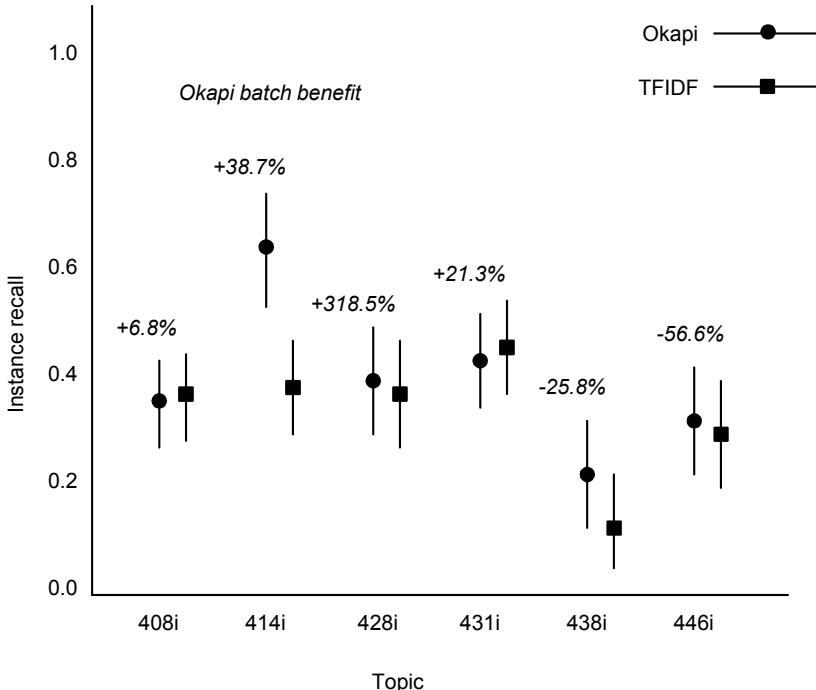
2.1 *Do batch and user evaluations give the same results?*

In TREC-8 and TREC-9, Hersh and colleagues addressed the question of whether results from batch searching experiments are comparable to those obtained by real users. That is, are the results found when batch-style experiments and measures like mean average precision (MAP) are used congruent with those obtained by real users with retrieval tasks using the same systems. A continuing debate in the information retrieval (IR) field is whether the results obtained by batch evaluations, from the non-interactive laboratory setting, can be generalized to real searchers. Much evaluation research dating back to the Cranfield studies (Cleverdon and Keen, 1966) and continuing through TREC (Harman, 1993) has been based on entering fixed query statements from a test collection into an IR system in batch mode with measurement of recall and precision of the output. It is assumed that this is an effective and realistic approach to determining the system's performance (Sparck-Jones, 1981). Some have argued against this view, maintaining that the real world of searching is more complex than can be captured with such studies. They point out that relevance is not a fixed notion (Meadow, 1985), interaction is the key element of successful retrieval system use (Swanson, 1977), and relevance-based measures do not capture the complete picture of user performance (Hersh, 1994).

To test this question, three-part experiments were performed. First, previous TREC Interactive Track data were used to determine the "best-performing" system under batch experiments. This was done by creating a test collection that designated documents with instances as relevant and running traditional batch experiments to generate MAP results. These experiments found that Okapi weighting (Robertson and Walker, 1994) (used in TREC-8) and Okapi plus pivoted normalization weighting (Singhal, Buckley *et al.*, 1996) (used in TREC-9) performed much better than "baseline" term frequency/inverse document frequency (TF*IDF) weighting (Salton, 1991). Next, the interactive user experiments were carried out, with the best-performing system serving as the "experimental" intervention and the TF*IDF system as the "control." In the final step, the batch experiments were repeated with the data generated from the new interactive experiments, to verify the results held with the topics employed in the user experiments. (The experiments had to be done this way to accommodate the general approach of TREC in determining results from data generated in current experiments. Thus, the batch experiments calling for the data from the interactive experiments could not take place until those experiments had been performed.)

In the TREC-8 instance recall task, Hersh *et al.* (Hersh, Turpin *et al.*, 2000a) found in the first batch experiment that Okapi weighting performed best, achieving an 81% better mean average precision (MAP) than TF*IDF weighting. For the user experiments that followed, Okapi weighting served as the experimental system and TF*IDF weighting as the control. All subjects used a standard natural language

interface to the MG system (Witten, Moffat *et al.*, 1994). Highly experienced searchers were again employed. The results showed a better average instance recall for Okapi over TF*IDF weighting, 39 vs. 33%. However, as seen in Figure 1, all of the benefit was seen in one query, and the results were not statistically significant. In the final batch experiments, the Okapi weighting achieved a 17.6% benefit with the data that had actually been used by the interactive searchers. These experiments also found a positive and statistically significant linear relationship for instance recall with the following variables:



*Figure 1. Instance recall (mean and standard deviation) achieved by systems employing Okapi and TF*IDF weighting for six interactive queries (Hersh, Turpin *et al.*, 2001). The benefit for the Okapi weighting in batch experiments is listed as a percentage value above the interactive mean and standard deviation.*

- The number of documents saved as having instances by users
- Document recall (defining a relevant document as one having one instance or more)
- The number of documents selected for viewing by the user (seen)

The same experiment was repeated in TREC-9 by Hersh *et al.* (Hersh, Turpin *et al.*, 2000b), using the question-answering task. In the initial batch experiments here, however, a combination of Okapi plus pivoted normalization (PN) weighting achieved the highest MAP over the TF*IDF baseline, 58%. The user experiments thus employed Okapi plus pivoted normalization weighting as the experimental

system. The same user types as well as the same natural language interface to the MG system were used. In the user experiments, the proportion of questions answered correctly was virtually identical between systems, 43.0% for TF*IDF weighting and 41.0% for Okapi plus pivoted normalization weighting (Table 2). In both the TREC-8 and TREC-9 experiments, use of the same user interface precluded measuring user satisfaction differences between systems. The results of the final batch experiments were comparable to those of the TREC-8 experiment: the Okapi plus pivoted normalization weighting achieved a 31.5% MAP improvement over the TF*IDF weighting.

Table 2. Question answering results in the TREC-9 Interactive Track for two weighting schemes (Hersh, Turpin et al., 2000b).

Question	TF*IDF			Okapi + Pivoted Normalization		
	Searches	#Correct	%Correct	Searches	#Correct	%Correct
1	13	3	23.1%	12	1	8.3%
2	11	0	0.0%	14	5	35.7%
3	13	0	0.0%	12	0	0.0%
4	12	7	58.3%	13	8	61.5%
5	12	9	75.0%	13	11	84.6%
6	15	13	86.7%	10	6	60.0%
7	13	11	84.6%	12	10	83.3%
8	11	0	0.0%	14	0	0.0%
Total	100	43	43.0%	100	41	41.0%

A follow-up experiment attempted to determine why the “improved” (Okapi weighting in TREC-8 and Okapi plus pivoted normalization weighting in TREC-9) systems failed to lead to better performance over the TF*IDF system in the hands of real users (Turpin and Hersh, 2001). Analysis of the search output found that the users retrieved more relevant documents as well as more instances in the top 10 documents with the improved systems. However, the overall number of relevant documents retrieved was comparable for both systems, although the improved system retrieved fewer nonrelevant documents. Further analysis showed that the average length of documents retrieved by the improved systems was over twice as long. This was to be expected, given that the document length normalization in Okapi and pivoted normalization weighting give additional weight to longer documents. The most likely explanation from these experiments was that the TF*IDF system provided users access to shorter documents, albeit fewer relevant ones, enabling users to achieve comparable performance with the outcomes measures: instance recall and correctness of answering questions.

These results have significant meaning for the myriad of batch experiments performed in TREC and other IR evaluations. Namely, the MAP of systems without human users in the loop is a very limited and possibly misleading measure of performance. Of course the TREC Interactive Track experiments are limited to

certain tasks, topics, users, and databases. However, further research must determine how well the results of batch experiments correlate with the performance of real users.

2.2 Relevance Feedback

Other groups have used the TREC Interactive Track to assess different research questions. Belkin *et al.* (2000) have attempted to determine the utility to users of relevance feedback. Again taking the perspective that something shown to have value in batch experiments will not necessarily benefit users, this group has assessed a variety of approaches to relevance feedback. Their initial experiments (TREC-5) looked at the value of basic relevance feedback, that is, the ability to designate documents as relevant and to add some of their terms to the query to generate a new list of document output. Their observational study found that users preferred interactive to automatic relevance feedback; that is, they wanted to use relevance feedback as a device to add new terms to their queries rather than have it blindly suggest more documents.

The next experiments (TREC-6) added more features to the system, including the ability to specify the use of documents for feedback in a positive or negative way. In addition, the user was given control over terms added to the query; that is, relevance feedback was used as a *term suggestion* device. Users were randomized to have access to only the positive feedback functions or to both the positive and negative feedback functions. There was a trend toward improved instance recall with the latter system as well as the number of documents saved, but the results were not statistically significant, possibly owing to the small sample size. A follow-up experiment with similar interfaces (TREC-7) showed similar results.

Further experiments (TREC-8) compared the use of positive relevance feedback in a term-suggestion manner with a technique called *local context analysis* (LCA) (Xu and Croft, 1996), which derives a set of terms from the top-ranking documents to suggest a different group of terms for use in the feedback process. Because the latter does not require the users to make relevance judgments about documents, there is less work for users. Although the instance recall results were virtually identical for the two systems, users selected more suggested terms in the LCA system for use in their queries, perhaps because they spent less time reviewing and designating relevant documents in the regular relevance feedback approach.

Other researchers have investigated relevance feedback as well. Robertson *et al.* (1998) found that it slightly enhanced instance recall in the Okapi system, though a simple TF*IDF approach outperformed both. Yang *et al.* (2000) assessed different approaches to relevance feedback. The main focus of their experiments was to compare the use of the whole document for relevance feedback versus selection by the user of specific passages in documents. They hypothesized that the user would do better with the latter, more focused approach. Their results showed no significant difference, but a definite trend in favor of the whole-document approach.

2.3 Presentation of Search Results

Other research in the TREC Interactive Track has looked at the presentation of search results. Wu *et al.* (2000) used the instance recall tasks to assess the values of users' clustering of documents (TREC-7) or classifications of terms (TREC-8). The value of clustering was assessed by comparing a system that grouped the document list by clusters with one that just provided a straight ranked list. The list approach yielded better instance recall than the cluster approach, but the difference was not statistically significant. Feedback showed that users liked the cluster approach but had difficulty understanding exactly how it worked.

To analyze the value of term classification, Wu *et al.* chose an approach that prompted the user to categorize the output of the original natural language query by one of the search terms. For the selected term, a set of related terms was shown from WordNet, a lexical resource (Fellbaum, 1998). WordNet contains *synsets* that represent all the polysems for a given word. After the user chose a category, the documents retrieved in the original search that matched each polysem in the set were shown under that term. Again, the results showed no differences between the two approaches.

In the TREC-9 question-answering interactive task, Wu *et al.* (2001) identified a user interface difference that affected results. They compared two systems that differed in how they displayed their document output (or "surrogates"). One system displayed "answer-indicative sentences" that showed three sentences containing the highest number of words from the query. The other displayed the title and first 20 words of the document. Users had a higher rate of answering questions correctly with the former system (65% vs. 47%). A follow-up experiment with a different locally developed test collection showed comparable results.

Other researchers have investigated presentation of search results as well. In the TREC-7 instance recall task, Allan *et al.* (1997) found that separate "aspect windows" allowing a user to maintain different queries and result sets did not improve performance over a baseline TF*IDF system, but a 3D interface highlighting documents belonging to the different instances produced an improvement that was statistically significant. Belkin *et al.* (2000) compared displays that showed single documents versus six documents tiled at a time, with no differences in the results.

3. QUESTION ANSWERING IN THE MEDICAL DOMAIN

The medical domain has also been an area that has attracted interest in question answering. The rationale for these studies is that the usual goal of using an IR system is to find an answer to a question. While the user must obviously find relevant documents to answer that question, the quantity of such documents is less important than whether the question is successfully answered. In fact, recall and precision can be placed among the many factors that may be associated with ability to complete the task successfully.

3.1 Success in answering questions

The first study by this group using the question answering approach compared Boolean versus natural language searching in the textbook *Scientific American Medicine* (Hersh, Elliot *et al.*, 1994). Thirteen medical students were asked to answer ten short-answer questions and rate their confidence in their answers. The students were then randomized to one or the other interface and asked to search on the five questions for which they had rated confidence the lowest. The study showed that both groups had low correct rates before searching (average 1.7 correct out of 10) but were mostly able to answer the questions with searching (average 4.0 out of 5). There was no difference in ability to answer questions with one interface or the other. Most answers were found on the first search to the textbook. For the questions that were incorrectly answered, the document with the correct answer was actually retrieved by the user two-thirds of the time and viewed more than half the time.

Another study compared Boolean and natural language searching of MEDLINE with two commercial products, CD Plus (now Ovid) and Knowledge Finder (Hersh, Pentecost *et al.*, 1996). These systems represented the ends of the spectrum in terms of using Boolean searching on human-indexed thesaurus terms (CD Plus) versus natural language searching on words in the title, abstract, and indexing terms (Knowledge Finder). Sixteen medical students were recruited and randomized to one of the two systems and given three yes/no clinical questions to answer. The students were able to use each system successfully, answering 37.5% correct before searching and 85.4% correct after searching. There were no significant differences between the systems in time taken, relevant articles retrieved, or user satisfaction. This study demonstrated that both types of system can be used equally well with minimal training.

Further research by this group expanded the analysis to include nurse practitioner (NP) students and a myriad of other possible factors that could influence searching. Each of these studies used only one IR system, the Ovid system used to search MEDLINE but with links to about 80 full-text journals.

The first study focused solely on NP students and used multiple-choice questions from the *Medical Knowledge Self-Assessment Program* (MKSAP, American College of Physicians, Philadelphia) (Rose, 1998). Each of the 24 subjects answered three out of the eight questions used. Before searching, 25 of the 72 questions (34.7%) were answered correctly, and after searching the total of correct responses increased to 42 out of 72 (58.3%).

The second study assessed both medical and NP students, with 29 subjects answering three questions each out of a group of 30 (Hersh, Crabtree *et al.*, 2000). The questions, which were phrased in a short-answer format, were obtained from three sources: MKSAP, the Cochrane Database of Systematic Reviews, and those captured from shadowing physicians in actual clinical practice (Gorman and Helfand, 1995). The main success-related results showed that medical students scored higher before and after searching but that both groups improved their scores by the same amount.

In the final study, probably the largest study of medical searchers to date, 66 medical and NP students searched five questions each (Hersh, Crabtree *et al.*, 2002).

This study used a multiple-choice format for answering questions that also included a judgment about the evidence for the answer. Subjects were asked to choose from one of three answers:

- Yes, with adequate evidence
- Insufficient evidence to answer question
- No, with adequate evidence

Both groups achieved a pre-searching correctness on questions about equal to chance (32.3% for medical students and 31.7% for NP students). However, medical students improved their correctness with searching (to 51.6%), whereas NP students hardly did at all (to 34.7%). Table 3 shows that NP students changed with searching from incorrect to correct answers as often as they did from correct to incorrect.

Table 3. Cross-Tabulation of Number and Percentage of Answers Correct vs. Incorrect Before and After Searching for All (A), Medical (M), and NP (N) Students (Hersh, Crabtree et al., 2002).*

Post search		
Presearch	Incorrect	Correct
Incorrect		
A	133 (41.0%)	87 (26.9%)
M	81 (36.3%)	70 (31.4%)
N	52 (51.5%)	17 (16.8%)
Correct		
A	41 (12.7%)	63 (19.4%)
M	27 (12.1%)	45 (20.2%)
N	14 (13.9%)	18 (17.8%)

*Percentages represent percent correct within each group of students.

Another group to assess ability to use an IR system successfully has been Wildemuth, Friedman, and associates. These researchers used a question-answering approach to assess INQUIRER, a system containing factual databases on several biomedical topics in a medical school curriculum (Wildemuth, deBlek et al., 1995). The study yielded the following findings:

- Personal knowledge scores (ability to answer questions before searching) were low.
- Database-assisted scores (ability to answer questions with searching) were substantially higher.
- There was no relationship between personal knowledge scores and database-assisted scores, that is, a searcher's prior knowledge was not associated with better ability to search.
- Database-assisted scores persisted at a higher level long after the course was over and personal knowledge scores had returned to lower levels.

A further analysis of the final point (in the bacteriology domain) showed that personal knowledge scores were low before the bacteriology course (12%), rose right after the course (48%), and decreased 6 months later (25%). However, the database-assisted scores rose linearly over the three assessments from 44% to 57% to 75% (deBliek, Friedman *et al.*, 1994). Thus information relevant to a problem can be retrieved from a database long after the ability to directly recall it from memory has faded.

A subsequent study from this research showed that when INQUIRER was incorporated into a clinical simulation, there was a trend toward better performance when the system was used and a statistically significant benefit when the record containing the correct answer was displayed (Abraham, Friedman *et al.*, 1999). Another study compared a Boolean and a hypertext browsing interface (Friedman, Wildemuth *et al.*, 1996). While both led to improvements in database-assisted scores, the hypertext interface showed a small but statistically significant benefit.

Osheroff and Bankowitz (1993) have also looked at the ability to answer clinical questions, although this investigation considered actual questions generated by real users. Of the 50 questions from eight users, 40% were deemed to be fully answered, 32% partially answered, and 28% not answered at all by the six information resources provided. More than half of the unanswered questions could be answered elsewhere by searchers using other information sources.

A variant of the task-oriented approach has also been applied to health care consumers searching the Web (Eysenbach and Kohler, 2002). This study of 17 users given health questions to search on found an answer was successfully obtained in 4 to 5 minutes. Although participants were aware of quality criteria for health-related Web sites (e.g., source, professional design, scientific language), they did not appear to apply these criteria when actually visiting sites.

3.2 *Factors Associated with Success*

The most comprehensive analysis of factors relating to searching success has been carried out by Hersh and colleagues. Three studies have investigated the factors associated with successful searching by medical and NP students (Rose, 1998; Hersh, Crabtree *et al.*, 2000; Hersh, Crabtree *et al.*, 2002). This section focuses on the most recent of these studies because it was the largest and most comprehensively analyzed.

The first task in looking for factors associated with successful searching is to develop a model of the various factors. Actual operationalization of such a model requires modification of the model to the specific aspects of the study. For example, there needs to be an ultimate measure of success in using the IR system: namely, one or more dependent (outcome) variables. Likewise, there will be specific independent (predictor) variables specific to the study, such as the demographic characteristics of the users, the type of retrieval system, and the instruments used to assess various user attributes.

Table 4 presents the model of factors for the study described in the third study by Hersh and colleagues (Hersh, Crabtree *et al.*, 2002), which was refined from their initial two studies (Rose, Crabtree *et al.*, 1998; Hersh, Crabtree *et al.*, 2000). In that model, the dependent variable was successful answering of the clinical question. The independent variables were grouped in several different categories. The first consisted of identifiers for the user and question searched, as well as the order of the question. The next category covered demographic variables, some of which were generic (e.g., age and sex), while others were specific to the study population (e.g., school enrolled and years worked as a nurse). There were also categories for computer experience, computer attitudes, and searching experience. The searching experience factors included not only general amounts of literature and Web searching but also specific knowledge of and experience with advanced features of MEDLINE.

*Table 4. Model of Factors Associated with Successful Question-Answering by Means of MEDLINE Searching and Evidence Judgment (Hersh, Crabtree *et al.*, 2002).*

Variable	Definition
<i>Basic</i>	
ID	User ID
Question	Question number
<i>Demographics</i>	
School	School student enrolled (medical or NP)
Age	(Years)
Sex	(Male, female)
MedSpec	Medical specialty will be entering (from list, M students only)
YrsNurse	Years worked as a nurse (years, NP students only)
NPSpec	Nurse practitioner specialty (from list, NP only)
<i>Computer experience</i>	
CompHrs	Computer usage per week (hours)
ProdSW	Use productivity software once a week (yes, no)
OwnPC	Own a personal computer (yes, no)
Modem	Personal computer has a modem (yes, no)
Internet	Personal computer connects to Internet (yes, no)
<i>Computer attitudes</i>	
PracHard	Practice easier or harder with computers (easier, harder)
EnjComp	Enjoy using computers (yes, no)
<i>Searching experience</i>	
LitSrch	Literature searches per month (number)
WebSrch	Web searches per month (number)

Variable	Definition
WebMed	Web searches for medical information per month (number)
TrainEBM	Ever had instruction in EBM (yes, no)
HrdMsh	Ever heard of MeSH terms (yes, no)
UsedMsh	Ever used MeSH terms (yes, no)
HrdSH	Ever heard of subheadings (yes, no)
UsedSH	Ever used subheadings (yes, no)
HrdExp	Ever heard of explosions (yes, no)
UsedExp	Ever used explosions (yes, no)
HrdPT	Ever heard of publication types (yes, no)
UsedPT	Ever used publication types (yes, no)
<i>Cognitive traits</i>	
VZ2	Spatial reasoning test (score)
RL1	Logical reasoning test (score)
V4	Vocabulary test (score)
<i>Search mechanics</i>	
Time	Time to complete question (minutes)
Stacks	Whether searcher went to stacks (true, false)
Sets	Sets in MEDLINE search (number)
Retrieved	Number of articles retrieved by user in terminal set(s)
Viewed	Total MEDLINE references viewed (number)
FTViewed	Full-text documents viewed (number)
<i>User satisfaction</i>	
Quis	QUIS average for this searcher (number)
<i>Relevance</i>	
Helpful	Citations helpful answer (number)
Justified	Citations justifying answer (number)
Precision	User's precision for retrieval of definitely or possibly relevant articles
Recall	User's recall for retrieval of definitely or possibly relevant articles
<i>Answers</i>	
Order	Order question done by this search (2-6; first search ignored because all did same one)
Answer	Answer to question (yes, no, or insufficient evidence to determine)
Type	Question type (therapy, diagnosis, harm, prognosis)
PreAns	Answer before searching (yes, no, or insufficient evidence to determine)
PreCorr	Answer correct before search (true, false)
PreCert	Certainty of answer before search (1 = high, 5 = low)
PostAns	Answer after searching (yes, no, or insufficient evidence to determine)

Variable	Definition
PostCorr	Answer correct after search (true, false)
PostCert	Certainty of answer after search (1 = high, 5 = low)
Preferred	Who user would seek for answer (from list)

The model also included assessment of cognitive factors, since these have been shown to be associated with searching performance not only in the studies of Saracevic *et al.* (Saracevic and Kantor, 1988b; Saracevic and Kantor, 1988a; Saracevic, Kantor *et al.*, 1988), but other work as well. Three factors were included in this model because they have been found to be associated with successful use of computer systems in general or retrieval systems specifically.

- *Spatial visualization*: The ability to visualize spatial relationships among objects has been associated with retrieval system performance by nurses (Staggers and Mills, 1994), ability to locate text in a general retrieval system (Gomez, Egan *et al.*, 1986), and ability to use a direct manipulation (3D) retrieval system user interface (Swan and Allan, 1998).
- *Logical reasoning*: The ability to reason from premise to conclusion has been shown to improve selectivity in assessing relevant and nonrelevant citations in a retrieval system (Allen, 1992).
- *Verbal reasoning*: The ability to understand vocabulary has been shown to be associated with the use of a larger number of search expressions and high-frequency search terms in a retrieval systems (Allen, 1992).

The cognitive traits were assessed by validated instruments from the Educational Testing Service (ETS) Kit of Cognitive Factors (Ekstrom, French *et al.*, 1976) (ETS mnemonic in parentheses):

- Paper folding test to assess spatial visualization (VZ-2)
- Nonsense syllogisms test to assess logical reasoning (RL-1)
- Advanced vocabulary test I to assess verbal reasoning (V-4)

Other important categories in the model included intermediate search results (i.e., results of searching performance that ultimately influence the user's ability to successfully answer the question). One of these is search mechanics, such as the time taken, the number of Boolean sets used in the searching process, the number of articles retrieved in the “terminal” set, and the number of MEDLINE references and full-text articles viewed by the user. Another intermediate category is user satisfaction. In this study, user satisfaction was measured with the Questionnaire for User Interface Satisfaction (QUIS) 5.0 instrument, which measures user satisfaction with a computer system, providing a score from zero (poor) to nine (excellent) on a variety of user preferences (Chin, Diehl *et al.*, 1988). The overall user satisfaction is determined by averaging the scores of all the preferences.

The next group of factors addressed the relevance of the retrieval set. These measures, including recall and precision, were considered to be intermediate outcome measures relative to the ultimate outcome measure of successfully answering the question. This is in distinction to the many retrieval studies that assess

recall and precision as the final outcome measures. The final category of factors contains variables associated with the answer, such as the answer itself, the question type, whether the user gave the correct answer before or after searching, and the user's certainty of the answer.

With this model in place, the investigators proceeded to collect data for all the variables, ultimately aiming to determine which were associated with successful searching. Subjects for the experiment came from a convenience sample of senior medical students from Oregon Health & Science University (OHSU) and NP students from OHSU and Washington State University, Vancouver. The general experimental protocol was to participate in three sessions: a group session for the administration of questionnaires and orientation to MEDLINE, the techniques of evidence-based medicine (EBM), and the experiment, followed by two hands-on sessions in which the subjects would do the actual searching, read the articles, and answer the questions.

At the group session, subjects were first administered a questionnaire on the personal characteristics and experience with computers and searching factors from Table 4. Next they were tested for the cognitive attributes measured by the ETS instruments. The session also included a brief orientation to the searching task of the experiment as well as a 30-minute hands-on training session covering basic MEDLINE and EBM principles. The following searching features were chosen for coverage: Medical Subject Headings (MeSH), text words, explosions, combinations, limits, and scope notes. The overview of EBM described the basic notions of framing the appropriate question, determining which evidence would be most appropriate for a given question, and identifying the best searching strategies for finding such evidence.

The hands-on sessions took place 2 to 4 weeks later in the OHSU library. All searching was done by means of the Web-based version of the Ovid system, accessing MEDLINE and a collection of 85 full-text journals. The logging facility of Ovid enabled all search statements to be recorded, as well as the number of citations presented to and viewed by the user in each set. In the two hands-on sessions, subjects searched six questions. For the first question of the first session, all users searched the same "practice" question, which was not graded. The remaining five questions (the last two from the first session and all three from the second session) were selected at random from the pool of 20 questions. Question selection was without replacement; that is, the same pool of questions was used for four consecutive searchers.

Before searching, subjects were asked to record a pre-search answer and a rating of certainty on a one (most) to five (least) certain scale for the questions on which they would search. They were then instructed to perform their searching in MEDLINE and to obtain any articles they wished to read, either in the library stacks or in the full-text collection available online. They were asked to record on paper their post-search answer, their certainty of the correctness of the answer (on a five of 1 to 5), which article(s) justified their answer, and any article that was looked at in the stacks or in full-text on the screen. Upon completion of the searching, they were administered the QUIS instrument.

Overall time taken was measured for each question. The search logs were processed to count the number of search cycles (each consisting of the entry of a search term or Boolean combination of sets) and the number of full MEDLINE

references viewed on the screen. After all the hands-on searching sessions had been completed, the actual answers to the questions were determined by the research team. Generalized estimating equations (GEEs) were used in statistical analysis because since each searcher had answered five different questions, individual questions were not independent (Hu, Goldberg *et al.*, 1998).

The recall-precision analysis was carried out by means of approaches most commonly used in the IR literature, including the use of domain experts to judge relevance, pooling of documents within a single query to mask the number of searchers who retrieved it, and assessment of interrater reliability. To calculate recall and precision from the myriad of Boolean sets generated by searching in each question, the researchers identified "end queries" of the search process, which were defined as the terminal point of a search strategy. This was the point at which the subject stopped refining (creating subsets of) a search and began using new search terms or new combinations of search terms. All relevance judgments were done by means of the MEDLINE record distributed in an electronic file, though judges were encouraged to seek the full text of the article in the library if necessary.

Sixty-six searchers, 45 medical students and 21 NP students, performed five searches each. There were 324 searches analyzed. Several differences between medical and NP students were seen (Table 5). Use of computers and use of productivity software were higher for NP students, but searching experience was higher for medical students. Medical students also had higher self-rating of knowledge and experience with advanced MEDLINE features. The NP students tended to be older and all were female (whereas only half the medical students were female). Medical students also had higher scores on the three cognitive tests. In searching, medical students tended to have higher numbers of sets but lower numbers of references viewed. They also had a higher level of satisfaction with the IR system, as measured by QUIS.

Table 5. Values of All Searching-Related Factors for All Searches, Stratified by Student Type (Hersh, et al., 2002).

Variable	Student type		
	All	Medical	NP
n	66	45	21
School (= M)	68%	100%	0%
Age (years)	34	31	41
Sex (= M)	35%	51%	0%
CompHrs (hours)	8.4	7.4	10.7
ProdSW (% true)	79	72	95
OwnPC (% true)	92	91	95
Modem (% true)	86	82	95

Variable	Student type		
	All	Medical	NP
Internet (% true)	73	67	85
PracHard (% true)	68	72	60
EnjComp (% true)	88	87	90
LitSrch (monthly)	5.7	7.1	2.6
WebSrch (monthly)	9.0	11.0	4.5
WebMed (monthly)	2.5	2.4	2.7
TrainEBM (% true)	57	53	67
HrdMsh (% true)	91	91	92
UsedMsh (% true)	73	75	67
HrdSH (% true)	87	85	92
UsedSH (% true)	60	60	58
HrdExp (% true)	79	83	72
UsedExp (% true)	49	60	25
HrdPT (% true)	56	69	28
UsedPT (% true)	22	29	5.0
VZ2 (score)	12.7	14.2	9.2
RL1 (score)	12.6	14.2	9.0
V4 (score)	22.6	23.1	21.6
Time (minutes)	32	30	39
Stacks (= Used)	28%	32%	20%
Sets (number)	19.1	21.6	13.7
Retrieved (articles)	26	23	32
Viewed (articles)	8.6	7.1	12.0
FTViewed (articles)	0.94	0.87	1.1
QUIS (score)	6.6	6.8	6.3
Helpful (articles)	2.3	2.3	2.3
Justified (articles)	1.6	1.7	1.4
Precision (calculated)	29%	30%	26%
Recall (calculated)	18%	18%	20%
Order (2-6)	4.0	4.0	4.0
PreCorr (% true)	32	32	31
PreCert (1-5)	3.2	3.2	3.2

Variable	Student type		
	All	Medical	NP
PostCert (1-5)	2.0	2.0	2.1
PostCorr (% true)	46	51	35

As noted earlier, the rates of correctness before searching for medical and NP students were virtually identical (32.3 vs. 31.7%). But following searching, medical students had a higher rate of correctness than NP students (51.6 vs. 34.7%). The goal of the statistical analysis was to determine the factors associated with successful searching, as defined by the outcome variable of correct answer after searching (PostCorr). After screening individual variables for their *p* values, using analysis of variance (ANOVA) techniques for continuous variables and chi-square tests for categorical variables, the researchers built a GEE model (Table 6). The final model showed that PreCorr, VZ2, Used2, and Type were significant. For the variable question type, questions of prognosis had the highest likelihood of being answered correctly, followed by questions of therapy, diagnosis, and harm. The analysis also found that the VZ2 and School variables demonstrated multicollinearity; that is, they were very highly correlated, and once one was in the model, the other did not provide any additional statistical significance. The variable VZ2 was included in the final model because it led to a higher overall *p* value for the model than School. Next, a similar analysis was done to find the best model using the 220 searches when the subject did not have the right answer before the MEDLINE search. The final best model was very similar to the model for all questions, with PreCorr obviously excluded. Again, VZ2 and School demonstrated high multicollinearity.

Table 6. Values of Searching-Related Factors Stratified by Correctness of Answer, Along with p Value of Screening for Statistical Analysis (Hersh, et al., 2002).

Variable	Incorrect	Correct	Screening <i>p</i> value
n	174	150	N/A
School (= M)	62%	77%	.01
Age (years)	34	33	.55
Sex (= M)	29%	42%	.03
CompHrs (hours)	8.0	8.8	.48
ProdSW (% true)	79	79	.95
OwnPC (% true)	93	91	.62
Modem (% true)	87	85	.74
Internet (% true)	73	72	.84
PracHard (% true)	68	69	.92

Variable	Incorrect	Correct	Screening <i>p</i> value
EnjComp (% true)	87	88	.88
LitSrch (monthly)	4.9	6.7	.16
WebSrch (monthly)	7.7	10.5	.05
WebMed (monthly)	2.4	2.7	.46
TrainEBM (% true)	59	56	.69
HrdMsh (% true)	90	93	.10
UsedMsh (% true)	70	77	.14
HrdSH (% true)	85	89	.18
UsedSH (% true)	56	64	.16
HrdExp (% true)	75	84	.02
UsedExp (% true)	42	57	.01
HrdPT (% true)	49	65	.01
UsedPT (% true)	16	28	.02
VZ2 (score)	12.0	13.4	.00
RL1 (score)	11.9	13.4	.10
V4 (score)	22.2	23.1	.21
Time (minutes)	33	32	.41
Stacks (= Used)	24%	33%	.10
Sets (number)	21	17	.29
Retrieved (articles)	27	25	.74
Viewed (articles)	9.1	8.0	.14
FTViewed (articles)	0.9	1.0	.52
QUIS (score)	6.6	6.7	.78
Helpful (articles)	2.4	2.2	.23
Justified (articles)	1.6	1.6	.57
Precision (calculated)	28%	29%	.99
Recall (calculated)	18%	18%	.61
Order (2-6)	4.0	4.0	.95
PreCorr (% true)	24	42	.00
PreCert (1-5)	3.1	3.3	.16
PostCert (1-5)	2.0	2.0	.66

In the recall-precision analysis, there were three relevance judgments made on question/document pairs on the scale of not relevant, possibly relevant, and definitely relevant. The reliability of the relevance judges, measured by Cronbach's α , was 0.81. Final document relevance was assigned according to the following rules: (1) if all judges agreed, the document was assigned that rating, (2) if two judges agreed, the document was assigned that rating, and (3) if all three judges disagreed, the document was assigned the possibly relevant rating. The relevance judgments were then used to calculate recall and precision for each user/question pair. As seen in Table 5, there was virtually no difference in recall and precision between medical or NP students. Likewise, Table 6 shows there was no difference in recall and precision between questions that were answered correctly and incorrectly.

A number of conclusions were drawn from this study. First, users spent an average of more than 30 minutes conducting literature searches and were successful at correctly answering questions less than half the time. Whereas medical students were able to use the IR system to improve question answering, NP students were led astray by the system as often as they were helped by it. The study also found that experience in searching MEDLINE and spatial visualization ability were associated with successful in answering questions. A final finding was that the often-studied measures of recall and precision were virtually identical between medical and NP students and had no association with correct answering of questions.

These results showed some similarities and some differences with respect to the prior studies. The original study, focused on NP students only, showed that attitudes toward computers were the strongest factor association with searching success (Rose, Crabtree *et al.*, 1998). None of the demographic, experience, or cognitive variables were associated with success. The second study found that the most predictive factor of successful question-answering was student type (medical vs. NP); spatial visualization showed a trend towards predicting successful answering, but it was short of statistical significance (Hersh, Crabtree *et al.*, 2000). The second study differed from the final in showing that both medical and NP students had their question-answering ability improved by the IR system. As with the final study, literature searching experience was associated with answering questions correctly. Factors that did not predict success in the second study included age, gender, general computer experience, attitudes toward computers, other cognitive factors (logical reasoning, verbal reasoning, and associational fluency), Myer-Briggs personality type, and user satisfaction with the IR system.

In summary, these studies demonstrated that students in clinical training were at best moderately successful at answering clinical questions correctly with the assistance of searching the literature. Possible reasons for the limited success of question answering include everything from inadequate training to an inappropriate database (i.e., a large bibliographic database instead of more concise, synthesized references) to problems with the retrieval system to difficulties in judging evidence.

4. COMMON THEMES AND FUTURE CHALLENGES

The user-oriented research described in this chapter shows that user evaluation is essential for understanding the efficacy of question answering systems. The results obtained in non-user evaluations may not easily translate to real-world settings. While batch-style evaluation is justified in the early “proof of concept” work in building question answering systems, user evaluation is essential for understanding their ultimate role in meeting the needs of those who will use such systems.

There are unfortunately no concrete conclusions or guidelines that emanate from the interactive evaluation that has been performed to date. This is partially due to the nature of interactive studies, i.e., introducing users into the process adds variance that is not present in system-oriented studies. This variance is present not only in individuals (e.g., users behave in unpredictable and/or inconsistent ways) but also at different sites (e.g., research groups must choose specific systems and user interfaces, plus they have hardware and/or network configurations that impact their results). As such, it is difficult to establish “truths” in the way that system-oriented evaluation approaches allow.

Another reason for the lack of concrete conclusions is that most evaluation studies have been done with small numbers of users, with a resultant lack of statistical power to truly detect differences. This is partially due to the nature of TREC, with its annual cycle that makes large-scale experiments challenging. Of course, even if large numbers of users can be recruited for individual studies, there are just too many permutations to study all aspects of a particular approach. For example, in assessing relevance feedback interactively, comprehensive experiments would need to compare strict document relevance designation with the ability to add suggested terms, compounded further by whether or not to allow users to change the weighting of suggested terms. Further permutations of the design might compare different user interfaces for adding terms and/or changing their weight, such as slider bars versus numerical input. No definitive guidelines for employing relevance feedback could reasonably be made without performing experiments on all of these different permutations, yet the resources to do so are likely prohibitive.

A further challenge in interactive question answering is that all experiments to date have focused on the use of conventional document retrieval systems for the task. No large-scale studies have been performed with systems explicitly designed for question answering described in the other chapters of this volume. Clearly such studies must be performed to provide the same sort of “reality check” that interactive studies have provided for convention IR systems and approaches.

Despite the challenges of user-oriented evaluation, effort must continue in this direction. Even the most elegant question answering algorithms must be usable by real humans for knowledge-based tasks. Further work should include studies that systematically evaluate individual features of these systems with adequate statistical power to determine the value of specific approaches.

5. REFERENCES

- Abraham, V., Friedman, C., et al. (1999). Student and faculty performance in clinical simulations with access to a searchable information resource. *Proceedings of the AMIA 1999 Annual Symposium*, Washington, DC. Hanley & Belfus. 648-652.
- Allan, J. (1997). Building hypertext using information retrieval. *Information Processing and Management*, 33: 145-160.
- Allen, B. (1992). Cognitive differences in end-user searching of a CD-ROM index. *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Copenhagen, Denmark. ACM Press. 298-309.
- Belkin, N., Cool, C., et al. (2000). Iterative exploration, design and evaluation of support for query reformulation in interactive information retrieval. *Information Processing and Management*, 37: 403-434.
- Belkin, N., Keller, A., et al. (2000). Support for question-answering in interactive information retrieval: Rutgers' TREC-9 interactive track experience. *The Ninth Text REtrieval Conference (TREC-9)*, Gaithersburg, MD. National Institute of Standards and Technology. 463-474.
- Chin, J., Diehl, V., et al. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. *Proceedings of CHI '88 - Human Factors in Computing Systems*, New York. ACM Press. 213-218.
- Cleverdon, C. and Keen, E. (1966). Factors determining the performance of indexing systems (Vol. 1: Design, Vol. 2: Results). Cranfield, England, Aslib Cranfield Research Project.
- deBlik, R., Friedman, C., et al. (1994). Information retrieved from a database and the augmentation of personal knowledge. *Journal of the American Medical Informatics Association*, 1: 328-338.
- Egan, D., Remde, J., et al. (1989). Formative design-evaluation of Superbook. *ACM Transactions on Information Systems*, 7: 30-57.
- Ekstrom, R., French, J., et al. (1976). *Manual for Kit of Factor-Referenced Cognitive Tests*. Princeton, NJ. Educational Testing Service.
- Eysenbach, G. and Kohler, C. (2002). How do consumers search for and appraise health information on the World Wide Web? Qualitative study using focus groups, usability tests, and in-depth interviews. *British Medical Journal*, 324: 573-577.
- Fellbaum, C., ed. (1998). *WordNet: An Electronic Lexical Database*. Cambridge, MA. MIT Press.
- Friedman, C., Wildemuth, B., et al. (1996). A comparison of hypertext and Boolean access to biomedical information. *Proceedings of the 1996 AMIA Annual Fall Symposium*, Washington, DC. Hanley & Belfus. 2-6.
- Gomez, L., Egan, D., et al. (1986). Learning to use a text editor: some learner characteristics that predict success. *Human-Computer Interaction*, 2: 1-23.
- Gorman, P. and Helfand, M. (1995). Information seeking in primary care: how physicians choose which clinical questions to pursue and which to leave unanswered. *Medical Decision Making*, 15: 113-119.
- Harman, D. (1993). Overview of the First Text REtrieval Conference. *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, PA. ACM Press. 36-47.
- Hersh, W. (1994). Relevance and retrieval evaluation: perspectives from medicine. *Journal of the American Society for Information Science*, 45: 201-206.
- Hersh, W. (2001). Interactivity at the Text Retrieval Conference (TREC). *Information Processing and Management*, 37: 365-366.
- Hersh, W., Crabtree, M., et al. (2002). Factors associated with success for searching MEDLINE and applying evidence to answer clinical questions. *Journal of the American Medical Informatics Association*, 9: 283-293.
- Hersh, W., Crabtree, M., et al. (2000). Factors associated with successful answering of clinical questions using an information retrieval system. *Bulletin of the Medical Library Association*, 88: 323-331.
- Hersh, W., Elliot, D., et al. (1994). Towards new measures of information retrieval evaluation. *Proceedings of the 18th Annual Symposium on Computer Applications in Medical Care*, Washington, DC. Hanley & Belfus. 895-899.
- Hersh, W. and Over, P. (2000). TREC-9 Interactive Track report. *The Ninth Text REtrieval Conference (TREC-9)*, Gaithersburg, MD. National Institute of Standards and Technology. 41-50.
- Hersh, W., Pentecost, J., et al. (1996). A task-oriented approach to information retrieval evaluation. *Journal of the American Society for Information Science*, 47: 50-56.

- Hersh, W., Turpin, A., et al. (2000a). Do batch and user evaluations give the same results? *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece. ACM Press. 17-24.
- Hersh, W., Turpin, A., et al. (2001). Challenging conventional assumptions of automated information retrieval with real users: Boolean searching and batch retrieval evaluations. *Information Processing and Management*, 37: 383-402.
- Hersh, W., Turpin, A., et al. (2000b). Further analysis of whether batch and user evaluations give the same results with a question-answering task. *The Ninth Text REtrieval Conference (TREC-9)*, Gaithersburg, MD. National Institute of Standards and Technology. 407-416.
- Hu, F., Goldberg, J., et al. (1998). Comparison of population-averaged and subject-specific approaches for analyzing repeated binary outcomes. *American Journal of Epidemiology*, 147: 694-703.
- Lagergren, E. and Over, P. (1998). Comparing interactive information retrieval systems across sites: the TREC-6 interactive track matrix experiment. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbroune, Australia. ACM Press. 162-172.
- Meadow, C. (1985). Relevance. *Journal of the American Society for Information Science*, 36: 354-355.
- Mynatt, B., Leventhal, L., et al. (1992). Hypertext or book: which is better for answering questions? *Proceedings of Computer-Human Interface* 92. 19-25.
- Osheroff, J. and Bankowitz, R. (1993). Physicians' use of computer software in answering clinical questions. *Bulletin of the Medical Library Association*, 81: 11-19.
- Over, P. (2000). The TREC Interactive Track: an annotated bibliography. *Information Processing and Management*: 369-382.
- Robertson, S. and Walker, S. (1994). Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland. Springer-Verlag. 232-241.
- Robertson, S., Walker, S., et al. (1998). Okapi at TREC-7: automatic ad hoc, filtering, VLC, and interactive track. *The Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, MD. National Institute of Standards and Technology. 253-264.
- Rose, L. (1998). *Factors Influencing Successful Use of Information Retrieval Systems by Nurse Practitioner Students*. School of Nursing. M.S. Thesis. Oregon Health Sciences University.
- Rose, L., Crabtree, K., et al. (1998). Factors influencing successful use of information retrieval systems by nurse practitioner students. *Proceedings of the AMIA 1998 Annual Symposium*, Orlando, FL. Hanley & Belfus. 1067.
- Salton, G. (1991). Developments in automatic text retrieval. *Science*, 253: 974-980.
- Saracevic, T. and Kantor, P. (1988a). A study in information seeking and retrieving. II. Users, questions, and effectiveness. *Journal of the American Society for Information Science*, 39: 177-196.
- Saracevic, T. and Kantor, P. (1988b). A study of information seeking and retrieving. III. Searchers, searches, and overlap. *Journal of the American Society for Information Science*, 39: 197-216.
- Saracevic, T., Kantor, P., et al. (1988). A study of information seeking and retrieving. I. Background and methodology. *Journal of the American Society for Information Science*, 39: 161-176.
- Singhal, A., Buckley, C., et al. (1996). Pivoted document length normalization. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland. ACM Press. 21-29.
- Sparck-Jones, K. (1981). *Information Retrieval Experiment*. London. Butterworths.
- Staggers, N. and Mills, M. (1994). Nurse-computer interaction: staff performance outcomes. *Nursing Research*, 43: 144-150.
- Swan, R. and Allan, J. (1998). Aspect windows, 3-D visualization, and indirect comparisons of information retrieval systems. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia. ACM Press. 173-181.
- Swanson, D. (1977). Information retrieval as a trial-and-error process. *Library Quarterly*, 47: 128-148.
- Turpin, A. and Hersh, W. (2001). Why batch and user evaluations do not give the same results. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, LA. ACM Press. 225-231.
- Wildemuth, B., deBlek, R., et al. (1995). Medical students' personal knowledge, searching proficiency, and database use in problem solving. *Journal of the American Society for Information Science*, 46: 590-607.

- Witten, I., Moffat, A., et al. (1994). *Managing Gigabytes – Compressing and Indexing Documents and Images*. New York. Van Nostrand Reinhold.
- Wu, M., Fuller, M., et al. (2000). Using clustering and classification approaches in interactive retrieval. *Information Processing and Management*, 37: 459-484.
- Wu, M., Fuller, M., et al. (2001). Searcher performance in question answering. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, LA. ACM Press. 375-381.
- Xu, J. and Croft, W. (1996). Query expansion using local and global document analysis. *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Zurich, Switzerland. ACM Press. 4-11.
- Yang, K., Maglaughlin, K., et al. (2000). Passage feedback with IRIS. *Information Processing and Management*, 37: 521-541.

WILLIAM OGDEN, JAMES MCDONALD, PHILIP BERNICK
AND ROGER CHADWICK

HABITABILITY IN QUESTION-ANSWERING SYSTEMS

Abstract: This chapter discusses our work in evaluating question-answering systems. We begin by reviewing evaluation concepts and methods that have been used in the past. We then consider more recent efforts to evaluate search engines that have contributed to the techniques we currently use. Finally we discuss our techniques and discuss results from user studies we have conducted. Implications are discussed for the future design of habitable question-answering systems.

1. INTRODUCTION

1.1 *Background*

For several decades researchers have worked hard to design and build tools for information retrieval. Some attempts involved building natural language interfaces to these systems. By the 90's, many of the problems associated with natural language interfaces had been identified, but the expense and difficulty of resolving them was so great that many developers refocused their energy elsewhere. Some of the redirected effort was spent developing tools for publishing, and locating and navigating information on the web.

In the mid-1990's several tools were available: web crawlers that collected links and information published on the web into a database, and search engines for navigating or querying those databases. Current versions of these tools enable users to form queries using natural language, and significant energy is being expended to build and refine them. Many refinements are in the area of question-answering systems. While search engines typically provide users with links to potentially useful source information, question-answering systems perform more sophisticated processing of user queries and source data, and attempt to answer queries directly. As these systems mature it is appropriate to look at methods for evaluating and identifying ways to improve them. These methods are the focus of this chapter.

1.2 Usability and Habitability

Habitability, in the context of information retrieval systems, is a term that has been used by Watt (Watt, 1968) and others to describe how well a system supports the language people use to interact with the system. For example, if there are several ways to ask a question, a habitable system language supports all of them by providing a useful response to each. While people may not get the information they are looking for, habitable systems make the reasons for failure clear. Looking for habitability is a particularly useful way to evaluate natural language systems because these systems attempt to support all of the ways that people naturally use language to interact with the system. By looking at how systems fail to be habitable we can identify ways to improve them. Present day search engines and question-answering systems are similar to earlier natural language interfaces (even when no natural language processing component is present) because queries often take the form of natural language. For this reason habitability is at the center of our work.

We use the term *habitability* to mean more than usability. Habitability describes how easily, naturally, and effectively people can use language to express themselves within the constraints of a system language. Habitable systems enable users to express everything that is needed for a task using language they think the system will understand.

Ogden (Ogden and Bernick, 1997) describes habitability in the context of four domains: *conceptual*, *functional*, *syntactic* and *lexical*, that impact system habitability. To successfully use a system a user “must learn to stay within the limits of all four domains, and habitable systems must provide information about their capabilities in all four” (pp. 138-39). Evaluating system habitability requires observing how people use a system and how the system responds to and supports their search strategies.

Information retrieval systems generally, and question-answering systems particularly must:

1. Respond to people’s queries in useful ways, and
2. Support search strategies that people are already familiar with or can learn easily.

The goal of the user studies described below is to discover habitability problems and their solutions

1.3 Important Evaluation Considerations

The goal of our evaluation effort is to find out how people actually use the system being evaluated, and/or to find out how well a system meets users’ needs. As a result, choosing users and tasks for system evaluation is particularly important.

1.3.1 Choosing Users for System Evaluation

System habitability depends on how well a system language matches a user’s knowledge in the domain of discourse. Studies of systems that have been designed

for unique user populations select participants carefully or train them in the subject domain. But question-answering systems may have no such defined user population, and there may be no constraints on the kinds of questions users can ask. For obvious reasons, evaluating systems designed for general user populations is more difficult than evaluating systems designed for restricted populations.

1.3.2 Choosing User Tasks for System Evaluation

Evaluation tasks, like users, must resemble the tasks real systems might be used to solve. Tasks for question-answering systems are sometimes selected from searches people have already performed, sometimes generated by experimenters in order to exercise greater control over elements of the system that get used, or to compare two or more systems. Participants may also be allowed to generate their own queries. Participant-generated tasks can provide a higher level of validity because they may more closely resemble tasks an actual system might encounter. On the other hand, participants may have a difficult time coming up with appropriate questions, or questions that exercise a sufficient number of system components.

1.3.3 User Training

If actual users are not available, or if a system is designed for users who need to be trained to use it, study participants might need to be trained both on knowledge of the domain and on how to use the system. Training affects how participants perform in system evaluations, so the training must reflect the kinds of experience users would get with an actual system.

1.3.4 Data Collection and Analysis

A variety of methods for data collection are available. These include user observations in the workplace, interviews, and laboratory experiments that record everything that happens during a session using audio, video, and think-aloud techniques. Some researchers, like Lewis and Rieman (1994), distinguish between two sorts of data: process data, and bottom-line data. Process data consist of observations of what participants are thinking about and doing during a study. Bottom-line data include the end results, tell us what happened, and are represented in terms of numbers, i.e., how long it took to process a query, how long it took to complete a task, user satisfaction ratings, numbers of results returned, accuracy of results, etc.

Measuring the accuracy or success of a system is an important consideration. It is difficult to determine if a correct answer has been returned by a system without doing an exhaustive search of the data, which, for most systems, is tedious and difficult. Methods for estimating accuracy have been developed for the Text Retrieval Conferences (TREC), but they require a fixed set of questions and independent human assessment. For habitability evaluations, it is important to allow for question variability. Measuring system accuracy is much more important for comparison studies where one is looking for comparison measures. For habitability

evaluations, where the goal is to find ways of improving habitability of a system, user-ratings of the success of an interaction may be more diagnostic. The user may be the best judge of the information that satisfies a given question.

Think-aloud methods can also be used to complement data collected through observation. In the context of our studies, thinking aloud involves having participants articulate what it is they are doing and why, and can help clarify what is observed during the course of the study.

2. RELATED WORK

There is a growing body of work that looks at how people search for information and tools that help them. Several studies look at search engine interfaces to electronic information systems, and identify general characteristics that useful interfaces ought to have.⁴⁵ Researchers in these studies looked primarily at bibliographic databases and the web interfaces to them. They did not, however, address search engines generally, or how people use them to locate information on the web. Hölscher and Strube (2003) looked at user behaviors to determine useful elements to include in search engine interfaces but did not look at the implemented characteristics of specific search engines.

Other studies have explored user behaviors during browsing or navigating the web, but none of them have focused on search engines.⁴⁶ Choo, Detlor and Turnbull (2003) looked at the information seeking behavior of workers over a two week period using surveys, interviews and client-side logs. They characterized several information seeking behaviors of web users, and summarized them using a model of behavioral modes and moves. Navarro-Prieto, Scaife and Rogers (2003) identified cognitive strategies related to web searching by comparing web searchers with high and low experience. They concluded that expert searchers plan ahead in their searching behavior based on their knowledge about the web, while novices hardly plan at all and are driven by external representations, for example, by what they see on the screen. Of interest here, but still missing, are examples of how specific interface characteristics supported or interfered with this behavior.

Other researchers have collected large datasets derived from the logs of Internet search engines like Excite and AltaVista.⁴⁷ While these studies offer a picture of the actual queries that search engine users produce, and how the engine responds to those queries, the relationships between a query and the user's actual information need are difficult to determine because the data source consisted of a query log. There were no users who could be asked about the tasks or task goals that motivated their queries, or how well the search results met their goals. None of these studies looked at system habitability.

⁴⁵ Examples of these include studies by Marchionini, Dwiggins, Katz and Lin (1993); Shneiderman, Byrd, and Croft <http://www.dlib.org/dlib/january97/retrieval/01shneiderman.html>.

⁴⁶ These studies include Cockburn and Jones (1996); Catledge and Pitkow (1995); Tauscher and Greenberg (1997).

⁴⁷ See for example Jansen, Spink, and Saracevic (1998); Jansen, Spink, Bateman, and Saracevic (1998); Spink, Wolfram, Jansen, and Saracevic (2001).

Other studies have looked at user behaviors and queries, but these have often been in the context of understanding behaviors so that interfaces could be designed to support them, rather than identifying system characteristics that impact those behaviors. For example, O'Day and Jeffries (1992) looked at the search behaviors of people (those with an information need and those conducting searches to satisfy that need) to identify behaviors that an electronic system would need to support. Spink *et al.* (2001) looked more closely at the queries produced by search engine users.

Missing from these studies is research that tells us about current systems from the perspective of usability, and more particularly, from the perspective of habitability. Furthermore, few of the methodologies used in those studies can help answer questions of system habitability. As part of his dissertation research, Bernick (2003) performed a comparison study of two generally available search-engine interfaces. This study, which attempted to identify system characteristics that impact habitability, is described in the following section.

3. A COMPARISON STUDY

For many years the Text Retrieval Conferences (TREC), have provided a venue for researchers working on problems associated with information retrieval to build, demonstrate, and test techniques and technologies. In the past, the TREC Interactive Track has used experimenter-generated tasks across systems to evaluate how well (how quickly and effectively) they retrieve information. While much of this work has focused on problems associated with text retrieval and summarization, and in building tools that can meet users' needs, less time has been spent looking at how the technologies are used once they are made more widely available. It is important to identify user needs prior to implementing a technology, but it is equally important to look at how the implementation of features designed to meet those needs are used once they are in the hands of users. As a result, Bernick (2003) used a variation of the Guidelines from the TREC 2002 Interactive Track as the basis for this comparison study⁴⁸.

The methodology for this comparison study varied from TREC in two ways. First, the TREC interactive track has a goal of identifying systems that produce efficient search results and accurate information. Less attention has been given to engine interfaces or user interaction. The comparison study looked primarily at user interaction; participant's determined whether results were useful or not, and task performance was not constrained by task time. Second, the TREC interactive track uses a database consisting of information derived solely from the gov domain, whereas the comparison study used two popular web search engines (Google and Ask Jeeves) whose data are derived from all available web domains.

Bernick attempted to identify characteristics that impact habitability by asking the following questions:

- **Strategies:** Are there interface characteristics that support or prevent search strategies, or help users work with different search strategies?

⁴⁸ <http://www-nlpir.nist.gov/projects/t11i/guidelines.html>.

Looking at strategies involved looking at how users approached search tasks, and evaluating whether those strategies changed over time or with different engines.

- **Query Formulation:** Are there interface characteristics that impact query formulation?

Evaluating query formulation involved looking at how participants used vocabulary, the number of query terms that were used and the number of unique terms, whether tasks were paraphrased, and how queries were modified.

- **Query Results:** Are there interface characteristics that facilitate or impede successful searches?

Query results were useful for identifying cues that participants attended to. The comparison study looked at how study participants used cues, i.e., how participants evaluated query results, located information and selected the sites that were visited.

- **Unsuccessful Search Tasks:** Are there interface characteristics that contribute to or mitigate failure?

In the comparison study, success or failure on a particular task was determined by the participant performing the task. Participants would usually indicate this by stating that they had either found what they needed to satisfy task criteria, or that they had not found what they wanted but were done looking. Data collection for the comparison study involved recording the sessions with software that provided a record both of the participant's interaction with the system and verbal comments made while performing tasks.

Bernick found that most participants were successful in most searches. The most successful search strategies involved using enough key terms from the task, or paraphrasing the search task with enough specificity to produce results, identifying the links to information sources that did have information related to the information need, and continuing to search that information source for information that satisfied the participant's interpretation of the task's information need. This suggested two significant problems with the study.

1. Experimenter generated tasks were cast in a way that kept researchers from understanding how people go about satisfying their own information need because the task influences the way participants formulate the query.
2. Task participants using experimenter-generated tasks don't own the information need. Rather, they are trying to satisfy an artificial information need, and may be attempting to satisfy the researcher rather than themselves.

Most participants in the comparison study used strategies that resembled their initial strategy for all tasks; none seemed to change their strategy significantly over the course of the study. This would suggest that a person's past experience with similar tools influences the ways in which they use a new tool. Participants formulated queries by using key words from the task or by paraphrasing key words from the task, by forming natural language queries based on the task, or by using terms that were designed to locate a site, or at least a source, of useful information.

The choice of vocabulary used was determined by a participant's immediate information need, but didn't always reflect the information needed to satisfy task criteria.

Most importantly, the vocabulary used by participants in the comparison study tended to reflect the information need posed by the task, but not always, and queries often used vocabulary from the experimenter-generated tasks. While the vocabulary used to describe a task was often understood and used in the same sense by the task author, experiment participant, and web authors, Bernick tried to explain task failure in part by appealing to the use of different word senses by web authors and task participants. However, it may simply be that the information need posed by the experimenter-generated task in addition to the language used in the task description produced results that were difficult for a particular participant to understand. In other words, since the participant didn't truly own the information need, they may not have really understood the task in a useful way, and hence, couldn't understand the results. The study design makes it difficult to pinpoint when the participant was actually working outside their domain of expertise.

The comparison study was useful for gaining insight into the characteristics of search engines that help people locate information, some characteristics that might not be so useful, and for suggesting other areas and methods of inquiry. Most importantly, it demonstrated that other methods were needed in order to evaluate question-answering systems from the perspective of habitability.

From the comparison study it was clear that methods would need to be developed that:

1. Enable evaluation rather than comparison.
2. Provide insight to the relationship between how a user thinks a question-answering system works and the queries they produce for it.
3. Help us understand:
 - a. The sorts of knowledge that people need to have to formulate successful queries.
 - b. How much benefit people get from knowledge of a system's structure and content.
 - c. Whether people who get training do better, and if so, for how long.
 - d. The training that helps most.

4. IMPROVING METHODS FOR QUESTION-ANSWERING SYSTEM EVALUATION

The Bernick study looked at question-answering as an activity that could be accomplished with web search tools. There has been recent progress in developing systems that are designed to do a better job of natural-language question analysis and to return results that more directly answer users' questions. These systems attempt to let users ask questions the way they normally would and, unlike most web search engines, return answers, not web documents. We took the lessons learned from the Bernick study and applied them to the development of a methodology for

studying two natural-language question-answering systems currently available as web demonstrations.

By looking at how systems fail to be habitable we can identify ways to improve them. Our method is focused on creating situations in which we can observe users' natural question-asking behavior. The goal of the user studies described below is to discover habitability problems and their solutions. The results of the Bernick study led us to modify our method to allow participants to generate their own information needs. While this makes system comparisons more difficult, it allows us to better focus our evaluation on one system at a time.

A key interest of this approach will be: Does the design of the interface influence the types of questions people ask? Habitability is about discovering the ways in which people ask questions so that systems can answer them. If the design of the system influences the questions people ask, then it is important to determine those elements within the system that are responsible. In particular, if the system is designed to best answer full natural-language questions, then the system's interface displays and feedback should be designed to encourage this type of input.

4.1 LCC Study

The Language Computer Corporation (LCC) has developed state-of-the-art technology for question answering (Moldovan *et al.*, 2002; also this volume). The purpose of this study was to examine the degree to which naïve users found the LCC system habitable, to identify interface issues, and to propose changes to the interface that would enhance usability. Our main concern was the development of a methodology that would lead to improvements in the system and to identify the useful characteristics of the system.

In this study we used participants' self-generated information needs as the sources of questions for the system. One of the problems in evaluating natural language systems is giving them questions to ask. The phrasings used in the questions provided often influence participant queries. Some methods (Ogden and Bernick, 1997) attempt to circumvent this by presenting "big picture" problems to be solved via decomposed questions. However, knowledge of system domain can greatly influence problem decomposition and problem-solving ability, and this is a potentially confounding variable. Having the participants generate their own questions from a short list of potential topics precludes problem-statement phrasing influences, provides a rich source of unique queries for analysis, and allows a better determination of answer satisfaction. The participant is the best judge of the information that satisfies a given question. Such information cannot always be gleaned directly from an initial question phrasing. For example, the question "how do I care for roses" might indicate a need for factual information on gardening, or it might be a request for a list of available products. The participant is in a unique position to determine if the information need is met by the system's answer. One drawback of this methodology, however, is the large number of unanswerable questions generated. This can also be considered an advantage, however, since

unanswerable questions reveal a great deal about the system interface, which is a major concern of habitability.

4.1.1 Method

Seven graduate psychology students at New Mexico State University participated in this study. All were experienced computer users and also experienced with search engine tasks. Participants were paid for their participation. Each student participated in a single session lasting between one and a half and two and half hours.

Participants were informed that their voice and computer screen data would be recorded and gave their consent. A brief survey was completed by each participant containing demographic questions and ratings of computer and search engine experience. Instructions on think-aloud verbal protocol were then given. After being shown a list of general topic categories, participants were requested to generate twelve questions of interest and rank them in order of interest. Of these questions, the eight rated highest in participant interest were selected for the search task. Participants also rated the questions for “breadth” or scope. Participants then queried the LCC system’s internet demo page⁴⁹, for answers to each of their eight questions. Thus, a given information need would consist of one question and as many queries to the LCC system as required to complete the task. Audio and computer screen data were recorded using background software. Participants were instructed to verbalize their thoughts as they performed the task (think-aloud protocols). Instructions were given to terminate a specific search when the participants received what they considered to be satisfactory answers, when they came to believe that the system could not answer a question, or if they became too frustrated to continue with the given question. At the conclusion of each question, participants rated their satisfaction with the results, usefulness of the system, and change in scope (breadth) of the queries. At the conclusion of all eight questions, participants answered questions regarding the nature of their experience with the LCC system during the session.

4.1.2 Results

The video/audio files captured during the sessions were edited into short clips covering specific questions or queries. An index to the HTML log files, which contained links to each of the video clips, is available online⁵⁰.

4.1.3 Query Types

Since this study was primarily concerned with the behavior of the participants in how they approached question-answering systems, an analysis was performed on query types. Queries were categorized as directives (e.g. ‘show me’), keyword, or natural language type questions. There were 146 queries, of which 52% were keyword type and 46.6% were natural language question type. The mean number of

⁴⁹ http://www.languagecomputer.com/demos/question_answering/internet_demo/index.html

⁵⁰ <http://troia.nmsu.edu:8001/transcripts/>.

queries per information need was 2.61 ($SD = 1.93$). The tendency for the subjects to migrate either to or from keyword usage required further analysis. Despite the emphasis on using natural language with LCC, subjects tended to use a large percentage of keyword queries. Comments from users indicate some confusion about the necessity of using natural language questions. Participants had a tendency to use keywords due to their experiences with other Internet search engines such as Google and Yahoo. The LCC system is designed to work better with natural language questions, so it is unfortunate, but informative, that participants in this study did not always use questions. Developers of question-answering systems need to identify ways to encourage the use of natural language instead of keywords.

4.1.4 Answer Success

Since the LCC system uses the Internet database, we did not attempt to independently assess the extent to which the database contained an answer to each question. Instead, successful answers were measured by user ratings of their satisfaction with the results obtained from the system. Ratings were given using a 7-point scale. If success is defined as a rating above 4, 67.9% of the 56 generated information needs were considered successful and 32.1% unsuccessful.

4.1.5 User Comments

User comments were solicited from the seven participants at the conclusion of the study. Some of these comments are paraphrased below. Note that some of the comments have 'done' appended to show cases where changes were incorporated into the LCC system.

"Which features of the retrieval system made it more useful?"

- Question based queries
- Short descriptions [of results]
- The natural language method was nice but it didn't work well.
- You could type phrases
- Suggestions to misspellings
- The system seemed capable of handling fairly narrow searches via the use of questions instead of just phrases.

"Were there any features of the retrieval system that could be improved?"

- Organize the results chronologically or alphabetically
- Avoid chat groups and personal emails [subjects did not like results that were not credible]
- Bold the keywords
- Include the website for each result [done]
- Have a general menu with topics like "education"
- Add [instructions ?] that you can use keywords not just questions.
- No system feedback when you click on a link.
- Didn't like that a new window opens
- Web address not given to you in results [done]

Broader searches
No repeat websites
Have it look for keywords, not the question.
“The whole darn thing”
It seemed a little slow, reported status in a small blue font easy to miss so I thought nothing was happening.

Because the experimenter prompted the participants for comments, they may be exaggerations and should be carefully considered. However, it is clear that many users were uncomfortable or confused about the capabilities of the system.

4.1.6 General Observations and Conclusions

The users in this study generated a large number of time-critical questions, but the LCC system had no time processing capability. This created a significant number of problems. Apparently participants believed the system would be able to understand phases like, “yesterday”, “today”, and “currently”. For example, one participant entered the following set of questions before giving up:

Did the Bruins win today?
Did the Boston Bruins win their last game?
Did the Bruins win on 02/26/03?
Where can I find the Boston Bruins record for 2003?

This example is important for two reasons. First, the function necessary to answer these questions should be included in the capabilities of a natural-language question-answering system because users expect them. Second, and more importantly, the system should provide a feedback mechanism to alert users when they ask questions that are outside of the system’s capabilities. The system, like most traditional information retrieval systems, always returns some answer. The natural-language user can become more confused than the traditional user because the system’s response suggests that it understood the question, when in fact it did not. Natural language systems have many such “hidden” capabilities. If these capabilities were made visible, other missing capabilities might be easier to detect. In short, better feedback and/or better visibility of systems capabilities should be provided.

Other commonly occurring problems have simpler solutions:

1. Spell checking: The spell checker often came up with suggestions for spelling changes that seemed unnecessary. For example, the system suggested ‘France’ in response to ‘france’, although the results were not sensitive to case. There were other examples where spell checking seemed to need fine-tuning
2. Repeated web Sites in results: Several results were often presented which referred to the same web page, thus being redundant.
3. The ‘Power Answer’ button could be misleading. Some participants thought that performing a ‘power answer’ was somehow more than just asking for an

answer. A simpler button that indicated ‘search’ or ‘answer’ would be simpler and avoid confusion and wasted time.

4. Phrasing of question: There appeared to be considerable difficulty determining the best way to ask for information. Participants tried natural language questions and keyword phrases. For some questions it seemed difficult to phrase the information need as a question.
5. While it seems to be unimportant from the system’s perspective, participants often worried over whether or not to use a question mark. They often repeated queries with or without the punctuation if the system did not give them successful results. Again, users were confused about the hidden or missing capabilities of the system.

4.2 START study

START, a web-based question answering system, has been online since December, 1993. START was developed by Boris Katz and his associates in the InfoLab Group at the MIT Artificial Intelligence Laboratory (Katz *et al.*, 2002). The purpose of this study was to examine the degree to which naïve users found START habitable, to identify interface issues, and propose changes to the interface that would enhance habitability. The method was similar to that used in the LCC study reported above. We had participant’s self-generated information needs as the source of questions for the system.

4.2.1 Method

Three female graduate students between the ages of 18 and 37 were paid to participate in this study. Each participant’s session lasted between one and a half and two and a half hours. All participants were experienced computer users and experienced with search engine tasks.

After being shown the categories and example questions from the START web site, participants were asked to generate twenty questions of interest, distributed among the five categories. Participants ranked their questions in order of interest to themselves and rated each question for breadth. No training was given to participants beyond a review of the START home page and the categories and example questions,. Participants queried the START system for answers to each of their twenty questions in the order in which the questions were generated. Thus, a given information need consisted of one question and as many queries to START as required. Audio and computer screen data were recorded. Participants were instructed to verbalize their thoughts as they performed the tasks. Instructions were given to terminate a specific search if the participant received a satisfactory answer, came to believe that the system could not answer the question, or became too frustrated to continue with the search. At the conclusion of each question, the participant rated her satisfaction with the results, usefulness of the system, and change in scope (breadth) of the query. At the conclusion of all twenty questions, participants answered questions regarding the nature of their experience with the START system.

4.2.2 Results

Overall, participants had a difficult time understanding the conceptual and functional domains in the twenty-question session. A common comment was that they could not figure out how the system wanted them to phrase questions. Participants switched back and forth between using natural language questions and short ‘keyword’ questions. A summary of questions and ratings is provided online⁵¹. Participants were asked to rate how satisfied they were with the results they obtained on a 7 point scale. The mean satisfaction rating was 3.07, and mean usefulness was 3.65. Out of 60 questions, 35% were judged to be answered correctly by START, 10% were answerable by START but the right question was not asked, 47% were unanswerable by START, and 8% were judged to be partially answerable. These judgments were made after reviewing each question and extensively probing the START system for possible answers. There were 3.35 queries per question on average (standard deviation of 2.24), with fewer (mean 1.76, standard deviation 1.04) queries on the answered questions, as would be expected. Of the queries, 60% were natural language questions, 34% were short keyword phrases, and 6% were directives (e.g. ‘show me’).

4.2.3 Discussion

The results are perhaps best discussed in terms of specific examples, which illustrate the interface issues. Several key issues were examined and each is discussed below.

4.2.4 Failure Analysis

Some of the most useful results of this study came from a detailed failure analysis. The use of self-generated tasks provides a rich source of unique examples of queries. There are several types of failures to be considered. For example, some questions are answerable, but a satisfactory answer is not obtained. There are also answered question which required an unnecessary number of restructured queries. The researchers examined each question submitted by participants and submitted queries to START as needed in order to understand what went wrong. Causes of query failure were not readily apparent during the sessions, leading to much confusion about how the system functioned.

As an example of the failure to answer, one participant asked, “How many total episodes of ‘Excel Saga’ are there?” We subsequently found that the simple query ‘excel saga’ produced information that includes a direct answer to the question, although the answer is embedded in summary text and not directly accessed as a database item. The participant failed despite entering the query [“Excel Saga”]. We found that this query can produce a good result, with the answer embedded within the summary text. However, the START’s automatic processing of ellipsis prevented the query from retrieving the results during the test session. This example, and others, led us to conclude that the ellipsis processing of START was doing much more harm than good, at least in our study, and should be re-evaluated. The same

⁵¹ URL http://troia.nmsu.edu:8001/start-transcripts/start_subjcts.html

type of ellipsis interference problem also occurred in response to the question “Population of croatia”. The initial query submitted without the prior queries resulted in a good answer. However, the participant rephrased the question in order to produce an answer because, due to the sequence of queries and perhaps timing, some not readily apparent ellipsis function resulted in an erroneous ‘I don’t know’. By examining the queries one by one in this manner we were able to glean useful information about interface characteristics that could lead to improvements. It is our belief that the self generated nature of the questions employed contributed greatly.

4.2.5 *Ellipsis*

We concluded that the use of ellipsis in the system as implemented did more harm than good. The fact that the ellipsis function is hidden from the user and is not easy to discover was a source of confusion for participants. Queries that would have returned successful results, as illustrated in the previous paragraph, resulted in failure due to specific sequences, and perhaps the timing of queries submitted. In some cases participants were confused by feedback that, in it’s natural-language generative form, indicated a response to a previous question. Beginning with an examination of these inappropriate and odd results, we were able to identify the use of ellipsis as the source of the confusion. Perhaps ellipsis processing could be signaled to the user by showing the expanded interpretation of the users question. Also we suggest that ellipsis processing only occur when the system could find and answer to the previous query.

4.2.6 *Feedback*

Verbal protocols provide a rich source of behavioral understanding, although much is still left to inference. It became apparent both in real time and when reviewing the recorded videos that users were confused by the feedback provided by the system. The feedback from START is often “generated” natural language, with variability in responses for the same query. This variability is intended to add “naturalness” to the interface, but may actually contribute to user confusion. Users commented that they could not determine what START wanted for input. The rephrasing that was prompted by the feedback may have been misguided. Consider the question “Who created the clarinet”.

User: who created the clarinet

System: I’m afraid I can’t help you with that.

User: who made the first clarinet

System: Sorry, I don’t know who made the first clarinet.

User: where did the first clarinet come from

System: I don’t know.

User: where did the first musical instrument come from

System: Sorry, I don’t know the answer.

User: where was the first clarinet made

System: Sorry, I don't know where the first clarinet was made.

Upon further scrutiny, we concluded that in addition to the issue of variability in response, the feedback was insufficient in discriminating between possible failure types. For the question "Who is credited with the idea of chaos theory", the system understands "chaos theory" but doesn't have the ability to look up 'who is credited with'.

We are currently conducting a follow-up study with START to address this issue of variable feedback.

4.2.7 *Fallback*

Using the methodology of failure analysis, we probed the system extensively for answers. In many cases a direct answer was not possible using START, even though useful information was available in the database using simple queries. As an example, the question "What is SKA music" does not produce a result, however "What is SKA" does. We suggest that there are many cases (e.g. "When did the Korean war happen") where falling back to the information available on the main query terms would produce satisfactory results.

4.2.8 *Conclusion*

The use of self-generated questions, good data capturing, and detailed failure analyses provide a methodological structure that can yield practical results with natural language systems. Several key interface issues were uncovered using these methods. The developers of the START system have begun to address many of these.

5. EXPLAINING HABITABILITY: A TWO-STEP MODEL OF THE COGNITIVE PROCESS

Observing users struggling with question-answering systems provides insights into the thought processes that must occur in order to gain a useful answer. Here we propose a model of these processes to focus our discussion of question-answering system design.

Our model of a successful interactive information retrieval process from an online source consists of two steps. First, the user imagines how the target information is represented in the computer database. The user first has to know what an answer looks like and must believe that the system contains it. Then the user needs to know how the retrieval system will access and return the target. (The user needs to understand enough about the mechanism of the systems to control which information is retrieved.)

For web-based retrieval, users first have to imagine that there exists a web document or set of documents with the information that they need. They then must think of a set of keywords or phrases (the set of instructions the system uses) that uniquely identify and return the documents. Good retrieval systems are designed (or

have evolved) to map the keywords and phrases that users typically think of on to retrieval engines that return good (or useful) documents.

It is unlikely that most users understand this two-step process, at least not explicitly. However, we suggest that successful users employ this process, and that systems that make it easier for them to do so will have greater success. People fail either because they do not have a clear idea of how an answer is represented by the system, or do not understand how to control the search engine well enough to retrieve the target information. Users find it difficult to determine whether the information is missing or if they simply do not know how to retrieve it.

5.1 Implications for system design

The ways that users formulate queries when interacting with information retrieval systems is determined both by their expectations about what questions systems may be able to answer (what they think the system knows) and how they think the system will use their input to retrieve the answers.

There are two ways a user can fail and a system can help in both cases. The necessary information could be in the database, but the user does not have a clear idea how it is represented. Alternatively, the user might know how the information is represented, but does not know how to make the system work. The system can help in both cases by transforming the user's query into terms or procedures that are more likely to match the user's intentions. A natural language question-answering system can be helpful in transforming the user's question into the language contained in the data. The problem comes when a transformation alters the user's intention and the user does not have access to the interpreted or transformed version of the query.

This is why we feel that feedback and visibility are critical design features for natural language systems. Unfortunately these features are often overlooked. Habitability requires that users have accurate expectations about what they can or cannot ask. Making visible the hidden capabilities of natural-language interfaces remains one of the biggest challenges for the design of future question-answering systems.

6. REFERENCES

- Bates, Marcia J. "An Exploratory Paradigm for Online Information Retrieval." *Intelligent Information Systems for the Information Society*. Ed. B.C. Brookes. Amsterdam: North-Holland, 1986. 91-99.
- Bernick, Philip. "Habitability in Search Engine Interfaces: Characteristics Identified Through Formative Evaluation Techniques." Doctoral dissertation, New Mexico State University, Las Cruces, NM, 2003.
- Catledge, Lara D., and James E. Pitkow. "Characterizing Browsing Strategies in the World-Wide Web." *Computer Networks and ISDN Systems*. 27.6 (1995): 1065-73.
- Choo, C.W., B. Detlor, and D. Turnbull. "Information Seeking on the Web- An integrated model of browsing and searching." *Proceedings of the Annual Meeting of the American Society for Information Science (ASIS)*, 23 June 2003 <<http://choo.fis.utoronto.ca/fis/respub/asis99/>>
- Chu, H., and Rosenthal, M. 1996. Search Engines for the World Wide Web: A Comparative Study and Evaluation Methodology. In *ASIS 1996 Annual Conference Proceedings*, 23 June 2003 <http://www.asis.org/annual-96/ElectronicProceedings/chu.html>
- Cockburn, A., & Jones, S. (1996). Which Way Now? Analysing and Easing Inadequacies in WWW Navigation. *International Journal of Human-Computer Studies*, 45(1), pp. 105-129.

- Hölscher, Christoph and Gerhard Strube. "Web Search Behavior of Internet Experts and Newbies." *Proceedings of the 9th International World Wide Web Conference (WWW9)*. 23 June 2003 <http://www.www9.org/w9cdrom/81/81.html>
- Jansen, B.J., A. Spink, A., J. Bateman, and T. Saracevic. "Real Life Information Retrieval: A Study of User Queries on the Web." *SIGIR Forum*, 32.1 (1998): 5-17.
- Jansen, B.J., A. Spink, and T. Saracevic. T. "Failure Analysis in Query Constructions: Data and Analysis from a Large Sample of Web Queries." *Digital Libraries 98*. New York: ACM, 1998. 289-90.
- Katz, Boris, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland and Baris Temelkuran. "Omnibase: Uniform Access to Heterogeneous Data for Question Answering." *Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems (NLDB 2002)*, June, 2002.
- Lewis, Clayton and John Rieman. "Task-Centered User Interface Design: A Practical Introduction." available via anonymous ftp at: <ftp.cs.colorado.edu> in: /pub/cs/distrib/clewis/HCI-Design-Book. 1994.
- Marchionini, G., S. Dwiggins, A. Katz, and X. Lin. "Information Seeking in Full-Text End-User-Oriented Search Systems: The Roles of Domain and Search Expertise." *LISR*, 15 (1993): 35-69.
- Moldovan, Dan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu. Performance Issues and Error Analysis in an Open-Domain Question Answering System. *40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, July 2002, Philadelphia, Pennsylvania.
- Navarro-Prieto, R., M. Scaife, and Y. Rogers. "Cognitive Strategies in Web Searching." *Proceedings of the 5th Conference on Human Factors and the Web*, 23 June 2003 zing.ncsl.nist.gov/hfweb/proceedings/navarro-prieto/index.html
- O'day, Vicki, and Robin Jeffries. "Orienteering in an Information Landscape: How Information Seekers Get From Here to There." *HPL-92-127* Hewlett-Packard:n.p, 1992.
- Ogden, William C., and Philip Bernick. "Oleada: User-Centered Tipster Technology for Language Instruction." Proceedings of the Tipster Phase II 24 Month Workshop. Tysons Corner, VA, 1996.
- Ogden, William C., and Philip Bernick. "Tabula Rasa Meta-Tool: Text Extraction Toolbuilder Toolkit." Technical Report MCCS-97-305, Computing Research Laboratory, New Mexico State University, Las Cruces, NM, 1997.
- Ogden, William. C., and Philip Bernick. "Using Natural Language Interfaces." Handbook of Human Computer Interaction. Eds. Helender, M., T. Landauer, and P. Prabhu, 2nd Edition, Amsterdam: North Holland, 1997, 137-61.
- Spink, A., D. Wolfram, B.J Jansen, and T. Saracevic. "Searching the Web: The Public and Their Queries." *Journal of the American Society for Information Science and Technology*, 52 (2001): 226-34.
- Tauscher, Linda and Saul Greenberg. "How people revisit Web pages: Empirical findings and implications for the design of history systems." *International Journal of Human Computer Studies*, 47(1): 97-138, 1997.
- Watt, W.C. "Habitability." *American Documentation*, July (1968) 338-51.

6

PERSPECTIVES ON QUESTION ANSWERING

STEVEN J. MAIORANO

QUESTION ANSWERING: TECHNOLOGY FOR INTELLIGENCE ANALYSIS

Abstract: This chapter presents a former analyst's perspective on Question Answering (QA) systems. Although the author's background is in intelligence analysis, he is currently engaged in sponsoring basic research in linguistic pragmatics. Given this background, the author believes that today's technology has the potential to help fulfil the vision of providing a highly capable tool enabling an analyst – any analyst, be they journalists or financial experts on Wall Street – the ability to interact with data in a meaningful way, much as one would conduct a dialogue with another person, in order to answer questions or aggregate relevant information in the attempt. Currently, analysts derive most of the information they need from a large corpora of documents. The preferred means of doing so, for the most part, is a three-step process of collect, filter, and then process manually or semi-automatically. The predominant paradigm employed usually involves the Web, an information retrieval (IR) engine such as Google, and intense reading. Their real need, however, is to obtain specific answers to specific questions. The loftiness of the goal notwithstanding, the author believes that technology is progressing to the point where QA systems can address this need. Although this chapter is more a vision statement and less a research roadmap per se, the author does suggest areas of research that are pertinent to the vision, and discusses the possible impact a dialogue-based QA system might have on analysts in general. To this end, two scenarios are presented in the Appendix.

1. BACKGROUND

The eighth Text Retrieval Conference (TREC-8; Voorhees and Harman, 1999) sponsored its inaugural Question Answering (QA) Track in November 1999 which was significant for several reasons. It was a first step in the post-TIPSTER (TIPSTER) era in which information extraction (IE) and information retrieval (IR) technologies emerged from their individual sandboxes of the Message Understanding Conference (MUC) and TREC evaluations to play together. The National Institute of Standards and Technology (NIST) together with other US Government research offices added a QA Track to TREC in order to provide a problem and forum in which traditional IR techniques would be coerced into partnership with different natural language processing (NLP) technologies. Having researchers develop systems to answer specific questions of fact (“*Who was the sixteenth president of the U.S.?*” “*Where is the Taj Mahal?*” etc.), was thought to be an appropriate way to accomplish this coercion, and, at the same time, set the stage for a much more ambitious evaluation.

The QA Track was an unqualified success in terms of achieving the above-stated goal. The top-performing systems incorporated MUC-like named-entity recognition technology in order to categorize properly the questions being asked and anticipate

correctly the type of responses that would be appropriate as an answer. Traditional IR techniques alone were insufficient. In addition, participants were required to be more creative with their indexing methods. Paragraph indexing combined with Boolean operators proved to be surprisingly efficient. The subsequent TREC QA evaluations have become more difficult in terms of the question type and answer format, and this has necessitated that systems attempt to incorporate even more NLP techniques. Some top-performing systems in fact have begun to derive semantic information in various ways from diverse sources. In short, the QA Tracks have not only moved TREC out of the shadow of TIPSTER, but have also helped demonstrate that returning a collection of relevant documents is a limited view of what “search strategy” means. In many instances, a specific answer to a specific question is the epitome of a user’s “search.”

Factual Knowledge

- What is the CPI?
- What are the countries with the highest GDP?
- Who is the governor of Minnesota?
- Whose portrait is in the \$50 bill?
- Which picture won the Oscar for best picture in 1978?
- In which country is Azeri spoken?
- Who is Heidi Fleiss?
- Who won the Nobel Prize for economics in 1980?
- Which great baseball player died in March 1999?

Explanatory Knowledge

- What is a bear market?
- What are recognized to be the predominant causes of the rise of the Taliban in Afghanistan?

Mention was made above of a more ambitious evaluation. TIPSTER had many success stories, but the implicit goal of natural language understanding was not one of them. It was thought that having systems tackle tasks involving the extraction of events and relationships and fill complex template structures would, *a priori*, necessitate solving the natural language understanding problem, or at least major portions of it. Extraction developers, however, discovered that partial parsing and syntax alone (pattern recognition) would score well in the MUCs. The syntactic approach – what Jerry Hobbs (Hobbs, 1998) characterized as getting what the text “wears on its sleeve” – extracted entity names with human-level efficiency. However, the 60% performance ceiling on the more difficult event-centered Scenario Task is indicative of what was not successfully extracted or even attempted, such as: those items that crossed sentence boundaries and required robust coreference resolution, evidence merging, inference, and world knowledge; that is, natural language understanding.

This view of the extraction work carried under the TIPSTER Program is shared by others outside of the IE mainstream such as Enrico Franconi (Franconi, 2003) who discusses the recent state of Description Logics as applied in NLP applications. He states that “the interest of the applied computational linguistic community in Description Logics began to drop [in the early 1990s], as did its interest in well-founded theories on syntax and semantics. [At present] there is no major applied project in natural language processing making use of Description Logics. This is due to the positive achievements in real applications of the systems based on shallow analysis and statistical approaches to semantics, initiated by the applications in the message understanding area.”

Semantics under TIPSTER was, in short, put on the back burner. The contention here is that QA properly conceived and implemented as a mid- to long-term program (five to eight years) puts natural language understanding in the forefront of researchers and systems’ developers. As Wendy Lehnert (Lehnert, 1978) stated over 20 years ago: “Because questions can be devised to query any aspect of text comprehension, the ability to answer questions is the strongest possible demonstration of understanding.” The ultimate QA capability facilitates a dialogue between the user and machine, the ultimate communication problem and all that that entails. There are many technological issues that need to be addressed along the way, and the Advanced Question Answering for Intelligence (AQUAINT) Program sponsored by the Advanced Research and Development Activity (ARDA) provides the framework within which the necessary research breakthroughs will, hopefully, be made.

2. ASPECTS OF QUESTION ANSWERING

2.1 *Knowledge Types*

The TREC-8 QA Track dealt with what could be called “factual knowledge,” answering specific questions of fact. TREC-9 (Voorhees and Harman, 2000) required that answer strings be reduced from 50 and 200 bytes to 20. TREC-8 avoided difficult questions of motivation and cause – Why and How – limiting most of the questions to What, Where, When, and Who. A more ambitious evaluation would require moving beyond questions of fact to “explanatory knowledge.”

Anyone who has considered a typology⁵² of questions readily understands that questions are not always in the form of a question. But answers sometimes are.

⁵² The examples provided here are discussed in terms of lexical categories which only scratches the surface of any proposed typology. Wendy Lehnert discusses 13 conceptual question categories in her taxonomy. (“The Process of Question Answering,” [Lehnert, 1978]) In the context of MUC where slot fills in general answer the question Who did What to Whom When Where and sometimes How, the How question would most likely be considered in terms of **instrumentality**, e.g., How did the dignitary arrive? (By what means did he come?), or **causal antecedent**, e.g., How did the F-15 crash? (What caused the plane to crash?). These categories may cover the predominant number of cases in a bounded MUC task, but not in an open domain. After Lehnert, one would need to deal with such categories as: **Quantity** – How often does it rain in Seattle? (Everyday. Twice a day. Most evenings between November and May.); **Attitude** – How do you like Seattle? (It’s fine. I like rain. I haven’t

Consider a telephone conversation that begins with the pseudo-question “Well, I need your name to handle this matter.” Since the questioner obviously is eliciting a name, a rhetorical response might be “What, you expect me to provide personal information?” Also, sometimes the person with the answers asks the questions. The previous examples point to other challenges in turn. The problem is not simply one of dealing with more difficult questions; the data source is not limited to narrative texts or factual news articles. Speech is also a part of the ultimate QA terrain. Furthermore, the “search strategy” cannot be thought of only as one user trying to acquire answers from a system. Communication in the “real world” does not work that way. Communication is interactive and necessitates clarification upon clarification. The same should be true for our QA framework.

Further analysis of the question typology reveals the sort of knowledge characterized by DARPA’s High-Performance Knowledge Base (HPKB) Program (Cohen *et al.*, 1998): “Modal knowledge.” The program’s knowledge base technology employed ontologies and associated axioms, domain theories, and generic problem-solving strategies to handle proactively crisis management problems. In order to evaluate possible political, economic, and military courses of action before a situation becomes a crisis, one needs to ask such hypothetical questions as “What might happen if Faction Leader X dies?” Or, “What if NATO jets strafe Serb positions along Y River?” More difficult still are counterfactual questions such as “What if NATO jets do not strafe Serb positions along Y River?” In addition to the problems surrounding knowledge base technology, like knowledge representation and ontological engineering, questions of this type require sophisticated logical and probabilistic inference techniques with which the text-processing community has not traditionally been involved. However, just because this has not been traditionally the case does not mean that this state of affairs should continue. Since knowledge bases (KBs) form a repository from which possible answers, factual or hypothetical, can be deduced, and since unstructured text provides much of the factual fodder for the encoding in KBs – much of it derived from human judgments involving inference in the first place – there is a need for greater cooperation between the text-processing and knowledge base communities.

Take for example the hypothetical situation in which the following pieces of information have been extracted from different sources:

- 1) *Steve is scheduled to travel from his office to Princeton on September 18th to meet with George Miller.*
- 2) *He has purchased a ticket on Amtrak.*
- 3) *Hurricane Isabel is forecast to hit the East Coast on September 18th.*
- 4) *The storm hits North Carolina on the morning of September 18th.*

Question: Did Steve discuss – or what is the likelihood that Steve discussed – WordNet in September?

seen anything yet.); **Emotional/Physical State** – How is John? (A bit queasy after the 6-hour flight in coach.); **Relative Description** – How smart is John? (Not smart enough to fly business.); **Instructions** – How does one get to Seattle? (Take a left onto Constitution and go west for 3,000 miles.)

Let us assume that IE systems are able to obtain most, if not all, of the above-stated facts regarding person, location, and time. The merging of and reasoning over the information is a difficult issue. For example, the system would, among other things, need to know that George Miller is the “father” of WordNet; that Amtrak is a mode of transportation (whose operating efficiency is less than optimal under even the best weather conditions); that both North Carolina and Princeton are on the East Coast; that these geographical locations are within an area that could be affected by a hurricane with Isabel’s characteristics; and that “He” in sentence 2 refers to “Steve,” and “storm” in sentence 4 refers to “Hurricane Isabel.” Also, if the system is “smart” enough to infer that Steve probably did not make a trip to Princeton on the 18th, it would still need to reason further about Amtrak re-starting service after the storm passed enabling Steve to re-schedule for another day in September.

As Pearl (1988) states, probabilistic reasoning techniques are pertinent to “almost any task requiring that conclusions be drawn from uncertain clues and incomplete information.”

Finally, there is a fourth type of knowledge we will characterize as “serendipitous knowledge.” One could equate this search strategy to a user employing a search engine to browse document collections. However, a sophisticated QA system coupled with sophisticated data mining, text data mining, link analysis, and visualization/navigation tools would transform our QA system into the ultimate human-machine communication device. These tools would provide users with greater control over their environment. A query on, e.g., Chinese history, might lead one to ask questions about a Japanese school of historiography because of unexpected links that the “knowledge discovery” engine discovered, and proffered to the user as a valuable path of inquiry. That path of inquiry, moreover, could be recorded for future use, and traversed by others for similar, related, or even different reasons.

2.2 Data Types

Text and speech are not the only sources of information. Databases abound on the Web and in any organizational milieu. A QA system must not require the user to be educated in SQL, but must enable the user to query any data source in the same natural and intuitive manner. The system should also handle misspellings, ambiguity (“*What is the capital of Switzerland?*”), world knowledge (“*What is the balance on my 401K plan?*”). Other data – transcribed speech and OCR output – is messier. And, most of the world’s data is not in English. TIPSTER demonstrated that retrieval and extraction techniques could be ported to foreign languages. This being the case, a TREC-like QA track for answering questions of fact in foreign languages will be initiated soon, especially in those languages for which extraction systems possess a named-entity capability.

2.3 Systems Evolution

In the new world of e-commerce and knowledge management, the term “infomediary” has come to replace “intermediary” in the old world of bricks and

mortar and the physical distribution of goods. A QA system viewed as a search tool would require an infomediary to handle the profiling function of traditional IR systems. One can envision perhaps intelligent agents that would run continuously in the background gathering relevant answers to the original query as they become available. In addition, these agents would provide ready-made QA templates a la "AskJeeves," but generated automatically and stored in a FAQ archive. This archive would actually constitute a knowledge base, a repository of information to retain and augment analytical expertise.

Figure 1 illustrates the typical architecture of QA systems, as was implemented by the majority of TREC participants. A QA system, no matter how precise and sophisticated, cannot satisfy the intense need for specialized information experienced by professional analysts if every question is posed in isolation. With the increasing availability of on-line information, either on the Internet or within Intranets, the availability of QA systems that allow user interaction expressed in natural language has become critical. A major obstacle in building robust, user-friendly QA systems is the need to enable a conversation with the user in which clarifications, follow-up questions and context specification are made possible.

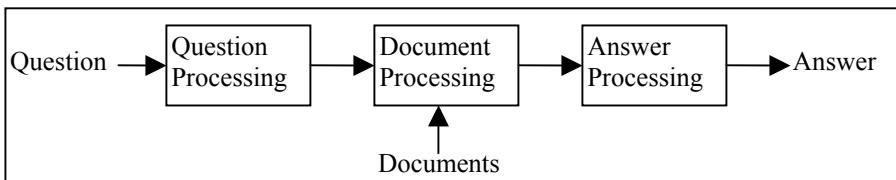


Figure 1. The Architecture of a Typical QA System.

With an interactive QA system, regardless of the domain of interest, an analyst may pose a series of questions using natural language and the QA system returns a precise answer to each question, retrieving information from text collections as well as knowledge bases, either linguistic in nature or encoded with domain-specific expertise. Information requests can range from finding a specific fact (e.g. "*What is the situation with Prime Minister Mori today?*"), to planning rescue operations at an American Embassy (e.g. "*If Americans won't leave or cannot get to the evacuation site, where can they go for safety?*"), or to preparing a report on a complex issue (e.g. "*What is the import-export trend between Japan and the U.S. over the last six months?*"). However, due to the various ambiguities inherent to natural language, clarifications or specifications need to be added in order to obtain the desired answer. Moreover, the analyst, when asking a question, addresses only one aspect of a larger task (s)he needs to resolve. These aspects are more than often related, therefore the questions should be processed in a context that is enabled only by considering QA sessions, made possible by the dialogue shell implemented in, e.g., the Language Computer Corporation (LCC) AQUAINT project called Just-In-Time Interactive Question Answering (JITIQA).

The ultimate goal of a QA system is to provide the answer to questions of variable degrees of complexity. A basic taxonomy of answer types, especially for factual questions, can be easily assembled as was proposed by the TREC

participants. However, information analysts are rarely interested in trivia-type information; they are more likely to need specialized information on a subject they know quite well. Their expertise is apparent through the questions they pose and especially through the line of reasoning they use in their quest for information. Moreover, an interactive QA system should recognize the expected answer type without relying on the analyst to define it or requiring the analyst to burden the question with unnecessary elaborative information as in normal conversation. Indeed, the analyst should be able to pose the question in the same manner as when conversing with another expert analyst. This entails the recognition of answer types from an enhanced answer taxonomy.

One of the challenges of having a wide-coverage answer taxonomy is that sometimes the expected answer type incorporates too much world knowledge and thus cannot be categorized in the way general ontologies allow it. For example, the expected answer type of the question: "*What was the import-export trend between Japan and the US over the last six months?*" can be recognized in a multitude of expressions, such as *deficit*, *surplus* or even *leveling*. Moreover, the trend might be a surplus for some sectors and a deficit for others; thus, the different lexico-semantic expressions should not be interpreted as conflicting information, but rather as information with a greater degree of granularity or specificity. Only specialized knowledge about import-export balances can determine the correctness of the answer type. The JITIQA project incorporates both specialized knowledge and world knowledge in determining an enhanced answer taxonomy. The validity of this enhanced answer taxonomy will be determined by monitoring feedback from the analyst. Furthermore, the resulting enhanced answer taxonomy as well as enhanced question taxonomy will be encapsulated in a dialogue shell that can be imported by any AQUAINT QA system wishing to use the enhanced taxonomies to improve performance. The question and answer taxonomies of the AQUAINT LCC system is interfaced with the taxonomies provided by the JITIQA interactive component shown below (Figure 2).

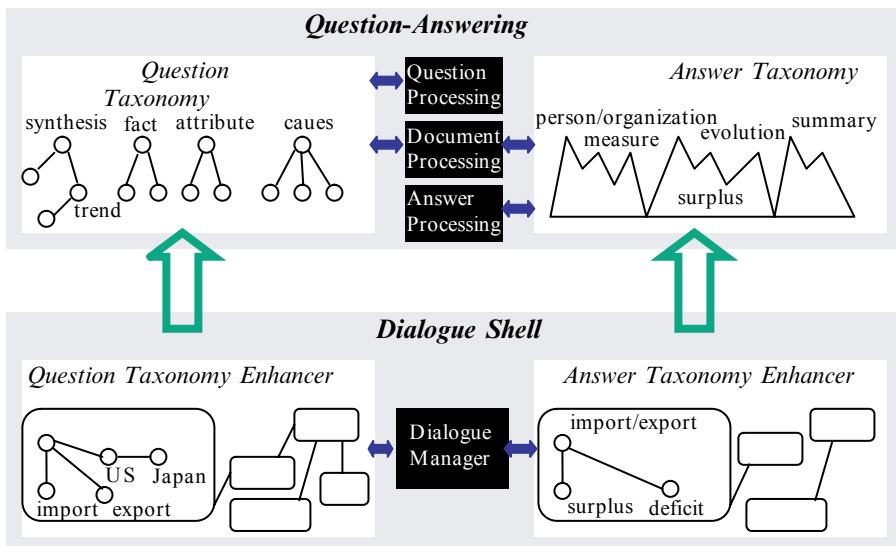


Figure 2. Enhancing answer taxonomy via multiple interactions with QA System.

The enhanced answer taxonomy has two important functions: (1) it expands the search space of the QA system, raising the certainty that all available data sources are mined for the answer; and (2) it allows the QA system to conclude that no or inconclusive relevant data exists within the data sources available when good candidate answers cannot be posited; so rather than being coerced into offering an absurd answer for the sake of providing an answer, the system can degrade gracefully and opt for not returning an answer at all. As shown in Figure 2, the recognition of surplus as a possible answer is derived from information in the enhanced answer taxonomy guided by a script-like structure encoded in the enhanced question taxonomy (Harabagiu *et al.*, 2000a).

2.4 Asking the Right Question

Whether or not progress in the natural language understanding disciplines and IR can reach the point where both logical and non-logical inference can be done automatically or semi-automatically, a QA system that enables human-computer interaction or dialogue is one of the ultimate aims of AQUAINT. Interaction is the essential process by which an analyst can ask the right question whether the system disambiguates terms and reformulates a question automatically or it prompts the user for semantic clarification or it provides alternate hypotheses based upon a link analysis and inference mechanism that is still immature. Furthermore, this capability is indispensable in seeking answers amongst all the above-mentioned knowledge types. Many times the key to the answer is asking the right question. One thinks of the genius of a philosopher such as Kant who asked questions of a more fundamental and powerful sort than his predecessors, challenging assumptions that had hitherto gone unchallenged. One wonders about the conventional question “*How*

do we use technology?" Asking instead "*How does technology use us?*" is more than semantic chicanery. Perhaps what we needed to think about in the early twentieth century was not how to drive cars, but what they would do to our air, landscape, social relations, and cities.

Different questions are ways of challenging assumptions. For example, take the question "*How do we provide more health care?*" The implication of course is that more is better. But what if more medicine is not necessarily good for our health and actually bad? Then the question is wrong, the search strategy artificially constrained, experiments ill-motivated, and policy misguided.

It is unlikely that a dialogue between a professional analyst and a QA system will involve only factually based questions. Many times interactions between an analyst and the QA system contain almost no factual questions, but, in fact, are dominated by complex, open-ended, abstract and vague questions. Moreover, the account of the context of the previous dialogue adds to the complexity of processing questions in an interactive QA environment. For example, the following question presented in Scenario 5 from ARDA BAA 01-01 is too vague unless the context of the conversation is known, e.g., civil war erupted in Mythia, rescue helicopters are likely to be met with resistance and the airport is controlled by a hostile military group:

Q: If Americans won't leave or cannot get to the evacuation site, where can they go for safety?

To extract and formulate a useful answer to this counterfactual question, the QA system needs to break it down in several other questions that may bring forward relevant information. Examples of such questions are: "*Where are the safe locations in Mythia?*" "*Which locations in Mythia are unsafe?*" and "*In which foreign cities caught in the midst of civil war did the Americans take refuge?*"

The processing of complex questions involves recognition by a dialogue management function – such as the LCC component shown in Figure 2 above – of question intent, motive, objective and contextual implication. Moreover, said dialogue manager comprises operators that allow comparisons, contrasts, examinations or matches between entities, events or opinion positions. Operators that search for similar entities, events or opinions are sometimes triggered by such surface cues as the pronoun *other*. For example:

Q: What OTHER recent purchases and significant purchases has company ABC, its subsidiaries or its parent firm made in OTHER foreign companies that are capable of manufacturing OTHER components and equipment needed to produce chemical weapons?

Q: Is there any OTHER evidence that suggests that the two terrorist groups may be planning a joint operation?

The processing of the two previous questions necessitates the extraction of all company acquisitions of company ABC and of all terrorist activities of the two groups respectively. To this end, the current JITIQA project is developing innovative technology that combines information extraction techniques with the

lexico-semantic feedback loops that contributed directly to the accuracy of LCC's predecessor system, Falcon (Harabagiu *et al.*, 2000b).

2.5 Mining Patterns of Activity

When using an interactive QA system, analysts use a combination of their own expert knowledge and knowledge inherent to the environment or context of the task. To provide valuable knowledge inherent to the environment, the interactive QA system must have access to several forms of information: (a) search profile of the analyst; (b) search traces of other analysts working on the same or similar tasks; (c) patterns of lexico-semantic unifications between questions and answers.

Access to textual data (or other structured data) is enabled by indexing mechanisms and associated retrieval techniques. Similarly, indexing and retrieval techniques need to be developed for analyst profiles and question-answering sessions. Most importantly, this information must be combined and presented to the analyst as search suggestions. The amount of information to show in a suggestion is always a tradeoff. To avoid distracting the analyst from their primary task it is important to display a minimal amount of information, but at the same time the whole point is to provide the user with a rich alternative information source. Furthermore, the very existence of a suggestion places an increased cognitive and perceptual load on the reader just to evaluate whether to get more information.

One way to address this tradeoff is by creating a *ramping interface* (Figure 1) where information is conveyed in stages. Each stage of a ramping interface provides a little more information at the cost of the analyst spending a little more of their attention to read and understand it. The idea is to get useful information to an analyst quickly, while at the same time allowing them to bail out on unwanted suggestions with as little distraction as possible. For example, when typing the question: "*Does this country have a Weapons of Mass Destruction program, and if so, what is its status?*" the analyst will be presented with information through the ramping interface that will guide the quest for information. The different stages of the ramping interface are determined by the search patterns other analysts have used when looking for the same or similar information.

The ramping interface can also be used to learn from more experienced analysts or even to trace a successful search at later times. Building a ramping interface presents two challenges. First, new indexing mechanisms of QA sessions must be built and techniques of retrieving the most similar QA interaction must be studied and implemented. To our knowledge, this research is novel and departs from traditional information retrieval or dialogue processing research.

Secondly, the patterns of activity must be established across multiple analysts that use the QA system for the same or similar topics. Mining search patterns from data collected when tracing all QA sessions must be studied. This involves combining in new ways information extraction, data mining and semantic processing.

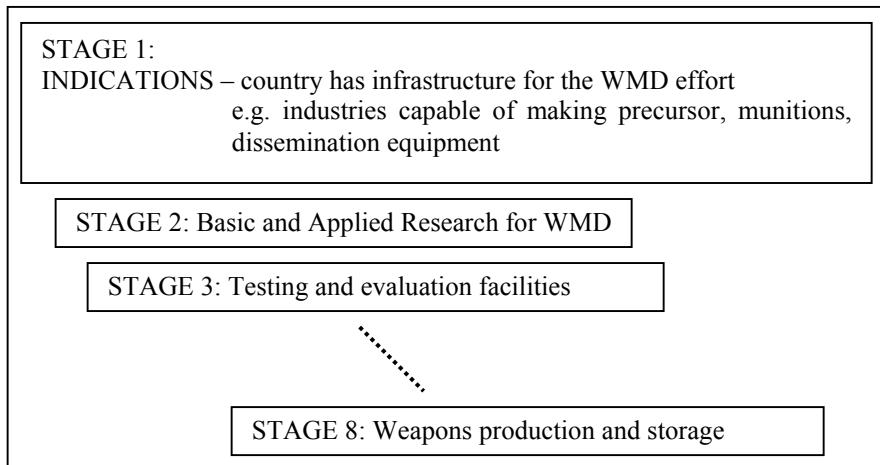
Ramping Interface

Figure 3. Ramping Interface.

3. RESEARCH ISSUES

3.1 *Research in NLP*

We do not wish to list or presume to recognize all the techniques that may be useful in accomplishing natural language understanding. Some of the obvious ones, however, can be suggested. For example, there are a host of disciplines in the area of pragmatics, such as: discourse structure, modeling speakers' intents and beliefs, situations and context – in short, everything that theoretical linguists have not formalized completely in terms of what it is to communicate and what Carnap would have considered to have been idiosyncratic, that is, not amenable to logical notation and manipulation.

Because the term “inference” is usually associated with the work of the knowledge base community manipulating axioms via logical operations such as “modus ponens,” let us examine for a moment non-logical inference which we characterize as making explicit that in text which is ambiguous or implicit. Horn (2001) states that “the absolute symmetry definable between affirmative and negative propositions in logic is not reflected by a comparable symmetry in language structure and language use.” In other words, language is messy. (See the examples in Figure 2 below.) Simply understanding the intended use of a polysemous word or resolving anaphoric references requires non-logical inference

derived from linguistic cues and context. Moreover, context itself, is a complex notion given that both the author-speaker and interpreter have wide latitude in constructing natural language context, and bring much to the table in terms of shared background knowledge in doing so. (Pinkal (1985) says that given “the individual biographies of the participants, the physical environment, the social embedding, the cultural and historical background, and...facts and dates no matter how far removed in time and space... ‘context’ can be the whole world in relation to an utterance act.”)

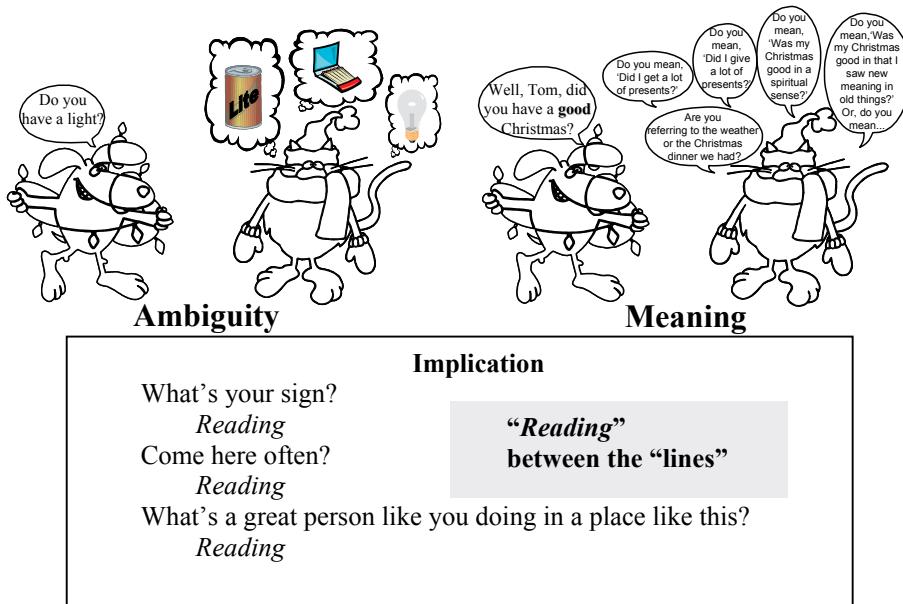


Figure 4. Ambiguity and Meaning.

Recent advances in NLP have made it possible to develop dialogue systems for many applications. Typically, a dialogue is managed by processing hand-crafted knowledge templates that model the characteristic events associated with the application. This framework is not viable in an interaction with an open-domain QA system. The LCC JITIQA project is currently implementing the dialogue architecture illustrated in Figure 5. Before submitting a question directly to the QA system, the question is parsed and transformed into a semantic representation that captures the relevant concepts and their interconnections. Then the question is analyzed to determine its degree of complexity and its relation to the previous dialogue. If the question is deemed to be new or unrelated to the prior utterances, it is presented to the QA system that returns a set of answers. If the question is contextually related, however, the question – or a dialogue manager-generated

answer – is first grounded contextually in the existing dialogue and a new question, containing the contextual information, is sent to the QA system. A template of the information-seeking task is built based on the question complexity and on the semantics of the answers. Each interaction fills a new template that is organized into a hierarchical structure by the operators defined according to a template algebra similar to the one reported in (Abella and Gorin, 1999). This organization enables analytical dialogue management capable of handling complex strategies. The decision maker associates actions with dialogue motivators thus generating follow-up questions or presenting information to the user. The dialogue motivators provide the decision maker with the dialogue strategies that govern its behavior. The template algebra provides the building blocks needed to create new dialogue motivators and analyze them. For example, one of the dialogue motivators detects when a topic switch has occurred and pushes all sub-dialogue templates on a stack.

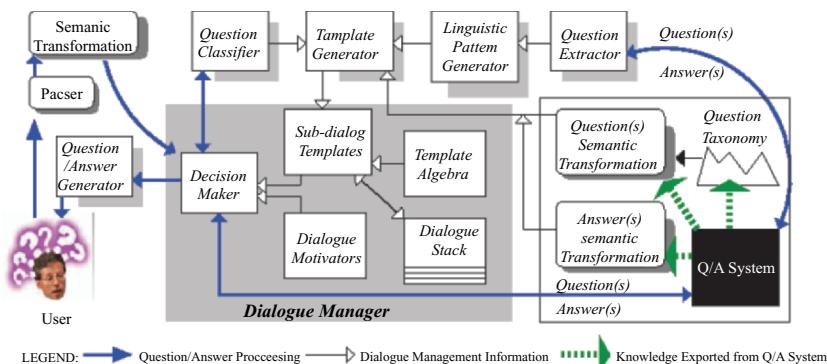


Figure 5. The Architecture of an Interactive QA system.

There is also a second usage of the QA system in the JITIQA interactive QA architecture. That is, the QA system is not employed to find answers to a user's question, but rather to provide information that enables the automatic generation of task templates. The information provided by the question classifier when combined with the semantic representations of the answers does not always generate templates encompassing all the typical events for a given problem. Therefore, the collection needs to be mined for additional related information. To mine the text, linguistic patterns typical for the domain of interest are generated following a method reported in (Harabagiu and Maiorano, 2000). The patterns are then mapped into a sequence of questions that are posed to the QA system. The additional template information is then acquired by IE techniques applied to the resulting answer passages. A by-product of this process is the acquisition of lexico-semantic information that is used in the generation process in conjunction with the production rules learned from the semantic representations.

3.2 *Research in IR*

In the area of IR, conceptual indexing and value filtering would be an important component to any QA system. Also, applying Bayes' algorithms in order to get beyond pattern recognition by grasping concepts beneath the surface (syntax) of text is a promising area of research; and, not only for the text processing community. The same techniques might help tackle the thorny issue identified in the HPKB Program that language-use linguists following Wittgenstein (Wittgenstein, 1958) have understood for years: Many words or concepts are not easily defined because they change canonical meaning in communicative situations and according to context and their relationship with other terms. Bayes' formulations can be trained to look specifically at these associative meanings.

Progress in traditional IR approaches have enabled computers to rank automatically retrieved data more efficiently. Despite this fact, people are still by far the best judges of what is relevant. For this reason, a QA system would benefit greatly from incorporating automatically the relevant judgments of analysts by implementing special search agents. To this end, the JITIQA project is trying to answer three specific questions regarding the interaction between the analyst and QA system in order to design and implement an enhanced QA system:

1. How does the use of just-in-time interactive search agents affect the way in which an analyst uses information provided by a QA system? Imagine a meticulous analyst who wants to be sure that she reads every piece of information related to her work. To make sure that nothing is missed, she searches her old memos, notes and reports after each answer is received from the QA system. A search agent would have a similar effect.

Of course, no one is as meticulous as the analyst described. Sometimes the analyst wants a piece of information and, therefore, in this instance the QA system is the appropriate tool. Other times a person will not bother to perform a search because of lack of time, because the information she has is already "good enough," or because she expects a search will not turn up anything useful. Such action (or inaction) is in keeping with Zipf's Principle of Least Effort which states that people will try to minimize their total future work, given their best estimates at the time.

When applied to the information search and QA domain, this theory suggests that if the cost of finding and using information is more than the expected value of that information, then the search will not be performed. This result could be the case for several reasons. First, the desired information may not be important enough. For example, the analyst perhaps thinks she "remembers it well enough" or there might be little reward for accuracy. The analyst might also think the search will not be fruitful and thus the expected value is low even if an unexpectedly good result would be worthwhile. Finally, the analyst could be under time pressure or dealing with a poor interface, and thus the effort for finding an answer is too costly.

Search agents greatly reduce the cost of searching for answers by doing most of the work automatically. The downside is that questions and queries are automatically generated and therefore will not be as exact as human-generated questions. This low-cost search for answers is not just a time-saver, it has the qualitative effect

of providing information in situations where the analyst is not personally willing or able to perform the search for answers. Additionally, it impacts the quality of the dialogue templates, and thus of the overall dialogue management.

2. How can a just-in-time interactive search agent automatically find information that would be useful for an analyst by looking at the current environment? The perfect QA system would instantly be able to know whether an answer would be useful to an analyst. The next best thing would be an answer engine that would know the analyst's task, what information she already knows, her current thinking, and how she processes new information. With such knowledge, a search agent would easily deduce whether certain information would be useful. Unfortunately, we have neither the ability to prognosticate nor to read minds. A search agent must make do with whatever limited information it can sense automatically from the analyst's computational environment.

Because it is impossible to directly sense whether information derived from a document or a structured knowledge base is useful, features from the context of the dialogue developed in the QA session need to be used as predictors or proxies of usefulness. For the search agents currently under development, several measures of similarity between the current context and a given answer will be used as proxies of usefulness.

Search agents need both ranked-best evaluation and filtering. Answer engines usually produce a ranked-best list of answers for a given question (e.g. TREC-like results). The absolute relevance score of a hit is not important as long as this relative ranking is correct. Search agents need a combination of ranked-best evaluation and filtering. They need to display the top few hits, but also need to display an absolute confidence in the relevance of a suggestion and to suppress the display of low-quality suggestions. This need is similar to the problem faced by text-filtering systems such as automatic newspaper-clipping services. However, these systems assume that a stream of documents is compared to a long-lasting set of query profiles, such as a list of keywords expressing user interests. Search agent queries are not long-lasting, so most of the machine-learning techniques used by these text-filtering systems cannot be applied.

Search agents learn relevance measures by extending learning techniques that extract correct answers. The fact that questions handled by search agents are not restricted to a number of predefined domains is a challenging feature. Developing a machine learning approach for open-domain QA agents has several advantages: (1) it eliminates the need for a knowledge engineer to craft the rules that extract answers and (2) it can scale up to a large number of new, unseen questions and adapt new knowledge for their resolution.

3. How should a just-in-time interactive search agent present potentially useful information? The most important design constraint for search agents is that reading the provided information should be a secondary task for the analyst. The design of a transparent interface must take into account two limitations on cognitive processes: *focused attention* and *divided attention*. Focused attention is the ability to attend to intended stimuli and tasks while ignoring others. The other side of the coin is *divided attention*: the ability to switch between tasks or stimuli. Generally speaking,

it is easier to focus one's attention when features of the environment being attended to are dissimilar to the distractions. Unfortunately, there is a tradeoff between focused and divided attention: the same similarity in display that makes it harder to focus on one stimulus and ignore the other makes it easier to switch focus between the two stimuli.

To make divided attention and task switching easier, a search agent should display information in a way that is congruous with the QA context to which it relates. Even when the interface is designed to help divided attention, a search agent will still increase the cognitive load. When the information is valuable, the tradeoff is worthwhile, but false positives will still be a problem. False positives are also guaranteed: a search agent will never reach 100 percent useful information without perfect information about the task that is investigated during the QA session. Given this limitation, information should be conveyed in stages through a technique called "ramping interface", first introduced in (Rhodes and Maes, 2000). (See discussion and diagram in Section 2.5.) Each stage of a ramping interface provides a little more information, at the cost of a little more attention required from the analyst.

The first stage can be referred to as the no-action, no-information stage. The search agents analyze the question or answer and decide whether to search for any suggestion at all. The decision is based on the analysis of the context of the conversation. If the information retrieved is above a certain relevance threshold, the ramping interface automatically jumps to the second stage and shows a suggestion note. In keeping with the philosophy that jumping to the next stage should be commensurate with the amount of work or attention required by the current stage, no user thought or action is required to display the suggestion note.

At this point the analyst may ignore the suggestion entirely, and the interaction has cost nothing more than some screen real estate and very minor cognitive load. If the analyst wishes to go to the next stage, she can quickly view a row of filled-in circles indicating the relevance value for that suggestion.

The next stage is reached by reading the suggestion description, which requires further attention from the analyst. Information in the suggestion is as concise as possible to allow rapid scanning for context. The interface box may contain many different kinds of information to try to contextualize the suggestion. At the final stage, the interface displays the most relevant keywords and their organization in the queries used by the search agents. The analyst needs only to click on a hyperlink and obtains the suggested document containing relevant information, pertaining to the answer she seeks.

3.3 *Research in AI*

Many questions require for their answer deep inferences from known facts or hypotheses. To manage these inferences, more research will be needed particularly in complementing logic-based inferences with probabilistic models. These probabilistic models require knowledge representations that can be easily derived from text. An example of such representations is the template format introduced in the MUC evaluations. Moreover, these templates should be open-domain, frame-like

structures determined by open-domain dialogues with the QA system. Open-domain dialogue templates can be generated by combining four knowledge sources: (1) the semantic transformations of the user's question; (2) the semantic transformations of the answers returned by the QA system; (3) a question taxonomy; and (4) information indicating the degree of complexity of a question.

Template instances are generated by matching linguistic patterns against dialogue utterances. A WordNet-based automatic method of obtaining linguistic patterns was presented in (Harabagiu and Maiorano, 2000). We propose to use this method as it provides a more flexible approach than the semi-automatic methodology presented in (Bagga et. al., 1997) or the corpus-based method from (Yangarber et. al., 2000), that requires a pre-filtered set of texts.

The application of linguistic patterns generated by this procedure creates template instances. Figure 4 illustrates a dialogue template as well as a linguistic pattern that fills its slots.

<i>Template</i>	<i>Linguistic Rule</i>
Template-Name: RELATION Entity-1: Sakakibara[PERSON] Entity-2: Joseph Stiglitz[PERSON] Entity-3: IMF[ORGANIZATION] reform Relation-type: joint research project	Person-1 and Person-2 <start-verb><work-relation-noun> Text Sakakibara and Joseph Stiglitz, former chief economist of the World Bank, are about to embark upon a joint research project into Asian economic growth and IMF reform.

Figure 6. Examples of dialogue template and linguistic rule that instantiates its slot fillers.

Six relations and two operators structure the template instances into a template algebra. Each template is defined to have three fields: a Name, identifying the discourse template that it instantiates, a *slot-body*, representing all the slot names, and each slot may have a slot-value. All slots that are filled with values are collected into a *filled-slot-body*. Sometimes, slot-values are templates themselves. The values of discourse templates may be only other discourse templates. The six relations are:

- **Equality(T_1, T_2)** holding when both template instances T_1 and T_2 have the same name and the same filled- slot-bodies.
- **Restriction(T_1, T_2)** holding when T_1 and T_2 have the same name but *filled-slot-body*(T_1) Management.
- **Containment(T_1, T_2)** holds when $Name(T_1) \in filled-slot-body(T_2)$, thus they do not share the name. Moreover, $filled-slot-body(T_1) \subset filled-slot-body(T_2)$.
- **Generalization(T_1, T_2)** is defined as $Containment(T_2, T_1)$.
- **Symmetric-Generalization(T_1, T_2)** holds if both $Generalization(T_1, T_2)$ and $Generalization(T_2, T_1)$ are true.
- **Containment-Generalization(T_1, T_2)** holds if there is a subpart $b \in filled-slot-body(T_1)$ such that $Symmetric-Generalization(b, T_2)$ is true.

The two operators from the template algebra are the *unification* and the *projection*. The *unification* generates a new template instance that has slot values resulting from the unification of the argument slots. The projection operator generates a template containing only certain required (projected) slots. The projection operator is extremely important as it detects ambiguities in the dialogue. For example, Figure 5 illustrates the projection of template T_1 through template T_2 , resulting in an ambiguous construct.

AMBIGUITY is a special template that does not have any slot-name, but is filled with other templates.

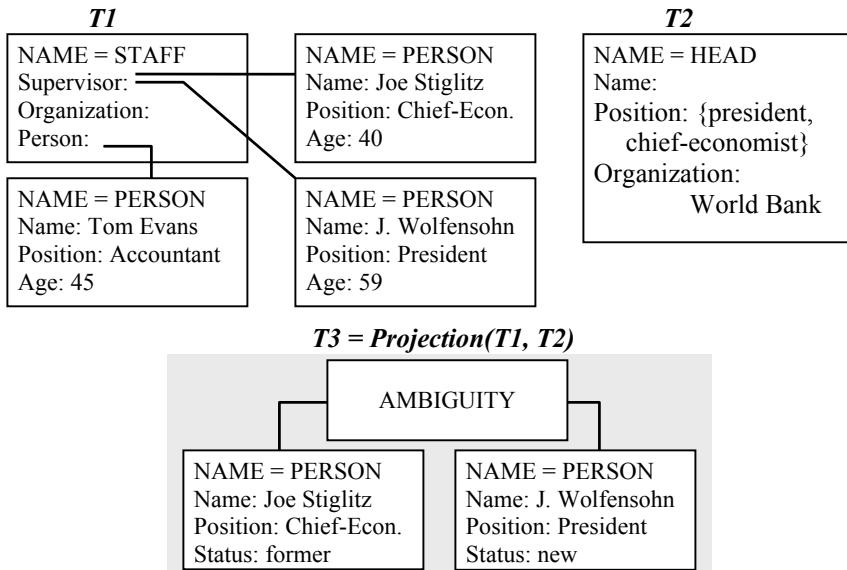


Figure 7. Ambiguity obtained through template projection.

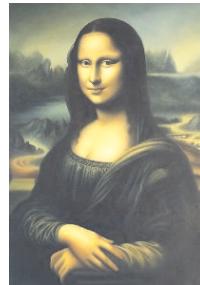
4. POSSIBLE IMPACT

More and more frequently one hears the call for analysts to “think out of the box.” It is a cliché endemic not just to the intelligence community that requires “dots to be connected,” but also to the commercial sector or wherever creative thinking is valued highly. Except perhaps for the extraordinary polymath, we contend that, taken literally, this is impossible to do. Cognitively we are not adequately equipped. Analysts are overloaded with data; following the time-honored division-of-labor management philosophy, they are restricted to narrow domains; and psychologically – due to the data overload in the first place – they are predisposed to look for corroborating evidence for a single hypothesis, and are fortunate if there is sufficient time to do this in an exhaustive fashion. At a deeper level of methodology there is another problem that we call the “text immutability” factor.

The Gricean maxims (Grice, 1975) detail how people in conversation are able to communicate effectively through the process of negotiation and repair among other things. Anyone who deals with written material is put at an immediate disadvantage in that negotiation and repair are not tools available for clarifying meaning. Text is the paradigmatic WYSIWYG world. Take the cartoon below. Although the text snippet's context is severely constrained, it is generous compared to that of a database and similar to what a person would encounter in a newspaper article. The point, however, is that there are a number of questions one would like to ask such as: When was the book published? Is this a book review? What are the author's credentials in economics? How figurative or literal is the use of the term 'demolish?' Moreover, if the author is doing more than demolishing, what could that possibly be? Something constructive? One can query the author of the book and the article ad nauseum, but the best one can expect in return is an enigmatic smile!

Immutability

Rothschild's new book "Economic Sentiments" does something more than merely demolish hostile reactions to Smith and Condorcet....



Finally, unlike the above-mentioned questions, many of the questions that preoccupy intelligence analysts are open-ended and unanswerable. They are the stuff that motivate the class of intelligence community (IC) reporting called National Intelligence **Estimates**. This is an aptly named and revealing moniker. There is no definitive answer. The questions belong to that class of work Pearl says "requires that conclusions be drawn from uncertain clues and incomplete information."

The picture that emerges is one of an analyst who is the product of our scientific age and imbued with its scientific prejudice: In objective inquiry, evidence and facts are like objects just "out there" waiting to be discovered. But scientific analysts have their own biases that shape the questions they ask. Furthermore, the objects are embedded in a context that, as was noted previously, is more likely to be global rather than local in its scope. If an analyst, moreover, is dealing with a database rather than text, context is almost nonexistent because the information has gone through the process of "information laundering." In order to cope with the preponderance of data bombarding us today, it is necessary to capture, encapsulate, and abstract these data so that the resulting formulated information and knowledge is manageable. This usually means that the information has become quantified so that it can be crunched on a computer and graphed. The negative consequence of this

process is that the information – whether it has been quantified or not – is almost totally decontextualized; the original meaning is lost or, at best, difficult to recover; and, since it is tersely encapsulated, there is no room for the type of narrative information not amenable to abstraction: history, ideals, and passions. (Can suicide bombers be rationally or quantifiably understood and explained?)

The intelligence analyst is not much different from the typical consumer and user of data. With one caveat: there is more secrecy and compartmentalization of information so collaboration and interaction are much more constrained. Therefore, the individualistic predisposition to working alone in a narrow domain is buttressed by an infrastructure that fosters “within-the-box thinking” notwithstanding the many creative thinkers there are.

This being the case, it is clear that the ultimate QA system is one way to compensate for an individual’s limitations in terms of experience, expertise, and bias. A dialogue-based tool that enables interaction, even in a virtual sense, is a tool for not just searching, but for clarifying, exploring alternate hypotheses, obtaining different perspectives, asking the right questions. In keeping with the spirit of this paper, perhaps it is more suitable to ask if this is a better way to maximize “within-the-box thinking?”

Picture the physical environment of today’s symbolic analysts. It is the picture of individuals in cubicles tethered to computers. It is hard to visualize a scene such as the one from the first day of the Cuban Missile Crisis on October 16, 1962. The transcription from *The Kennedy Tapes* (May and Zelikow, 1997) is as follows:

*What kind of world do we live in? [geopolitical situation]
In Cuba what action do we take?
What do we expect Castro will be doing after you attack these missiles?
Does he survive as a political leader?
Is he overthrown?
Is he stronger? Weaker?
How will he react?
How will the Soviets react?
How could Khruschev afford to accept this without some kind of rebuttal? So where?
How do we react in relation to it?
What happens when we mobilize?
How does this affect our allies’ support of us in relation to Berlin?*

Aside from admiration at the mental dexterity of Secretary of Defense McNamara, one can see, these questions are not eliciting facts; they are open-ended and hypothetical for planning purposes; the stuff of National Intelligence Estimates. And, although the transcription of the other participants is not included here, these questions initialized a long, intense dialogue. It is the kind of scene one would wish to see an analyst be able to simulate with a computer tool that would enable interaction with data.

5. CONCLUSIONS

When trying to reformulate ideas for a technology program and determine what is worthy for support in research, we often return to philosophy and history. And when the topic under consideration is dialogue it is impossible not to re-visit Plato. In his day, philosophy could only be carried out by means of a community of life and dialogue between masters and disciples. Hadot (2002) writes that a true dialogue was possible only if the participants want to dialogue. There was an implicit agreement between the participants which is renewed at each stage of the discussion; neither one of the participants imposes his truth upon the other. On the contrary, dialogue teaches them to put themselves in each other's place and thereby transcend their own point of view. The participants discuss within themselves a truth that is independent of them so (as in all ancient philosophy) philosophy consists in a movement by which the individual transcends himself toward something which lies beyond him. Plato called it logos. This something was not an absolute knowledge, but the agreement established between participants by transcending particular points of view. We would add that this agreement is not static. It is valid for a specific point in time, and not the stuff of database fodder. The validity of the agreement or understanding between the participants has a short shelf life. Imagine the dynamism of the previously mentioned Cuban Missile Crisis Scenario.

6. ACKNOWLEDGEMENTS

This essay has profited greatly by the encouragement of the editors, substantive discussions with University of Texas at Dallas Prof. Sanda Harabagiu, the perspicuous observations of two colleagues, Dr. Jean-Michel Pomarede and Wiley Harris, and UTD graduate student Finley Lacatusu's attention to detail.

7. REFERENCES

- Abella, Alicia and Allen L. Gorin. (1999) Contract Algebra: Analytical Dialogue Management. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99), pages 191-199, 1999.
- AQUAINT Program <http://www.ic-arda.org/InfoExploit/aquaint/>
- Bagga, Amit, Joyce Yue Chai and Alan Biermann. (1997) The Role of WordNet in The Creation of a Trainable Message Understanding System. In Proceedings of the 14th Conference on Artificial Intelligence (AAAI/IAAI-97), 941-948.
- Cohen, P., Schrag, R., Jones, E., Pease, A., Lin, A., Starr, B., Gunning, D., and Burke, M. (1998) The DARPA High Performance Knowledge Bases Project. AI Magazine, Winter, 1998.
- Franconi, Enrico. (2003) "Natural Language Processing." In The Description Logics Handbook edited by Franz Bader et al., published by Cambridge University Press, Cambridge 2003, p. 451.
- Grice, Paul J. (1975) Logic and Conversation. In P. Cole and J. Morgan, editors, Syntax and Semantics Vol. 3: Speech Acts, pages 41-58. Academic Press, New York, 1975.
- Hadot, Pierre. (2002) What Is Ancient Philosophy? translated by Michael Chase, published by The Belknap Press of Harvard University Press, Cambridge Mass. 2002.
- Harabagiu, Sanda and Steven Maiorano. (2000) Acquisition of Linguistic Patterns for Knowledge-Based Information Extraction. In Proceedings Second International Conference on Language Resources and Evaluation (LREC-2000), Athens, Greece, 2000.

- Harabagiu, Sanda, Marius Pasca, and Steven Maiorano (2000a) Experiments with Open-Domain Textual Question Answering, in Proceedings of COLING-2000, August 2000, Saarbrucken Germany, pages 292-298.
- Harabagiu, Sanda, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu (2000b). FALCON: Boosting Knowledge for Answer Engines, in Proceedings of the Text Retrieval Conference (TREC-9), November, 2000.
- Hobbs, Jerry. (1998) Private communication.
- Horn, Lawrence R. (2001) A Natural history of Negation, CSLI Publications, Stanford.
- Lehnert, Wendy (1978). The Process of Question Answering: a computer simulation of cognition. Lawrence Erlbaum.
- May E. and P. Zelikow (1997) The Kennedy Tapes edited by Ernest May and Philip Zelikow published by The Belknap Press of Harvard University Press, Cambridge Mass. 1997 p. 115.
- MUC Message Understanding Conference http://www.itl.nist.gov/iaui/894.02/related_projects/muc/
- Pearl, Judea (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, San Mateo, CA.
- Pinkal, Manfred (1985). "Kontextabhangigkeit, Vagheit, Mehrdeutigkeit." In Schwarze, Christoph and Wunderlich, Dieter (editors), Handbuch der Lexikologie, Konigstein: Athenaum Verlag. The quotation used is translated and cited in: Quasthoff, Uta M. "Context." In: Asher, R.E. (editor) Encyclopedia of Language and Linguistics, Pergamon Press, 1994, 730-737.
- Rhodes, B.J. and P. Maes. (2000) Just-in-time information retrieval agents. IBM Systems Journal, 39(3&4), 2000.
- TIPSTER Text Program http://www.itl.nist.gov/iaui/894.02/related_projects/tipster/
- Voorhees, Ellen M. and Donna Harman (1999) NIST Special Publication 500-246: The Eighth Text REtrieval Conference (TREC 8).
- Voorhees, Ellen M. and Donna Harman (2000) NIST Special Publication 500-249: The Ninth Text REtrieval Conference (TREC 9).
- Wittgenstein, Ludwig (1958) Philosophical Investigations, translated by G.E.M. Anscombe, Macmillan Publishing Co., New York.
- Yangarber, Roman Yangarber, Ralph Grishman, Pasi Tapanainen and Silja Huttunen. (2000) Unsupervised discovery of scenario-level patterns for information extraction. In Proceedings of the Conference on Applied Natural Language Processing and the North American Chapter of the Association for Computational Linguistics, (ANLP-NAACL-2000), pages 282-289.

APPENDIX

The following scenarios are presented here to illustrate the nature of the human-computer interaction that is envisioned as the holy grail for the AQUAINT Program.

BACKGROUND:

Jackie Ryan is an economist with fifteen years' experience in analyzing macroeconomic policy in the Asian-Pacific region, and bilateral trade issues between the US and countries of that region. However, she has no in-depth familiarity with any of the cultures in the area although she did spend one year in Japan as part of her graduate training where she learned some rudimentary Japanese. In the past, JR has reported on issues concerning the monetary policy of Indonesia, Malaysia, and China. She wrote economic analyses pertaining to US-Japan trade in the high-technology sector as part of the US Government's semi-conductor negotiations with Japan in the late 1980s. Like many other economic analysts, JR also reported on aspects of the "Asian financial crisis" of 1997. (In her case it was the response of Indonesia to bailout measures; Japan played a role, but this was not the focus of JR's work.)

During the past three months, JR has started concentrating on Japanese economic and trade issues such as bid-rigging in the construction industry, Japanese economic reform initiatives and monetary policy to combat its recession, and proposed financial sector deregulation. Currently, she is investigating Japan's relationship vis-à-vis the World Bank and International Monetary Fund.

At the time of JR's assignment to the Japan desk, she followed the routine she had used in the past when taking on a new domain and topic: JR used the single keyword, "Japan," to query the in-house repository of newswire, video clips, broadcast transcripts, and internal reports for the past three years in order to cast as broad a net as possible over all items relating to Japan since her last encounter with Japan in 1997. JR also created a search profile of her interests so that new incoming data would be added to the initial archive of 500,000 retrieved documents. The video clips constituted a separate 50-GB repository. JR, moreover, chose to keep her existing profile of the past three years since it dealt with questions of international purport and might be relevant to her current assignment.

To complement her three-year collection of data, JR also has at her disposal several other legacy data and knowledge bases, including the CIA World Factbook, maintained by various offices organized by country. JR is confident that with the aid of the suite of IT tools she has used in the past together with the recently installed QA system, she will be able to craft timely and comprehensive analytic pieces.

SCENARIO 1:

JR is investigating Japan's relationship to the World Bank and IMF. Through her previous work on the "Asian financial contagion," she is knowledgeable about the international levers of power and influence. JR's initial reporting deals with the Japan side of the equation. She recalls the Japanese role vis-à-vis the Indonesian bailout, but recalls only vaguely the names of the ministries, leaders, banks, financial think-tanks and the like. The functions of the various principal players and entities are known precisely to JR.

The types of questions posed by JR to the QA system are matters of fact. Contextually, the facts may differ in terms of time, but, basically, the questions are what JR considers to be straightforward. Since the QA technology is relatively immature and unsophisticated in terms of its overall systemic complexity, JR directs her questions to the specific data repository wherein she thinks intuitively the answer resides. There is not yet a "brokering" capability by which questions are automatically routed. The outline of her initial paper has been completed, but she requires some precise answers to some specific questions to finalize the task. JR remembers that the head of the World Bank was "Wolfson," or something akin to that. She queries one of the databases:

"Who was the head of the World Bank?"

The system immediately responds: "President James Wolfensohn." It also returns the answer "chief economist Joseph Stiglitz."

JR realizes that the query term "head" is a bit vague, but is satisfied that the name of the person she was seeking has been provided. She is also pleased to see the name Stiglitz because she knows that he no longer holds that position, and this is

information that might be useful to her, if not now, then perhaps in a later report. JR does wonder, however, about the trustworthiness of the database because she herself was a bit lax in the care-and-feeding duties of her own office's factbase. Therefore, JR gives the same query to the 500,000 collection of retrieved documents to corroborate the time frame for Wolfensohn's tenure. JR receives the same answer, but this time she looks at the source document to check the date of publication. Sure enough the answer is correct. In addition, the same document highlights the fact that Wolfensohn is in his second term so JR knows that he not only "was," but still "is" the World Bank president which was going to be her next question.

Next, JR moves on to the IMF. She knows that Camdessus resigned as managing director, but she does not know who succeeded him.

Because she knows that Camdessus' resignation was fairly recent and, therefore, the factbase may not have been updated, she queries the document collection:

"Who is the head of the IMF?"

The system returns the answer: "In March 2000, the IMF chose as its head Horst Kohler..."

JR is satisfied with this answer, but looks at the source document to see what other useful information might be gleaned. The above passage continues to say: "...former head of the European Bank for Reconstruction and Development." Useful information indeed. JR knows that the IMF traditionally picks a European for its top post, but she is not very knowledgeable about that region having spent most of her time analyzing Asia. Knowing Kohler's background could hardly be considered gratuitous information she feels.

Since Kohler is completely unknown to her, JR decides to query the 50 GByte repository of video clips to see if there is a picture that might help her visualize the kind of person the new IMF head is. Although European bankers are somewhat reticent personalities and not known for the media visibility of their US counterparts, there is a brief interview of Kohler with German television at the G-8 economic summit meeting held in Cologne in February 1999. The interview is in German with close captioning in English.

Next, JR moves on to detect and verify the names of the Japanese leaders and their current positions. In all cases, there are no video clips of these bureaucrats, but the occasional newspaper and magazine picture is returned by the system. She obtains the necessary information with such alacrity that she takes a brief moment to run an experiment. Implementing a strategy that predates the QA system, JR uses the IR engine in her suite of tools to search for Eisuke Sakakibara whom she knows to have been a vice minister in Japan's powerful Ministry of Finance (MOF) and has recently become head of a new center affiliated somehow with Keio University, a prestigious private university strong in economics. The traditional search engine retrieves 1,000+ documents referring, ostensibly, to Sakakibara. Since this is too much data to peruse in the hope that she'll learn serendipitously the name of the new center at Keio, JR tries "Sakakibara" and "Keio." However, she obtains the same number of documents, many of them discussing Keio University with no mention of Sakakibara. After skimming 10 documents, she quits and uses the QA system.

“Who is Sakakibara?”

In short order the QA system responds with ten answers, eight of which refer to him as an MOF vice minister, and two which mention him as head of the Global Security Center (GSec) at Keio University. Further reading of the source documents reveals that Sakakibara resigned from MOF in July 1999, and became the head of GSec in April 2000. In addition, JR learns that in the interim between his resignation and taking up his new post, “Sakakibara and Joseph Stiglitz, former chief economist of the World Bank, are about to embark upon a joint research project into Asian economic growth and IMF reform.”

“Aha!” she thinks, “Now I know everything. This really closes the loop on my report.” JR also makes a mental note, albeit a vernacular but poignant one, about the QA system: “I like it. It’s gotta a chance!”

SCENARIO 2:

It is a few years later, and the QA system has undergone several iterations of improvement which make it a sophisticated analytic aid. JR herself has grown along with the system; covering the same account, she has become much more knowledgeable of the domain and, therefore, her information needs have gone beyond just requests for simple facts. Although decidedly in the “non-techie” category of user, JR has noticed several obvious aspects of the QA system. She no longer has to direct her questions to separate data stores, but asks once, and receives answers that seemingly emanate from text, databases, and other forms of media and languages en masse. JR deduces this from the fact that some answers are short as if from a factbook or gazetteer, and others are longer as if they were summaries from several documents. And, of course, the Japanese material is obvious, if not completely recognizable to her. The system, moreover, has incorporated visualization and navigation tools which are not intrusive, but transparent and intuitive to use and, overall, promote the feeling of her being in control of her search space. The most notable feature of the new system, however, is, for lack of a better term, its interactive nature. If questions are unclear, the system prompts her for clarification. Although – she would be quick to point out to anyone who is interested – the system seemingly has been requiring clarifications less and less frequently. Not to be too anthropomorphic about it, but JR feels the system anticipates her intentions more effectively. No doubt the result of the QA system “learning” from her six-year log of questions to the system. It must be due to something like that because she knows that her questions have not become any less ambiguous.

The repository of data that JR now has on hand has grown ten-fold, and the incoming daily take from the retrieval system is growing at an alarming rate. JR is following the same issues.

She first asks, however, the QA system to track an event that just began unfolding a week ago. Although it is a political leadership struggle, JR is interested only so far as the outcome might effect economic policy. She requests:

“What is the situation with Prime Minister Mori on November 14, 2000?

JR asks this in order to establish a baseline event. The system returns information regarding reports on a potential vote of no-confidence. Some of the information is in Japanese and Chinese, probably from Xinhua she thinks. JR then requests that the system follow this event as it unfolds and update her daily with only new information. The system acknowledges that it will conduct event-tracking and updating. It asks whether or not video clips are to be included in the daily updates. JR responds "No." She is not **that** interested. She does, however, want essential items from the Japanese source documents extracted and translated to get a flavor of the personalities appearing in the daily reports. These names can be used to update the office factbase devoted to Japanese leadership. JR will have these extracted names routed to her separately in order to validate them before they are inserted into the factbase.

Since the QA system has access to a model of Japanese factional politics, she asks it to predict not what the outcome of a new election would be, but whether the vote of no-confidence will come about at all. The system, knowing that the Liberal Democratic Party may not be able to hold together a coalition allowing it to remain in power if a general election is called, predicts that the threatened vote of no-confidence from within the LDP will not happen. The degree of confidence is given to be 70%.

"What if there is not a no-confidence motion?"

The system responds: "The LDP will remain in power, but not with the current Prime Minister."

JR figures that to be about right. The disaffected LDP faction member(s) will cut a backroom deal, and that faction's leader will be the next Prime Ministerial candidate. In any event, it's clear to JR that Mori's days as PM are numbered. On to more important matters she thinks to herself.

"What is the import-export trend over the last six months?" JR congratulates herself on remembering to specify a time frame.

The system responds: "Do you want statistics for Japan and the US?"

Admonishing herself for yet another non-specific question, JR replies "Yes," and then quickly adds "broken down by sector." She knows that while the US economy is booming and Americans are not paying attention to the growing trade imbalance, it is more important to know in which sector the imbalance is greatest.

The system returns the desired trade figures in tabular format. The system knows that the primary sector can be further divided into such items as raw materials and agricultural goods, and that different ministries and agencies compile statistics on the same sectors and products. As has been the trend for several years now, the US shows an export balance in wheat, corn, and especially soybeans. What JR cannot immediately figure out is why the system returned two different tables for agriculture, especially since the information is usually merged. Then she notes that there's no mistake: The Ministry of Agriculture, Forests, and Fisheries compiles statistics on a calendar-year basis while those of Ministry of International Trade and

Industry's (MITI) are gathered on a fiscal-year schedule. She can employ usefully both sets of data.

What JR is really curious to know, however, is why, except for some agricultural products, the US consistently runs a trade deficit with Japan in most sectors, especially manufacturing? She has only been covering Japan a few years, but this trend seems more like an historical pattern; and, the history of Japan is a bit off JR's radar scope. She does know, however, that such things rarely arise due to a country's macroeconomic or microeconomic maneuvering alone. This is more of a politico-economic issue. Hoping that her data repository contains appropriate internal reports by other analysts that cover events which predate the relatively recent six year-old vintage of items in the repository, JR queries the QA system:

"What are the historical or political reasons for the current trade imbalance between the US and Japan?"

The system returns what looks to be a summary from multiple analytic reports: "Due to war in Korea, Japan became the home for US troops and military bases. In addition, the US sought to contain China and the Soviet Union. The US became the biggest market for Japanese producers... Conclusion: The US at this time considers political and security issues to be more important than economic or trade issues."

The system also suggests an hypothesis or linkage for JR to ponder: "Taking into account the sequence of your previous questions overall, suggest you consider the idea that Japan's prime ministers are weak due to the country's emphasis on economic matters (?)"

More than plausible JR thinks to herself. She then asks for "justification" in order to trace the reliability of the sources, the integrity of the summarization, and, above all, the certainty of the reasoning and conclusion. The QA system displays five "think pieces" all written within the last six years.

Two of the pieces were written by a colleague covering security issues between the US and Asia. The system highlights relevant portions of the text discussing student demonstrations over renewal of the US-Japan Security Treaty, demonstrations against US warships' carrying of nuclear weapons into naval bases in Japan such as Yokosuka. Both reports contain sections on the origin and evolution of the Security Treaty.

Another report discusses Japan's trade with China, and other Southeast Asian countries. The reference to China states that "Japan, although intent on increasing its trade with China, pursues this policy with circumspection since it still relies upon the goodwill of the US which provides shelter under its nuclear umbrella and guarantees Japan's security in the region."

The remaining two reports are on the rape of a school girl by US soldiers in Okinawa. These reports discuss at length the preponderance of US military bases on Okinawa and the rest of Japan.

The system displays the discrete portions of the relevant text in such a way that JR is able to view them as premises in a line of reasoning. The snippets of text are extracted out of the global context of the reports, but form a valid local context in the generated summary. The conclusion, as far as JR can tell, seems well-grounded and logical. Which, of course, doesn't make it the whole truth, but does make the

conclusion in itself another valid premise for the system's suggested hypothesis with regards to the center of power in Japan. Certainly it was the dynamism of its industrial policy that attracted JR to Japan in the first place, and not its political structure which seemed weak and ineffectual by comparison.

JR realizes that if what the system has hypothesized about the locus of power in Japan being centered in such bureaucracies as MOF and MITI is correct, then her report of three years ago on Japan's relationship to the World Bank and IMF and the plan that Sakakibara helped establish for aiding Southeast Asia could be viewed in a whole new light.

"I am interested in discussing the topic of acquiring political hegemony by economic means. Could we begin with some historical precedents?"

"Yes, but may I suggest you make a pot of coffee?"

REVERSE-ENGINEERING QUESTION/ANSWER COLLECTIONS FROM ORDINARY TEXT

Abstract: Researchers have begun to investigate the use of statistical and machine learning methods for question answering. These techniques require training data, usually in the form of question/answer sets. In this chapter, we describe a reverse-engineering procedure that can be used to generate question/answer sets automatically from ordinary text corpora. Our technique identifies sentences that are good candidates for question/answer extraction, extracts the portions of the sentence corresponding to the question and the answer, and then transforms the information into an actual question and answer. Using this procedure, a collection of questions and answers can be automatically generated from any text corpus. One key benefit of this automatic procedure is that question/answer sets can be easily generated from domain-specific corpora, creating training data which could be used to build a Q/A system tailored for a specific domain.

1. INTRODUCTION

Anyone who uses the World Wide Web will appreciate the need for systems that can accept a question expressed in natural language and find the answer in text, such as web pages or newspaper articles. There are numerous applications for question answering systems, including web searches, database queries, fact extraction, and intelligent user interfaces. In the last few years, there has been a surge of interest in computational question answering systems, both in the information retrieval community (TREC-8 Proceedings, 1999; TREC-9 Proceedings, 2000) and the natural language processing community (Charniak *et al.*, 2000; Hirschman *et al.*, 1999; Ng *et al.*, 2000a; Riloff and Thelen, 2000; Wang *et al.*, 2000).

One of the most promising directions for question answering research is the use of statistical methods and machine learning. Statistical techniques have produced good results for many natural language processing and information retrieval tasks, such as part-of-speech tagging (Weischedel *et al.*, 1993; Church, 1989), parsing (Charniak, 1993), word sense disambiguation (Yarowsky, 1992; Yarowsky, 1995), and document retrieval (Turtle and Croft, 1991). The main bottleneck behind these methods is that they require large amounts of training data. Researchers have begun to use statistical and machine learning methods for question answering and have acquired some question/answer collections for training (Section 1). But most of the existing Q/A collections are problematic for various reasons. For example, these collections may be relatively small in size, protected by companies as proprietary information, expensive to obtain, or limited to specific topics. For learning algorithms to be maximally effective, large volumes of training data are needed and the training data should be

similar to the application domain for which the resulting Q/A system will be used. For example, a Q/A system to access medical literature must have the medical vocabulary and specialized jargon needed to effectively recognize medical information. A question answering system trained on general news articles probably would not work well on medical literature.

Relying on existing question/answer collections as training data is easiest when appropriate collections are available, but that is not always the case. As an alternative, we have developed a reverse-engineering technique that can automatically generate questions and corresponding answers from ordinary texts. Our technique opens the door for question/answer collections to be created automatically. These question/answer collections could be used to augment and enhance existing resources, or to train Q/A systems for specialized domains.

This chapter is divided into three main sections. In the first section, we give an overview of the state-of-the-art in statistical and machine learning approaches for computational question answering. In the second section, we describe our reverse-engineering procedure, which uses a shallow parser and heuristic rules to transform ordinary sentences into questions and answers. In the third section, we present examples of questions and answers generated from four text corpora representing three domains, and show that the resulting Q/A collections produce different statistical profiles for the three domains.

2. BACKGROUND

There has been much activity in recent years in training different parts of question answering systems and on building resources automatically to improve question answering performance. In this section, we describe some of the Q/A data sets that have been used and give an overview of recent research on statistical and machine learning approaches for Q/A. In Section 2.1, we describe the three main Q/A data sets. Section 2.2 presents a general overview of how most question answering systems work, focusing on two types of information, answer type and answer context, which are necessary to pick out short answers from a large text collection. Section 2.3 describes recent research in machine learning techniques for question answering, as well as automatic methods for building resources for Q/A systems.

2.1 *Q/A Data Sets*

In recent years, three main collections of questions and answers have been used by researchers to train and test learning algorithms for question answering.

TREC Q/A Collection

Since 1999, the TREC organizers have sponsored a question answering track (TREC-8 Proceedings, 1999; TREC-9 Proceedings, 2000; TREC-10 Proceedings, 2001; TREC-11 Proceedings, 2002). Many Q/A researchers participate in this international evaluation of question answering systems. The task of the TREC Q/A track is to find the answer to factoid questions using a document collection

consisting of roughly 500,000 newswire documents. For each question, systems are allowed to return a ranked list of 5 short (either 50-character or 250-character) responses. As a service to track participants, AT&T provides top documents returned by their retrieval engine for each of the TREC questions.

Every year, the TREC organizers create a new test set of questions and answers. In early evaluations, the questions were back-generated by participants from answers they found (manually) in the text collection. For later evaluations, the questions and answers have been collected from on-line search data, thus reflecting more closely the information needs of real users. After a TREC evaluation is completed, the test set becomes a valuable resource that can be used as training data for subsequent research. Several Q/A researchers have used previous TREC Q/A collections as training data for their learning algorithms.

Reading Comprehension

(Hirschman *et al.*, 1999) first proposed the idea of *reading comprehension* as an evaluation task for question answering systems. In the reading comprehension task a short text is presented, followed by one or more questions. The goal is to find the answers to the questions within the text presented (e.g., Figure 1). Two reading comprehension data sets have been used by Q/A researchers:

1. **Remedia Reading Comprehension Exams:** These tests were developed by the Remedia Corporation. There are 120 stories written for 3rd to 6th graders. Each story has five questions attached, each asking one of “Who”, “What”, “When”, “Where”, and “Why”. An answer key is provided.
2. **CBC Reading Comprehension Exams:** These texts were collected from the Canadian Broadcasting Corporation web page for kids (<http://cbc4kids.ca/>). There are 259 stories that seem to be aimed at elementary and middle school students (eight to thirteen year olds). The stories are often based on newswire articles and fall into the following domains: politics, health, education, science, human interest, disaster, sports, business, crime, war, entertainment, and environment. A group at The MITRE Corporation has created 8-12 questions and an answer key for each story.

The reading comprehension task differs from the TREC Q/A task because the space in which to look for answers is much more constrained (i.e., just a single document). This simplifies the search in some respects, but (Light *et al.*, 2001) shows that it can be more difficult to find an answer in a single document than from a large text collection. The reason is answer redundancy: there are often many instances of the correct answer in a large text collection, but there is usually only one opportunity to find the answer in a specific document. Results from (Ravichandran and Hovy, 2002) and (Brill *et al.*, 2002) are consistent with this observation, suggesting that searching on the Web may lead to better performance than searching in the TREC collection because the Web is so much larger.

Mars Polar Lander - Where Are You?

(January 18, 2000) After more than a month of searching for a signal from NASA's Mars Polar Lander, mission controllers have lost hope of finding it. The Mars Polar Lander was on a mission to Mars to study its atmosphere and search for water, something that could help scientists determine whether life ever existed on Mars. Polar Lander was to have touched down December 3 for a 90-day mission. It was to land near Mars' south pole. The lander was last heard from minutes before beginning its descent. The last effort to communicate with the three-legged lander ended with frustration at 8 a.m Monday. "We didn't see anything," said Richard Cook, the spacecraft's project manager at NASA's Jet Propulsion Laboratory." The failed mission to the Red Planet cost the American government more than \$200 million dollars. Now, space agency scientists and engineers will try to find out what could have gone wrong. They do not want to make the same mistakes in the next mission.

- When did the mission controllers lose hope of communicating with the lander?
(Answer: 8AM, Monday Jan. 17)
- Who is the Polar Lander's project manager?
(Answer: Richard Cook)
- Where on Mars was the spacecraft supposed to touch down?
(Answer: near Mars' south pole)
- What was the mission of the Mars Polar Lander?
(Answer: to study Mars' atmosphere and search for water)

Figure 1. Sample CBC reading comprehension exam.

Reading comprehension tasks are appealing for other reasons as well. Since they are administered to children at graded levels, computer performance can be graded as well, with progressively more difficult exams being given. Computer performance could also be correlated with human performance at different grade levels.

<p>Question : Who invented eyeglasses?</p> <p>Explanation : Marco Polo reported seeing many pairs of eyeglasses worn by the Chinese as early as 1275, 500 years before lens grinding became an art in the West.</p> <p>Answer : The Chinese</p> <p>Wrong Answers: The Japanese, Marco Polo, Benjamin Franklin</p>
--

Figure 2. Typical Trivia Game Database Entry.

Trivia Questions

A third data set comes from trivia games (Figure 2). Like the questions in the TREC collection, the trivia game questions are factoids that typically require a noun phrase as the answer. Data sets from the trivia games Phishy (MacDonald, 1999) and

TriviaSpot (TriviaMachine Inc., 1999) have been used for training and testing. Typically, each Q/A entry in the trivia game includes a question, a short answer, a few incorrect answers, and a short explanation which may justify the answer, provide additional information, or explain why another answer is wrong. Trivia games are available in large quantities, after obtaining appropriate license agreements. However, they are only applicable to short-answer question answering, and the domains covered by trivia questions (e.g., entertainment, sports, etc.) may be quite different from the intended application domain.

Table 1. Distributions of initial unigrams for questions in five Q/A collections, with the five most common for each dataset in bold.

#Questions	TREC-8 200	TREC-9 693	CBC 651	Phishy 4,857	TriviaSpot 10,959
What	.299	.433	.235	.292	.334
Who	.234	.162	.147	.109	.202
How	.154	.075	.180	.047	.029
Where	.104	.101	.118	.018	.026
When	.095	.069	.101	.001	.006
In	.025	.004	.012	.097	.057
Which	.045	.004	.014	.314	.145
Why	.010	.003	.141	.002	.003
Total	.970	.876	.945	.881	.802
(other)	.030	.124	.125	.119	.198

These data sets can also potentially be useful for cross-development. For example, (Ittycheriah *et al.*, 2001) uses trivia questions to train a component of their system for the TREC evaluation. For this cross-development to be successful, there must be a considerable amount of commonality between the training and test sets. Table 1 looks at the distribution of initial unigrams (usually WH-heads) for questions in five of the data sets we've discussed.⁵³ The top five words for each set are shown in boldface. Table 1 shows that there are substantial differences between the data sets, even for something as superficial as the distribution of WH-heads in the questions. These data sets may still be useful, but clearly there are differences. In an ideal world, developers would have access to Q/A training sets that have the same properties and distributions as the Q/A sets on which the system will ultimately be used. In Section 3, we will present our research which will enable developers to create Q/A training sets from arbitrary text collections using a reverse-engineering algorithm. In Section 4, we show substantive differences for Q/A sets representing different domains.

⁵³ The Remedia data set is not shown because its questions have a fixed uniform distribution across ‘who’, ‘what’, ‘where’, ‘when’, and ‘why’.

2.2 Question Answering Architecture

Question answering systems are still in the early stages of development, so it is difficult to characterize any architecture as the exemplar or standard architecture. Nevertheless, there appears to be a growing consensus on the type of information used by question answering systems and on the common steps performed by question answering systems⁵⁴.

Broadly, most question answering systems break down their task in terms of three levels of increasing specificity of search. One example of a typical Q/A architecture is shown in Figure 3. The first step is document retrieval, where documents likely to contain an answer to the proposed question are collected. The next is sentence retrieval where the document or documents retrieved are searched to select the sentence which best answers the question. For some tasks this is sufficient. For example, early TREC evaluations accepted 250-byte answers, which in some cases may be a single sentence. However, the ultimate goal of question answering requires a higher level of specificity, where the exact short answer is retrieved. Therefore, many question answering systems have an additional step where the sentence is searched to extract the best word or phrase which answers the question. In this chapter, we analyze work on learning for the two final stages: sentence retrieval and short answer extraction.

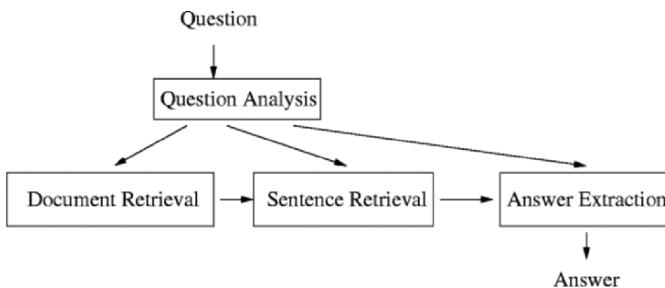


Figure 3. Basic Architecture of a Typical Question Answering System.

To perform sentence retrieval and short answer extraction, two types of information must be identified:

- *Answer Type*: the kind of answer the question expects
- *Answer Context*: context(s) in which the answer is likely to appear

There is a wide array of methods for answer type prediction; nearly every question answering system has a different set of answer types and methodology. In spite of this diversity, there are two major trends, the first towards detecting named entity types (e.g. predicting that the answer is a person), and the second towards

⁵⁴ One system with a remarkably different architecture is (Moldovan *et al.*, 2003) which uses a logic prover as an integral piece of the question answering process.

recognizing semantic classes using the WordNet ontology (e.g. predicting that the answer is a kind of color). These preferences can then be used in a variety of ways in sentence retrieval and short answer extraction. Typically, hand-coded constraints describe which answer types correspond with which questions, using hand-written question analyzers.

Answer context is also used in a wide variety of ways. The most common way to use context is to find sentences that share words or structure with the question, typically using word overlap to find the sentence which best matches the question. (Harabagiu *et al.*, 2000) presents an intriguing alternative where the parse structure of the question is used as a partially underspecified logical form which candidate answers attempt to prove in an abductive logic framework.

2.3 Statistical and Machine Learning Approaches to Question Answering

2.3.1 Answer Type Methods

Answer typing methods predict the type of expected answer based on features in the question. In most cases, these techniques predict the type of named entity that is most likely to answer the question, such as: person, location, city, country, organization, number, percent, date, time, duration, measure, money. Non-statistical Q/A systems use hand-coded rules and correspondences to predict, for example, that a ‘who’ question will probably be answered by people or organizations. Sometimes, a very precise type is able to be inferred (e.g. city), and sometimes only a more general type (e.g. location) is inferable.

Statistical techniques automatically learn the rules to match questions with their prospective answers. A simple technique, (Mann, 2001), uses a mutual information (MI) method to learn from an unannotated set of trivia questions and answers, by applying a named entity tagger to the answers and producing an answer type distribution. The question head word is assigned a distribution over named entities, according to the mutual information between the head word and the named entity tags marked by the tagger.

Table 2. Named Entity tags with highest Mutual Information with ‘Who’ questions.

NE Tag	MI(who ; NE)
Person	6.166
Location	2.186
Organization	1.333
Name	1.174
Country	0.583

Table 2 gives an example of the data this process learns. In the case of ‘who’ questions, this model would prefer people, but would also accept organizations and locations as answers, with empirically estimated preferences. This method, and the algorithms that will be discussed next, estimate similar values and hold two

advantages over non-statistical methods: 1) their preference ordering and strengths are estimated from data and 2) they can be easily adapted to new answer types and data sets.

A more sophisticated model, (Ittycheriah *et al.*, 2001; Ittycheriah, 2003), uses a set of trivia questions that have been annotated as to which named entity class the answer belongs. Using this data, a Maximum Entropy model is estimated from a variety of features taken from the question, as shown in Table 3. The features include an “Expanded Hierarchy” feature that ascends the WordNet hierarchy and uses all hypernyms of the word as features. The “Mark Question Word” feature identifies the question word and its place in the sentence (bqw = beginning of the sentence question word). This model estimates how good an answer type is given a question. One interesting aspect of the model is that bigrams and unigrams from the entire question are used.

*Table 3. Features used in a whole question Max-Ent model (Ittycheriah *et al.*, 2001).*

Unigrams	What year did World War II start?
Morphed, Part-Of-Speech	what{wp} year{nn} do{vbd} World{np} War{np}
	II{np} start{nn}
Bigrams	what{wp} what{wp} _year{nn} what{wp} _do{vbd} what{wp} _world{np} etc.
Expanded Hierarchy	What{wp} year time period measure Abstraction year{nn} do{vbd}
Mark Question Word	What_bqw year time period measure Abstraction year{nn} do{vbd}

Another system which uses annotated data is (Hermjakob, 2001) which trains a deterministic machine-learning based shift-reduce parser, Contex, to improve accuracy on detecting the type of the question (the ‘Qtarget’). The parser is trained on 3000 Wall Street Journal sentences, and a Question Treebank of around 1000 previously asked TREC questions, which have been annotated with the desired answer type. This work presents the possibility of using sophisticated linguistic analysis to determine the answer type using a complete parse as well as learning techniques to improve its performance.

The prior approaches train on questions with answers, in some cases annotated with their proper named entity type. In contrast, Prager *et al.* (2002; 2003) introduce a probabilistic co-occurrence model for estimating the likely answer type for a question from unrestricted text. They extract a multi-word term which explicitly names an answer type (e.g. **wingspan**) and tabulate which named entity type co-occurs most frequently with those words in the corpus. They use three different spans of context (phrasal, sentential, and document co-occurrence) to assess named entity match. These three spans are then combined to determine a final distribution (Table 4), where the type of the proposed answer is ranked. For example, **wingspan** is judged first to be a length, then a speed, and finally a weight.

Table 4. Question Term / Answer Type correspondences (Prager et al. 2002).

Question Term	Ranked 1 st		Ranked 2 nd		Ranked 3 rd	
	Class	S	Class	S	Class	S
Wingspan	LENGTH	0.06	SPEED	0.12	WEIGHT	0.06
Cancer	DISEASE	0.39	MEDICAL	0.26	CONSTELL	0.22
Culture	DURADATE	0.19	NATIONAL	0.16	THING	0.14
Season	SPORTSTEAM	0.34	TIMEOFTHEYEAR	0.19	WEIGHT	0.12
Bone	MUSICAL	0.38	DISEASE	0.28	MEDICAL	0.12
Plant	POWER	0.52	PROVINCE	0.08	ROLE	0.07

These previous methods have used words in the question as hooks to named entity classes. An alternative is to link words into a hierarchy, and when a question word is found, look for answers which are children of that word in the hierarchy. For example, if the question is judged to be looking for a color, take all the colors as possible answers.

Recently there has been work in building ontologies for these purposes. Girju, (2001) uses lexical rules proposed in (Hearst, 1992) to dynamically build an ontology for a domain. Mann (2002a) applies a syntactic pattern to induce a proper noun ontology for improved performance on answering questions about people. For example, it learns a large set of poets and actresses, and then can use this information when given a question starting with “What actress...”. Fleischman and Hovy (2003) extend this method and describes an algorithm for improving the precision of the extracted ontology. These methods create generic lexical resources that can be used not only for the specific task of question answering, but for other application problems as well.

2.3.2 Context-Based Methods

Knowing the type of answer expected by a question narrows down the possibilities, but other information must be used to find the precise answer. In particular, words in the question often indicate which instance of the answer type is the correct one.

Non-statistical approaches typically measure overlapping features between the question and the candidate answer sentence in order to identify the sentence that is most likely to contain the correct answer (Ittycheriah *et al.*, 2001). Harabagiu *et al.* (2000) present an alternative approach which uses an inference method to unify the question with an undefined variable in the candidate sentence.

There has been some work that takes this type of question/explanation word match approach, places it in a machine learning context, and attempts to improve the performance by optimization. Radev *et al.* (2000) approach the problem of picking the best span (a 50-byte sentence fragment). Their system combines a non-statistical answer type matcher with word-overlap features, and then uses a logistic regression optimization algorithm trained on 38 questions to combine the following features:

- Ranking of span by word overlap among all spans
- Ranking of span by word overlap among the spans in its passage
- Number of other spans in the current passage
- Number of words in the span that do not appear in the query
- Ranking of answer-type in span relative to preferred answer type
- Average distance between beginning of span and query words
- Passage relevance

A statistical estimator then uses these features to determine which span (50-byte sentence fragment) best answers the question.

Ng *et al.* (2000b; 2001) use a similar approach for the reading comprehension Q/A task. They collect a number of word overlap features and type-match features and apply a decision tree learning algorithm (C5) to estimate the confidence of each sentence being the correct answer. Wang *et al.* (2000) also use learning for sentence retrieval in reading comprehension exams. They use a variety of machine learning approaches on roughly the same feature set, including neural networks, a neuro-fuzzy classifier, a decision tree, and a genetic algorithm.

Ittycheriah (2003) describes a method for picking the correct sentence given a set of lexical sentence features, as well as an approach for using dependency structure to pick appropriate sentences. This follows work described in (Strzalkowski *et al.*, 1997; Harabagiu *et al.*, 2000).

Mann (2002b) pursues an approach which looks not only at word overlap, but at structural properties of the question and candidate sentence. In this model, depicted in Figure 4, the match between two phrases is related to the distance between anchors (matching words across phrases) and the types of intervening words. The features in the model are de : the distance between the candidate answer and the anchor, dq : the distance between the question word and the anchor in the question, and r : whether the answer and the question word are on reversed sides of the anchor. A statistical model is then built which combines type-preference information with these phrasal features. The statistical estimates are derived from a trivia game data set.

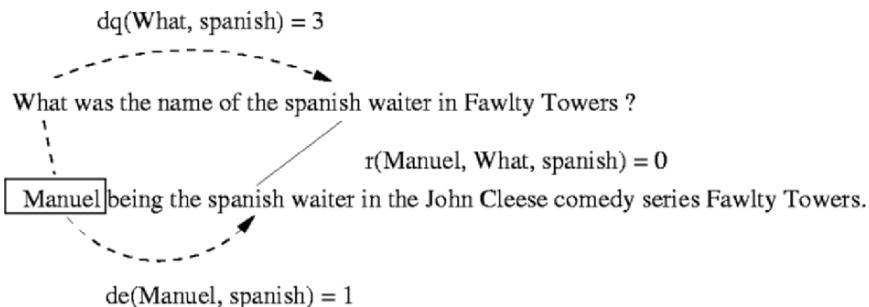


Figure 4. Example alignment in (Mann, 2002).

Another line of research attempts to learn lexical variation or phrasal synonymy to answer questions. Identifying phrasal synonymy is necessary to untangle phrases that entail the same relationships between constituents. For example, the phrase “Francis Scott Key wrote the ‘Star Spangled Banner’” describes the same relationship as “The ‘Star Spangled Banner’ was penned by Francis Scott Key”. Phrasal synonymy has been explored by Jacquemin *et al.* (1997), and Barzilay and McKeown (2001). Information extraction methods have also been developed to learn sets of phrasal patterns that can extract specific kinds of information or relationships (e.g., Riloff and Jones, 1999; Riloff, 1996; Yangarber *et al.*, 2000).

Lin and Pantel (2002) have addressed the problem of phrasal synonymy for question answering. In this work, the matrix phrase from each question is extracted and a large corpus is searched to find similar phrases. For example, for the question “Who is the author of ‘The Iron Lady’?”, the underlying relationship is “X is the author of Y”. They only examine phrases which have two slots (X and Y). Their system extracts paths from a dependency tree and uses a mutual information measure to compare paths by looking at the potential slot-filters in the path. Paths which have more slot-filters in common are more likely to be similar. Table 5 illustrates the synonyms learned for the “X writes Y” and “X manufactures Y” phrases.

Table 5. Inference Rules learned by Lin and Patel (2001).

Rank	X writes Y	X manufactures Y
1.	X co-authors Y	X produces Y
2.	X is co-author of Y	X markets Y
3.	X writes Y	X develops Y
4.	X edits Y	X is supplier of Y
5.	Y is co-authored by X	X ships Y
6.	Y is authored by X	X supplies Y
7.	X tells story in Y	Y is manufactured by X
8.	X translates Y	X is maker of Y
9.	X writes in Y	X introduces Y
10.	X notes in Y	X exports Y

Ravichandran and Hovy (2002) pursue a different strategy. Instead of using the phrase embedded in the question, (e.g., “X writes Y”), they take a given X_i and Y_i pair and use a corpus to find the patterns which are most likely to yield this pairing. In particular, they use a suffix tree to find the shortest phrases that contain both of the desired terms. They then calculate the precision of the phrase by searching a corpus for one of the terms and counting the number of times where X_i was paired with something other than Y_i . They do not use parse information, and consequently they can learn ungrammatical patterns (Table 6).

Table 6. Surface Text Patterns (Ravichandran and Hovy 2002).

<i>confidence</i>	BIRTHYEAR (e.g. X born on Y)	<i>confidence</i>	DISCOVERER (e.g. X discovered Y)
1.0	X (Y -)	1.0	When X discovered Y
.85	X was born on Y,	1.0	X's discovery of Y
.60	X was born in Y	1.0	X, the discoverer of Y
.59	X was born Y	1.0	X discovers Y
.53	Y X was born	1.0	X discoverer Y
.50	- X (Y	1.0	X discovered Y, the
.36	X (Y -	.95	Y was discovered by X
.32	X (Y)	.91	of X's Y
.28	born in Y, X	.90	Y was discovered by X in

2.4 Closely Related Work

Fujii and Ishikawa (2001) study a related problem, which picks out term descriptions from a multiple choice question. Their approach builds a form of encyclopedic knowledge from the web and uses it to support or reject possible multiple choice answers. Berger *et al.* (2000) studies another related problem: question answering scenarios where multi-paragraph retrieval is the desired output. Their system would match a question to the most appropriate FAQ-answer (sometimes many paragraphs long). In their model, a machine translation approach is used, and the text most likely to have been generated by the question is chosen as the answer.

2.5 Summary

In many of these approaches, the work is not fully automated, and needs hand-analysis before it can be used in a question answering engine. In others, while the analysis can be done for unannotated text, it is not clear that it can perform as well as supervised systems. In both cases, the goal of training a complete question answering system from raw text is unfulfilled. An automatic way of producing Q/A data sets could be beneficial for many of these learning systems.

3. REVERSE ENGINEERING QUESTION/ANSWER COLLECTIONS FROM ORDINARY TEXT

As we have just seen, one of the most active and promising directions for Q/A research is the application of statistical and machine learning techniques. These methods, however, require large amounts of training data. A few Q/A systems have begun to incorporate statistical methods by taking advantage of existing Q/A collections, such as those described in Section 2. These data sets are a good starting

point, but most of them are relatively small in size. For statistical techniques, the general rule is: the more, the better. It would be desirable to have thousands, if not hundreds of thousands, of questions and answers for training. An additional complication is that some Q/A collections, such as the trivia game data sets, are not freely available (e.g., license agreements are necessary and/or they may be expensive to purchase), which hampers the ability of researchers to experiment with them.

As an alternative to manually generated Q/A collections, we have developed a *reverse-engineering algorithm* that can automatically generate questions and answers from ordinary text. Our goal is to use this reverse-engineering process to automatically generate a large collection of question/answer pairs that can be used as training data for Q/A research. Since this technique requires raw text as input, the amount of training data that can be generated is limited only by the availability of ordinary text corpora. Another benefit of this approach is that the reverse-engineering process can be applied to domain-specific text collections to generate specialized Q/A collections that could be used to build domain-specific question answering systems.

In the following sections, we give an overview of the reverse engineering process, describe the three main steps of the reverse engineering algorithm, give examples of questions and answers produced by this technique, and discuss the strengths and weaknesses of this approach.

3.1 Overview of the Reverse Engineering Process

The goal of our research is to reverse engineer questions and answers from ordinary sentences. As a simple example, consider the following sentence:

S: *Hillary Clinton lives in New York.*

If we recognize that New York is a location, then we can create a *Where* question from this sentence and identify its answer:

Q: *Where does Hillary Clinton live?*

A: *New York*

This example illustrates the general idea behind our approach: take ordinary sentences and create questions and answers from them. Figure 5 shows the three main steps involved in the reverse engineering process, which begins with an unannotated text corpus (e.g., a collection of news articles). First, heuristic rules identify *candidate sentences* that contain a relationship associated with a general question class. Second, lexico-syntactic rules extract the portions of each candidate sentence that correspond to the question and the answer. Third, linguistic rules transform the question portion of the sentence into an actual question. In the next three sections, we describe each of these steps in more detail.

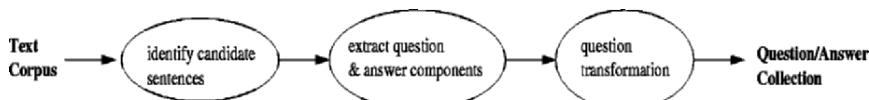


Figure 5. The Reverse Engineering Process.

3.2 Step 1: Identifying Candidate Sentences

The first part of the reverse engineering process is to recognize which sentences contain information that can be re-expressed as a question and answer. Initially, we decided to focus on five general question classes: *Who*, *Where*, *When*, *What*, and *Why* questions. We have found several types of heuristics that can identify sentences containing information associated with these question classes. Below, we describe each type of heuristic and give examples of how they apply to real sentences.

WH-words: The presence of a WH-word often reveals a relationship associated with the corresponding question class. For example:

At First Christian Church, where relatives of the missing awaited news of their loved ones, ...

Q: Where did relatives of the missing await news of their loved ones?

A: First Christian Church

Wei, who heads the Communist Party's anti-corruption organ, is a rising star in China.

Q: Who heads the Communist Party's anti-corruption organ?

A: Wei

Named Entities: Sentences containing named entities can be used to reverse engineer *Who*, *When*, and *Where* questions. For example, in the sentence below the date *July 1987* can yield a *When* question, and the location *Mecca* can yield a *Where* question.

In July 1987, nearly 400 pilgrims were killed in Mecca in Iranian-inspired riots.

Q: When were nearly 400 pilgrims killed in Mecca in Iranian-inspired riots?

A: July 1987

Q: Where were nearly 400 pilgrims killed?

A: Mecca

Semantic Class Members: Sentences that contain words belonging to certain semantic classes can be used to create questions. This is similar to the Named Entity rule above, except that this rule looks for common (non-proper) nouns that are

defined with a semantic label in a dictionary (e.g., WordNet (Miller, 1990)) or a semantic lexicon created automatically (e.g., Phillips and Riloff, 2002; Thelen and Riloff, 2002; Caraballo, 1999; Riloff and Jones, 1999; Riloff and Shepherd, 1999; Roark and Charniak, 1998; Hearst, 1992). For example, in the sentences below the word *intellectuals* refers to a person so it can yield a *Who* question, and the word *airport* refers to a location so it can yield a *Where* question.

Intellectuals organized a labor union in the industrial city of Shanghai in the 1920s.

Q: Who organized a labor union in the industrial city of Shanghai in the 1920s?

A: Intellectuals

However, Lopez' relatives rejected that his murder had any connection with drug trafficking discovered at the airport.

Q: Where was drug trafficking discovered?

A: the airport

Lexical Phrases: Many question classes are strongly correlated with easily identifiable keywords or phrases. For example, the phrases “located in” and “found in” can be used to construct *Where* questions. Similarly, the words “because” and “the reason” can be used to construct *Why* questions. For example:

The Great Wall and Dragon Khan are located in China.

Q: Where are the Great Wall and Dragon Khan?

A: China

The FDIC said it decided to pay off depositors because no other institution wanted to buy the failed bank.

Q: Why did The FDIC say it decided to pay off depositors?

A: because no other institution wanted to buy the failed bank.

Common noun phrases: In some syntactic positions, inanimate noun phrases⁵⁵ can be used to construct *What* questions. In particular, inanimate noun phrases that function as the subject or object of a verb can be used in this way. For example:

Thai helicopter gunships deterred renegade Karen fighters from crossing into Thailand Monday despite the rebels' vow to continue raiding refugee camps.

Q: What deterred renegade Karen fighters from crossing into Thailand Monday?

A: Thai helicopter gunships

⁵⁵ That is, noun phrases that are not possible answers for *Who* questions.

Q: What did the rebels vow to continue raiding?

A: refugee camps

These types of heuristics can identify sentences that are likely to be transformable into a question and answer. But not every sentence identified by these heuristics will produce a coherent question and answer. The next step of the reverse engineering process pushes the candidate sentences through a second set of lexico-syntactic rules, which filter out inappropriate sentences and extract the question and answer components from the good sentences.

3.3 Step 2: Extracting questions and answers

Most of the candidate sentences identified in Step 1 can be successfully transformed into a question and answer. But some of them cannot, due to lexicalized phrases, idiosyncratic expressions, or unusual syntactic constructions. Step 2 of the reverse engineering process applies a set of lexico-syntactic Q/A rules to the candidate sentences. These rules serve two purposes: (1) they check that the sentence is amenable to successful question/answer construction (sentences that are not transformable are discarded), and (2) they extract the portions of the sentence that correspond to the question and the answer.

Each candidate sentence first passes through a set of *filtering rules*, which discard sentences that do not seem to contain viable questions and answers. The filtering rules look for generic expressions, phrases with specialized meanings, and syntactic constructions for which the question or answer is missing. As an example, here are some of the filtering rules for candidate *Where* sentences. A candidate sentence is not viable if:

- The sentence begins with the word “Where” or is a question already.
- The word preceding the word “where” is a verb or preposition (in this case, the sentence usually does not contain a location).
- The word “where” appears in an expression such as “except where” or “point where”.
- The location has an indefinite article. This eliminates generic expressions, such as “in a place where ...”.
- The *Where* expression is negated (e.g., “not where”, “not found in”, etc.).

Sentences that successfully pass the filtering rules are then pushed through a set of *Q/A extraction rules*, which extract the question and answer portions of the sentence. The scope of the question and answer are not always well defined (i.e., different human judges might extract different portions of the sentence). For example, the sentence below will produce a *Where* question that has four arguably good answers.

S: *Outside Junction City, Kansas, where the Ryder truck that carried the bomb was rented, ...*

Q: Where was the Ryder truck that carried the bomb rented?

A: “Outside Junction City”, “Outside Junction City, Kansas”, “Junction City”, or “Kansas”?

For our purposes there does not need to be a single correct answer. Our goal is to extract a question and answer pair that makes sense and is factually consistent with the original sentence. In the previous example, any of the four possible answers would produce a reasonable question/answer pair.

Every sentence is first parsed by Sundance, a shallow parser developed at the University of Utah, which identifies syntactic constituents and separates the sentences into clauses. Most of the extraction rules then identify a noun phrase as the answer and extract an entire clause as the basis for the question, although some rules extract longer questions and answers. To illustrate, some of the Q/A extraction rules for *Where* questions are listed below. The rules are ordered, so that only the first matching rule is applied.

- If the word “where” appears in a relative clause construction, the NP preceding “where” becomes the answer and the clause following “where” becomes the question (e.g., “The U.S., where wages have hovered...”).
- If a known location (e.g., Named Entity) is in a prepositional phrase (PP), then the location becomes the answer and the portion of the clause preceding the PP becomes the question (e.g., “John flew to Mexico”).
- If the sentence contains a location phrase such as “found in NP”, then the NP becomes the answer and the verb phrase and its subject become the question (e.g., “The weapons were found in Lima”).

3.4 Step 3: Creating questions

The final step of the reverse engineering process is to transform the question portion of a sentence into an actual question. This process is usually a straightforward syntactic transformation, based on the question class. For example, consider the sentence below, which could be used to generate three questions:

The attackers robbed the National Bank on November 7.

Q1: Who robbed the National Bank?

A1: the attackers

Q2: What was robbed by the attackers?

A2: the National Bank

Q3: When did the attackers rob the National Bank?

A3: November 7

For some cases, such as Q1, the original sentence structure remains the same except that the answer is replaced by the question word. If the answer was a direct

object or prepositional phrase, as in Q2, then the sentence must be syntactically rearranged to begin with the *Wh-* word. In many *When* and *Where* questions, such as Q3, an auxiliary verb (e.g., “did”) must be added and the verb tense modified accordingly. These transformations are fairly simple and mechanical, producing syntactically valid questions in most cases.

3.5 Summary

In this section we presented a reverse-engineering procedure that can be used to construct question and answer pairs from ordinary sentences. One possible concern is that our Q/A collection will be strongly biased by the heuristics that we use. We will try to minimize this bias by using a wide variety of heuristics that can be instantiated in many different ways. The five types of heuristics that we presented here can provide a good foundation for a relatively diverse Q/A collection, but we believe that additional types of heuristics will also be discovered as this research progresses. Furthermore, since the reverse-engineering process only needs raw text as input, we should be able to generate an enormous Q/A collection, which will help produce a data set with broad coverage and good diversity. In the next section, we show examples of questions and answers produced by this process from four different corpora, and we present data analyses showing that Q/A collections generated for different domains can exhibit very different properties.

4. EMPIRICAL INVESTIGATIONS

4.1 Data Sets and Examples

To investigate the feasibility of the reverse engineering process, we generated questions and answers from four text corpora: the Penn Treebank Wall Street Journal corpus (Marcus *et al.*, 1993), Reuters texts (Reuters Ltd., 1997), the MUC-4 terrorism corpus (MUC-4 Proceedings, 1992), and a collection of press releases from pharmaceutical companies. We gathered the pharmaceutical texts from the medical/pharmaceutical section of Yahoo’s financial industry news pages.⁵⁶

So far, we have designed and implemented rules for *Who*, *Where*, and *Why* questions using three types of heuristics: *WH-words*, *Named Entities*, and *Lexical Phrases*. We are still in the process of refining our rule set, and there are more rules that we plan to add. But even with the relatively small rule set that we’ve implemented so far, we have been able to generate substantial sets of questions and answers.

Table 7 shows the size of the four corpora and the number of questions and answers generated for each question type. The number of questions and answers

⁵⁶ Located at <http://biz.yahoo.com/news/medical.html>. The articles were dated March, April, and June 2002.

varies considerably because the corpora vary in size and because the rules apply to some texts more frequently than others.

Table 7. Statistics about our data sets.

Corpus	#Texts	#Who Questions	#Where Questions	#Why Questions
<i>WSJ</i>	2500	2550	2607	623
<i>Reuters</i>	6325	734	1629	367
<i>Pharmaceutical</i>	2523	2342	2020	236
<i>Terrorism</i>	1700	1067	3695	561

Figure 6 contains examples of questions and answers generated from each corpus. The nature of the questions and answers varies greatly across the corpora, illustrating the strong domain differences in both the nature of the language and the focus of the topics. Some of the questions sound a bit awkward because of the reverse-engineering process, but most of them represent legitimate questions and answers.

Figure 6 shows questions and answers that make sense outside the context of the original document, but many question/answer pairs that we generate are not fully specified outside of their original context. Figure 7 shows two examples that illustrate the most common problem that we encounter: relative references. Pronouns and other anaphora (e.g., “the continent”, “tonight” and “he”) are extremely common and produce underspecified questions (and sometimes answers) when extracted from the original documents.

Fortunately, this issue is not as problematic as it might seem because our goal is to generate training data, not a factual database. There is nothing incorrect about the questions or answers that we generate, they are simply underspecified, so we believe that they can still be valuable for learning. For example, from the Q/A pairs in Figure 7 the system could learn that heads of state have dined in San Jose and that the Banco Bilbao Vizcaya has a director general. Since our goal is to generate an extremely large Q/A collection, statistical techniques should be able to find useful associations even if some of the data is not fully specified.

Ideally, references should be resolved before the reverse engineering process begins. But coreference resolution is far from a solved problem, and incorrect resolutions could make the data less reliable. As a middle ground, we hope to incorporate a low-recall, high-precision anaphora resolution algorithm to correctly resolve the “easy” cases, leaving the more uncertain cases unresolved.

Penn Treebank Wall Street Journal corpus

- Q: Who follows bank stocks?
 A: money manager
 Q: Where is Dick Lobo, the general manager of WTVJ, the NBC-owned station?
 A: Miami
 Q: Why did net sales of bond funds for the month total \$1.1 billion, down two-thirds from \$3.1 billion in August?
 A: The major reason: heavy outflows from high-risk, high-yield junk bond funds.

Reuters corpus

- Q: Who was selling dollars and buying yen before the end of the Japanese fiscal year on March 31 and undermining the U.S. currency?
 A: some Japanese companies
 Q: Where does oil account for up to two-thirds of total exports and as much as 60 pct of government revenues?
 A: Ecuador
 Q: Why did Thermo Process Systems Inc say its proposed acquisition of the Surface Combustion Division of privately-held Midland-Ross Corp. has been terminated?
 A: because mutually satisfactory terms could not be established.

Pharmaceutical corpus

- Q: Who has experienced difficulties with the other drugs for insomnia currently on the market?
 A: the elderly and female patients
 Q: Where does it helps break down large biomolecules called glycosaminoglycans?
 A: in the lysosomes of cells
 Q: Why has Anti-angiogenic therapy recently been a hot area in cancer research?
 A: because it appears to bypass the major problem with chemotherapy – the tumor cells' ability to mutate and develop resistance to the drugs.

MUC-4 terrorism corpus

- Q: Who is leader of the Salvadoran Leftist Party National Revolutionary Movement?
 A: Hector Oquelí
 Q: Where is the Miraflores neighborhood?
 A: Lima
 Q: Why is Reynoso known in Lima as the escape king?
 A: because he has escaped from Peruvian jails a number of times disguised as a woman or a police officer

Figure 6. Examples of reverse-engineered questions and answers.

- Q: Where were the 16 heads of state of the continent scheduled to dine tonight?
 A: downtown San Jose
 Q: Where was he formerly director general?
 A: Banco Bilbao Vizcaya

Figure 7. Questions with contextual problems.

4.2 Domain-specific Q/A Analysis

One of the main benefits of being able to generate question/answer collections automatically is that we can generate Q/A collections for specialized domains. These question/answer pairs could then be used to train and build domain-specific Q/A systems. One could argue that a general-purpose question-answering system should be able to answer questions about any topic. While that may be true to some extent, many Q/A applications involve domains that have specialized language needs, including vocabulary, phrases, and named entities. For example, medical applications use medical terminology and jargon that is not usually present in general-purpose dictionaries and resources. Even a domain like terrorism, which may not seem specialized or esoteric at first glance, depends on the recognition of atypical vocabulary. Recognizing the names of terrorist organizations is essential. However, terrorist organization names are not likely to be present in a general-purpose dictionary and they are not likely to be recognized by general-purpose named entity recognizers because they often use acronyms (e.g., FMLN) or have unusual names (e.g., “The Extraditables”).⁵⁷

The corpora that we use in our experiments represent three distinct domains: U.S. business-oriented news (WSJ and Reuters), U.S. pharmaceutical industry news (pharmaceutical press releases), and Latin American terrorism (MUC-4 corpus). The questions and answers produced from these corpora reveal strong domain-specific differences. For example, Q/A systems typically assume that the answer to a *Where* question will be a geographic location. While this is often true, within a specific domain there may be other strong possibilities as well. For example, *Where* questions produced from the WSJ and Reuters corpora often have answers that are financial entities rather than geographic locations, such as:

Q: Where would they continue making markets in the Pacific-listed options?

A: the CBOE⁵⁸

Q: Where were traders still betting that the earthquake disrupted the San Francisco area’s large oil refining plants?

A: European markets

In the pharmaceutical domain, *Where* questions are often answered by body parts or biological entities. For example:

Q: Where does it helps to break down large biomolecules called glycosaminoglycans?

A: in the lysosomes of cells

⁵⁷ They could be recognized as proper names, but they would not be distinguished from other proper names, such as government agencies or sports teams.

⁵⁸ The CBOE is an options exchange.

Q: Where are plasma TG frequently found?

A: in a triad with low HDL-C and small LDL particles

These examples suggest that strong domain biases do exist. To investigate this hypothesis further, we looked more closely at the *Where* questions and answers generated from our corpora. We gathered all of the *Where* questions and answers produced by the WH-word heuristic (i.e., from sentences that contain the word *where*) because there is no reason to believe that this heuristic would be biased toward answers of any particular type.⁵⁹ We manually labeled the answer to each question according to two criteria: (1) whether it is a proper name (PN), such as “Japan”, or a general noun phrase (GN), such as “the beach”, and (2) the semantic category of the noun phrase. Table 8 shows the semantic categories that we used, along with sample answers matching each type.

Table 8. Semantic Classes Used in Manual Answer Tagging.

Tag	Semantic Class	Sample Answers
Loc	Geographic location	<i>Asia, Detroit, this country, the zone</i>
Org	Organization	<i>House Banking Committee, the United Nations</i>
Fin	Financial entity	<i>European markets, CBOE, its music business</i>
Body	Body part	<i>fetal skin, the inner ear</i>
Bldg	Building	<i>the laundromat, the hotel</i>
Person	People	<i>its patients, the forgotten people of Africa</i>
Date	Date/time	<i>January, 1972</i>
Number	Numeric value	<i>.28</i>
Other	Anything else	<i>Hymowitz v. Lilly, his desk, my MBA class, cultures</i>

Table 9 shows the breakdown of answers associated with these *Where* questions. For example, PN-loc refers to proper names that are locations, such as “Japan”, and GN-loc refers to general noun phrases that are locations, such as “the zone”. Proper name locations make up the majority of answers in the business and terrorism domains (62% and 72%, respectively), but surprisingly only 22% of the answers in the pharmaceutical domain. In fact, the most common semantic class for *Where* questions in the pharmaceutical collection is proper name organization. An example question with a PN-organization as its answer is shown below:

Q: Where has he been a member of the faculty since 1975?

A: The Salk Institute

⁵⁹ For example, gathering sentences generated by the Named Entity heuristic would obviously bias the answers to be Named Entities.

Overall, 72% of the pharmaceutical *Where* questions were answered by entities that are not locations, as well as 37% in the business domain and 13% in the terrorism domain. Assuming that most answers to *Where* questions are geographic locations is clearly not a sufficient strategy. In the terrorism domain, geographic locations are clearly dominant, with a few other semantic classes showing up occasionally. In the business domain, geographic locations are the most frequent class, but financial entities also seem to be common answers. In the pharmaceutical domain, organizations are the most frequent answer type, along with a wide variety of other types, including body parts, which were never the answer to *Where* questions in the other domains.

Table 9. Semantic classes of Where answers.

BUSINESS		PHARM		TERRORISM	
<i>Class</i>	<i>freq</i>	<i>class</i>	<i>freq</i>	<i>Class</i>	<i>freq</i>
PN-loc	.62	PN-org	.22	PN-loc	.72
GN-fin	.11	PN-loc	.19	GN-loc	.15
GN-other	.07	GN-other	.16	PN-org	.09
PN-org	.07	GN-body	.13	PN-bldg	.01
PN-other	.04	GN-loc	.09	GN-org	.01
GN-bldg	.04	GN-fin	.06	GN-bldg	.01
PN-fin	.02	GN-person	.06	GN-other	.01
PN-date	.01	GN-bldg	.03		
GN-number	.01	PN-other	.03		
		GN-date	.03		
Locs	.62	Locs	.28	Locs	.87
Non-Locs	.37	Non-Locs	.72	Locs	.13
PNs	.77	PNs	.44	PNs	.82
GNs	.23	GNs	.56	GNs	.18

Table 9 also shows the percentage of answers that were proper names vs. general noun phrases. In the business and terrorism domains, 82% of the answers were proper names, but in the pharmaceutical domain only 44% were proper names. These numbers reveal that Q/A systems need both named entity recognition for proper names, as well as semantic class labels for common noun phrases. WordNet (Miller, 1990) is the most widely used resource for semantic knowledge, but a general resource such as WordNet would probably not be sufficient for domains with specialized vocabulary, such as medical domains. For specialized domains, it is probably advantageous to use a domain-specific semantic dictionary (such as a medical thesaurus) if one is available. Alternatively, several techniques have been developed in recent years to learn domain-specific semantic lexicons (e.g., Thelen and Riloff, 2002; Phillips and Riloff, 2002; Caraballo, 1999; Roark and Charniak, 1998).

5. SUMMARY

Automatically generating large question/answer collections can provide a new stimulus for research in statistical and machine learning methods for question answering. When building a Q/A system for a specific domain, it is desirable to train the system on data sets that are representative of that domain. Q/A sets generated from a domain-specific corpus will reveal distributions, vocabulary, and patterns that are strongly associated with that domain. Domain-specific Q/A collections could be used as stand-alone training data, or they could be used to augment more general Q/A collections, combining the best of both worlds.

We can imagine many ways in which this data could be useful. As we showed in Section 4, different domains yield different expectations and probabilities about answer types. We should also be able to measure statistical correlations between words in the question and the answer. For example, the Q/A pair below might allow a system to learn statistical associations between “flight attendants”, “terminals”, and “airport”, which could be useful for identifying promising answer contexts or ranking candidate answers.

Q: Who had been handing out information brochures at American’s terminals at the Dallas-Fort Worth airport?

A: 17 flight attendants

To date, we have implemented several rules and heuristics to generate question/answer collections for three major question classes: *Who*, *Where*, and *Why* questions. We plan to implement additional rules for these question types, and add rules for *What* and *When* questions in the future. We will continue to extend and diversify the set of heuristics that we use, so that our reverse engineered Q/A collection will contain a healthy mix of different kinds of questions and answers. We also plan to experiment with additional text corpora, to identify differences across domains and text sources.

6. ACKNOWLEDGEMENTS

This research was supported by the National Science Foundation under award IRI-9704240.

7. REFERENCES

- Barzilay, Regina and McKeown, Kathleen R. (2001). Extracting paraphrases from a parallel corpus. In *Proceedings of ACL/EACL*, Toulouse, France.
- Berger, A., Caruana, R., Cohn, D., Freitag, D., and Mittal, V. (2000). Bridging the lexical chasm: Statistical approaches to answer-finding. *Proceedings of the 23rd Annual Conference on Research and Development in Information Retrieval (ACM SIGIR)*, pages 192–199.
- Brill, E., Dumais, S., and Banko, M. (2002). An analysis of the askmsr question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing.*, pages 257–264.

- Caraballo, Sharon (1999). Automatic acquisition of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.
- Charniak, E. (1993). *Statistical Language Learning*. The MIT Press, Cambridge, MA.
- Charniak, E., Altun, Y., de Salvo Braz, R., Garrett, B., Kosmala, M., Moscovich, T., Pang, L., Pyo, C., Sun, Y., Wy, W., Yang, Z., Zeller, S., and Zorn, L. (2000). Reading Comprehension Programs in a Statistical-Language-Processing Class. In *ANLP/NAACL Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*.
- Church, K. (1989). A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing*.
- Fleischman, M. and Hovy, E. (2003). Offline strategies for online question answering: Answering questions before they are asked. In *The Annual Meeting of the Association for Computational Linguistics*, page (to appear).
- Fujii, Atsushi and Ishikawa, Tetsuya (2001). Question answering using encyclopedic knowledge from the web. In *Workshop on Open-Domain Question Answering at ACL*.
- Girju, Roxana (2001). Answer fusion with on-line ontology development. In *Student Research Workshop Proceedings at The 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*.
- Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., nu, M., Surdea, Bunescu, R., Girju, R., Rus, V., and Mor, P. (2000). Falcon: Boosting knowledge for answer engines. *Proc. of TREC-9*.
- Hearst, Marti (1992). Automatic acquisition of hyponyms from large text corpora. *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*.
- Hermjakob, Ulf (2001). Parsing and question classification for question answering. In *Workshop on Open-Domain Question Answering at ACL*.
- Hirschman, L., Light, M., Breck, E., and Burger, J. (1999). Deep Read: A Reading Comprehension System. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.
- Ittycheriah, A. (2003). A statistical approach for open domain question answering. In Harabagiu, S. and Strzalkowski, T., editors, *Advances in Open Domain Question Answering*. Kluwer.
- Ittycheriah, A., Franz, M., Zhu, W-J., and Ratnaparkhi, A. (2001). Question Answering Using Maximum Entropy Components. *Proceedings of the Second Meeting of The North American Chapter of the Association of Computational Linguistics*, pages 33–39.
- Jacquemin, Christian, Klavens, Judith, and Tzoukermann, Evelyne (1997). Expansion of multi-word terms for indexing and retrieval using morphology and syntax. In *Proceedings of ACL/EACL*, Barcelona, Spain.
- Light, Marc, Mann, Gideon S., Riloff, Ellen, and Breck, Eric (2001). Analyses for elucidating current question answering technology. *Journal of Natural Language Engineering*.
- Lin, Dekang and Pantel, Patrick (2002). Discovery of inference rules for question/answering. *Journal for Natural Language Engineering*.
- MacDonald, G. (1999). Phishy web trivia.
- Mann, Gideon S. (2001). A statistical method for short answer extraction. In *Workshop on Open-Domain Question Answering*, pages 23–30.
- Mann, Gideon S. (2002a). Building a proper noun ontology for question answering. In *Proceedings of SemaNet02: Building and Using Semantic Networks*, Taipei, Taiwan.
- Mann, Gideon S. (2002b). Learning how to answer questions using trivia games. In *Proceedings of the Nineteenth International Conference on Computational Linguistics (COLING 2002)*.
- Marcus, M., Santorini, B., and Marcinkiewicz, M. (1993). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2): 313–330.
- Miller, G. (1990). Wordnet: An On-line Lexical Database. *International Journal of Lexicography*, 3(4): 235–312.
- Moldovan, D., Clark, C., Harabagiu, S., and Maiorano, S. (2003). Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL 2003*, pages 166–172.
- MUC-4 Proceedings (1992). *Proceedings of the Fourth Message Understanding Conference (MUC-4)*. Morgan Kaufmann, San Mateo, CA.
- Ng, H.T., Teo, L.H., and Kwan, J.L.P. (2000a). A Machine Learning Approach to Answering Questions for Reading Comprehension Tests. In *Proceedings of EMNLP/VLC-2000 at ACL-2000*.
- Ng, Hwee Tou, Kwan, Jennifer Lai Pheng, and Xia, Yiyuan (2001). Question answering using a larger text database: A machine learning approach. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

- Ng, Hwee Tou, Teo, Leong Hwee, and Kwan, Jennifer Lai Pheng (2000b). A machine learning approach to answering questions for reading comprehension tests. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 124–132.
- Phillips, W. and Riloff, E. (2002). Exploiting Strong Syntactic Heuristics and Co-Training to Learn Semantic Lexicons. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.
- Prager, J. M., Chu-Carroll, J., Brown, E. W., and Czuba, K. (2003). Question Answering by Predictive Annotation. In Harabagiu, S. and Strzalkowski, T., editors, *Advances in Open Domain Question Answering*. Kluwer.
- Prager, John, Chu-Carroll, Jennifer, and Czuba, Krzysztof (2002). Statistical answer-type identification in open-domain question answering. In *Human Language Technologies Conference*.
- Radev, Dragomir R., Prager, John, and Samn, Valeria (2000). Ranking suspected answers to natural language questions using predictive annotation. In *Proceedings of the Sixth Applied Natural Language Processing Conference*, pages 150–157.
- Ravichandran, Deepak and Hovy, Eduard (2002). Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*.
- Reuters Ltd. (1997). Reuters-21578, Distribution 1.0. <http://www.research.att.com/~lewis>.
- Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049. The AAAI Press/MIT Press.
- Riloff, E. and Jones, R. (1999). Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.
- Riloff, E. and Shepherd, J. (1999). A Corpus-based Bootstrapping Algorithm for Semi-Automated Semantic Lexicon Construction. *Journal for Natural Language Engineering*, 5(2):147–156.
- Riloff, E. and Thelen, M. (2000). A Rule-based Question Answering System for Reading Comprehension Tests. In *ANLP/NAACL Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*.
- Roark, B. and Charniak, E. (1998). Noun-phrase Co-occurrence Statistics for Semi-automatic Semantic Lexicon Construction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics*, pages 1110–1116.
- Strzalkowski, T., Lin, F., Perez-Caraballo, J., and Wang, J. (1997). Building effective queries in natural language information retrieval. In *ANLP*, pages 299–306.
- Thelen, M. and Riloff, E. (2002). A Bootstrapping Method for Learning Semantic Lexicons Using Extraction Pattern Contexts. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.
- TREC-10 Proceedings (2001). *Proceedings of the Tenth Text Retrieval Conference*. National Institute of Standards and Technology, Special Publication 500-250, Gaithersburg, MD.
- TREC-11 Proceedings (2002). *Proceedings of the Eleventh Text Retrieval Conference*. National Institute of Standards and Technology, Special Publication 500-251, Gaithersburg, MD.
- TREC-8 Proceedings (1999). *Proceedings of the Eighth Text Retrieval Conference*. National Institute of Standards and Technology, Special Publication 500-246, Gaithersburg, MD.
- TREC-9 Proceedings (2000). *Proceedings of the Ninth Text Retrieval Conference*. National Institute of Standards and Technology, Special Publication 500-249, Gaithersburg, MD.
- TriviaMachine Inc. (1999). TriviaSpot.com. www.triviaspot.com.
- Turtle, Howard and Croft, W. Bruce (1991). Efficient Probabilistic Inference for Text Retrieval. In *Proceedings of RIAO 91*, pages 644–661.
- Wang, W., J., Auer, Parasuraman, R., Zubarev, I., Brandyberry, D., and Harper, M.P. (2000). A Question Answering System Developed as a Project in a Natural Language Processing Course. In *ANLP/NAACL Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*.
- Weischedel, R., Meteer, M., Schwartz, R., Ramshaw, L., and Palmucci, J. (1993). Coping with Ambiguity and Unknown Words through Probabilistic Models. *Computational Linguistics*, 19(2):359–382.
- Yangarber, R., Grishman, R., Tapanainen, P., and Huttunen, S. (2000). Automatic Acquisiton of Domain Knowledge for Information Extraction. In *Proceedings of the Eighteenth International Conference on Computational Linguistics (COLING 2000)*.

- Yarowsky, D. (1992). Word sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*, pages 454–460.
- Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*.

MARK MAYBURY

NEW DIRECTIONS IN QUESTION ANSWERING

Abstract: Next generation question answering systems are challenged on many fronts including but not limited to massive, heterogeneous and sometimes streaming collections, diverse and challenging users, and the need to be sensitive to context, ambiguity, and even deception. This chapter describes new directions in question answering (QA) including enhanced question processing, source selection, document retrieval, answer determination, and answer presentation generation. We consider important directions such as answering questions in context (e.g., previous queries, day or time, the data, the task, location of the interactive device), scenario based QA, event and temporal QA, spatial QA, opinionoid QA, multimodal QA, multilingual QA, user centered and collaborative QA, explanation, interactive QA, QA reuse, and novel architectures for QA. The chapter concludes by outlining a roadmap of the future of question answering, articulating necessary resources for, impediments to, and planned or possible future capabilities.

1. INTRODUCTION

Although Albert Einstein stated “I never think of the future. It comes soon enough.”, John F. Kennedy noted that “Change is the law of life. And those who look only to the past or present are certain to miss the future.” Question answering has engaged human interest for centuries, including the Greek contribution of Socratic questioning in which deep, systematic, and comprehensive questioning seeks to discover the truth or plausibility of things. Some of the first computational endeavors of early artificial intelligence scientists was a computer program that could answer questions.

We might expect future question answering systems will be able to provide capabilities such as being able to answer a broader and deeper set of questions, extract answers from more varied sources, provide more richly justified answers, support a broader range of users of differing backgrounds, ages, levels of expertise, and interest. The nature of human question answering and information trends suggests the future will include advances to challenges such as:

- the scale of dealing with large collections such as the web
- real-time sources and continuous changes in knowledge, such as rapidly forming knowledge in biology, medicine, or technology
- the heterogeneity of questions and sources (e.g., multilingual, multimedia, multisource)
- dealing with sources of varying pedigree and ambiguity (e.g., with implicit intensions)

- dealing with questions and answers and sources that may be imprecise, indirect, or outright deceitful
- dealing with young and old users, novice and experts,
- answering questions sensitive to linguistic, task, domain, and social context

Articulating the future of question answering is more challenging than many fields because of the interdisciplinary nature of the field. It lies at the intersection of several scientific fields including natural language processing (understanding and generating natural language text), information retrieval (query formulation, document analysis, relevancy feedback), and human computer interaction (interface design, user modeling). Figure 1 illustrates the relation of the three areas and their intersection in systems that support question answering. Several additional scientific disciplines may support question answering are not shown, such as knowledge representation and reasoning for question and answer analysis, or recommender technology to find preferred answers, or multimodal information processing to help extract answers from audio or video sources, or information visualization for results display. Thus forecasting the future of QA is akin to forecasting the futures and intersection of several disciplines which might interact over time (naturally or by human intention) in interesting and possibly unforeseeable ways.

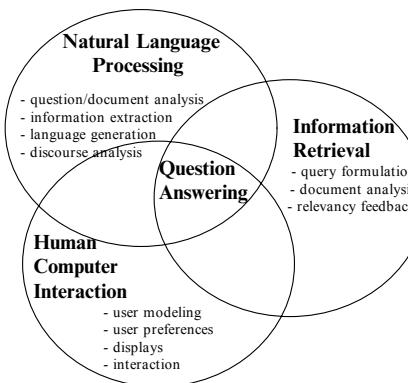


Figure 1. Relationship of Question Answering to other Disciplines.

Drawing upon several workshops (Maybury *et al.*, 2002; 2003) and a resultant collection (Maybury, forthcoming), this chapter summarizes, synthesizes, and assesses contributions of the future of question answering. This chapter considers a number of new directions in turn and concludes with a roadmap for the future.

2. OVERVIEW OF FUTURE CAPABILITIES

Conventional question answering systems consist of some basic elements. Regardless of their underlying data and knowledge sources, methods (e.g., hand coded, machine learned or hybrid), architectures (e.g., rules, blackboards, case-based, agent-based), as Figure 2 illustrates, most current question answering systems

have some mechanism for understanding a user question, selecting source(s) in which to find an answer, retrieving and extracting answers, selecting the most appropriate answer(s), and generating a presentation of that answer. Notably, during each of these processing steps there may need to be interaction with the user or other processing components. Elements from current systems provide primarily English, factoid based answers from text sources. Question answering systems are advancing along several simultaneous dimensions, detailed in the subsequent sections. For example, users will be able to pose questions in a range of languages and search multilingual sources enabling cross-lingual question answering. In addition, users will be able to pose questions in a variety of media and modalities (e.g., textually, graphically, geospatially, auditorily) and receive multimodal (e.g., audio, visual, haptic) and multimedia (e.g., text, graphics, speech) answers. Imagine in the future someone asking a cooking or cultural question answering system how an ethnic dish tastes, smells, or feels? Systems will have an increasing amount of sources and presentational resources available to them to synthesize the most effective response.

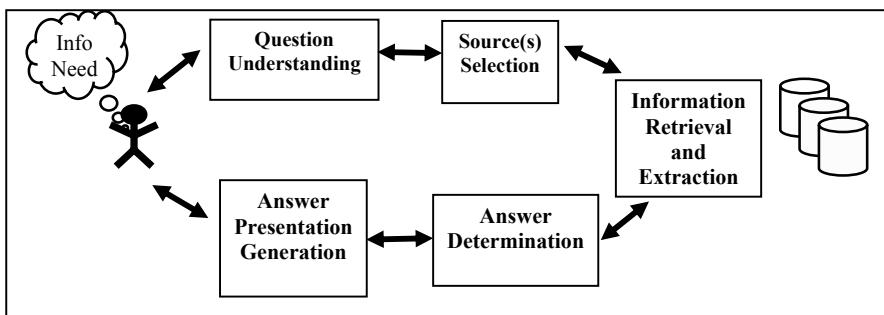


Figure 2. Elements of Question Answering.

Table 1 summarizes the current state of the art in a range of component areas as well as new functional areas. For example, the processing of questions will move beyond current isolated language processing to include context-sensitive question interpretation using methods such as discourse analysis, question decomposition, and semantic or conceptual question analysis. Queries will be enriched with multimedia and multimodal capabilities (e.g., gestural and/or geospatial query) and will support increasingly complex and context dependent situations. Whereas current systems exploit pre-defined knowledge and data sources, in the future systems will need to not only discover but also evaluate and assess over time the quality, veracity, reliability, and value of sources. Once selected, information will be extracted from these sources in increasingly complex ways to include extraction of patterns beyond entities, relations, and events to include arguments and rhetorical structure. Answer selection from the many partial, incomplete, or contradictory answers will require not only an ability to discern quality but also the ability to aggregate, integrate, and fuse answers across heterogeneous sources. And when the answer is generated for the user, it will not only include the answer and source but it

will be an interactive presentation that is interactive, context sensitive, and tailored to the user, environment (e.g., device, bandwidth, mobility), and task.

In addition to these basic component advances, a number of new functional question answering capabilities will occur in the future. One area of current active development is answering questions about term definitions or for biographical characterizations of individuals or organizations. Learning patterns and rhetorical structure both for effective extraction and coherent generation are important current research topics. Another new functional area is answering questions about events and time, including the generation of time lines and stories. Also important will be question answering related to multiple perspectives or opinions on a particular topic. Important capabilities will be the discovery of ranges of viewpoints, the detection of opinion change over time, and the automated interpretation and generation of argument from explored hypotheses and presented evidence. Because diversity in viewpoints can be cross cultural, increasingly there will be a need to support multilingual question answering, supporting both cross-lingual query and multilingual answer discovery, trans-lingual integration, and presentation. And because answers will be found often in heterogeneous sources, we will need an ability to support multimedia (i.e., text, graphics, pictures, sound) and multimodal (i.e., visual, auditory) questioning and answering. Finally, because of the steady increase of web content, methods to exploit this massive corpus and harness its power (e.g., to build language models will become an important future direction.)

Because of the increasingly complexity of these system, usability and habitability will become key to success. This means a system will need exquisite self-knowledge and the ability to introspect to understand when it is unable to answer a question. And because some answers will require extensive work to compute them, the system should know what answers it has previously given if not also what answers other systems have provided so that it can reuse and build upon past experience. Systems will also need to be enriched with models of their user and discourse so they can tailor their interaction. Finally, we will need to go beyond the current corpus based evaluation methods to develop new user, task, and environment centered evaluations and toward rapid and diagnostic evaluations to support effective discovery (Breck *et al.*, 2000).

Table 1. New Directions in Question Answering.

AREA	CURRENT CAPABILITIES	REQUIREMENT GAP
<i>Question processing</i>	Lexical, syntactic, and semantic analysis of isolated natural language queries.	Discourse analysis. Question sequence analysis. Decomposition of complex questions into subquestions. Semantic or conceptual search, geospatial/temporal search, multimodal query, and media specification. Richer question taxonomies.

AREA	CURRENT CAPABILITIES	REQUIREMENT GAP
<i>Source selection</i>	Pre-determined knowledge and data sources including lexical (e.g., WordNet), ontological (e.g., CYC), domain, and a range of documents including structured (e.g., location gazetteer), semi-structured (e.g., encyclopedias, e.g., CIA Fact Book), and unstructured (e.g., news like WSJ).	Automated discovery of relevant knowledge and data sources, provenance/pedigree analysis, assessment of veracity and quality of sources.
<i>Information retrieval and extraction</i>	Information retrieval which trades precision and recall (e.g., 80% precision with low recall) and information extraction at ~90+% precision and recall for entities (e.g., people, locations, organizations), ~70+% for relations and event extraction, ~60% for multilingual event extraction.	Higher performing information retrieval and information extraction. Deeper understanding to include argument extraction, rhetorical structure recovery, importance analysis, contextual interpretation.
<i>Answer determination</i>	Selection of most prominent answer.	Dealing with partial, incomplete, contradictory, or negative (null) answers. Aggregation, integration, and fusion across heterogeneous sources.
<i>Answer presentation generation</i>	Presentation of answer and source context.	Answer explanation generation and answer justification that is interactive, context sensitive, and tailored to the user, environment, and task.
<i>Definitional and Biographical QA</i>	Preliminary prototypes for definition and biography QA.	Learn patterns of good definitions and biographies from example texts.

AREA	CURRENT CAPABILITIES	REQUIREMENT GAP
<i>Event and Temporal question answering</i>	Limited.	Time line creation, Story generation.
<i>Multiple perspective question answering</i>	Limited.	Opinionoid question answering. Opinion classification and clustering. Detection of opinion change over time. Automated analysis/generation of argument.
<i>Multilingual question answering</i>	Answer extraction in single languages (e.g., English, Japanese).	Multilingual and mixed language questions. Integration of responses across multilingual sources.
<i>Multimedia and Multimodal question answering</i>	Retrieval of captioned images related to factoid questions (e.g., "What does Castro look like?").	Answering questions regarding entities and their properties, (e.g., temporal, spatial, semantic) relations, and events over time involving multimedia (text, audio, video) and multimodal (visual, auditory) queries, sources, and answers.
<i>Methods and systems for question answering on the web</i>	Exploitation of web to build language models, exploit answer redundancy.	Use of web for system extension. Source assessment.
<i>Usability and habitability of question answering systems</i>	Limited self knowledge of system performance boundaries.	Ability to predict success and explain failure and/or capability boundaries. Consistency in competency across questions and answers.
<i>Re-use in question answering</i>	Limited reuse	Reuse of questions, answers, question-answer relations, sources, and justifications.
<i>Interactive and/or dialogue based question answering</i>	Resolution of anaphora and ellipsis in question sequences.	Rich representations and reasoning about discourse and user models.

AREA	CURRENT CAPABILITIES	REQUIREMENT GAP
<i>Evaluation of question answering</i>	Corpus based evaluation, initially on factoid QA and extending to additional question classes.	User, task, and environment centric evaluations. Real-time evaluations.

3. ENHANCED QUESTION PROCESSING

One of the most important aspects of question answering is understanding the question. Researchers have been exploring increasingly sophisticated question analysis. This includes moving from more simple taxonomies of question to support factoid question answering, to richer question classifications. Methods to enrich questions, such as query term expansion, have also been used to get access to more material that is relevant to the query. Richer semantic analysis of natural language queries also can enhance performance, supporting more conceptual search than simple matching of query keywords. Interaction with the user, while expensive, can significantly enhance performance, for example, by clarifying of vague questions or sense selection if polysemous query terms are used. Analysis of large corpora, such as queries from search engine web logs, can inspire increasing the range and sophistication of question taxonomies.

Working on real-time questions in the domain of Reusable Launch Vehicles (aerospace engineering), Diekema *et al.* (2003) found a range of complex questions. In addition to classical who, what, when, where, and why questions, they also found questions dealing with quantification that look for a specific amount, such as cost, weight, number, maximum, volume, etc. (e.g., What is the highest temperature the space shuttle undersurface experiences during its mission?), conditional questions indicated by phrases such as in addition to, aside from, other than, etc. (e.g., Aside from contact of two tiles that can be damaging, are there any other reasons why insulating tiles on Reusable Launch Vehicles must be isolated from one another?) and alternative questions where the questioner provides several alternatives, one of which needs to be proven true (e.g., Are Thermal Protection systems of spacecrafts commonly composed of one panel or a collection of smaller tiles?).

In addition to richer processing of questions, others have looked into better prediction of the success of a given query. For example, Croft and Cronen-Townsend (2003) aim to predict question quality by predicting the quality of retrieved passages or documents. They found strong correlation with average precision for TREC Queries. Figure 3 displays the clarity scores (a scale of 0 to 3) versus average precision (a scale of 0 to 1) for a set of location questions. The upper left and lower right quadrants represent problem cases. In the upper left quadrant of high clarity and low average precision, this represents answerless questions where there is a coherent context such as “What was Poe’s birthplace?” where “birthplace” and “Poe” do not co-occur and so the retrieved candidate passages are bad. In contrast, a question such as “Where was Poe born?” actually performs well and predicts well. In contrast, the lower right quadrant represents low clarity, high

average precision characteristic of very rare, often few correct passages as in the case of the question “What is the location of Rider College?” which had one passage containing the correct answer. Since this passage is ranked first, the average precision was 1, however, thus it is not possible to increase language coherence among passages.

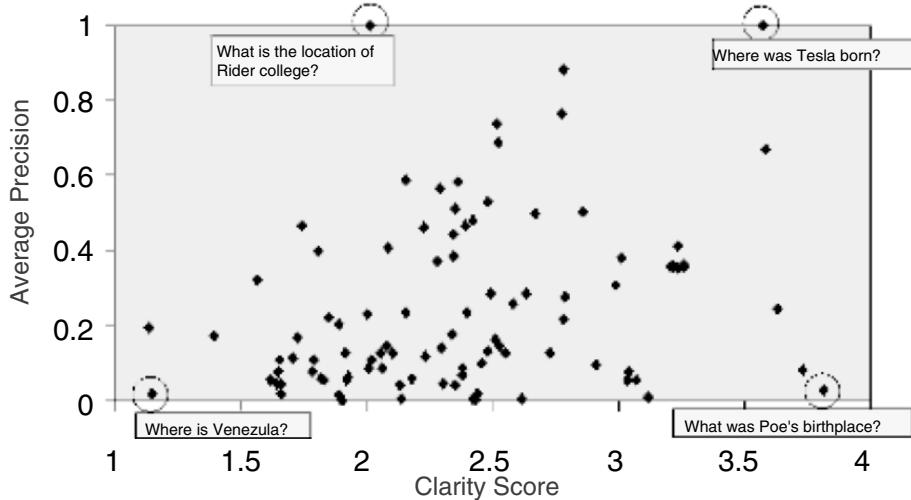


Figure 3. Clarity for QA.

An experiment of 445 well-formed questions selected randomly from an Excite log, clarity scores were found to be significantly correlated with human predictions of question quality. Also, also found that if they perform query expansion, they could use a modified clarity score to guess if the expansion helped. They compute improvement in ranked list clarity scores and if the difference is higher than some threshold, they choose relevance model results. Choosing expansion using this method produces 51% of optimal improvement for TREC-8, though this had more impact than in TREC-7 where expanded query performance is less mixed.

4. QUESTION ANSWERING IN CONTEXT

Important clues from context can also be used to enhance query processing. This can include context provided by the user query itself, previous queries, day or time, the data, the task, location of the interactive device (e.g., kiosk vs. desktop vs. palmtop), and so on. Thus if a user asks “What time is it?”, the answer should be distinct based upon the time zone of the recipient, based on their location. The elliptical follow up question “What month?” should be interpreted in the context of the first question. And if the query “When did the bombing occur?” is followed by “Who caused it?”, this anaphoric reference “it” needs to be interpreted in the context of the prior question.

In addition to the simple context of the previous question, a richer context model could be created that includes what questions have been asked by this user and others, today and before, on other topics. For example, if a user asks about travel locations a system might be able to generalize a model of what questions are asked, e.g., hotel cost, size of bed, if it has a pool, if pets are allowed, etc.

A range of contextual processing beyond the local discourse context might involve dynamically constructing a model of the users interests, beliefs/biases, capabilities, etc. The question answering system might provide distinct answers to a novice user vs. expert domain analyst vs. information access intermediary.

Contact can be exploited both for input analysis and output generation. For example, upon generation a system can refer to previously provided answers or parts of them. In a multimedia artifact, a system could utilize cross modal references to refer to parts of a longer answer to a question (e.g., an event timeline together with descriptive text).

5. SCENARIO BASED QUESTION ANSWERING

Liddy (2003) defines scenario based question answering as “sequences of contextually inter-related questions and responsive answers of varying complexity that need to be understood within a larger model of the user’s information needs/requirements.” She characterizes the context of the scenario as incorporating the background of the analyst, the customer for their product, the intended use of the information, the type, content, and form of the end product, and time frame available for the analysis. She articulates an evidence-based intelligence analysis process model which consists of spiral steps including understanding the task (and the associated product), discovery (including search and inquiry, marshalling evidence, determining key questions, and hypothesis generation), analysis and synthesis (differentiating facts and inference, identifying assumptions, and constructing arguments) and presentation of findings and judgements, including clarification of suppositions and uncertainty.

As Figure 4 illustrates, Prange (2003) exemplifies a question answering scenario that occurs within some overarching context and/or operational context (exemplified by the bounding box). In this scenario which focuses on analyzing an unknown individual observed to meet with a Foreign Minister, two analysts collaborate together and generate a sequence of interrelated questions. First they ask who the advisor is (a definitional question), next they ask what we know about them (a biographical question), then what his or her views or perspectives are, then what influence they have on the leader (causal question), and finally if this signals that policy changes are coming (a predictive question).

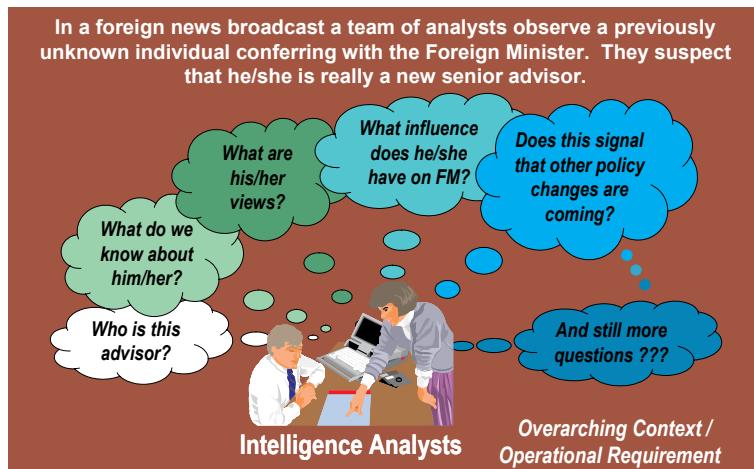


Figure 4. Scenario Based QA.

Prange notes a key task within a scenario is the need to decomposing complex questions into a series/sequence of simpler questions whose answers can be found. He also points out that either the analyst, perhaps with future system assistance, needs to create either explicitly or implicitly, a detailed, complex plan to solve the question answering task at hand.

6. EVENT AND TEMPORAL QUESTION ANSWERING

Event and temporal question answering is the automated interpretation of questions regarding events and their temporal attributes and relations and extraction of answers and their presentation, possibly using devices such as time lines or event diagrams. Questions about events and time are prevalent in logs of internet search engines as in the following (errorful) human examples:

1. When is the 1999 Superbowl?
2. When did Johanne Guterberg invent printing with movable type and completes the first bible?
3. When is our last day before winter break?
4. What happens when warm and cold fronts meet?
5. What type of dancing did people do during the Shakesperean/Elizabethan times?
6. What is the technology timeline in Europe after 1400 A.D.?
7. How long does alcohol stay in your body after you've consumed it?
8. Where can I find the schedule of the stock exchanges around christmas holidays?
9. Where can I buy cds on line and have them delivered to me within an hour?
10. How many minutes, seconds, days, months, hours until the new millenium?
11. What are some long and short term effects of asthma?

Questions 1 and 2 are factual temporal questions, the second one actually a compound temporal question. Question 3 requires looking up a date prior to the start of an event. Question 4 relates to the physical and temporal co-occurrence of two natural phenomena whereas question 5 is about events occurring within a time period. Question 6 interestingly requests a specific temporal display as an answer for a time after a particular point. Question 7 asks about a temporal epoch or duration whereas question 8 asks for a particular temporal artifact (a schedule) but uses a vague temporal specification (“around”) associated with an event. Question 9 also specifies a temporal duration with a fixed, not to exceed endpoint. Question 10 requires a computation (from the current time) and question 11 requires both temporal understanding of short and long and causal analysis of the effects of a condition.

Answering these real questions requires representing and reasoning about time including temporal points and durations, temporal indexicals (i.e., resolving now and then), absolute and relative times, relations among events (e.g., before, after, during, within, around), as well as supporting temporal computations and/or inference.

Pustejovsky *et al.* (2003) reports on the temporal specification, TimeML, a standard for markup of events, their temporal anchoring, and how they are related to each other in documents. TimeML captures some basic event distinctions such as occurrence, perception, reporting, and so on. It extends the TIMEX2 annotation to provide temporal functions (e.g., “three years ago”) and anchors to events and other temporal expressions (e.g., “three years after the Gulf War”). It also identifies signals determining interpretation of temporal expressions such as temporal prepositions (e.g., “for”, “during”, “on”, “at”) and temporal connectives (e.g., “before”, “after”, “while”). Further TimeML identifies event expressions to include tensed verbs (e.g., “has left”, “was captured”, “will resign”), stative adjectives (e.g., “sunken”, “stalled”, “on board”), and event nominals (“merger”, “Military Operation”, “Gulf War”). Finally, TimeML creates dependencies between events and times to include anchoring (“John left on Monday”), orderings (“The party happened after midnight.”) and embedding (“John said Mary left.”). Properties associated with annotated words allow for the specification of, for example, verb tense (e.g., past, present, future) and aspect (e.g., progressive, perfective, perfective, progressive).

TimeML supports the specification of a number of key links between events. A TLINK or Temporal Link represents the temporal relationship holding between events or between an event and a time, and establishes a link between the involved entities. This link makes it explicit if the events are simultaneous (happening at the same time), identical (referring to the same event), before, after, immediately before the other; immediately after the other, including, being included in the other, holding during the duration of the other, being the beginning of the other, being begun by the other, being the ending of the other, being ended by the other.

In contrast to a temporal link, an SLINK or Subordination Link is used for contexts introducing relations between two events, or an event and a signal. For example, a Modal link is a relation introduced mostly by modal verbs (should, could, would, etc.) and events that introduce a reference to a possible world (e.g., “John should have bought some wine.”). A factive is where certain verbs introduce

an entailment (or presupposition) of the argument's veracity such as words like forget, regret or manage (e.g., "John forgot that he was in Boston last year."). A Counterfactive is an event that introduces a presupposition about the non-veracity of its argument as in forget (to), unable to (in past tense), prevent, cancel, avoid, decline (e.g., "John forgot to buy some wine."). Evidential relations are introduced by reporting or perception events as in "Mary saw John carrying only beer." Finally, Slinks can be negative evidential (e.g., "John denied he bought only beer.") or simply negative ("John didn't forgot to buy some wine.").

Finally an ALINK or Aspectual Link represents the relationship between an aspectual event and its argument event. Examples of possible aspectual relations are initiation ("John started to read."), culmination ("John finished assembling the table."), termination ("John stopped talking.") and continuation ("John kept talking."). Figure 5 illustrates a text in which time expressions, events, and temporal relations are annotated (DCT means document creation time).

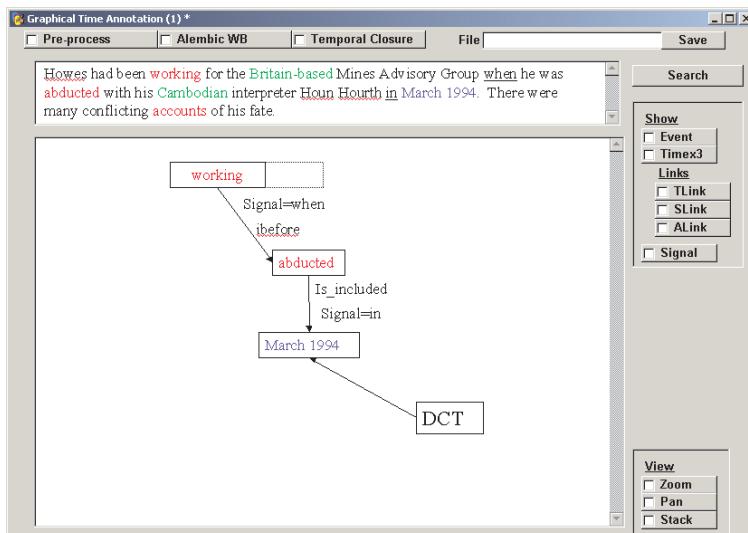


Figure 5. Visual Display of Text Annotated using TimeML.

These careful distinctions (in both question and answer) are necessary in order to get at the answers to temporal questions of the kind in #1-11 above. Because temporal annotation can be time consuming and result in very detailed tagging, Pusteyovsky *et al.* are presently creating a graphical annotation tool for dense annotation tasks (such as TimeML), called the TimeML Annotation Graphical Organizer (TANGO). TANGO will incorporate an interactive closure algorithm into the annotation environment, to help compute event and temporal relationships automatically. When completed, this tool will be publicly released along with a gold standard corpus (TIMEBANK) of 300 annotated documents.

Schilder and Habel (2003) report a temporal expression tagger which chunks nominal chunks with 98% precision and 96% recall and verbal chunks with

95% precision and 93% recall on a set of 10 articles from the Financial Times Deutschland. However, for nominal event extraction performance dropped to 66% precision and 44% recall, although they used only a few morphological and context cues. Finally, they evaluated linking of temporal and event information and were able to achieve precision of 78% with perfect recall.

An important area for future research is not only temporal question and document analysis but also the intelligent presentation of event and temporal information. This might include display of extracted events and temporal knowledge on traditional time lines, link-node diagrams of events and state changes, or histograms. But it might also include new kinds of artifacts such as interactive event displays, organizational charts animated over time (if showing succession events), or event displayed on a cartographic background as they unfold over time. Indeed the incorporation of space as well as time in question answering is a topic to which we turn next.

7. SPATIAL QUESTION ANSWERING

A currently underinvestigated question answering areas is spatial question answering. This is somewhat surprising in light of the popularity of travel planning and route navigation sites. Consider these queries from a popular log involving spatial properties, relations or reasoning:

1. How many miles around the Earth?
2. What is the longitude and latitude of the dominican republic?
3. Where can I find Native American Jewelery around Boston?
4. Where can i find asthma specialist in or near Virginia Beach?
5. How can I determine which airport is closest to my house - Dulles or BWI?
6. Where can I find maps of London England and the area around there?
7. What cloud is closer to earth, cumulus or cirrus?
8. How can I convert a street address into latitude and longitude?
9. How do i convert latitude/longitude to geocodes?
10. What date(s) is the sun directly overhead at 23.5 south latitude?

The first two questions are spatial factoid questions that you might find in a dictionary or encyclopedia. Questions 3-6 require local knowledge (and knowledge of the user's location for question 5) and an ability to reason about nearness. Question 7 requires a computation or at least comparison and questions 8 and 9 are spatial transformations. Question 10 may require some simple orbital reasoning if the answer is not on the web in a table somewhere.

In addition to supporting a range of spatial queries, spatial input devices might augment a traditional query box to specific spatial attributes, extent, or spatial relations. For example, the user might require drawing tools to show a shape or spatial layout. For example the user might use multimedia query such as "Find me images where Bill Clinton is standing in this relation (drawing it) to George Bush." or "Have any bombing occurred in this area?" accompanied by user drawing a bounded area on a digital map. While there has been extensive work in human

computer interaction in gesture recognition and geospatial interaction, the integration and extension of this work to support spatial question answering is a new research opportunity.

8. “OPINIONOID” QUESTION ANSWERING

Whereas question answering systems initially focused on so called factoid questions, *opinionoid question answering* is the automated detection of opinions (of individuals, groups, or institutions) in response to questions about perspectives of entities. Thus, whereas initial QA systems sought a single factual answer to a question, in this case multiple answers to a question or all distinct answers to a question are sought.

The logs of a popular Internet search engine provide ample evidence for a range of opinion questions as shown in these naturally occurring examples:

1. opinions on topics, e.g., “where can i find an opinionated essay on iran contra affair?”
2. opinions about individuals, e.g., “any opinions on Zefer?”
3. opinions about groups, e.g., “what do people think about the Palestinians?”
4. opinions from a group, e.g., “Wherican i find christian opinions?”
5. opinions of a group about a topic, e.g., “What were the opinions of the people in the south about slavery before the civil war?”
6. specific pro or con opinions, e.g., “where can i find some opinions against communism?
7. confirmation/denial regarding social attitudes, e.g., “Do Jamaican’s hate American’s?”
8. request to justify or explain an opinion or belief, e.g., “why do canadians hate the usa?”
9. request to classify an opinion or belief, e.g., “Are my opinions more democratic or republican?”
10. request for range of opinions, e.g., “different opinions about the government?”

Note how question 5 requires reasoning about time as well as opinion. Users can even ask for opinions about (legal) opinions, as in the user who asked “Was the Supreme court biased from the time period of 1800 through 1860?”. As this naturally occurring evidence suggests, opinion questions will serve as a rich source of future research.

For example, Wiebe *et al.* (2003) explore answering questions from multiple perspective or viewpoints. Wiebe argues there are two ways of expressing private states which include:

1. explicit mentions of private states and speech events (e.g., “The United States *fears* a spill-over from the anti-terrorist campaign”)
2. expressive subjective elements (e.g., “The part of the US human rights report about China is *full of absurdities and fabrications.*”).

In an annotation study, Wiebe found that annotators agreed on about 85% of direct expressions of opinion, about 50% of indirect expressions of opinion, and achieved up to 80% Kappa agreement on the rhetorical use of perspective.

Whereas a factoid question typically has one right question (e.g., there is only a single current President), multiple perspective processing allows for different answers to the question “Was the 2002 election process in Zimbabwe fair?” if the source is in Africa (“yes”) or in Europe and North America (“no”). Also, we can imagine these opinions and the underlying subjective elements thereof having various strengths.

Figure 6 illustrates the Multiple Perspective Question Answering (MPQA) annotation tool built within GATE that enables rapid markup up of both expressive subjective elements such as “to put it mildly” or “odium” as well as explicit mentions of private states and speech events (e.g., “said”, “surprised”, “criticism”, and “objected”). These can be used to infer various attitudes, viewpoints, or perspectives of individuals, groups, or organizations in response to questions about opinions. They can also be used to infer evolution or changes in opinion over time.

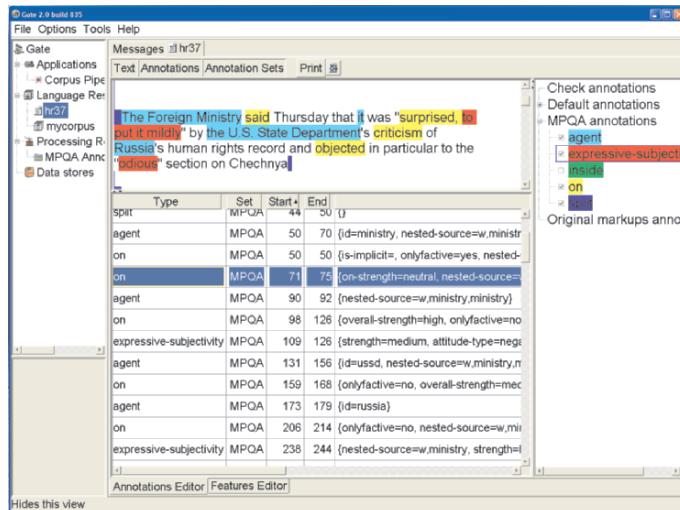


Figure 6. MPQA Opinion Annotation for QA.

Cardie *et al.* (2003) describe the associated annotation scheme for low-level representation of opinions, and note the results of inter-annotator agreement studies using the opinion-based annotation framework. This includes a set of opinion-oriented template relations that identify each expression of opinion in a text along with its source (i.e., the agent expressing the opinion), its type (e.g., positive, negative, uncertain), and its strength (e.g., strong, weak). They propose the use of opinion-oriented “scenario templates” to act as a summary representation of the opinions expressed in a document, a set of documents, or an arbitrary text segment.

For example, the OnlyFactive attribute indicates whether, according to that source, only objective facts are being presented (OnlyFactive=yes), or whether some emotion, evaluation, etc., is being expressed (OnlyFactive=no). For the OnlyFactive attribute, two trained annotators achieved a Kappa (Cohen, 1960) value of 0.80 for 82 of the judgments for which both were certain using a corpus of 114 annotated documents.

The authors trained naive Bayes and k-nearest-neighbor classifiers for the task of annotating low level opinion features using simple lexical features and syntactic features from a partial parser. They found that “ten-fold cross-validation results showed improvements in F-measure for both algorithms over a simple (knowledge-based) baseline system (from 56.7 to 58.0 for naive Bayes, and to 66.4 for k-nearest neighbor.”

As a longer range vision, we might expect a system to be able to automatically extract evidence and argument chains from a variety of perspectives and generate pro and con argumentation for different positions. This will require new advances in automated argument analysis and generation to explore alternative competing hypothesis, but can draw upon recent advances in structured argumentation (Lowrance *et al.*, 2000). Indeed, argumentation in the great tradition of Socratic dialogue, may become an important separate area of question answering research.

9. MULTIMODAL QUESTIONING AND ANSWERING

An analysis of a large web log illustrates that web searchers frequently look for media including photographs (of people, places, objects, animals, events), video (of objects, events, shows, music, sports, dance, robots, famous people), maps (of places, topics, events, directions, travels, demographics), special maps (for street navigation, aviation, sea navigation, stars), radio stations, audio transcripts, and so on. Consider these actual user queries from a popular web search engine:

1. Where can I find pictures of The Battle of Bunker Hill?
2. Where can i find photos of families at christmas time?
3. Where can I view the video tape of JKF shot?
4. Where can I download surgery videos?
5. Where can I find defensive basketball videos?
6. Where can i find a good latin dance workout video?
7. Maps of Champlains first Voyage?
8. Where can i find constellation maps?
9. Where can I find maps of the 1904 World's Fair?
10. Where can I find “Hitchikers guide to the galaxy” radio transcripts?
11. Transcripts of Nightly Business Report?
12. How can I listen to radio broadcasts from around the world?

The user question, the sources, or the resultant answer may be in the form of a range of *media* (e.g., text, graphics, audio, video input or output) and may also be *multimodal* (i.e., involving one or more human sense systems such as auditory, visual, tactile, haptic, or olfactory). For example, a visionary question such as “What

is it like to canoe down the Amazon River in the rain forest” could someday be answered by showing the visual, auditory, olfactory, and even tactile experience of being on the river. In short, *multimodal question answering* is the automated processing of multimodal queries, analysis of multimodal sources, and generation of multimodal responses.

In addition to traditional text sources such as documents or web pages, the user may want to query audio and/or video repositories. For example, Gish *et al.* (2003) are developing answer-spotting technology to provide analysts with answers from a spontaneous speech database in multiple languages. Following automatically generation of topic and semantic category labels for conversations, an analyst can ask the question such as “How many conversations about the Car topic talked about Saabs and not BMWs?” They can follow this up by asking “Of these how many were male speaking to male?” or “Which are the speech utterances that refer to Saabs?”

Users may also want to ask questions about images or videos. Smeaton and Qigley (1996) and others have used language analysis of captions to retrieve associated images and Smeaton *et al.* (2001), Light and Maybury (2003), and others similarly use closed captions or speech transcripts associated with video to retrieve relevant passages. However, transcription errors as well as expressiveness limitations of natural language means that some visual content will not be captured in the associated language.

Accordingly, Katz *et al.* (2003) explore the integration of video and language analysis to support question answering about moving objects that appear within surveillance video. For example, their prototype system, called Spot, analyzes objects and trajectories from surveillance footage taken around the MIT AI Lab in Technology Square in Cambridge, Massachusetts. Using MIT’s START Natural Language System, Spot can interpret a natural language query such as “Show me all cars leaving the garage.” and respond with a video clip showing only cars exiting the garage (see keyframes in Figure 7); all other vehicular and pedestrian traffic is discarded. Spot can currently answer a variety of interesting questions including:

- “Did any cars leave the garage towards the north?”
- “Display cars exiting the garage towards the south.”
- “Show me cars entering Technology Square.”
- “Give me all southbound cars.”

This team is researching a Common Linguistic Visual Representation (CLiViR) is designed to supports tasks such as event recognition, question answering, natural language generated event description, and event filtering (e.g., “Notify me whenever a black sedan pulls up into the driveway.”).



Figure 7. Spot's answer to "Show me all cars leaving the garage".

10. MULTILINGUAL QUESTIONING AND ANSWERING

Multilingual question answering is the automated processing of multilingual queries, analysis and retrieval of multilingual sources, and generation of multilingual responses. Given that much of the web is not in English, it is increasingly important to support cross lingual QA. Many search engines already allow the user to specify their query language and/or target source language to retrieve documents in a particular language.

Magnini *et al.* (2001) describe a Italian/English system in which a question can be posed either in English or Italian, the search can be performed in either language, and the answer is given in the language of the question. Their approach includes the use of WordNet 1.6 in which Italian synsets were created in correspondence with English synsets, importing lexical and semantic relations from the corresponding English synsets. While the resulting system answered only 10% of the questions from the TREC main task, over 50% of the error rate was the consequence of highly restrictive document retrieval (no more than 10 documents per question and no documents over 2000 words long).

Fukumoto *et al.* (2003) describe the Question Answering Challenge (QAC1), a 16 system Japanese question answering evaluation that involved both single and multi-answer factoid types of question (e.g., who, what, when, where, how and why) from two years of the Mainichi Newspaper. The systems could retrieve the answers, in the form of MUC-like named entities, from the newspaper or other information sources such as an encyclopedia, thesaurus, corpus of data and so on. There were three tasks: providing the best five answers (task 1), providing the best single answer (task 2), and then answering a question followed by a series of follow up questions (which contained anaphora and ellipsis). 200 questions were prepared for task 1, 40 for task 3. The most accurate of 14 systems evaluated in Task 1 achieved a mean reciprocal rank of 0.61 with the worst performer obtaining an 0.1 MRR. The best system returned correct answers in the first rank for more than half the questions and in up to the fifth rank in more than $\frac{3}{4}$ of the questions. The most accurate of 14 systems evaluated for task 2 achieved a .36 mean F-score. Only six systems were

evaluated on task 3 and 14 of the 40 questions could not be answered by any system. The best system achieve a .14 MF.

In summary, while there have been several initial attempts to create multilingual systems, much work remains to be done. Automated processing of multilingual queries, analysis and retrieval of multilingual sources, and generation of multilingual responses all remain exciting research areas.

11. EXPLANATION, JUSTIFICATION, AND REASONING

With increasing sophistication of question answering systems, there is an concomitant need for justification of claims, explanation of source selection, and answer generation. This is particularly true if there are sophisticated underlying processing occurring such as aggregation across answers or inference such as generalization across retrieved answers. For example, McGuiness (2003) argues for the need to provide understandable explanations for derived query answers in an intuitive and accessible fashion. She argues that the key to trust of answers is understanding and explanations of information provenance (e.g., source, recency, authoritativeness) and derivation history. A questioner may need to inspect a deductive proof trace, however, this may need to be in an abstracted and more intuitive and accessible form.

In the future, powerful reasoning engines that will support the complex inferential needs of dealing with incomplete, contradictory, and hypothetical reasoning will be required. Machine learning algorithms that will assist in the acquisition of ever changing linguistic, task, and domain knowledge will also be essential.

12. USER CENTERED AND COLLABORATIVE QA

The increasing complexity of sources and systems is also driving both a system need and a user desire for more intimate interaction. In the future this will include increased emphasis on usability and habitability, implying an increased need for systems to predict success and explain failure and/or their own capability boundaries. It will equally mean consistency in competency across questions and answers (i.e., if a system justifies a previous answer with a temporal relational, then a user should be able to use that relation in posing a follow up question).

Tailoring the presentation is increasingly important which will entail more effective modeling of individual user capabilities (cognitive and perceptual), interests, goals and preferences. Just as we want to personalize interaction to the individual, we also want to discover and leverage groups of questioners. This might first start by reusing answers to questions previously asked by others (see the next section). But this might develop further into mechanisms for team discovery. Social network analysis might become important as a method to identify those with similar information needs.

Finally, one way in which to overcome system knowledge limitations is to transform the user-system interaction from ask and tell into a more symbiotic, mixed

initiative relationship in which clarifications, refinements/generalizations, source selections and so on are performed in a joint fashion. In this model a system might learn from user questions and choices and automatically alert of newly discovered sources or documents, anticipate future information needs and preload or preprocess them, or seek to automatically decompose more complex questions into more likely answerable ones.

13. REUSE

One future issue that has been elaborated by Light *et al.* (2003) and Ittycheriah *et al.* (2003) is the reuse of previously computed answers to questions. Humans often capture frequently answered questions (FAQs) to avoid rework or to distribute more broadly knowledge from precious expert human talent. They argue that an “advanced QA system should be able to accumulate questions, answers, and other auxiliary information. This information could then be “reused” to enable the system to better answer future questions. In this way, a system could duplicate a human’s ability to gain knowledge and proficiency in an area as she or he answers questions”. An taxonomy of question reuse and a corpus of one hundred reuse examples was created through the analysis of one million web queries from the summer of 2000 as well as the creation of 221 questions on 29 topics generated by researchers and graduate students in 2003. As summarized in Figure 8, they found question oriented, answer oriented, and combined reuse.

Consider question oriented reuse. For example, some questions are simply syntactic variants of one another (e.g., “Which team won the Stanley Cup in 1994?” is equivalent to “Who won the Stanley Cup playoffs in 1994?”). If the system detects this, it can simply look up the previously computed answer.

Others questions may be subparts of those that have already been answered (e.g., the answer to “When does the summer 2000 Olympics start?” is a part of the answer to question “When are the Summer 2000 Olympics?”). Other questions may be embedded in others (e.g., the answer to the question “How much Vitamin D do you need to absorb the recommended amount of calcium?” requires the answer to the question “How much calcium should an adult woman get in her diet?” for a female questioner). Some questions contain facts, such as “How many Movie awards did “Gone with the Wind” win?” contains the fact that “Gone with the Wind” is a movie. Finally, questions can be clarified (e.g., “Who poses the biggest threat to the United States?” might subsequently be clarified by a question from the same user such as “What countries are developing weapons of mass destruction?”).

Another class of reuse is related to answers. If a question has been previously asked or is simply a variant of a previous one, the answer can be looked up as opposed to be recomputed. Effort can also be avoided by inferring answers from multiple answers or learning the limits of an answer (e.g., the answer from a previous question can limit the interpretation of a subsequent one). Finally, reuse that is a combination of question and answer includes when the same facts are appropriate for a different question, evidences (answers to one question may provide

evidence to another), topic characterizations, and answer revision (previous answers may need to be revised based on subsequent answers).

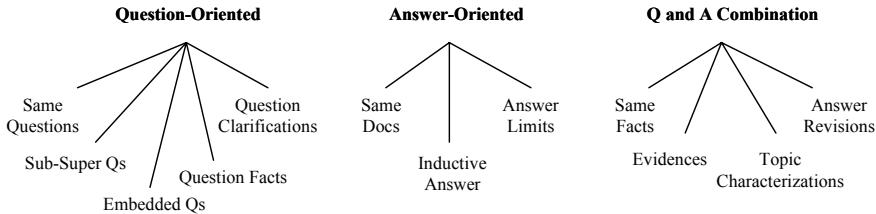


Figure 8. Reuse Taxonomy.

14. NOVEL ARCHITECTURES

A number of researchers are beginning to explore architectures beyond those described in Figure 2. This includes hybrid analysis methods (e.g., statistical and logical), knowledge sources (e.g., dictionaries, encyclopedia, the web) as well as underlying architectures (e.g., blackboards, agents).

For example, the HITIQA system, Small *et al.* (2003) fused the SMART and InQuery question answering systems using linear classifier models. This kind of global or topic independent fusion beat SMART 10 times out of 15 and beat InQuery 6 out of 15 times. Local or topic dependent fusion beat (or tied) InQuery 7(7) out of 15 times. Local fusion beat (ties) even an oracle 5(9) of 15.

To explore if synergy could be obtained by combining multiple question answering systems, Burger *et al.* (2003) used an unweighted centroid method to selecting the best answer for each question given by the 67 submitted TREC QA track runs. Unfortunately, this method did not outperform the top system(s).

However, the highly modular Catalyst architecture (Mardis and Burger, 2002) includes flexible data and control flow and persistent storage (similar to ATLAS stand-off annotation). The system enables composition of language processes joined by a distributed dataflow network.

In the context of PIQUANT, Chu-Carroll (2002) and Welty *et al.* (2003) are exploring the parallel invocation of multiple question answering strategies and systems to increase performance. Figure 9 shows the overall system architecture of PIQUANT in which multiple answering methods (e.g., statistical and rule-based) as well as multiple knowledge source (e.g., a corpus, wordnet, CYC, or the web) are mixed and match to seek synergistic performance. On the TREC-11 corpus, they found they could boost question answering performance by 20% by using multiple sources and by another 7% by using multiple strategies (statistical and rule based). They were able to raise percentage of correct answers from 28.2% with an average precision of .466 to 35.6% with a .586 precision. PIQUANT utilizes CYC to verify answers, such as testing if a quantity falls within acceptable range.

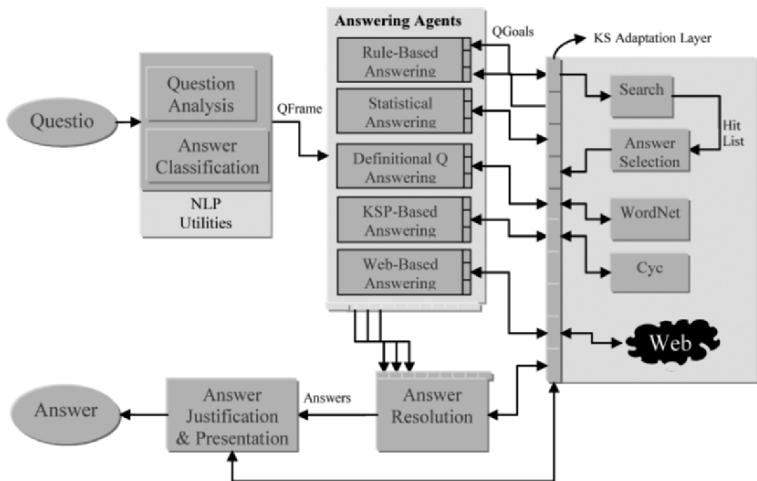


Figure 9. IBM's Agent-based, Multisource QA System, PIQUANT.

15. A ROADMAP TO THE FUTURE OF QUESTION ANSWERING

Considering each of these areas, Figure 10 provides a roadmap of the future of question answering, articulating necessary resources for, impediments to, and planned or possible future capabilities. Figure 10 was jointly created by participants of the LREC 2002 Q&A workshop (Maybury *et al.*, 2002) and further refined at the AAAI Spring Symposium on New Directions in Question Answering (Maybury *et al.* 2003). The roadmap is divided into three lanes dealing with data and resources necessary to develop QA systems, the metrics and methods needed to evaluate them, and the algorithms and application systems that would result. The roadmap runs from 2003 until 2008. Existing capabilities near the bottom of the time line are italicized. Each lane leads to outcomes indicated by sign posts.

Sign posts along the road indicate intermediate outcomes, such as a typology of users, a typology of questions and answers, a model of QA tasks (from both a system and user perspective), QA reuse across sessions, and interactive dialogue. Roadblocks along the way include the need to manage and possibly retrain user expectations, the need for reusable test collections and the need for evaluation methods. Overall workshop participants felt that general natural language processing and inference were limiters to progress, and so these were represented as speed limits signs on the left hand side of the road map. Here also we can see an arrow that indicates that feasibility testing and requirements determination are continuous processes along the road to productive, quality QA.

On the right hand side of the road map we can see the progression of question and answer types. Questions progress from simple factoid questions asked in isolation in text to multimodal and context specific questions, asked within a scenario and across many documents. Questions will progress from simple factbased

questions to how to, why then, and what-if questions. The sophistication of answers will also progress, starting out as simple facts but moving to scripted or templated answers and then progress further to include multimodal, context specific answers.

Related fields such as high performance knowledge bases, topic detection and tracking, databases, virtual reference desks, and user modeling were noted as having particular importance for solving the general QA problem which will require cross community fertilization. An overall, long-term outcome of QA systems is that they become high quality and enhance productivity.

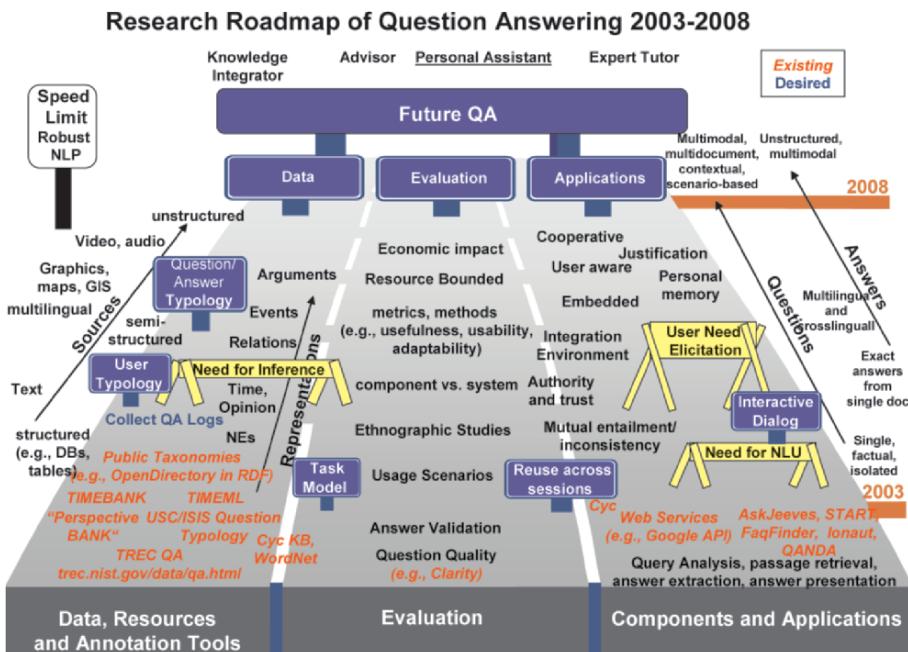


Figure 10. Question Answering Roadmap.

16. SOME LONG RANGE QUESTION ANSWERING VISIONS

What will the user experience of question answering be like in the long term future? No one really knows, however, some possibilities include personal assistants, embedded intelligence, and embodied agents. One possibility is the emergence of personal assistants which are an extension of oneself. These assistants would develop a detailed model of user capabilities, preferences and interests, including their cognitive style. For example, if they are an auditory/linguistic, or visual/kinesthetic learner, a question regarding how to fix a device might be answered in a personalized manner (e.g., with a text, video, or virtual animated

device). They might become your personal tutor or trainer. Eventually, these assistants might develop to become stand in agents or avatars for human users.

In contrast to this user centered vision, we might focus instead on the future of the surrounding environment. Thus, another possibility is that sophisticated question answering abilities will become increasingly embedded in appliances, buildings, cars, clothing and so on. This embedded intelligence will have the ability to assist users, guide use, and enhance performance. Question answering functions might include explanations of how to use, repair, maintain, or simply understand the operation of these devices.

Another possibility is the emergence of embodied agents, either virtual avatars or physical humanoid robots. For example, these could act as guides through real "or virtual shops, museums, or factories. They might provide a range of question answering facilities including justification, explanation, and perhaps even prediction.

One issue will become increasingly prevalent will be the best use of human and system with respect to specific tasks. Whether the question answering intelligence is user centered, environment embedded, or agent embodied, a crucial issue will be discovering how the system and user can interact as cooperative, symbiotic, mixed initiative partners.

17. ACKNOWLEDGEMENTS

I would like to thank each of the contributors to the LREC Workshop, AAAI Spring Symposium, and the collection New Advances in Question Answering for their unique collective insight into the future of question answering.

18. REFERENCES

- [AQUAINT] ARDA AQUAINT Program - <http://www.ic-arda.org/InfoExploit/aquaint/index.html>.
- [ARDA] ARDA QA Roadmap http://www.nplir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc
- Bos, J. and Gabsdil, M. 2000. "First-Order Inference and the Interpretation of Questions and Answers", Proceedings of Gotelog 2000, Goteborg, Sweden, pages 43-50.
- Breck, E. J., Burger, J. D., Ferro, L., Hirschman, L., House, D., Light, M., Mani, I. 2000. How to Evaluate your Question Answering Systems Every Day ... and Still Get Real Work Done. Proceedings of LREC-2000, Second International Conference on Language Resources and Evaluation 1495-1500. Las Palmas, Spain.
- Burger, J. D., Ferro, L., Greiff, W., Henderson, J., Light, M., and Mardis, S. 2003. MITRE's Qanda at TREC-11" Proceedings of the 11th Text Retrieval Conference.
- Cardie, C., Wiebe, J., Wilson, T. and Litman, D. 2003. Combining Low-Level and Summary Representations of Opinions for Multi-Perspective Question Answering. Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 20-27.
- Chu-Carroll, J., Prager, J., Welty, C., Czuba, K., Ferrucci, D. A Multi-Strategy and Multi-Source Approach to Question Answering. In the Eleventh Text Retrieval Conference (TREC 2002), Gaithersburg, Maryland, November 2002. <http://trec.nist.gov/pubs/trec11/papers/ibm.prager.pdf>.
- Croft, B. and Cronen-Townsend, S. 2003 Predicting Question Quality. AQUAINT 18 Month meeting, San Diego, CA.
- Diekema, A. R., Yilmazel, O., Chen, J., Harwell, S., He, L., and Liddy, E. D. 2003. What do You Mean? Finding Answers to Complex Questions. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 87-93.

- Dragomir, R., Fan, W., Qi, H., and Grewal, A. 2002. "Probabilistic Question Answering on the Web", Proceedings, 11th International WWW Conference (Honolulu, Hawaii, May 2002).
- Eichman, D. 2003. Issues in Extraction and Categorization for Question Answering. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 107-115.
- Fukumoto, J., Kato, T., and Masui, F. 2003. Question Answering Challenge (QAC-1) An Evaluation of Question Answering Tasks at the NTCIR Workshop 3. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 122-133.
- Gish, H., Van Sciver, J. and Hain, R. R. 2003. Question Answering from Spontaneous Speech Data (Answer Spotting). AQUAINT 18 month review.
- Ittycheriah, A., Latto, A., Light, M., and McCracken, N. 2003. A Classification Scheme and Corpus for Reuse. From ARDA NRRC Workshop on Reuse in Question Answering.
- Katz, B., Lin, J., Stauffer, C. and Grimson, E. 2003. Answering Questions about Moving Objects in Videos. 2003. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 145-152.
- Liddy, L. 2003. ARDA NRRC Workshop on Scenario Based Question Answering.
- Light, M. and Maybury, M. 2002. Personalized Multimedia Information Access: Ask Questions, Get Personalized Answers. *Communications of the ACM* 45(5): 54-59. In Brusilovsky, P. and Maybury, M. (eds). Special Section on The Adaptive Web.
- Light, M., Ittycheriah, A., Latto, A., and McCracken, N. 2003. Reuse in Question Answering: A Preliminary Study. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. May 24-26. pp. 78-86.
- Magnini, B., Negri, M., Prevete, R. and Tanev, H. 2001. Multilingual Question/Answering: The Diogene System. Proceedings of the Tenth Text Retrieval Conference (TREC 10). Gaithersburg, MD. 2001. <http://trec.nist.gov/pubs/trec10/papers/IrstQA.pdf>
- Mardis, S. and Burger, J. 2002. Qanda and the Catalyst Architecture. In AAAI Spring Symposium on Mining Answers from Text and Knowledge Bases. Stanford, CA.
- Maybury, M. T., Sparck Jones, K., Voorhees, E., Harabagiu, S., Liddy, L., and Prange, J. Tuesday May 28, 2002. Workshop on Strategy and Resources for Question Answering in conjunction with the Third International Conference on Language Resources and Evaluation (LREC). Palacio de Congreso de Canarias, Canary Islands, Spain.
- Maybury, M. T., Day, D. Prange, J., Pusteyovsky, J. and Wiebe, J. (eds.) 2003 AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26.
- Maybury, M. T., forthcoming. *New Directions in Question Answering*. AAAI/MIT Press.
- McGuiness, D. 2003. Knowledge Representation for Question Answering. In AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. May 24-26. pp. 75-77.
- [NRRC] ARDA NRRC Summer 2002 workshops on temporal, multiple perspective, and reuse in question answering - nrrc.mitre.org.
- Prange, J. 2003. AQUAINT Program Overview. AQUAINT 18 Month PI Meeting. San Diego, CA. 10-12 June.
- Small, S., Strzalkowski, T., and Ting, L. 2003. HITIQA: A Data Driven Approach to Interactive Question Answering. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 94-104.
- Smeaton, A. and Qigley, I. 1996. Experiments on using semantic distances between words in image caption retrieval. Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-96) ACM Press, NewYork, 1996, pp. 174-180.
- Smeaton, A. Over, P. and Taban, R. The TREC-2001 video track report. In Proceedings of the Tenth Text REtrieval Conference (TREC 2001).
- [TREC] Text Retrieval and Evaluation Conference. QA Collections trec.nist.gov/data/qa.html.
- Welty, C., Prager, J., Chu-Carroll, J., and Ferrucci, D. 2003. Hybridization in QA. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 116-121.

- Lowrance, J. D., Harrison, I. W., and Rodriguez, A. C. 2000. Structured Argumentation for Analysis, in Proceedings of the 12th International Conference on Systems Research, Informatics, and Cybernetics: Focus Symposia on Advances in Computer-Based and Web-Based Collaborative Systems, Baden-Baden, Germany, pp. 47-57, Aug 2000.
- Wiebe, J., Breck, E., Buckley, C., Cardie, C., Davis, P., Fraser, B., Litman, D., Pierce, D., Riloff, E., Wilson, T., David Day, D., and Maybury, M. 2003. Recognizing and Organizing Opinions Expressed in the World Press. AAAI Spring Symposium on New Directions in Question Answering. Stanford, CA. May 24-26. pp. 12-19.

INDEX

- Advanced Question Answering for Intelligence (AQUAINT) xviii, 102, 135, 163, 183, 184, 307-309, 342, 428, 479-484, 498
- Advanced Research and Development Activity (ARDA) xviii, xxi, 102, 163, 182, 307, 309, 342, 428, 479, 485
- anaphora resolution 55, 71-93, 194, 523
- anchor terms 392-394, 405
- annotations 41, 50, 110, 125, 126, 144, 195, 205, 309-548
- answer
- candidates 5-30, 58-64, 276-277, 307-340, 386-396
 - chunks/snippets 259-261, 270-279, 363-378
 - classification of 31, 36, 92, 395
 - complex 104, 108-117, 142-144
 - direct 104-107, 216, 469, 471
 - extraction of 5-25, 195-221, 233, 244, 250, 505, 510, 511, 538
 - generation of 178, 185, 344, 350, 389, 551
 - key 15, 16, 260, 414, 415, 423, 507
 - image as 27-32
- incorrect 5, 100, 345, 387, 509
- numeric 29, 236, 246, 249, 254
- passage 5, 113, 123, 125, 208, 334, 489
- patterns 13-15, 58-65, 385-400
- phrase-level 350, 359, 369
- answer/asking point 350, 358-379
- selection of 391-403
- statistical 39-41, 64-67, 270, 276-279, 400, 403
 - knowledge-based 386, 390, 391, 404-405
- sentence 13-16, 54-59, 384-400
- space 165-167, 172-179
- structure of 104, 142-145, 216
- tagging 35-65, 526
- unsupported 62, 421
- validation of 92
- yes/no 156
- appositions 9, 10, 61-65, 76, 85, 135, 217, 219
- Artificial Intelligence (AI) 230, 468, 492, 533
- AskJeeves 230, 287, 418, 419, 482
- assessors 12, 144, 249, 384, 412-429, 432, 434
- attributes 10, 46, 47, 109-124, 159-181, 542-548
- mismatch of 176, 177
- values of 161-181, 268

- automated reasoning 4, 5, 7, 11, 20
 automatic indexing 198
- Bayes Theorem 39, 490, 548
 bigrams 50-52, 233, 290, 512
 biography 37, 99-111, 177, 276, 297, 343, 537
 Boolean queries 13, 24, 231, 269
 bootstrapping 286
- case restoration 355, 372-380
 causal relations 124, 125
 Center for Nonproliferation Studies (CNS) 19-22, 135, 182, 184
 chunks 36, 37, 54, 58-65, 191, 193, 203, 395, 396, 544
 clarification 165, 169, 175, 294, 480-484, 501, 539, 541, 552, 553
 classification/categorization
 accuracy of 301, 342
 algorithm for 293
 semantic 103-137, 209-210, 307-345, 511-527
 hierarchical 49
 clustering 165-173, 276, 296-299, 439, 538
 KNN clustering 107, 119
 coercions 101, 132-145, 477
 COGEX 5, 6, 10, 11
 cognitive factors 445, 451
 collocations 133, 165, 315, 320
 complexity
 cognitive 185
 computational 81, 84, 92
 confidence 111, 169, 183, 309-345
 confirmation 155-158, 166, 229, 546
 conflicts 172-180, 370
 confusion 38, 60, 466-470
 constraints 77-88, 350-372
 content terms/words 229-253
 context
 analysis of 80, 132, 335, 541
 global 329-331, 503
 information 166, 179, 489
 local 35, 75, 233, 329-332, 438, 503
 representation of 126
- context change theory 151
 context-free grammars 267
 conversation 151-161, 480-495
 co-occurrences 133, 138, 262, 318, 357, 512
 co-reference 71-93, 351-365
 resolution 55, 71, 128-130, 194
 Cranfield paradigm 411-415
 cross-validation 342, 396, 548
- data sources 159, 422, 460, 480-484, 535, 537
 decision trees 54, 135-137, 301, 341, 342, 386, 514
 Defense Advanced Research Projects Agency (DARPA) 151, 158, 168, 187, 194, 372, 410, 480
 definite descriptions 72-81
 dependencies
 links 204, 353-360, 368, 369, 379
 relations 203-207, 356
 trees 355, 368, 515
- Dialog Act Markup in Several Layers (DAMSL) 152-159
- dialogue 102, 120, 129, 139-143, 149-185, 477-486, 493-497
 acts 150-158
 Backward-Looking Function 152-157
 Forward-Looking Function 152-155
 clarification 173-178, 184
 human-computer 150-163, 484, 534
 management of 149, 159-161, 165, 178, 484-491
 models 159-160
 motivators 489
 spoken 151
 strategies 173-177, 489
- dictionary 240-245, 519-527
- discourse
 analysis 230, 355, 534-536
 context 73, 541
- distance
 metrics 21, 36, 231
 penalty 57, 58

- documents
analysis 89, 239, 286, 534, 545
classification of 301
collections of 12-15, 21-26,
88-93, 321-324,
367-380, 416-421
ranking 60, 297
retrieval 164-167, 230-251,
416-422, 431-432
selection of 198-202
domain knowledge 20, 117, 119, 120,
551
domain-independent 160, 221,
286, 287, 302, 349-354,
414, 428
domain-specific 18-22, 104, 354,
482, 505, 517, 525-528
open domain 229-256, 479-493
pharmaceutical domain 525-527
- EM algorithm 291-294, 301
encyclopedia 3, 56, 60, 66, 67, 194,
234, 245, 416, 431, 516,
537, 550, 553
end-to-end QA systems 182, 307,
385, 386, 401-404
entities
identification of 195, 233, 238,
244, 251
profiles 352-379
numeric 60, 245
entropy 35-53, 383-405
evaluation
automatic 246-254, 277, 278,
280, 288
baseline for 369, 548
batch experiments 435-438
benchmarks 286, 358, 368, 373,
374, 409-411
community 410-411
comparative 79, 277-280, 420
end-to-end 184
interactive 431, 452
large-scale 184, 409, 428
methodology of 184, 409-416,
536, 554
planning of 428, 429
qualitative 183, 385
- quantitative 183, 185, 307
system-oriented 452
expected answer type (EAT) 20-26,
103-142, 190-215
- failure analysis 255, 469, 471
features
binary-valued 395
exclusion of 372
ranking of 358, 359, 364, 366, 378
selection of 44, 45, 330, 403, 404
vector of 134
feedback 35, 67, 232, 239, 245, 296,
308-309, 435-452, 464-472
negative 438
filtering 24, 165, 197, 262, 333, 352,
418, 432, 490, 491, 520, 549
fitness scores 299-300
F-measure/F-score 50, 52, 351, 374,
548, 550
focus
matching 58
signatures 109, 117, 142-144
frames 114-117, 124, 165-181, 185,
446
conflict 175-178
attributes of 168-170, 178-181
generic 168-179
goal 165, 171-177
groups of 180, 181, 185
merging 173
near-miss 174-178
types of 169-172, 177, 180, 181
FrameNet 100, 114, 115, 126, 134,
137, 142, 170
fuzzy set model 231
- gazetteer 4, 427, 501, 537
General Architecture for Text
Engineering (GATE) 171,
547
Google 18, 19, 25-27, 165, 184, 269,
270, 275, 285, 298, 461,
466, 477
grammar
links 359, 360, 365-371
rules 6, 193, 198, 267, 268
features of 133

- relations 349, 350, 370
- template-based 193
- habitability 457-472
- headlines 178, 179, 240, 245, 301
- Hidden Markov Model (HMM) 355, 372-378
- High-Quality Interactive Question Answering (HITIQA) 149, 162-187, 553
- human assessment/opinions 249, 415, 459
- Human Language Technology (HLT) 168
- hypernyms 8-20, 50-60, 317-318, 336-337
- hyponyms 58-60, 338-339
- IdentifiFinder 171, 187, 351, 386, 392
- implicatures 130-144
- inference 100-102, 388-389, 480-492
- information
 - analysts 149, 483
 - content 151, 154
 - density of 72, 88, 90
 - redundancy 378
 - seeking 162, 175, 176, 460, 489
- information extraction 349-378, 409-416, 477, 485, 486, 534, 537
- information retrieval 165, 166, 229-253, 267-269, 411-416, 431-452, 466-472
 - evaluation of 256, 287, 411, 422
 - interactive 471
- Information Technology (IT) 99
- InfoXtract 349-354, 359-368, 374, 379
- InQuery 165, 187, 232, 233, 385, 553
- intelligence analysis 182, 477, 495, 541
- intentions 99-145
 - intentional structure 104, 113, 117, 131, 138-145
- interaction with structured data (ISD) 160-163, 173, 175
- inter-annotator agreement 136, 547
- interrogative forms 156, 206-208
- inverse document frequency (IDF) 89, 270, 300, 365, 435-437
- inverted index 36, 265
- justification 5, 6, 28, 33, 196, 298, 336, 419, 503, 537, 538, 551, 556
- keywords
 - extraction of 129
 - indexing 349, 358, 359, 372
 - selection of 21
- knowledge
 - bases 343-346, 383-387, 480-499
 - representation of 33, 230, 287, 480, 492, 534
 - resources 232, 352, 354, 404, 405
 - sources 4, 14, 242, 302, 342-344, 493, 534, 553
 - world 5-8, 77-78, 478-483
- learning algorithms 192, 505-507, 514, 551
- lexical analysis 267-268
 - chains 5-8
 - database 4, 215
 - entities 310, 314
 - features 192, 363, 548
 - knowledge 75, 77, 354
 - lookup 215, 354
- lexicons 4, 210-212, 215, 352, 363, 365, 519, 527
- lexico-syntactic rules 517, 520
- linear classifiers 402, 553
- linguistic features 61, 62, 346
 - modules 214, 351, 355
 - patterns 5, 6, 8, 102, 104, 190, 489, 493
 - structures 113, 115, 117, 354, 355
 - variations 189-193, 221
- Link Grammar 205
- logic prover 4-11, 20, 21, 33, 510
- logical form 136, 189, 238, 355, 511
 - metonymies 136-138
 - reasoning 444, 445, 451

- machine learning 354-362, 505-516, 528
Machine Translation (MT) 288-290, 396-399, 409-410
Statistical (SMT) 62-65, 288-301, 397, 399
models 63, 232, 288-290
Mean Reciprocal Rank (MRR) 15-18, 250-255, 278-280
MEDLINE 440-451
Message Understanding Conference (MUC) 73-79, 194-195, 410-415, 522-525
meta-keywords 229-233, 236, 242, 247, 256
meta-search 19, 302
metonymy 132-138, 143-145
metrics 15, 21, 30, 36, 106, 107, 182, 410, 414, 431, 554
morpho-syntactic tagging 207, 222, 223
multimedia 3, 4, 27-33, 535-545
answers 27, 31
documents 3, 4, 31, 33
objects 27-31
multiple perspectives 536, 538, 546, 547
MultiText 259, 260, 265, 274, 277, 278
mutual information 43, 511, 515
named entities 350-377, 401, 511, 522, 525, 526, 550
classes 512, 513
extraction of 194, 196, 202, 221, 259, 276, 309, 414, 477, 527
matching 57, 58
tagging 35, 221, 349, 351, 354, 358, 366, 367, 372-378, 386, 392, 511
types of 191, 209-212, 217, 350, 358-361, 369, 371, 510, 512
named relations 337-338
National Aeronautics and Space Administration (NASA) 248, 249, 379
National Institute of Standards and Technology (NIST) 182, 183, 249-253, 419-429
National Science Foundation (NSF) 302, 528
natural language
document retrieval 286-287
generation 191, 193, 470, 534, 549
models 232, 290, 300, 301, 358, 372, 373, 536, 538
processing (NLP) 409-414, 477-479
queries 287, 439, 462, 536, 539, 549
questions 285-294, 300-303, 463-469
understanding 161, 229-230, 255-256, 478-487
negative evidence 121, 544
negotiation 167, 495, 498
neural networks 232, 514
n-grams 64, 167, 168, 276, 277
noisy channel 286-301, 383, 396, 400
nominalizations 75, 81, 82, 125-138
Okapi 435-438
ontologies 19-29, 315-318, 332-346
paraphrases 191-193, 285-302, 331-343, 385-390
parsing 85-88, 204-208, 280-281
semantic 349-379, 405
shallow 142, 194, 221, 349-378, 506, 521
part-of-speech tagging 79-87, 352-358
passages
retrieval of 12-13, 20-28, 259-281
selection of 259-261, 277, 281, 390
pattern matching 229-238, 259-281, 356-362, 422-423
Penn Treebank 37, 110, 522, 524
PERP 415
phrasal synonymy 515
PIQUANT 342-346, 553-554
PIRCS 229-256

- pivoted normalization weighting 435-437
- PowerAnswer 3-34
- pragmatics 78, 99-102, 130, 190, 203, 352, 355, 477, 487
- predicate-argument structures 110-112, 124, 134, 142
- predicates 112-114, 123-125, 132-138
- predictive annotation 307-345
- prepositional phrases 194-198, 268-269, 355-361, 521-522
- probabilistic models 231-232, 294-302
- probability distribution 40-44, 289-300
- PropBank 100, 110-112, 134, 137
- proper name/noun 74-81, 197-214, 233, 299, 310-341, 351-372, 513-527
- pseudo-documents 338
- QALC 189-224
- QASM 285-302
- QA-tokens 287, 309-341
- queries
- expansion of 20, 21, 36, 66, 67, 89, 233, 296, 340, 363, 365, 540
 - formulation of 13, 23, 231, 320, 342, 462, 534
 - modulation 285
- questions
- analysis of 81-83, 191-222, 229-256, 267-279, 308-341
 - binary 44
 - categorization of 210-220, 261, 270-281
 - complex 101-121, 138-145, 184, 485, 536, 539, 542, 552
 - conditional 539
 - contextual 126-130
 - decomposed 118, 138, 143, 464
 - definition 11-16, 113-131, 218-221, 327-342
 - factoid 103, 109, 111, 132, 142, 385-405, 538-547
 - focus 45-49, 60-64, 109-119, 126, 137
 - follow-up 125, 130, 139, 173, 482, 540, 550, 551
 - generation 123, 124, 132
 - hypothetical 480
 - opinionoid 533, 538, 546
 - predictive 139-145, 541
 - self-generated 470, 471
 - trivia 508-512
 - types of 207-212, 220, 233-254, 307-342, 419-428, 444-449, 465, 478, 522, 528
 - unanswerable 464, 465
 - web-reinforced 275, 278
 - what-is* 334, 336
 - where* 113, 238, 245, 296, 299, 332, 374, 377, 517-527
 - words 45-65, 200-204, 240-246, 512-514, 521
- question answering
- interactive 138, 139, 149-182, 411, 431, 452, 482-489, 533
 - multilingual 533
 - multimedia 27-31
 - sessions 482, 486, 491, 492
 - tasks 80-93, 230-254, 409-424, 554
 - trainable 35, 394
 - web-based 285, 368, 468
- question-answer collections 62, 505-509, 516, 517, 522-528
- question-answer pairs 384-399, 517, 521-525, 538
- ramping interface 486-492
- reading comprehension 54, 423, 507, 508, 514
- redundancy 259-278, 401-405
- reformulations 189-191, 387-393, 404-405
- regular expressions 197, 274, 277, 357
- relevance feedback 232, 239, 435, 438, 452
- relevance judgments 411-422, 432, 438, 447, 451
- reverse engineering 505-523

- scenarios 119-123, 140-144, 159-186, 477-501, 541-554
script judgments 227, 278
seeds 19, 111, 168, 292
segmentation 36, 87, 116, 122, 203-207
semantics 152-179, 245, 425, 489
 concepts 100, 104, 109, 111
 constraints 24, 25
 features 134, 136, 192
 knowledge 78, 86, 87, 100, 211, 212, 527
 mismatches 390
 networks 108
 patterns 111
 relations 13, 14, 21, 100, 321, 337, 357, 370, 386, 387, 404, 538, 500
 shallow 79, 111, 112, 192, 396
 structures 100, 104, 105, 367
 tagging 287
sentence
 analysis 202, 240-251, 355
 boundaries of 40, 41, 54, 89, 355, 478
 ranking 55-58, 89, 229, 233
 retrieval 253, 384-385, 510-514
 selection 191-214, 269
 weighting 233, 245
SGML tags 239, 245
signatures 104, 108-112, 117-128, 137-144, 168
 topic 117-127, 137, 141, 142
snippets 259-280, 358-370
 extraction of 358
 retrieval 27, 363, 378
speech acts 156-158
speech recognition 161, 162, 223, 372, 373, 396
spell checker 467
SQL 481
standard deviation 436, 469
START Natural Language System 230, 468-471, 549
stopwords 165, 247, 264, 272, 274, 279
structural relations/constraints 120, 360, 371
subdocuments 229, 233, 239, 251-253, 256
summaries 152-179, 270-276
supervised learning 354, 373
support vector machines (SVM) 301
synonym expansion 262, 379, 387
synsets 7, 8, 49, 108, 115, 198, 317, 318, 336, 439, 550
syntactic analysis 85, 202, 204, 355
 features 135, 202, 548
relations 203-212
structures 77-86, 190-219
variants 424, 425, 552
templates 193-195, 351-353, 414-415, 488-494
algebra 489-494
 matching 366-369
temporal expressions 76, 81, 82, 542-545
 grounding of 114
term frequency (TF) 240, 263, 300, 321, 395, 435-439
term weighting 201, 230, 233, 239, 270, 271, 278, 280, 308
terrorism domain 194, 526, 527
text collections/corpora 3-17, 112-119, 505-509
 large corpora 160, 422, 477, 539
TextMap 385-389, 404, 405
TimeML 543, 544
TIPSTER 168, 477-479, 481
tokens 240, 271-274, 287, 320-325, 335, 339, 345
training data 38-65, 505-528
TREC - Text Retrieval Conference 409-428, 431-439, 477-482, 506-512
ad-hoc experiments 251, 256
Interactive Track 411, 431-439, 461
Question Answering Track 277-281, 409-428, 431, 478, 553
troponyms 78, 136, 137
type constraints 369-371
unigrams 51, 65, 509, 512

- users
 expectations of 4, 554
 experiments 435-437
 interfaces 350, 367, 431, 437,
 439, 445, 452, 505
 satisfaction 140, 143, 145,
 437-451, 459
- vector space model (VSM) 269, 363
- verb groups 355-378
- visualization 165-180, 445, 451, 481,
 501, 534
- vocabulary 166, 196, 313, 414, 444,
 445, 462, 465, 506, 525-528
- Wall Street Journal (WSJ) 90, 110,
 245, 251, 434, 512, 522-524,
 537
- weapons of mass destruction 18-22,
 170, 175, 486, 552
- web
 data 261, 270-280
 search engines for 4, 19, 23, 28,
 276, 285, 461, 463, 548
- wh*-phrases 317, 359, 360
- wh*-words 209, 296, 299, 315-318,
 361, 518, 522, 526
- WordNet 4-20, 46-67, 106-115,
 132-137, 165-177, 296-299,
 315-320, 336-346,
 Extended 4, 6, 108, 168
 glosses in 5-7, 14, 62, 108, 123,
 390
- Yahoo 12-14, 17, 165, 466, 522

Text, Speech and Language Technology

1. H. Bunt and M. Tomita (eds.): *Recent Advances in Parsing Technology*. 1996 ISBN 0-7923-4152-X
2. S. Young and G. Bloothooft (eds.): *Corpus-Based Methods in Language and Speech Processing*. 1997 ISBN 0-7923-4463-4
3. T. Dutoit: *An Introduction to Text-to-Speech Synthesis*. 1997 ISBN 0-7923-4498-7
4. L. Lebart, A. Salem and L. Berry: *Exploring Textual Data*. 1998 ISBN 0-7923-4840-0
5. J. Carson-Berndsen, *Time Map Phonology*. 1998 ISBN 0-7923-4883-4
6. P. Saint-Dizier (ed.): *Predicative Forms in Natural Language and in Lexical Knowledge Bases*. 1999 ISBN 0-7923-5499-0
7. T. Strzalkowski (ed.): *Natural Language Information Retrieval*. 1999 ISBN 0-7923-5685-3
8. J. Harrington and S. Cassiday: *Techniques in Speech Acoustics*. 1999 ISBN 0-7923-5731-0
9. H. van Halteren (ed.): *Syntactic Wordclass Tagging*. 1999 ISBN 0-7923-5896-1
10. E. Viegas (ed.): *Breadth and Depth of Semantic Lexicons*. 1999 ISBN 0-7923-6039-7
11. S. Armstrong, K. Church, P. Isabelle, S. Nanzi, E. Tzoukermann and D. Yarowsky (eds.): *Natural Language Processing Using Very Large Corpora*. 1999 ISBN 0-7923-6055-9
12. F. Van Eynde and D. Gibbon (eds.): *Lexicon Development for Speech and Language Processing*. 2000 ISBN 0-7923-6368-X; Pb: 07923-6369-8
13. J. Véronis (ed.): *Parallel Text Processing. Alignment and Use of Translation Corpora*. 2000 ISBN 0-7923-6546-1
14. M. Horne (ed.): *Prosody: Theory and Experiment. Studies Presented to Gösta Bruce*. 2000 ISBN 0-7923-6579-8
15. A. Botinis (ed.): *Intonation. Analysis, Modelling and Technology*. 2000 ISBN 0-7923-6605-0
16. H. Bunt and A. Nijholt (eds.): *Advances in Probabilistic and Other Parsing Technologies*. 2000 ISBN 0-7923-6616-6
17. J.-C. Junqua and G. van Noord (eds.): *Robustness in Languages and Speech Technology*. 2001 ISBN 0-7923-6790-1
18. R.H. Baayen: *Word Frequency Distributions*. 2001 ISBN 0-7923-7017-1
19. B. Granström, D. House and I. Karlsson (eds.): *Multimodality in Language and Speech Systems*. 2002 ISBN 1-4020-0635-7
20. M. Carl and A. Way (eds.): *Recent Advances in Example-Based Machine Translation*. 2003 ISBN 1-4020-1400-7; Pb 1-4020-1401-5
21. A. Abeillé: *Treebanks. Building and Using Parsed Corpora*. 2003 ISBN 1-4020-1334-5; Pb 1-4020-1335-3
22. J. van Kuppevelt and R.W. Smith (ed.): *Current and New Directions in Discourse and Dialogue*. 2003 ISBN 1-4020-1614-X; Pb 1-4020-1615-8
23. H. Bunt, J. Carroll and G. Satta (eds.): *New Developments in Parsing Technology*. 2004 ISBN 1-4020-2293-X; Pb 1-4020-2294-8

Text, Speech and Language Technology

24. G. Fant: *Speech Acoustics and Phonetics. Selected Writings.* 2004 ISBN 1-4020-2373-1; Pb 1-4020-2789-3
25. W.J. Barry and W.A. Van Dommelen (eds.): *The Integration of Phonetic Knowledge in Speech Technology.* 2005 ISBN 1-4020-2635-8; Pb 1-4020-2636-6
26. D. Dahl (ed.): *Practical Spoken Dialog Systems.* 2004 ISBN 1-4020-2674-9; Pb 1-4020-2675-7
27. O. Stock and M. Zancanaro (eds.): *Multimodal Intelligent Information Presentation.* 2005 ISBN 1-4020-3049-5; Pb 1-4020-3050-9
28. W. Minker, D. Bühler and L. Dybkjaer (eds.): *Spoken Multimodal Human-Computer Dialogue in Mobile Environments.* 2004 ISBN 1-4020-3073-8; Pb 1-4020-3074-6
29. P. Saint-Dizier (ed.): *Syntax and Semantics of Prepositions.* 2005 ISBN 1-4020-3849-6
30. J. C. J. van Kuppevelt, L. Dybkjaer, N. O. Bernsen (eds.): *Advances in natural Multimodal Dialogue Systems.* 2005 ISBN 1-4020-3932-8
31. P. Grzybek (ed.): *Contributions to the Science of Text and Language. Word Length Studies and Related Issues.* 2006 ISBN 1-4020-4067-9
32. T. Strzalkowski and S. Harabagiu (eds.): *Advances in Open Domain Question Answering.* 2006/2008 ISBN 978-1-4020-4744-2; Pb 978-1-4020-4745-9