ANSWER EXTRACTION FOR SIMPLE AND COMPLEX QUESTIONS

SHAFIQ RAYHAN JOTY
Bachelor of Science in Computer Science and Information Technology
Islamic University of Technology (Bangladesh), 2005

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

ANSWER EXTRACTION FOR SIMPLE AND COMPLEX QUESTIONS

SHAFIQ RAYHAN JOTY

Approved:

Name	*Signature*	*Rank*	*Date*
Dr. Yllias Chali Supervisor:			
Dr. Sajjad Zahir Committee Member:			
Dr. Howard Cheng Committee Member:			
Dr. Giuseppe Carenini External Examiner:			
Dr. Stacey Wetmore Chair, Thesis Examination Committee:			

I dedicate this thesis to my beloved parents whose endless support and inspiration has always been with me at each and every step of my life.

Abstract

When a user is served with a ranked list of relevant documents by the standard document search engines, his search task is usually not over. He has to go through the entire document contents to find the precise piece of information he was looking for. *Question answering*, which is the retrieving of answers to natural language questions from a document collection, tries to remove the onus on the end-user by providing direct access to relevant information. This thesis is concerned with open-domain question answering. We have considered both simple and complex questions. Simple questions (i.e. factoid and list) are easier to answer than questions that have complex information needs and require inferencing and synthesizing information from multiple documents.

Our question answering system for simple questions is based on *question classification* and *document tagging*. *Question classification* extracts useful information (i.e. answer type) about how to answer the question and *document tagging* extracts useful information from the documents, which is used in finding the answer to the question.

For complex questions, we experimented with both empirical and machine learning approaches. We extracted several features of different types (i.e. lexical, lexical semantic, syntactic and semantic) for each of the sentences in the document collection in order to measure its relevancy to the user query. One hill climbing local search strategy is used to fine-tune the feature-weights. We also experimented with two unsupervised machine learning techniques: k-means and Expectation Maximization (EM) algorithms and evaluated their performance. For all these methods, we have shown the effects of different kinds of features.

Acknowledgments

I take much pleasure to express my profound gratitude to my supervisor Dr. Yllias Chali for his persistent and inspiring supervision and helping me learn the ABC of Natural Language Processing. It would not have been possible to complete this work without his encouragement, patience, suggestions, generosity and support.

I also thank my M.Sc. supervisory committee members Dr. Howard Cheng and Dr. Sajjad Zahir for their valuable suggestions and guidance. I wish to thank Dr. Giuseppe Carenini for serving as the external examiner. I thank Dr. Stacey Wetmore for her kind consent to chair my thesis defense.

My cordial thanks to NSERC and the University of Lethbridge for the financial and travel support. I am also thankful to all my fellow researchers and faculty members in the Department of Mathematics and Computer Science for their spontaneous cooperation and encouragement. Furthermore, I would like to express thanks to my friends Salimur Choudhury, Munir Hossain, Sardar Haque, Sadid Hasan, Mohammad Islam and Iftekhar Basith for their friendship and encouragement, and for helping me keep sanity as I completed this thesis.

I wish to thank Dr. Dekang Lin, Dr. Satoshi Sekine, Dr. Sameer Pradhan, and Dr. Chin-Yew Lin for supporting this thesis by giving their tools and data.

Above all, my sincere gratitude to the Almighty who creates and makes things happen.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The size of the publicly indexable world-wide-web has provably surpassed several billions of documents and as yet growth shows no sign of leveling off. Dynamic content on the web is also growing as time-sensitive materials, such as news, financial data, entertainment and schedules become widely disseminated via the web. Search engines are therefore increasingly challenged when trying to maintain current indices using exhaustive crawling. Even using state of the art systems such as AltaVista's Scooter which reportedly crawls ten million pages per day, an exhaustive crawl of the web can take weeks. This vast raise in the amount of online text available and the demand for access to different types of information have, however, led to a renewed interest in a broad range of Information Retrieval (IR) related areas that go beyond the simple document retrieval. These areas include question answering, topic detection and tracking, summarization, multimedia retrieval (e.g., image, video and music), chemical and biological informatics, text structuring, text mining, genomics, etc.

Automated Question Answering (QA)—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 six or seven years (Strzalkowski and Harabagiu, 2008). Nonetheless, the expectations are already tremendous, reaching beyond the discipline itself, which is a subfield of Natural Language Processing (NLP), 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-trival knowledge and intelligence.

Web search engines offer access to billions of web documents 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 the 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, more than 99% of the output is useless. Either relevant information is present among the very few hits, or the search is deemed 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 (*Who won the Nobel prize in peace in 2006?*) rather than general information (*Find anything about Nobel prize in peace in 2006*). With present search technology, there is a limited support for the 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 the usage of boolean operators actually has a very small impact on the results (Strzalkowski and Harabagiu, 2008).

Output granularity is another disadvantage of the search engines. As the web continues to grow, it is increasingly likely that a board 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 the keywords are highlighted. To actually find a relevant information, users must read the few 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 the natural language and 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.

QA research attempts to deal with a wide range of question types including: fact, list, definition, how, why, hypothetical, semantically-constrained, and cross-lingual questions. Search collections vary from a small local document collections, to an internal organization documents, to a compiled newswire reports, to the world wide web.

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.

Several academic and commercial web-based QA systems are available now. For instance, Ask Jeeves[1] search engine, automatic question answering engine[2], START[3] QA system at MIT, QuALiM[4] system by the University of Edinburgh are the well-known QA systems where users can write natural language questions in English. There are several multilingual QA systems that support questions in different languages. For example, AnswerBus[5] answers questions in English, French, Spanish, German, Italian and Portuguese.

[1] http://www.ask.com/
[2] http://www.answers.com/bb/
[3] http://start.csail.mit.edu/
[4] http://demos.inf.ed.ac.uk:8080/qualim/
[5] http://www.answerbus.com/index.shtml

The DFKI[6] system supports English, Spanish and German languages.

Considerable advances have been made in answer extraction accuracy over the last 6 years, but QA does not stop with answer extraction. Answering analytical, exploratory, hypothetical and other open-ended complex 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 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.

This thesis concerns open-domain question answering. The counterpart to open-domain is restricted-domain. Restricted-domain documents are of a known subject and sometimes include a known format or a knowledge base, such as the PLANES system (Waltz, 1978). The PLANES system's domain is aircraft maintenance and flight data, and uses a knowledge base contained in a database. When a user asks a question, the system turns their question into a database query. If a restricted-domain system does not use a knowledge base, the subject or topic of the data is known beforehand. In these cases, extraction patterns can be created such that the system can easily access the information contained in the

[6]http://experimental-quetal.dfki.de/redirect.jsp

documents of a certain subject. KAAS (Diekema, Yilmazel, and Liddy, 2004) is a system that performs such extraction techniques in a restricted-domain system.

Open-domain question answering is when any document set can be used and extraction techniques will not be tailored to the subject of the data. This means questions about any subject can be asked, which makes extracting information increasingly difficult since the Information Extraction (IE) systems are highly dependent on domain knowledge (Moldovan et al., 2000). To be an open-domain question answering system, the QA system must handle questions of different types: *simple* or *complex*. In this thesis, we deal with both simple and complex questions.

1.2 Question Answering

Some questions are easier to answer which we call *simple questions*. For example, the question, "Who is the president of Pakistan?" asks for a person name. This type of questions (i.e. factoid) requires small snippets of text as the answers. Again, the question "Which countries has Pope John Paul II visited?" asks for a list of small snippets (i.e. list questions) of text. Finding answers to these questions are easier than questions that have complex information needs. People frequently ask these questions and they want QA system that is more efficient than Internet search, but at the same time is flexible, robust and not fussy, just like Google. Hence, the first problem taken up in Chapter 4, is *Answering Simple Questions*.

The best factoid QA system (Moldovan, Clark, and Bowden, 2007) can now answer over 70% of arbitrary, open-domain factoid questions. Recent approaches to simple question answering have the basic structure shown in Figure 1.1. The question is the 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-occuring words with the query. 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. The next stage is to select the exact answer, given the information about the answer class and the top documents.

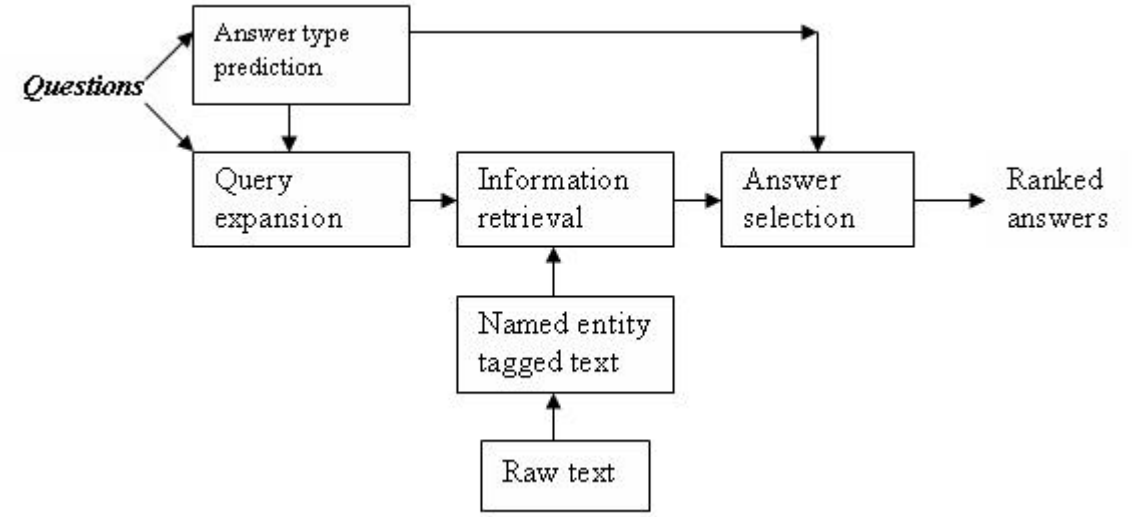

Figure 1.1: Common architecture of QA system

Research in open-domain QA is catalyzed by competitive evaluations conducted by the question answering track of the Text Retrieval Conference (TREC)[7], 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 questions. Moving beyond factoids, the current TREC QA systems can also provide composite answers to definition-style and biography questions, such as "What is photon" or "Who is Tom Cruise?", list questions such as "Name the books written by Paul Krugman", and even relationship questions that look for connections between specific entities, e.g. "What is Suresh Premachandran's con-

[7] http://www.trec.nist.gov/

nection to the Tamil Tigers?"

After having made substantial headway in factoid and list questions, researchers have turned their attention to more complex information needs that cannot be answered by simply extracting named entities (persons, organization, locations, dates, etc.) from documents. For example, the question: *"Describe the after-effects of cyclone Sidr-Nov 2007 in Bangladesh"* requires inferencing and synthesizing information from multiple documents. It is normally understood as an (intellectually challenging) human task, and perhaps the Google answer service[8] is the best general purpose illustration of how it works (Amigo et al., 2004). In this service, users send complex queries which cannot be answered simply by inspecting the first two or three documents returned by a search engine. For example (Harabagiu, Lacatusu, and Hickl, 2006), "What are the key activities in the research and development phase of creating new drugs?" or "What are the main events and important personalities in Myanmar leading up to and since the government changed in September 1988?" Answers to such complex information needs are provided by experts who, commonly, search the Internet, select the best sources, and assemble the most relevant pieces of information into a report, organizing the most important facts and providing additional web hyperlinks for further reading. This information synthesis task is understood, in Google Answers, as a human task for which a search engine only provides the initial starting point. The goal of this thesis is to develop a QA system that can accomplish information synthesis tasks.

From a computational linguistics point of view, *information synthesis* can be seen as a kind of topic-oriented, informative multi-document summarization, where the goal is to produce a single text as a compressed version of a set of documents with a minimum loss of relevant information. Unlike indicative summaries (which help to determine whether a document is relevant to a particular topic), informative summaries must be helpful to

[8] http://answers.google.com/

answer, for instance, factual questions about the topic.

Document Understanding Conferences (DUC)[9], sponsored by the Advanced Research and Development Activity (ARDA), is run by the National Institute of Standards and Technology (NIST) to further progress in summarization and enable researchers to participate in large-scale experiments.

The "definition" and "other" questions in the TREC-QA track and query-relevant summarization task of DUC exemplify the shift from simple information needs to more complex information needs. Therefore, the second problem, addressed in Chapter 5, is *Answering Complex Questions*.

1.3 Text Summarization

Though search engines provide a means to access huge volumes of information by retrieving the documents considered relevant to the user's query, the user still has to go through the entire document content to judge its relevance. This contributes towards a serious information overload problem. It has become a necessity to have tools that can digest the information present across various sources and provide the user with condensed form of the most relevant information (Kolla, 2004). *Summarization* is one such technology that can satisfy these needs.

Summarization can be defined in several ways: According to Mani and Maybury (1999), *"Summarization is the process of distilling the most important information from the source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks)"*. According to Mani (2001), "goal of summarization system is to take an information source, extract content from it and present the most important content to the user in a condensed form and in a manner sensitive to the user's or applications's need." Input to

[9]http://duc.nist.gov/

summarization process can be in different formats like text, video, audio, image, etc. We concentrate only on the textual format of the input hence we call *Text Summarization*.

Different summaries can be generated for the same input source depending on their functionality and usage (Mani, 2001). One of the most important factors is the *compression rate* which is the *ratio of the summary length to the source length*.

Single document summaries are generated from a single document and *multi-document summaries* are generated from a collection of documents. Multi-document summarization involves identification of the various concepts spread across the collection in order to obtain more compression and reduce redundancy.

Summaries can be *"indicative"*, highlighting the salient content of the document without much of an explanation or can be *"informative"*, explaining certain concept to the maximum possible detail at the given compression rate. Summaries can be generated by just copying and pasting the text from the source (*extracts*), or can be generated in abstractor's own words (*abstracts*). If the intended audience of the summarizer is a broad readership community, the summaries contain the information considered salient in the author's viewpoint which is known as *generic summaries*. *User-focused or query-relevant* summaries are generated to be read by a specific group of people having interests in a specific topic or concepts and they include information relevant to the user's interests irrespective of its salience in the document. Summaries can be *fragments* of sentences providing the gist of the document (useful for indexing); or can be highly polished *fluent text* that can be used as substitute for the actual documents, like abstracts of journal articles.

The summarizing operations can be applied on elements such as words, phrases, clauses, sentences or discourse. Elements can be analyzed at various linguistic levels: *morphological, syntactic, semantic and discourse/pragmatic*. Based on the level of linguistic analysis of the source, summarization methods can be broadly classified into two approaches (Mani, 2001):

Shallow Approaches These methods do surface level analysis of the document. They consider features such as word count, presence of cue phrases, position of sentence, to extract the salient sentences and re-arrange them to form a coherent summary.

Deeper Approaches These methods perform deeper syntactic and semantic analysis of the document content to identify the salient portions. They require highly domain-specific information to be able to perform deeper analysis.

Text Summarization is also increasingly exploited in the commercial sectors. Oracle's "Context" uses the summarization to mine textual databases. "InXight" summarizer [10] provides summaries for the documents retrieved by the information retrieval engine. Microsoft's word processor provides the autosummarize option to highlight the main concepts on the document. BT's ProSum, IBM's Intelligent Miner [11] are some of the other tools providing summaries to speed up the process of information access.

Several advanced tools have been developed in recent times using summarization techniques to meet certain requirements (Mani and Maybury, 1999). Newsblaster (McKeown et al., 2003) and NewsInEssence (Radev et al., 2001) allow the users to be updated about the interesting events happening around the world without the need to spend time searching for the related news articles. Meeting summarizer (Waibel et al., 1998) combines the speech recognition and summarization techniques to browse the contents of the meeting. Persival (McKeown et al., 1998), and Healthdoc (Hirst et al., 1997), aid physicians by providing a "recommended treatment" for particular patient's symptoms from the vast online medical literature. Broadcast news navigator (Maybury and Merlino, 1997) is capable of understanding the news broadcast and present the user with the condensed version of the news. IBM's Re-Mail (Rohall et al., 2004) can summarize the threads of email messages based on simple sentence extraction techniques.

[10] http://www.inxight.com/products/skds/sum/

[11] http://www-306.ibm.com/software/data/iminer/

Headlines, outlines, previews of movies, synopses, reviews, digests, biography, bulletins are different examples of summaries that we use in our daily activities.

1.4 State of the Art Question Answering Systems

Researchers working on question answering are trying new directions in QA research to see which methods provide the best results. The following are the types of systems that are currently being developed.

1.4.1 Knowledge Base Systems

Knowledge base systems involve extracting certain information beforehand and using it later. Any information that is extracted before a question is asked will not have to be extracted when that question is asked, thus providing a faster answer than if it had to be extracted on answer retrieval time.

One such system is MAYA (Kim et al., 2001) that creates a database of answers before any questions are asked. There are only fifteen types of entities this system considers as answers. Each passage that contains a possible answer (i.e. any of the fifteen entities) is kept, and when a question is asked, the answer that is contained in the passages most closely related to the question is given as the answer. Katz et al. (2003) developed a similar method of question answering that uses knowledge bases to compile facts about every subject before any definition questions are asked.

Clifton and Teahan (2004) built a knowledge base of questions from the document set. They use knowledgeable agents (Teahan, 2003) that are based on the knowledge grids proposed by Cannataro and Talia (2003). These knowledgeable agents go through the documents and form questions around entities they find. For instance, from the phrase,

"John Lennon died on December 8th, 1980 during a public dramatic interpretation of J.D. Salinger's Catcher in the Rye." their system forms the question-answer pair, *"When did John Lennon die?"* and *"December 8th, 1980"*. When a question is asked, the system will check whether it has the knowledge to answer the question by determining which questions they have identified match the incoming question.

1.4.2 Logical Form Representation Systems

In these systems, the question and the sentences that contain a possible answer are represented in a logical form. The systems use the logical form to determine whether the logical form of a possible answer follows from the logical form of the question. PowerAnswer 2 (Moldovan et al., 2004) is a QA system that uses logical proofs. Their method is outlined in Moldovan et al. (2002). The algorithm involves defining nouns as entities. These entities are then modified by verbs, adjectives and semantic categories, which are used to answer questions.

An example of how their system answers a question is (Dubien, 2005):

Question: *What is the Muslim Brotherhood's goal?*

Question Logical Representation (LR):

```
(exists x0 x1 x2 x3 (Muslim_NN(x0) & Brotherhood_NN(x1) &
nn_NNC(x2,x0,x1) & PURPOSE_SR(x3,x2)))
```

Their system defined x0 and x1 as *Muslim* and *Brotherhood* respectively, then combines them to make entity x2. Their system knows that a "goal" is equivalent to the Semantic Relation (SR) PURPOSE, and x3 will be the final goal for entity x2 (Muslim Brotherhood).

The next step is to turn passages with prospective answers into LR form. In this case the answer is contained in this passage:

> The Muslim Brotherhood, Egypt's biggest fundamentalist group established in 1928, advocates turning Egypt into a strict Muslim state by political means, setting itself apart from militant groups that took up arms in 1992.

And its logical form is:

```
(exists e1 e2 e3 e4 e5 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15
(Muslim_NN(x1) & Brotherhood_NN(x2) & nn_NNC(x3,x1,x2) & Egypt_NN(x4)
& _s_POS(x5, x4) & biggest_JJ(x5) & fundamentalist_JJ(x5) & group_NN(x5)
& SYNONYMY_SR(x3,x5) & establish_VB(e1,x20,x5) & in_IN(e1,x6) & 1928_CD(x6)
& TEMPORAL_SR(x6,e1) & advocate_VB(e2,x5,x21) & AGENT_SR(x5,e2) & PURPOSE
_SR(e3,e2) & turn_VB(e3,x5,x7) & Egypt_NN(x7) & into_IN(e3,x8)& strict_JJ
(x15,x14) & Muslim_NN(x8) & state_NN(x13) & nn_NNC(x14,x8,x13) & PROPERTY_
SR(x15,x14) & by_IN(e3,x9) & political_JJ(x9) & means_NN(x9) & MEANS_SR
(x9,e3) & set_VB(e5,x5,x5) & itself_PRP(x5) & apart_RB(e5) & from_IN
(e5, x10) & militant_JJ(x10) & group_NN(x10) & take_VB(e6,x10,x12)& up_IN
(e6,x11) & arms_NN(x11) & in_IN(e6,x12)& 1992_CD(x12) & TEMPORAL_SR(x12,e6)
```

The answer will be in a PURPOSE semantic relation and the system already knows that it will be the *Muslim Brotherhood*'s PURPOSE, which means that the answer to the question will be the other phrase involved in that PURPOSE relation. The other phrase refers to the phrase e3, which is *"turning Egypt into a strict Muslim state by political means"*. These pre-defined semantic relations make it possible to answer these types of questions.

1.4.3 Multi Corpus Systems

Some systems use secondary collections of documents, in addition to the primary collection in which the answer is to be found (Voorhees, 2003). Question answering involves finding the answer in the primary set of documents, so any answers found outside those documents can only supplement the answers from the primary set. The most popular corpus that is used to try to improve answer finding is the Internet.

BBN's system (Xu et al., 2002) first used the original corpus to find possible answers. Then it performed a search with Google[12] with the question. When a query is entered, Google displays the top ranked documents to that query, and for each document a summary containing key words from the query is generated. Their system would rank answers by how many times the answer is in the summary of the top 100 documents from the Google query. This is a similar method to Wu et al. (2003)'s web-proofing method, where their system created a query for Google and ranked prospective answers by how many times those answers occurred compared to other prospective answers.

Lin et al. (2003) and Wu et al. (2004) took the opposite approach with their systems and rather than using the original corpus to discover the answer, they used only the Internet. They then attempted to find the answer in the original set of documents. The EagleQA system (Chen et al., 2004) extracts answers from both the web, using the Google summaries, and by a query on the text of the question using the original corpus. The Google summary answers are used later to rank the answers extracted from the primary corpus.

[12] http://www.google.com/

1.4.4 Hybrid Systems

There are many question answering systems that have a serial architecture, such as AnswerFinder (Molla and Gardiner, 2004). A serial QA system consists of modules, each designed to perform a specific task. Data is passed from one module to the next.

Hybrid systems can also take advantage of multiple corpus methods as well. An example of one of these systems is PIQUANT II (Chu-Carroll et al., 2004). PIQUANT II makes use of multiple methods of finding answers. They developed their system so that they could use any method of ranking and extracting answers and could find the answer in any corpus. For each method, it would return the top ranked answers to the system. These answers go through answer justification. In answer justification, their system ensures the answer is in the original set of documents and, if found, the answer is returned to the user.

TextMap (Echihabi et al., 2003) uses three different methods of extracting and ranking answers; knowledge-based, pattern-based and statistical-based. They use these methods on the original corpus as well as on the web, through a Google search. Then, if the top ranked answer is found on the web, it searches the list of answers from the original corpus and, if found, it will return the top answer and the document the answer is found in.

The University of Amsterdam's Quartz QA system (Jijkoun et al., 2003) (Jijkoun and de Rijke, 2004) (Ahn et al., 2004) uses multiple corpora as well as using different methods on these sources. They use a total of seven combinations of ways to extract answers. The sources include: Wikipedia [13], The Internet by using Google and Knowledge Base created from entities from the original corpus.

Their system uses these corpora, along with the original, to form a bank of answers that are all "proved" on the web, with a method similar to the one described by Magnini et al. (2002).

[13] http://en.wikipedia.org/

1.5 State of the Art Query-based Summarization Systems

Researchers all over the world working on query-based summarization are trying different directions to see which methods provide the best results. The following are different types of systems that are currently being developed.

1.5.1 Graph-based Method

The LexRank method addressed in (Erkan and Radev, 2004) was very successful in generic multi-document summarization. An extended version of the original LexRank method was introduced by (Otterbacher, Erkan, and Radev, 2005). As in LexRank, the set of sentences in a document cluster is represented as a graph, where nodes are sentences and links between the nodes are induced by a similarity relation between the sentences. Then the system ranked the sentences according to a random walk model defined in terms of both the inter-sentence similarities and the similarities of the sentences to the topic description or question. This idea is captured by the following mixture model:

$$p(s|q) = d \times \frac{rel(s|q)}{\sum_{z \in C} rel(z|q)} + (1-d) \times \sum_{v \in C} \frac{sim(s,v)}{\sum_{z \in C} sim(z,v)} \times p(v|q) \tag{1.1}$$

Where, $p(s|q)$ is the score of a sentence s given a question q, is determined as the sum of its relevance to the question and the similarity to the other sentences in the collection. C is the set of all sentences in the collection. The value of the parameter d which is called "bias", is a trade-off between two terms in the equation and is set empirically. For higher values of d, we prefer the relevance to the question to the similarity to the other sentences.

The relevance of a sentence s to the question q is computed by:

$$rel(s|q) = \sum_{w \in q} log\,(tf_{w,s} + 1) \times log\left(tf_{w,q} + 1\right) \times idf_w$$

Where, $tf_{w,s}$ *and* $tf_{w,q}$ are the number of times word w appears in s and q, respectively and idf_w is the Inverse Document Frequency (IDF) of word w.

The system measures the cosine similarity weighted by word IDFs as the similarity between two sentences in a cluster:

$$sim(x,y) = \frac{\sum_{w \in x,y} tf_{w,x} \times tf_{w,y} \times (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} \times idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} \times idf_{y_i})^2}}$$

1.5.2 Summarization Based on Lexical Chain

The systems based on lexical chain first extract the nouns, compound nouns and named entities as candidate words (Li et al., 2007). Then using WordNet, the systems find the semantic similarity between the nouns and compound nouns. After that, lexical chains are built in two steps:

1. Building single document strong chains while disambiguating the senses of the words and;

2. building multi-chain by merging the strongest chains of the single documents into one chain.

The systems rank sentences using a formula that involves a) the lexical chain, b) keywords from query and c) named entities. For example, (Li et al., 2007) uses the following formula:

$$Score = \alpha P(chain) + \beta P(query) + \gamma P(nameentity)$$

where $P(chain)$ is the sum of the scores of the chains whose words come from the candidate sentence, $P(query)$ is the sum of the co-occurrences of key words in a topic and the sentence, and $P(nameentity)$ is the number of name entities existing in both the topic and the sentence. The three coefficients α, β and γ are set empirically. Then the top ranked sentences are selected to form the summary.

1.5.3 Summarization Based on QA System

These systems summarize the documents based on the use of a question answering system (Molla and Wan, 2006). At first, the topic sentences are converted to a sequence of questions as the underlined QA system is designed to answer only simple (i.e. factoid, list) questions. This is done in two steps:

1. Transforming the indicative forms into questions.

2. Splitting the complex sentences into several individual questions.

For example, the sentence "Include information regarding trends, side effects and consequences of such use" could be converted into three questions:

1. What are the trends of such use?

2. What are the side effects of such use?

3. What are the consequences of such use?

Then the QA system normalizes, classifies the question and finds the candidate answers along with the sentences in which the answers appeared. Instead of extracting the exact

terms as answers, the systems extract the sentences for each of the questions in the topic. In this way, the systems generate the summary.

1.5.4 Summarization Based on Statistical Models

There are systems that simplify the query sentences by converting one complex query sentence into several simple sentences (Harabagiu, Lacatusu, and Hickl, 2006) (Pingali, K., and Varma, 2007) (Toutanova et al., 2007). These systems rank the sentences based on a mixture model where each component of the model is a statistical model. For example, the mixture model introduced by (Pingali, K., and Varma, 2007) is as follows:

$$Score(s) = \alpha \times QIScore(s) + (1 - \alpha) \times QFocus(s, Q) \tag{1.2}$$

Where, Score(s) is the score for sentence s. Query-independent score (QIScore) and query-dependent score (QFocus) are calculated based on probabilistic models.

1.6 Our Approaches

Our approach for answering simple questions (Chapter 4) is based on document tagging and question classification. *Question classification* extracts useful information (i.e. answer type) from the question about how to answer the question. *Document tagging* extracts useful information from the documents, which will be used in finding the answer to the question. We used different available tools to tag the documents. Our system classifies the questions using manually developed rules.

TREC 2007 evaluation of our system (Chali and Joty, 2007b) shows that out of 360 *factoid* questions 93 are globally correct which is almost 26% accuracy. Our system is

ranked 4th among 51 participants in factoid question answering. The average F-measure of our system in the list questions is 0.132 which is ranked 6th among 51 participants. The average per-series score of our system is 0.1410 which is almost the same score as the 10th system in TREC 2007.

For complex questions (Chapter 5) we experimented with both empirical approach and machine learning approach. Our system (Chali and Joty, 2007a) that participated in DUC-2007 was based on the empirical approach. We then experimented with several statistical machine learning techniques for this particular problem and evaluated their results. Our empirical approach used six features in order to rank a sentence while our machine learning techniques used eighteen features to rank a sentence. We also experimented with the effects of different types of features in generating good summaries using each of the algorithms. To assess the contribution of different features in generating quality summaries the features are grouped into several classes. They are as follows:

Lexical: N-gram (N= 1, 2, 3, 4), LCS, WLCS score, skip bi-gram, head, head synonym and BE overlap.

Lexical Semantic: Synonym, hypernym/hyponym, gloss, dependency-based similarity and proximity based similarity.

Syntactic: Tree kernel-based syntactic feature.

Semantic: Shallow semantic tree kernel-based semantic feature.

Cosine Similarity: Graph-based similarity.

Our experiments show that including syntactic and semantic features improves the performance. Comparison among the learning approaches are also shown. Comparing with DUC 2007 participants, our systems achieve top scores and there is no statistically significant difference between our system and the DUC 2007 best system.

1.7 Statistical Machine Learning for NLP Problems

As a broad subfield of Artificial Intelligence, Machine Learning (ML) is concerned with the design and development of algorithms and techniques that allow computers to learn. The major focus of machine learning research is to extract information from data automatically, by computational and statistical methods.

Many important tasks within NLP require a substantial amount of knowledge, usually provided manually by practitioners within the field. However, with a shift to automatic processing of large amounts of language data, empirical methods have sought to reduce the dependence of manually embedding knowledge into NLP systems, and to let learning algorithms acquire the knowledge from available data. Data classification is a thoroughly active research topic within ML, and much research where ML and NLP have been combined is aimed at transforming a common NLP problem so that it can be represented as a classification problem. As a result, it is not uncommon to see that most well known ML techniques have been applied to just about every possible NLP task, where feasible.

Linear classifiers come in various flavours, but all are relatively simple to understand and are computationally efficient, particularly with 2-class problems. For example, linear threshold algorithms can calculate a weighted sum from input features, and then depending on whether that sum is greater or smaller than a certain threshold, a decision can be made as its class. Thresholds are determined through the use of training data. Such classifiers have been applied to text categorization (Dagan, Karov, and Roth, 1997) (Lewis et al., 1996), shallow parsing (Munoz et al., 1999) and Part of Speech (POS) tagging (Roth and Zelenko, 1998).

Decision trees are used to partition large samples of data into a hierarchical structure. Commonly associated as a tool for classification, they are also capable of generalizing seen data into sets of rules. They are generic enough to be applied to many subtasks of NLP,

and have been found in machine translation (Tanaka, 1996), parsing (Haruno, Shirai, and Ooyama, 1998), text categorization (Weiss et al., 1999) and word-sense disambiguation (Brown et al., 1991).

Clustering is a powerful method as it is one of the few that is fully unsupervised. It aims to discover natural partitions within data by grouping entities into clusters based on their similarity. This technique is utilized in syntactic and semantic classification (Hughes, 1994) and information retrieval (Ibrahimov, Sethi, and Dimitrova, 2001).

Statistical machine learning is a flavor of machine learning distinguished by the fact that the internal representation is a statistical model, often parameterized by a set of probabilities. For illustration (Berger, 2001), consider the syntactic question of deciding whether the word *chair* is acting as a verb or a noun within a sentence. Any fifth-grader would have little difficulty with the problem. But how to program a computer to perform this task? Given a collection of sentences containing the word *chair* and, for each, a labeling *noun* or *verb*, one could invoke a number of machine learning approaches to construct an automatic "syntactic disambiguator" for the word chair. A rule-inferential technique would construct an internal representation consisting of a list of lemmae, perhaps comprising a decision tree. For instance, the tree might contain a rule along the lines "If the word preceding *chair* is *to*, then *chair* is a verb." A simple statistical machine learning representation might contain this rule as well, but now equipped with a probability: "If the word preceding *chair* is *to*, then with probability *p chair* is a verb."

These applications of ML to NLP problems were quite successful. For example, given a reasonable number of instances of one particular type of entity in training data, learning algorithms such as Support Vector Machine (SVM) or Maximal Entropy (MAXENT) can recognize 80-90% instances of that type in test documents with similar accuracy. The earliest automatic POS tagging systems, based on an expert-systems architecture that requires the knowledge of English grammar, achieved a per-word accuracy of only around 77%

on the popular Brown corpus of written English. Surprisingly, perhaps, it turns out that a knowledge of English syntax is not at all necessary or even helpful in designing an accurate tagging system. Starting with a collection of text in which each word is annotated with its part of speech, one can apply statistical machine learning to construct an accurate tagger. A successful architecture for a statistical part of speech tagger uses Hidden Markov Models (HMMs), an abstract state machine whose states are different parts of speech, and whose output symbols are words. In producing a sequence of words, the machine progresses through a sequence of states corresponding to the parts of speech for these words, and at each state transition outputs the next word in the sequence. After automatically learning model parameters from this dataset, HMM-based taggers have achieved accuracies in the 95% range.

The success of Hidden Markov Model tagging supports the claim that knowledge-free (in the sense of not explicitly embedding any expert advice concerning the target problem) probabilistic methods are up to the task of sophisticated text processing and, more surprisingly, can outperform knowledge-rich techniques (Berger, 2001).

We follow this claim by experimenting with statistical machine learning techniques in the task of answering complex questions.

1.8 Thesis Outline

The remaining chapters of this thesis are organized as follows:

- Chapter 2 reviews the mathematical tools on which the QA systems rely.
- Chapter 3 is a discussion of various methods of tagging useful information in the documents.
- Chapter 4 addresses our approach for answering simple questions.

- Chapter 5 includes our approaches to answer questions which are complex in nature.
- Chapter 6 consists of concluding remarks about our findings and our views on the future of QA systems.

Chapter 2

Mathematical Machinery

2.1 Introduction

Our question answering systems rely on different tools and mathematical backgrounds. The QA system for simple questions uses information retrieval technique to retrieve the relevant passages to the query from the whole document collection. Hence, it demands that we give some background information about the information retrieval systems: the indexing methodology, searching and ranking strategy, querying options, etc. Again the QA system for complex questions is based on statistical machine learning techniques which requires the basics of Gaussian distributions, Bayesian networks and clustering techniques. This chapter reviews the mathematical tools and essential backgrounds on which the question answering systems rely: the Gaussian statistics, likelihoods, Bayesian classification, convexity, the k-means algorithm, the EM algorithm and information retrieval.

2.2 Gaussian Statistics

The normal distribution (also known as Gaussian distribution) (Bilmes, 1997), is an important family of continuous probability distributions, applicable in many fields. Each member of the family is defined by two parameters: the *mean* and *standard deviation*. The standard normal distribution is the normal distribution with a mean (μ) of zero and a variance (σ^2) of one. Figure 2.1 shows four normal distributions.

A *multivariate Gaussian distribution* is a specific probability distribution which is a generalization to higher dimensions of one-dimensional normal distribution defined above. The Gaussian probability density function (pdf) for the d-dimensional random variable **x** is

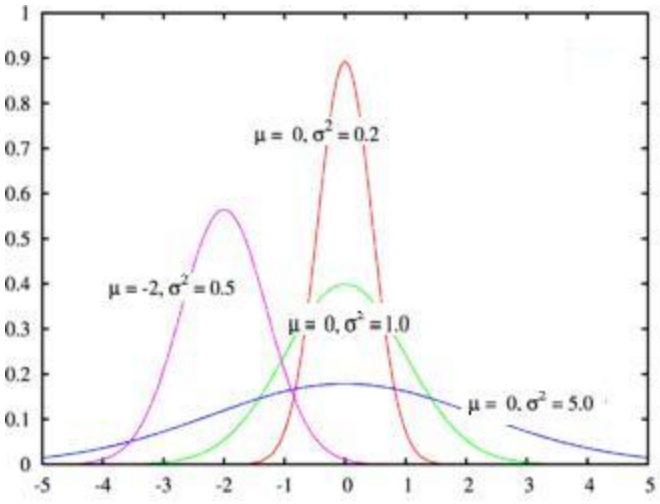

Figure 2.1: Example of four normal distributions

given by:

$$p_{(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\mathbf{x}) = \frac{e^{\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}}{\sqrt{2\pi}^d\sqrt{det(\boldsymbol{\Sigma})}} \tag{2.1}$$

where $\boldsymbol{\mu}$, the mean vector and $\boldsymbol{\Sigma}$, the covariance matrix are the parameters of the Gaussian distribution.

- The mean vector $\boldsymbol{\mu}$ contains the mean values of each dimension, $\mu_i = E(x_i)$, with $E(x)$ being the expected value of x. The mean vector $\boldsymbol{\mu}$ of dimension d is: $\boldsymbol{\mu} = [\mu_1, \mu_2, \cdots, \mu_d]^T$
- All of the variances c_{ii} and covariances c_{ij} are collected together into the covariance matrix $\boldsymbol{\Sigma}$ of dimension $d \times d$:

$$\sum = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,d} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ c_{d,1} & c_{d,2} & \cdots & c_{d,d} \end{bmatrix}$$

The covariance c_{ij} of two components x_i and x_j of x measures their tendency to vary together (i.e. co-vary),

$$c_{ij} = E\left[(x_j - \mu_j)(x_i - \mu_i)\right] \tag{2.2}$$

$$\sum = \mathbf{E}\left[(\mathbf{x} - \mathbf{E}[\mathbf{x}])(\mathbf{x} - \mathbf{E}[\mathbf{x}])^T\right] \tag{2.3}$$

2.2.1 Gaussian Modeling: Mean and Variance of a Sample

We can estimate the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of the Gaussian models from the data points:

$$Mean\ estimate: \qquad \hat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x_i} \tag{2.4}$$

$$Unbiased\ covariance\ estimate: \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{N-1}\sum_{i=1}^{N} (\boldsymbol{x_i} - \boldsymbol{\mu})^T (\boldsymbol{x_i} - \boldsymbol{\mu}) \tag{2.5}$$

2.2.2 Likelihood and Joint Likelihood

The likelihood of a sample point $\boldsymbol{x}_i$ given a model Θ is given by the value of the probability density function (pdf) for that data point. That means $p(\boldsymbol{x}_i|\Theta)$ (i.e. Eq 2.1) gives the likelihood of the data point $\boldsymbol{x}_i$ and the joint likelihood of a series of samples $X = \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N$ for a given model Θ (Θ is a Gaussian distribution) is given by:

$$p(X|\Theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i|\Theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.6}$$

We might choose to compute the log likelihood instead of likelihood as log-likelihood

turns the product into sum:

$$p(X|\Theta) = \prod_{i=1}^{N} p(\boldsymbol{x}_i|\Theta) \Leftrightarrow log\ p(X|\Theta) = log \prod_{i=1}^{N} p(\boldsymbol{x}_i|\Theta) = \sum_{i=1}^{N} log\ p(\boldsymbol{x}_i|\Theta)$$

In case of Gaussian distribution, it also avoids the computation of the exponentials. By applying log to the equation:

$$p(\mathbf{x}|\Theta) = \frac{e^{\frac{-1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}}{\sqrt{2\pi}^d \sqrt{det(\boldsymbol{\Sigma})}} \tag{2.7}$$

$$logp(\mathbf{x}|\Theta) = \frac{1}{2}\left[-dlog(2\pi) - log\left(det(\boldsymbol{\Sigma})\right) - (\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] \tag{2.8}$$

Note that, $log(x)$ is a monotonically growing function, the log-likelihoods have the same relations of order as the likelihoods:

$$p(\mathbf{x}|\Theta_1) > p(\mathbf{x}|\Theta_2) \Leftrightarrow logp(\mathbf{x}|\Theta_1) > logp(\mathbf{x}|\Theta_2)$$

2.2.3 Bayesian Classification

How can we tell that a certain feature vector $\boldsymbol{x}_i$ from a set of several feature vectors X belongs to a certain class q_k? We use the *Bayes' decision rule* to decide this:

$$X \in q_k\ if\ P(q_k|X,\Theta) \geq P(q_j|X,\Theta), \forall j \neq k$$

Which means, given a set of classes q_k, with their own parameters in model Θ, a set of one or more feature vectors X belongs to the class which has the highest probability once we know the sample X. $P(q_k|X,\Theta)$ is called the *a posteriori probability*, as it depends on having seen the sample, as opposed to the *a priori probability*, $P(q_k|\Theta)$ which does not

depend on any sample but depends on the parameters of the model Θ. One can calculate the posterior probability using Bayes' law as it says:

$$P(q_k|X,\Theta) = \frac{p(X|q_k,\Theta)P(q_k|\Theta)}{p(X|\Theta)} \tag{2.9}$$

where q_k is a class, X is a sample containing one or more feature vectors and Θ is the parameter set of all class models. If $p(X|\Theta) = const$ (that is the features are equi-probable), then $P(q_k|X,\Theta)$ is proportional to $p(X|q_k,\Theta)P(q_k|\Theta)$ for all classes:

$$P(q_k|X,\Theta) \propto p(X|q_k,\Theta)P(q_k|\Theta), \forall k$$

2.2.4 Discriminant Function

Discriminant function is a set of functions $f_k(x)$ that allows to classify a sample $\mathbf{x}$ into k classes q_k if:

$$\mathbf{x} \in q_k \Leftrightarrow f_k(\mathbf{x},\Theta_k) \geq f_n(\mathbf{x},\Theta_n), \forall n \neq k$$

Here, the k functions $f_k(\mathbf{x})$ are called discriminant functions. So, the *a posteriori probability* $p(q_k|x_i)$ is a discriminant function:

$$\begin{aligned}
\boldsymbol{x} \in q_k \quad &\Leftrightarrow \quad P(q_k|\boldsymbol{x}_i) \geq P(q_n|\boldsymbol{x}_i), \forall n \neq k \\
&\Leftrightarrow \quad p(\boldsymbol{x}_i|q_k)P(q_k) \geq p(\boldsymbol{x}_i|q_n)P(q_n), \forall n \neq k \\
&\Leftrightarrow \quad logp(\boldsymbol{x}_i|q_k) + logP(q_k) \geq logp(\boldsymbol{x}_i|q_n) + logP(q_n), \forall_n \neq k
\end{aligned}$$

2.2.5 Convexity

A function $f(x)$ is convex or concave up if:

$$f(\alpha x_0 + (1-\alpha)x_1) \leq \alpha f(x_0) + (1-\alpha)f(x_1) \;\; for\ all\ 0 \leq \alpha \leq 1$$

That means, if one selects any two points x_0 and x_1 in the domain of a convex function, the function always lies on or under the chord connecting x_0 and x_1:

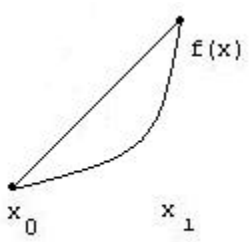

Figure 2.2: Convex function

2.2.6 Jensen's Inequality

Jensen's inequality is the most useful property of convex functions. If f is convex in x, then

$$f(E[x]) \leq E[f(x)]\ or\ f\left(\sum_x p(x)x\right) \leq \sum_x p(x)f(x) \tag{2.10}$$

where $p(x)$ is a pdf. For example, the following condition holds for any pdf p:

$$log\left(\sum_x p(x)x\right) \leq \sum_x p(x)log(x) \;\; since\ (-log)\ is\ convex$$

2.2.7 *Auxiliary Functions*

Auxiliary functions are simply pointwise lower (or upper) bounds on a function. For example, the function $f(x) = x - 1$ is an auxiliary function for $log\ x$ because $x - 1 \geq log\ x$ for all x. We are interested in the auxiliary functions that bounds the change in log-likelihood between two models. If Θ_1 is one model and Θ_2 is another model, then we are looking for an auxiliary function $A(\Theta_2, \Theta_1)$ such that:

$$L(s|\Theta_2) - L(s|\Theta_1) \geq A(\Theta_2, \Theta_1)\ and\ A(\Theta, \Theta) = 0$$

where L is the log-likelihood. This indicates that if we can find a Θ_2 such that $A(\Theta_2, \Theta_1) > 0$, then Θ_2 is a better model than Θ_1—in a maximum likelihood sense.

The EM algorithm introduced in the next section finds a local maximum model starts with a initial model Θ_1, then replace it by a superior model Θ_2 and repeat until no superior model can be found.

2.3 Unsupervised Clustering

The aim of cluster analysis is to divide the data or samples into a number of useful subsets based on the similarity of data points. The method used in order to select sentences for query-based summary extraction, is called *unsupervised learning*. That means, before the learning begins, it is not known how many subsets (clusters) there are or how they are distinguished from each other (i.e. the training data are not labeled with the class information).

For example, suppose we record the spectra of a hundred thousand stars (Russel and Norvig, 2003). Then how many different kinds (i.e. "red giant", "white dwarf" etc.) of

stars are there and which stars will fall in certain kind? The problem is that the stars do not have these labels with them. So, astronomers had to perform *unsupervised clustering* to identify these categories.

Unsupervised clustering begins with data (Russel and Norvig, 2003). Figure 2.3 shows data points, each of which specifies the values of two continuous attributes. The data points might correspond to stars and the attributes might correspond to spectral intensities at two particular frequencies.

Clustering techniques assumes that the data are generated from a mixture distribution, P. Such a distribution has k components, each of which is a distribution. A data point is generated by first choosing a component and then generating a sample from that component. Let the random variable C denote the component, with values 1,....,k; then the mixture distribution is given by :

$$P(\mathbf{x}) = \sum_{i=1}^{k} P(C = i)P(\mathbf{x}|C = i) \tag{2.11}$$

where **x** refers to the values of the attributes for a data point. For continuous data, each component distribution is a multivariate Gaussian and the mixture model is called **mixture of Gaussians**. The parameters of a mixture of Gaussians are:

$w_i = P(C = i)$: weight of component i in the mixture model.

$\boldsymbol{\mu}_i$: mean of component i.

$\boldsymbol{\Sigma}_i$: covariance matrix of component i.

Figure 2.3(b) shows a mixture of three Gaussians; this mixture is in fact the source of the data in 2.3(a). The unsupervised clustering problem, then is to recover a mixture model (i.e. parameters and weights of the components) like the one in Figure 2.3(b) from raw data like that in Figure 2.3(a).

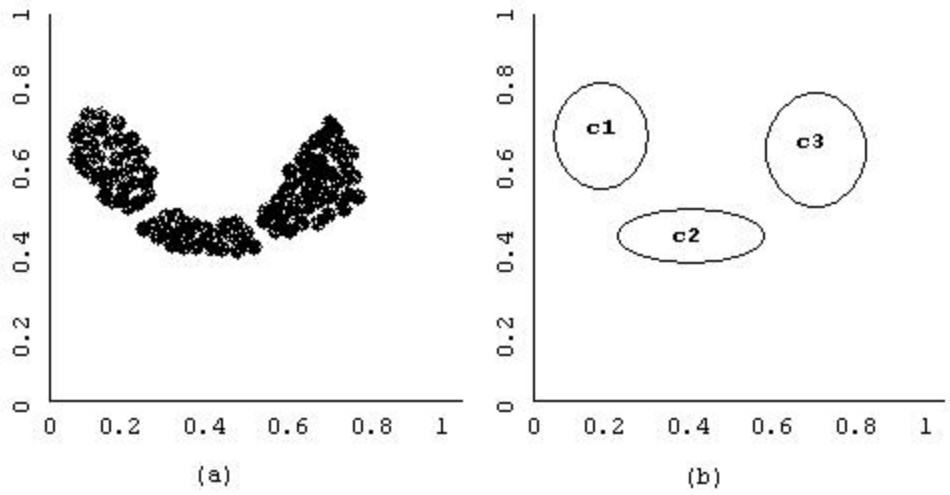

Figure 2.3: (a) data points (b) Gaussian mixture model

Clearly, if we knew which component generated each data point, we could just select all the data points from a given component and then applying the multivariate version of the formula— (2.4) and (2.5), we would fit the parameters of each component to the data. On the other hand, if we knew the parameters of each component, then we could at least in a probabilistic sense (Eq:2.9), assign each data point to a component. Unfortunately, we know neither the data assignments nor the parameters.

The basic idea of the unsupervised clustering algorithms is to pretend that we know the parameters of the model and then to infer the probability that each data point belongs to each component. Then, we refit the components to the data. The process iterates until convergence. Hence, we get the following steps for the unsupervised clustering algorithms:

- a set of models for the classes q_k (may not be Gaussian), defined by a parameter set Θ (means, variances, priors,...)
- a measure of membership, that measures to which extent a data point "belongs" to a model
- a way to update the model parameters as a function of the membership information.

The measure of membership either calculates the distance measure (k-means) or assign probability (EM). In the case of a distance measure, the models that are closer to the data characterize it better and in case of probability measure, the models with a larger likelihood for the data explain it better.

We have used k-means (hard clustering) and EM (soft clustering) unsupervised algorithms for clustering sentences according to their relatedness to the query in order to generate query-based summaries.

2.3.1 Number of Clusters in Unsupervised Learning

Both k-means and EM algorithms follow a simple and easy way to cluster a given data set through a pre-specified number of clusters k, therefore the problem of determining "the right number of clusters" has attracted considerable interest. There are several approaches (such as "iK-Means" by (Mirkin, 2005), Hartigan's method (Hartigan and Wong, 1979) etc.) to estimate the number of clusters. These methods may also give incorrect number of clusters.

However, in case of our problem, we simply assume that we have two clusters: 1. *Query-relevant cluster* that contains the sentences which are relevant to the user-questions, and 2. *Query-irrelevant cluster* that contains the sentences that are not relevant to the user-questions.

2.3.2 K-means Algorithm

K-means is a hard clustering algorithm that defines clusters by the center of mass of their members (Manning and Schutze, 2000). We start with a set of initial cluster centers and go through several iterations of assigning each object to the cluster whose center is closest.

After all objects have been assigned, we recompute the center of each cluster as the centroid or mean $\boldsymbol{\mu}$ of its members. The distance function is Euclidean distance.

In what follows we describe the k-means algorithm for estimating a mixture model following (Manning and Schutze, 2000):

Synopsis of the algorithm:

- Start with K initial prototypes $\boldsymbol{\mu}_k,\ k = 1, \cdots, K$.
- **Do:**
  1. For each data point $\boldsymbol{x}_n$, $n = 1, \cdots, N$, compute the squared Euclidean distance from the i^{th} prototype:

     $$d_k(\boldsymbol{x}_n) \ = \ \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2 = (\boldsymbol{x}_n - \boldsymbol{\mu}_k)^T (\boldsymbol{x}_n - \boldsymbol{\mu}_k)$$

     Note that, we have used squared Euclidean distance instead of true Euclidean distance. Since the square root is a monotonically growing function squared Euclidean distance has the same result as the true Euclidean distance but the computation overload is smaller when the square root is dropped.
  2. Assign each data point $\boldsymbol{x}_n$ to its closest prototype $\boldsymbol{\mu}_k$, i.e., assign $\boldsymbol{x}_n$ to the class q_k if

     $$d_k(\boldsymbol{x}_n) \ < \ d_l(\boldsymbol{x}_n), \forall l \neq k$$

  3. Replace each prototype with the mean of the data points assigned to the corresponding class;
  4. Go to 1.
- **Until:** no further change occurs.

The global criterion is:

$$J \ = \ \sum_{k=1}^{K} \sum_{\boldsymbol{x}_n \in q_k} d_k(\boldsymbol{x}_n)$$

which represents the total squared distance between the data and the models they belong to and k-means finds the local minimum.

Figure 2.4 and Figure 2.5 show an example of k-means applied to a sample of five clusters. The data-points are represented by randomly chosen 2 dimensional feature vec-

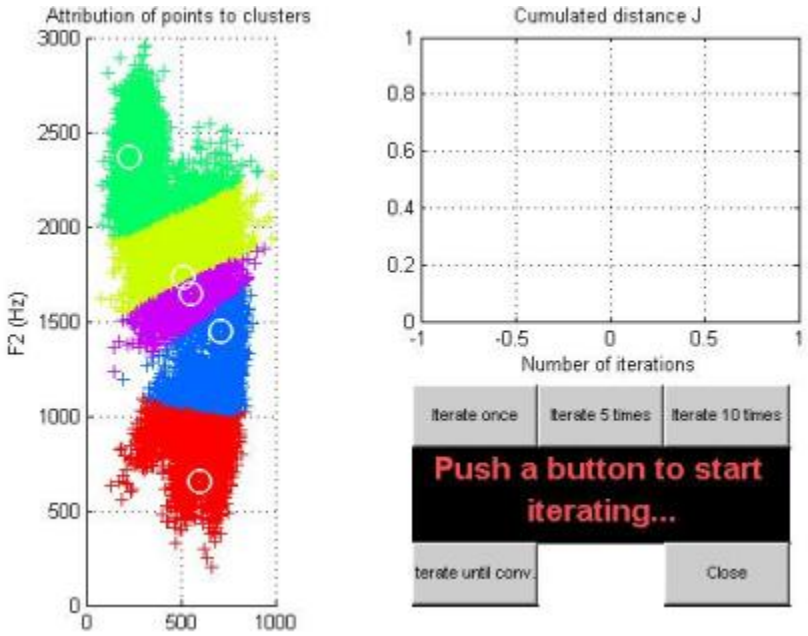

Figure 2.4: K-means initialization for five clusters

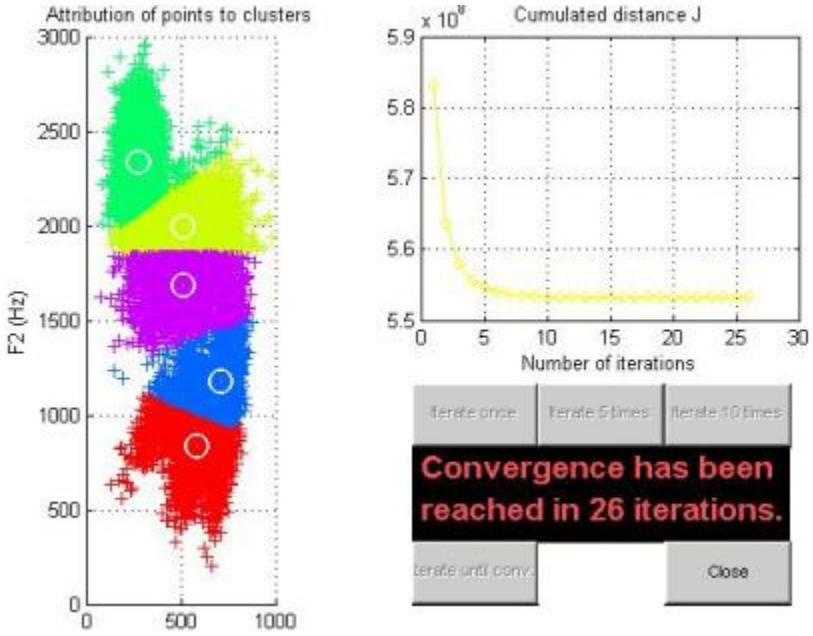

Figure 2.5: K-means convergence

tors. Figure 2.4 shows the initialization of clusters and Figure 2.5 shows the convergence situation after 26 iterations.

2.3.3 EM Algorithm for Gaussian Clustering

EM algorithm can be considered as a "soft" version of k-means algorithm described above (Manning and Schutze, 2000). As k-means, we start with a set of random cluster centers, c_1 and c_2 (figure 2.6). In k-means clustering, we would arrive at the final centers shown on the right side in one iteration. The EM instead does a soft assignment, which, for example, makes the lower right point mostly a member of c_2, but also partly a member of c_1. As a result, both cluster centers move towards the centroid of all three objects in the first iteration. Only after the second iteration we do reach the stable final state. EM finds the

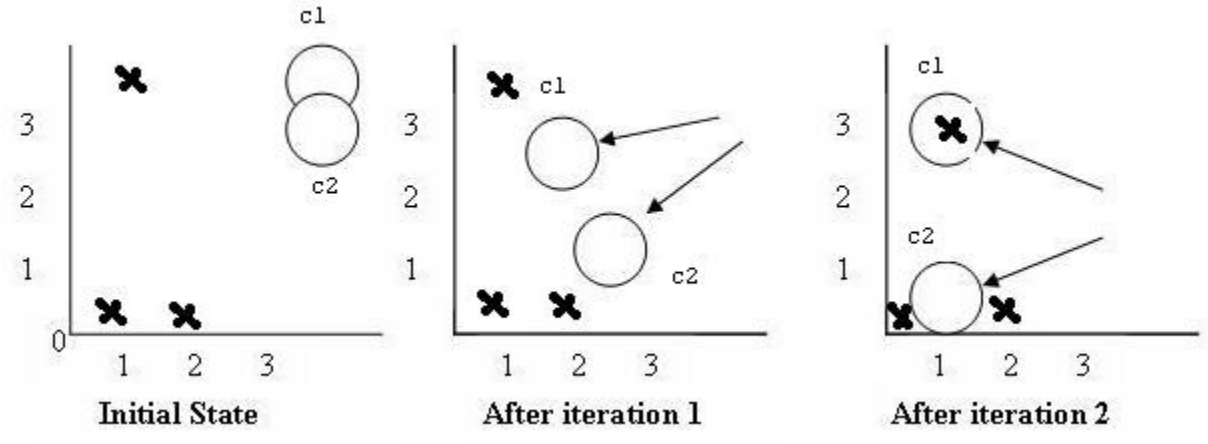

Figure 2.6: Soft clustering in EM

model which maximizes the likelihood of the data locally. Since, we are assuming that the data is generated by k-Gaussians, we wish to find the maximum likelihood model of the

form:

$$P(\boldsymbol{x}) = \sum_{i=1}^{k} P(C=i)P(\boldsymbol{x}|C=i) \quad (2.12)$$

$$= \sum_{i=1}^{k} P(C=i)P(\boldsymbol{x}|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) \quad (2.13)$$

In this model, we need to assume a weight $w_i = P(C=i)$ for each Gaussian, so that the integral of the combined Gaussians over the whole space is 1. We need to initialize the parameters: $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ so that, it finds a good local maximal.

EM algorithm is an iterative solution to the following circular statements (Manning and Schutze, 2000):

Estimate: If we knew the value of Θ (i.e. $w_i,\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i$), we could compute the expected values of the hidden structure (i.e. the probabilities to be in a cluster) of the model.

Maximization: If we knew the expected values of the hidden structure of the model, then we could compute the maximum likelihood value of Θ.

We break the circularity by beginning with a guess for Θ and iterating back and forth between an expectation and a maximization step, hence the name EM algorithm. In the expectation step, we compute expected values for the hidden variables $h_{i,j}$ which are cluster membership probabilities. Given the current parameters, we compute how likely it is that an object belongs to any of the clusters. The maximization step computes the most likely parameters of the model given the cluster membership probabilities.

In the following, we describe the EM algorithm for estimating Gaussian mixture following (Manning and Schutze, 2000).

Synopsis of the algorithm:

- Start with K initial Gaussian models: $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ $k = 1, \cdots, K$, with equal priors set to $P(q_k) = 1/K$.
- **Do:**
  1. **Estimation step:** compute the probability $P(q_k^{(i)}|\boldsymbol{x}_n, \Theta^{(i)})$ for each data point $\boldsymbol{x}_n$, $n = 1, \cdots, N$, to belong to the class $q_k^{(i)}$

$$P(q_k^{(i)}|\boldsymbol{x}_n, \Theta^{(i)}) = \frac{P(q_k^{(i)}|\Theta^{(i)})p(\boldsymbol{x}_n|q_k^{(i)}, \Theta^{(i)})}{p(\boldsymbol{x}_n|\Theta^{(i)})} = \frac{P(q_k^{(i)}|\Theta^{(i)})p(\boldsymbol{x}_n|\boldsymbol{\mu}_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)})}{\sum_j P(q_j^{(i)}|\Theta^{(i)})p(\boldsymbol{x}_n|\boldsymbol{\mu}_j^{(i)}, \boldsymbol{\Sigma}_j^{(i)})}$$

     This step gives a labeling of the data by telling to which extent a point $\boldsymbol{x}_n$ belongs to the class q_k. This represents a soft classification, since a point can belong, e.g., by 60% to class 1 and by 40% to class 2.
  2. **Maximization step:** - update the means:

$$\boldsymbol{\mu}_k^{i+1} = \frac{\sum_{n=1}^N \boldsymbol{x}_n P(q_k^{(i)}|\boldsymbol{x}_n, \Theta^{(i)})}{\sum_{n=1}^N P(q_k^{(i)}|\boldsymbol{x}_n, \Theta^{(i)})}$$

     - update the variances:

$$\boldsymbol{\Sigma}_k^{(i+1)} = \frac{\sum_{n=1}^N P(q_k^{(i)}|\boldsymbol{x}_n, \Theta^{(i)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(i+1)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(i+1)})^T}{\sum_{n=1}^N P(q_k^{(i)}|\boldsymbol{x}_n, \Theta^{(i)})}$$

     - update the priors:

$$P(q_k(i+1)|\Theta^{(i+1)}) = \frac{1}{N}\sum_{n=1}^N P(q_k^{(i)}|\boldsymbol{x}_n, \Theta^{(i)})$$

     Here, all the data points participate to the update of all the models, but their participation is weighted by the value of $P(q_k(i)|\Theta^{(i)})$.
  3. Go to 1.
- **Until:** the total likelihood increase for the training data falls under some desired threshold.

A key property of the EM algorithm is monotonicity: with each iteration of E and M steps, the likelihood of the model given the data increases. However, often it does not find the global maximum.

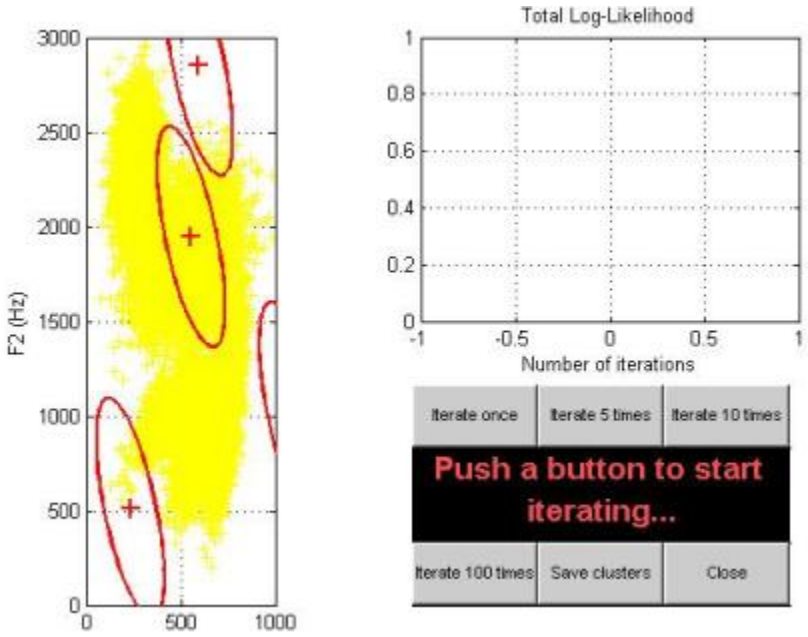

Figure 2.7: EM initialization for five clusters

The global criterion is the joint likelihood of all data with respect to all the models:

$$\begin{aligned}
\lambda(\Theta) &= log\ p(X|\Theta) \\
&= log \sum_{Q} p(X,Q|\Theta) \\
&= log \sum_{Q} P(Q|X,\Theta)p(X|\Theta) \quad (applying\ Bayes\ rule) \\
&= log \sum_{k=1}^{K} P(q_k|X,\Theta)p(X|\Theta)
\end{aligned}$$

Applying Jensen's inequality $\left(log \sum_j \lambda_j y_j \geq \sum_j \lambda_j\ log\ y_j\ \ if\ \sum_j \lambda_j = 1\right)$, we obtain:

$$\begin{aligned}
\lambda(\Theta) \geq J(\Theta) &= \sum_{k=1}^{K} P(q_k|X,\Theta) log\ p(X|\Theta) \\
&= \sum_{k=1}^{K} \sum_{n=1}^{N} P(q_k|\boldsymbol{x}_n,\Theta) log\ p(\boldsymbol{x}_n|\Theta)
\end{aligned}$$

Hence, the criterion $J(\Theta)$ represents a lower bound for $\lambda(\Theta)$ which is our desired auxiliary

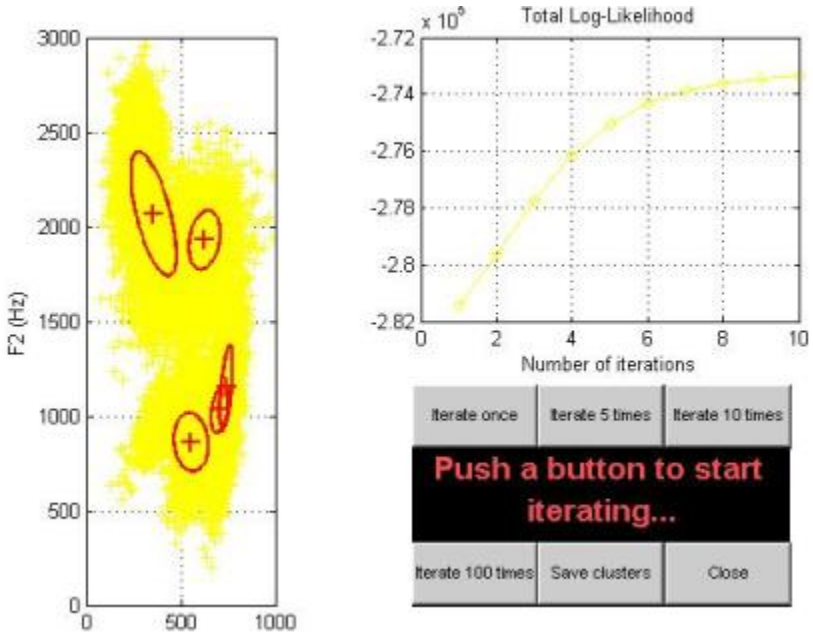

Figure 2.8: Clusters after 10 iterations

function. This criterion is locally maximized by the EM algorithm. Figure 2.7, Figure 2.8 and Figure 2.9 show examples of EM applied to a sample (10,000 data points) of five clusters as in k-means above. As in above, the data-points are represented by randomly chosen 2 dimensional feature vectors. Figure 2.7 shows the initialization of clusters by EM and Figure 2.8 shows the situation after 10 iterations. Note that, at each step, the total log-likelihood increases and after iteration 10 we have not got the convergence. Figure 2.9 shows the clusters after 110 iterations when we have got the convergence.

2.4 Information Retrieval

Question answering systems find the answer to a question in a collection of documents. In most cases, it is not feasible to process each document in a collection sequentially every time a new question is to be answered. An information retrieval system can be used to index the documents and allow a QA system to query the IR system, thus retrieving only

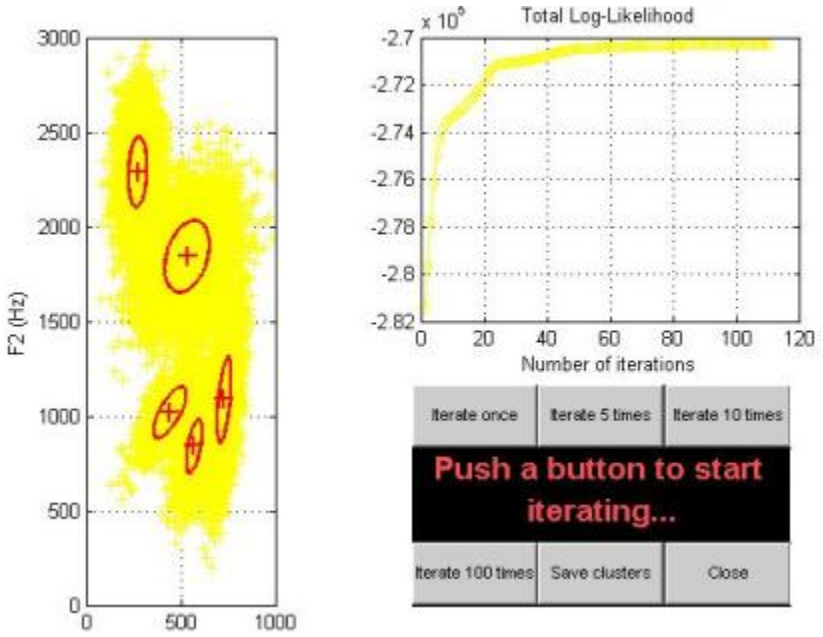

Figure 2.9: Clusters after 110 iterations

the documents that are relevant to the question.

2.4.1 Indexing Documents

Information retrieval system indexes all non-stop words in a document. The set of stop words is mostly made up of extremely common determiners, pronouns, and prepositions. Soem examples of stop words are: “the”, “that”, “an”, “by”, “at”, “from”, “how” etc. The IR systems exclude these stop words to save space or to speed up searches. For each word indexed, the number of occurrences of the word in each document is saved. To demonstrate how an inverted index stores words, here is an example with a collection that has 7 documents with a total of seven different words.

- Document D1 contains {W1, W2, W3}
- Document D2 contains {W1, W3, W4, W7}

W1	D1,D2,D5
W2	D1,D2,D4,D5,D7
W3	D1,D2,D4{2}
W4	D2,D4,D6{3}
W5	D3,D4,D5,D7{2}
W6	D4,D6
W7	D2,D3,D7

Table 2.1: Inverted file example

- Document D3 contains {W2, W5, W7}
- Document D4 contains {W2, W3, W3, W4, W5, W6}
- Document D5 contains {W1, W2, W5}
- Document D6 contains {W4, W4, W4, W6}
- Document D7 contains {W2, W5, W5, W7}

The inverted file entries for each word are shown in Table 2.1.

2.4.2 Queries

Once the documents are indexed, an information retrieval system allows you to retrieve documents relevant to a query. The two main ways to query the documents with an IR system are boolean and vector-space. The inverted file from Table 2.1 will be used as an example for the following sections.

Boolean

Boolean queries are made up of basic boolean operators: *AND*, *OR* and *NOT*. For instance, the query "*W4 AND W6*" will return the results of the intersect of the set of documents that contain W4 and the set of documents that contain W6. This query will retrieve documents D4 and D6.

Vector-Space

Vector-Space (Jurafsky and Martin, 2000, pages 647-651) queries, also called vector-cosine, are a way of ranking documents based on a query. In this ranking method, the documents and the query are represented as n-dimensional vectors with each dimension representing a word in the query. The rank of the document for the query will be the cosine of the angle between the query-vector and the document-vector.

The query of k terms will be represented as the vector $\boldsymbol{q}_k$, where $w_{i,k}$ is the weight of the i-th term in the query:

$$\boldsymbol{q}_k = (w_{1,k}, w_{2,k}, \ldots, w_{n,k})$$

Each document j will be represented as the vector $\boldsymbol{d}_j$, where $w_{i,j}$ represents the number of occurrences of term i in document j[1]:

$$\boldsymbol{d}_j = (w_{1,j}, w_{2,j}, \ldots, w_{n,j})$$

The formula for the vector cosine method for finding the rank is:

[1] $w_{i,j}$ may represent other measures such as $tf * idf$ of a word

$$cos\alpha = \frac{\boldsymbol{q}_k.\boldsymbol{d}_j}{|\boldsymbol{q}_k|\,|\boldsymbol{d}_j|} = \frac{\sum_{i=1}^{n} w_{i,j} \times w_{i,k}}{\sqrt{\sum_{i=1}^{n} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{n} w_{i,k}^2}}$$

where α is the angle between $\boldsymbol{q}_k$ and $\boldsymbol{d}_j$. Documents will not be considered if they do not contain any words from the query. This formula will yield the cosine of the angle between the query-vector and the document-vector. As an example, the documents will be ranked on the query *W3* and *W4* and *W6*. There are no *AND* and *OR* operations for a vector-space query. The inverted file in Table 2.1 will be used as the set of documents.

In this example, the occurrences of the terms will affect the weight. Since *W6* is only used two times, we will weight it more than the other terms and the query vector will be $\boldsymbol{q} = (1,1,2)$. The rank for each document will be calculated as follows:

$$\frac{\boldsymbol{q}.\boldsymbol{d}_1}{|\boldsymbol{q}|\,|\boldsymbol{d}_1|} = \frac{(1\times1)+(1\times0)+(2\times0)}{\sqrt{1+1+4}\times\sqrt{1+0+0}} = \frac{1}{\sqrt{6}} \approx 0.40824829$$

$$\frac{\boldsymbol{q}.\boldsymbol{d}_2}{|\boldsymbol{q}|\,|\boldsymbol{d}_2|} = \frac{(1\times1)+(1\times1)+(2\times0)}{\sqrt{1+1+4}\times\sqrt{1+1+0}} = \frac{2}{\sqrt{6}\times\sqrt{2}} \approx 0.57735027$$

$$\frac{\boldsymbol{q}.\boldsymbol{d}_4}{|\boldsymbol{q}|\,|\boldsymbol{d}_4|} = \frac{(1\times2)+(1\times1)+(2\times1)}{\sqrt{1+1+4}\times\sqrt{4+1+1}} = \frac{5}{\sqrt{6}\times\sqrt{6}} \approx 0.83333333$$

$$\frac{\boldsymbol{q}.\boldsymbol{d}_4}{|\boldsymbol{q}|\,|\boldsymbol{d}_6|} = \frac{(1\times0)+(1\times3)+(2\times1)}{\sqrt{1+1+4}\times\sqrt{0+9+1}} = \frac{5}{\sqrt{6}\times\sqrt{10}} \approx 0.64549722$$

The order in which the documents are returned is: D4, then D6, then D2, and then finally D1. Documents D3, D5 and D7 will not be considered because they do not contain any of the words of the query.

2.4.3 Lucene

We are using Java based Lucene[2] as our information retrieval system. Lucene is a group of tools that lets users create their own information retrieval system with the provided modules. Moldovan et al. (2004) notes that because of the Lucene system's ability to have a greater understanding of natural language, the passages retrieved can be more relevant to the query.

2.4.4 Case Folding and Stemming

Multiple words could be indexed as the same word in the inverted index. Case folding is when words that include upper case letters are indexed to the same entry as words without upper case letters. In stemming the words are indexed by their stems. A stem of the word is the part left after the affixes have been taken off. Affixes can take different forms and are letters that are used to modify the base form of a word. An example of this are the suffixes: "ly", "er", "ess", "ed", and "ing".

Lucene allows us to index the documents in a case folding and stemmed manner. Thus "World" would be indexed as the same word as "world" and "bank" will be indexed the same as the word "banking". This is useful because it will find more words that are related to the query term, but presents similar problems to case insensitivity. For stemming we are using the Porter Stemmer (Porter, 1980) in Lucene.

[2] http://jkarta.apache.org/lucene/

2.4.5 Paragraph Indexing

We indexed paragraphs instead of documents. A paragraph is a complete thought and most pronouns in a paragraph will represent the named entity that is already contained in the paragraph. A paragraph that does not explicitly contain the topic will rarely involve the topic. Indexing by paragraph helps us exclude the text that is unlikely to have the answer, hence minimizes the processing time for finding an answer. Harabagiu and Maiorano (1999) found that there is an increase in accuracy if documents are indexed by paragraph.

2.5 Chapter Summary

In this chapter, we described the mathematical theories and tools on which we built our QA systems. In the next chapter, we describe how the documents are preprocessed before extracting answers to the questions.

Chapter 3

Document Processing

Our *question answering* systems require different kinds of preprocessing of the documents. In this chapter, we give a detailed description of the different tags and parses that were done and the tools that were used to do the tagging and parsing [1].

We utilized different outside systems to aid in tagging documents including:

- WordNet (http://wordnet.princeton.edu/)
- Minipar (http://www.cs.ualberta.ca/ lindek/minipar.htm)
- OAK System (http://nlp.cs.nyu.edu/oak/)
- Lingpipe (http://www.alias-i.com/lingpipe/)
- Basic Element (BE) extractor (http://www.isi.edu/cyl/BE)
- Charniak Parser (http://www.cs.brown.edu/people/ec/software)
- ASSERT semantic role labeling system (http://cemantix.org/assert)

In preprocessing, we added different tags to the document sentences in order to add useful information to the words. We also did different kinds of parsing of the document sentences. The tags and parses that we did include:

- Tokenization and sentence splitting
- Part of speech tagging

[1] The examples in this chapter are taken from DUC 2007.

- Chunked part of speech tagging
- Word sense tagging
- Named entity tagging
- Co-reference resolution
- Basic element extraction
- Word dependency tagging
- Shallow Semantic parsing

First, we give an overview of the three most important tools our systems use to tag/parse sentences. Then, the sections describe how we used those tools to tag specific information to the sentences.

3.1 Overview of Selected Tools

3.1.1 WordNet 3.0

WordNet is a semantic lexicon for the English language. It groups English words (i.e. nouns, verbs, adjectives and adverbs) into sets of synonyms called synsets, provides short, general definitions (i.e. gloss definition), and records the various semantic relations between these synonym sets. Every synset contains a group of synonymous words or collocations (a collocation is a sequence of words that go together to form a specific meaning, such as ”car pool”); different senses of a word are in different synsets. The meaning of the synsets is further clarified with short defining glosses (definitions and/or examples). For

example, the noun *computer* has two senses and the synset and gloss definition for each of them are:

03082979 computer, computing machine, computing device, data processor, electronic computer, information processing system (a machine for performing calculations automatically)

09887034 calculator, reckoner, figurer, estimator, computer (an expert at calculation (or at operating calculating machines))

Each synonym set is identified by a *synset ID* (e.g. 03082979, 09887034). Most synsets are connected to other synsets via a number of semantic relations. The semantic relations include:

For nouns:

Hypernyms: Y is a hypernym of X if every X is a (kind of) Y (*pen* to *element*)

Hyponyms: Y is a hyponym of X if every Y is a (kind of) X (*pen* to *ballpoint pen*)

Coordinate terms: Y is a coordinate term of X if X and Y share a hypernym (*pen* and *marker*)

Holonym: Y is a holonym of X if X is a part of Y (*computer* to *platform*)

Meronym: Y is a meronym of X if Y is a part of X (*computer* to *CRT*)

For verbs:

Hypernym: the verb Y is a hypernym of the verb X if the activity X is a (kind of) Y (travel to movement)

Troponym: the verb Y is a troponym of the verb X if the activity Y is doing X in some manner (lisp to talk)

Entailment: the verb Y is entailed by X if by doing X you must be doing Y (sleeping by snoring)

Coordinate terms: those verbs sharing a common hypernym (*travel* and *journey*)

For adjectives:

Related nouns

Participle of verb

For adverbs:

Root adjectives

One useful relation is *hyponym/hypernym*. A *hyponym* for a word is a set of things that are instances of that word. For example, the hyponym set for sense one of *computer* includes different types of computers such as: *analog computer*, *digital computer*, *home computer*, *node, client, guest*, *number cruncher* etc. A *hypernym* set is the opposite of a hyponym set. Sense one of computer has a hypernym set of *machine, device, instrumentality, artifact,* etc. With hypernym sets and hyponym sets a hierarchy is formed, with hyponym set being more specific and hypernym being more general. A part of the hierarchy tree formed by these relations for first sense of *computer* is in Figure 3.1.

3.1.2 OAK System

OAK System (Sekine, 2002), from New York University, is a collection of tools that can tag documents in many different ways:

- Sentence splitter

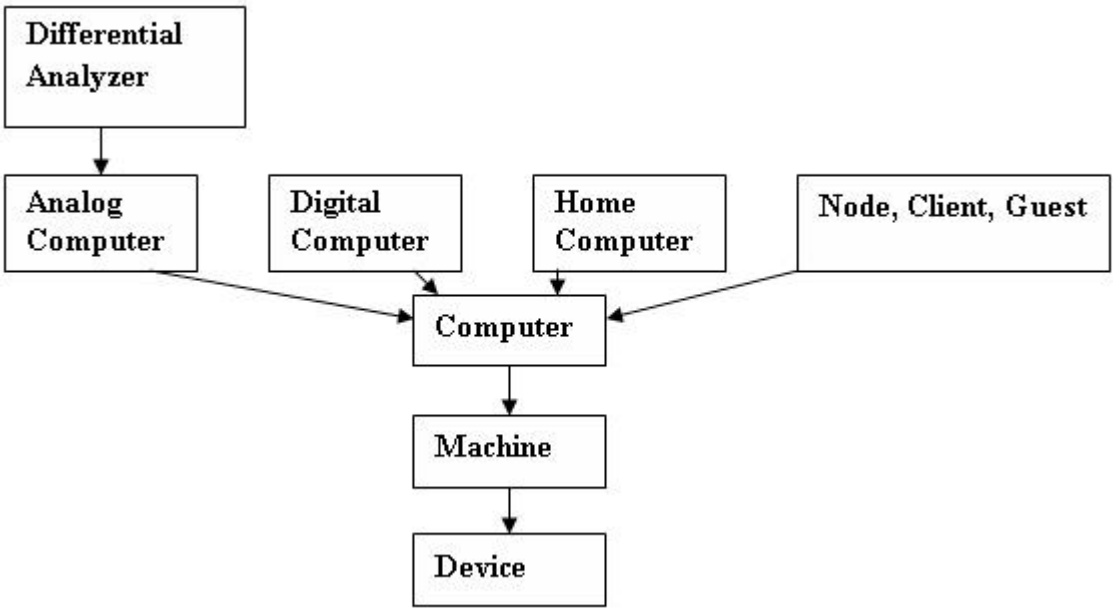

Figure 3.1: Hierarchy tree for computer

- Tokenizer
- Part of speech tagger
- Chunker
- Named entity tagger

3.1.3 Lingpipe

Lingpipe[2] developed by Alias-i, Inc [3] can tag documents in the following ways:

- Tokenizer

[2]http://www.alias-i.com/lingpipe/
[3]http://www.alias-i.com/

- Sentence splitter
- Named entity tagger
- Co-reference resolution

3.2 Tagging Documents

3.2.1 Tokenization and Sentence Splitting

Before text is tokenized and split into sentences, it is just a string of characters. Tokenization is splitting a string of characters into lexical elements such as words and punctuation. Sentence splitting separates the sentences (Jurafsky and Martin, 2000, page 180). We used OAK system for this purpose.

3.2.2 Co-reference Resolution

Knowing which terms are referring to which entities is very helpful in answering questions (Ageno et al., 2004). There are different tasks when resolving the references in a document. A person can be refered to by his full name, first name and last name or by a pronoun (pronoun resolution). For example, in the following passage, *Angelina Jolie* is referred to by her last name *Jolie* and pronouns *She, her*.

> *Angelina Jolie* lives on the edge. Jolie, 25, delights in making waves. *She* is the woman who literally jumped in a swimming pool in *her* ballgown after winning a Golden Globe Award for "Gia" (1998).

Being able to resolve the references could help the QA system discover that *Angelina Jolie's age is 25* again the pronoun resolution helps our systems to deduct that "Angelina Jolie" jumped in a swimming pool after winning a Golden Glove award for "Gia" in 1998.

One problem that arises is that not every instance of someone's name represents that person (Dubien, 2005). A person's name can be referring to one of the things they have made, as in this example:

> Even a minor Rothko on paper, not to mention a *Picasso* or a Miro, *generally tops $250,000*.

In this example, *generally tops $250,000* is not a fact about *Picasso*, but rather a fact about his paintings.

Co-reference Resolution with Lingpipe

Lingpipe resolves references to the named entities; person, location and organization. It gives each of these named entities (person, location or organization) in the document an ID and each reference to that entity will have the same ID. For example the above passage got tagged with Lingpipe:

⟨s i="0"⟩⟨ ENAMEX ID="0" TYPE="PERSON'⟩ Angelina Jolie ⟨/ENAMEX⟩lives on the edge. ⟨/s ⟩ ⟨ s i="0"⟩⟨ ENAMEX ID="0"TYPE="PERSON"⟩ Jolie ⟨/ENAMEX⟩, 25, delights in making ⟨ ENAMEX ID="1" TYPE="LOCATION"⟩ waves.⟨/ENAMEX⟩⟨/s ⟩⟨ s i="1"⟩⟨ENAMEX ID="0" TYPE="FEMALE_PRONOUN"⟩She⟨/ENAMEX⟩ is the woman who literally jumped in a swimming pool in ⟨ ENAMEX ID="0" TYPE="FEMALE _PRONOUN"⟩ her ⟨/ENAMEX⟩ ballgown after winning a ⟨ ENAMEX ID="2" TYPE= "PERSON"⟩ Golden Globe Award ⟨/ENAMEX⟩ for "⟨ ENAMEX ID="3" TYPE= "ORGANIZATION"⟩ Gia⟨/ENAMEX⟩" (1998).⟨/s⟩

In this passage, *Angelina Jolie* is given the reference ID *0*. Lingpipe resolves the pronouns *she* and *her* as a reference to *Anjelina Jolie*. It resolves also the last name *Jolie* as Anjelina Jolie. But, sometimes it fails to resolve the last name as well as pronouns.

Our system uses Lingpipe only for pronoun resolution. Before documents are tagged by OAK system, our system tags documents with Lingpipe and replace all pronouns with the entity they are representing.

The above passage will be changed to:

> Angelina Jolie lives on the edge. Jolie, 25, delights in making waves. *Angelina Jolie* is the woman who literally jumped in a swimming pool in *Angelina Jolie's* ballgown after winning a Golden Globe Award for "Gia" (1998).

Both our summarizing and question answering systems benefit from *co-reference resolution*. It helps our summarizing system find the similarity between the question and candidate sentence and rank accordingly. It helps our QA system when possible answers will be matched to patterns with entities from the question.

3.2.3 Stemming with OAK Systems

Stemming is the process for reducing inflected (or sometimes derived) words to their stem or root form. So, a stem of a word is the part left after the affixes have been taken off. Affixes can take different forms and are letters that are used to modify the base form of a word. Examples of these are the suffixes; "en", "ly", "er", "ess", "ed", and "ing". A stemming algorithm (for example, Porter stemmer) reduces the words "fishing", "fished", "fish", and "fisher" to the root word, "fish". We used the OAK system for stemming.

For example :

Input Passage Angelina Jolie lives on the edge. Jolie, 25, delights in making waves. She is the woman who literally jumped in a swimming pool in her ballgown after winning a Golden Globe Award for “Gia” (1998).

Stemmed Passage Angelina Jolie life on the edge. Jolie, 25, delight in make wave. She be the woman who literally jump in a swimming pool in her ballgown after win a Golden Globe Award for “Gia” (1998).

3.2.4 Part of Speech

Each word in a sentence is classified as a Part Of Speech (POS) that depends on the way the word is being used. For instance, the word *mail* can be used as a noun (*Did you receive that mail I sent you?*) or as a verb (*Could you mail me that book?*). The systems available can tag documents with part of speech with fairly high accuracy (almost 95%).

To be consistent, systems use sets of universal tags for parts of speech. Penn Treebank POS tag set (Marcus, Santorini, and Marcinkiewicz, 1994) is the most popular one and we used it because this Treebank was used to train the OAK system. Tables 3.1 and 3.2 include a list of the Penn Treebank tags and examples of words that can be tagged with them.

For example,

Input Passage Angelina Jolie lives on the edge. Jolie, 25, delights in making waves. She is the woman who literally jumped in a swimming pool in her ballgown after winning a Golden Globe Award for “Gia” (1998).

Stemmed POS tagged Passage Angelina/NNP Jolie/NNP life/NN on/IN the/DT edge/NN.

Tag	Description	Example
CC	Coordinating conjunction	*and, but, or*
CD	Cardinal number	*1, 2, two, 44*
DT	Determiner	*a, the, an*
EX	Existential *there*	*there*
FW	Foreign word	*moi, coupe, carpe*
IN	Preposition/subord. conjunction	*in, on, by*
JJ	Adjective	*red, mean, good*
JJR	Adjective, comparative	*faster, closer, taller*
JJS	Adjective, superlative	*fastest, closest*
LS	List item maker	*3, 1, Two*
MD	Modal	*should, can, may*
NN	Noun, singular or mass	*frog, dog, lamp*
NNS	Noun, plural	*frogs, dogs, lamps*
NNP	Proper noun, singular	*CNN, Mary*
NNPS	Proper noun, plural	*Carolinas*
PDT	Predeterminer	*all, both*
POS	Possessive ending	*'s*
PRP	Personal pronoun	*I, she, you*
PP$	Possessive pronoun	*their, your*
RB	Adverb	*slowly, never*
RBR	Adverb, comparative	*slower*
RBS	Adverb, superlative	*slowest*
RP	Particle	*up, off*
SYM	Symbol (mathematical or scientific)	*+,%*
TO	*to*	*to*
UH	Interjection	*um, ah, oops*
VB	Verb, base form	*sit*
VBD	Verb, past tense	*sat*
VBG	Verb, gerund/present participle	*sitting*
VBN	Verb, past participle	*sat*
VBP	Verb, non-3rd ps. sing. present	*sit*
VPZ	Verb, 3rd ps. sing. present	*sits*
WDT	*wh*-determiner	*which, that*
WP	*wh*-pronoun	*what, who*
WP$	Possessive *wh*-pronoun	*whose*
WRB	*wh*-adverb	*how, where*

Table 3.1: Penn Treebank POS tagset

Tag	Description	Example
#	Pound sign	#
$	Dollar sign	$
.	Sentence-final punctuation	.?!
,	Comma	,
:	Colon, semi-colon	; : ...
(	Left bracket character	
)	Right bracket character	
	Straight double quote	
‘	Left open single quote	‘
“	Left open double quote	“
’	Right open single quote	’
”	Right close double quote	”

Table 3.2: Penn Treebank POS tagset continued

Jolie/NNP, 25/CD, delight/NN in/IN make/VB wave/NN. She/PRP be/VB the/DT woman/NN who/WP literally/RB jump/VBP in/IN a/DT swimming/NN pool/NN in/NN her/PRP ballgown/NN after/IN win/NN a/DT Golden/NNP Globe/NNP Award/NNP for/IN “/“ Gia/NNP ”/” (/-LRB- 1998/CD)/-RRB- ./.

The two most popular POS taggers are the maximum entropy tagger (Ratnaparkhi, 1996) and the Brill tagger (Brill, 1994). The *maximum entropy* tagger uses probabilities to tag the document with the set of tags that are most likely to be correct. These probabilities are learned through supervised machine learning techniques. This tagger uses a decision tree to see all the possible tags for a sentence and finds the most probable tagging of the sentence as a whole.

The *Brill* tagger, also known as transformation-based tagging (Jurafsky and Martin, 2000, page 118), uses supervised machine learning as well, but learns sets of rules instead of probabilities. First it tags each word with the most probable tag for that word, and then it goes over the passage, applying the most probable set of rules for the situations.

For POS tagging, we are using the OAK System which uses a method similar to the

Brill tagger, but has 13% fewer errors (Sekine, 2002).

3.2.5 Chunked Part of Speech

Chunked part of speech is grouping words into different phrases. It is also referred to as a shallow parse since it is done with one pass (Jurafsky and Martin, 2000). Ramshaw and Marcus (1995) outline a transformation-based way of tagging chunked part of speech. This machine learning method learns rules for whether a word belongs in a noun phrase, verb phrase, or preposition phrase, given the part of speech tags of the word and the words that are already in these types of phrases.

For example,

Input Passage Angelina Jolie lives on the edge. Jolie, 25, delights in making waves. She is the woman who literally jumped in a swimming pool in her ballgown after winning a Golden Globe Award for “Gia” (1998).

Stemmed Chunked POS tagged Passage [NP Angelina/NNP Jolie/NNP life/NN] [PP on/IN] [NP the/DT edge/NN] ./. [NP Jolie/NNP] ,/, [NP 25/CD] ,/, [NP delight/NN] [PP in/IN] [VP make/VB] [NP wave/NN] ./. [NP She/PRP] [VP be/VB] [NP the/DT woman/NN] [NP who/WP] [ADVP literally/RB] [VP jump/VBP] [PP in/IN] [NP a/DT swimming/NN pool/NN] [PP in/NN] [NP her/PRP] [NP ballgown/NN] [PP after/IN] [NP win/NN] [NP a/DT Golden/NNP Globe/NNP Award/NNP] [PP for/IN] “/“ [NP Gia/NNP] ”/” (/-LRB- [NP 1998/CD])/-RRB- ./.

(Li and Roth, 2001) used this shallow parsing for question answering, instead of a deeper syntactic parse. They found that in certain situations, such as when lower quality text is used for extracting answers, a system using a shallow parse can be more effective and flexible at answering the questions. An example of lower quality text is when the text

was not edited for spelling and grammar. Our document collection contains both lower quality (BLOG06) and high quality texts (Newswire), so both shallow parse and syntactic parse are beneficial.

OAK performs a shallow parse of a document using chunked POS with a method similar to (Ramshaw and Marcus, 1995).

3.2.6 Named Entity Tagging

Named Entities (NE) are defined as terms that refer to a certain entity. For instance, *Canada* refers to a certain *country*, and *$200* refers to a certain quantity of money. The different classes of named entities our system recognizes are dependent on which named entities get tagged by the named entity tagger.

Our QA system used named entity tagging in order to extract the candidate answers. When our system classifies questions, it also discovers the answer type of the question. This answer type is often a named entity, and if that named entity is tagged, possible answers will be easier to extract with the answer extractor.

There are many ways to tag different named entities. One involves pattern matching from a list of examples of the entity. For instance, the entity *city* can be tagged by comparing an entity to a list of all cities. This will require some kind of look up table because running through the whole list for every new word will be time consuming (Dubien, 2005).

Another method of tagging certain named entities is by pattern matching using regular expressions. Where the named entity is a quantity (e.g. percentage), the pattern "*number%*" or "*number percent*" could be used to discover entities that are percentages.

Named Entity tagging using OAK System

We used OAK system to tag the documents with named entity. OAK system has 150 named entities that can be tagged. They are included in a hierarchy. They are found either by pattern matching from a list of examples of entities or by regular expressions. For instance, *city* is best found with a list of cities, since that is almost the only way to tell if an entity is referring to a city. Appendix C outlines all the 150 named entities that OAK system currently tags.

Since OAK tags all entities at the same time, some entities that can be more than one thing will only be classified as one of them. For instance, *Paris* could be the name of a *city* or the name of a *person.* OAK will only tag it as a *person.* As a result, our system cannot answer the question that asks for "Paris" as city.

An example of a chunked part of speech and named entity tagged sentence is:

> [NP The/DT shuttle/NN] [NP ⟨SPACESHIP Discovery/NNP ⟩] [VP is/VBZ scheduled/VBN to/TO dock/VB] [PP with/IN] [NP ⟨SPACESHIP Mir/NNP ⟩] ⟨DATE later/RB [NP this/DT week/NN ⟩] and/CC [VP retrieve/VB] [NP astronaut/NN ⟨MALE_FIRSTNAME Andrew/NNP ⟩ ⟨MALE_FIRSTNAME Thomas/NNP ⟩] ,/, [NP the/DT last/JJ] [PP of/IN] [NP seven/CD] [NP ⟨NATIONALITY American/NNP ⟩] [VP to/TO work/VB] [PP on/IN] [NP the/DT station/NN] [PP in/IN] [NP the/DT ⟨YEAR_PERIOD last/JJ three/CD years/NNS ⟩] ./.

Named Entity tagging with WordNet

As discussed previously, WordNet has sets for each synset called *hyponyms* which are words that are instances of that word. The hyponym set for computer has different kinds

of *computers*. This hyponym list can be loaded as a list of examples and a system can be made to tag these examples inside the document, similar to how OAK system tags NEs.

To use this method, our QA system should first know what word's hyponym list is going to be used to tag the document. Knowing the NE our system is looking for will allow our QA system to tag only the entities relevant to the question. These entities will be discovered with question classification module of our QA system. The hyponym list is then extracted and compared to the words of the document, and the entities that are in the list are tagged as possible answers.

3.2.7 Word Sense Tagging

Each word in WordNet has multiple senses for the different ways the word can be used. Word Sense Disambiguation (WSD) is the process of determining in which sense a word is being used. Once the system knows the correct sense that the word is being used, WordNet can be used to determine synonyms. This is useful for seeing if a word from the question is associated with a word in the passage by being in the same synonym set. To tag the correct sense we developed a WSD system (Chali and Joty, 2007c) and participated in SemEval 2007 WSD competition. Our system achieved encouraging result (60% accuracy) in SemEval-2007.

Our system creates a list of nouns in the passage. Each candidate word is expanded to all of its senses. We created a hash representation to identify all possible word representations, motivated from (Galley and McKeown, 2003). Each word sense is inserted into the hash entry having the index value equal to its synsetID. For example, athlete and jock are inserted into the same hash entry (Fig 3.2).

On insertion of the candidate sense into the hash we check to see if there exists an entry into the index value, with which the current word sense has one of the relations: 1) repeti-

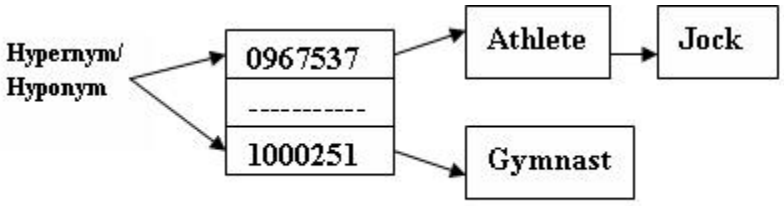

Figure 3.2: Hash indexed by synsetID

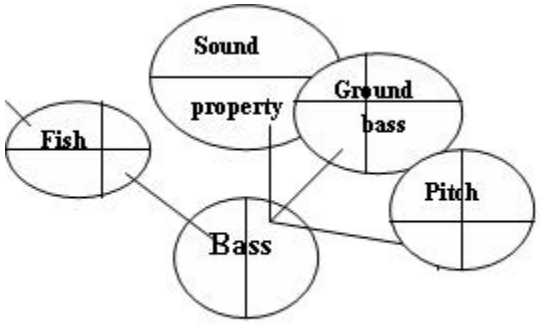

Figure 3.3: Partial disambiguation graph

tion, 2) synonym, 3) hypernym/hyponym and 4) gloss. No disambiguation is done at this point; the only purpose is to build a representation used in the next stage of the algorithm. This representation can be shown as a disambiguation graph (Galley and McKeown, 2003) where the nodes represent word instances with their WordNet senses and weighted edges connecting the senses of two different words represent semantic relations (Fig 3.3).

We use the intermediate representation (Fig 3.3) to perform the WSD. We add the weight of all edges leaving the nodes under their different senses. The one sense with the highest score is considered the most probable sense. For example, in Fig 3.3 bass is connected with three words: pitch, ground bass and sound property by its instrument sense and with one word: fish by its food sense. For this specific example all the semantic rela-

tions are of hyponym/hypernym type (score 0.33). So we get the score as in Table 3.3.

Sense	Mnemonic	Score	Disambiguated Sense
4928349	Musical Instrument	$3 \times 0.33 = 0.99$	Musical Instrument (4928349)
7672239	Fish or Food	0.33	

Table 3.3: Score of the senses

In case of tie between two or more senses, we select the one sense that comes first in WordNet, since WordNet orders the senses of a word by decreasing order of frequency.

3.2.8 Lexical Chain Extraction

Concepts of coherence and cohesion enable us to capture the theme of the text. Coherence represents the overall structure of a multi-sentence text in terms of macro-level relations between clauses or sentences (Halliday and Hasan, 1976). Cohesion, as defined by (Halliday and Hasan, 1976), is the property of holding text together as one single grammatical unit based on relations (i.e. ellipsis, conjunction, substitution, reference, and lexical cohesion) between various elements of the text. Lexical cohesion is defined as the cohesion that arises from the semantic relations between the words in the text (Morris and Hirst, 1991). Lexical cohesion among words are represented by lexical chains.

Several methods have been proposed to compute lexical chains (Barzilay and Elhadad, 1997), (Hirst and St-Onge, 1997), (Silber and McCoy, 2002), (Galley and McKeown, 2003). We followed the similar method as (Galley and McKeown, 2003), in which in the first step we perform the word sense disambiguation using the disambiguation graph (Section 3.2.7), then we form the lexical chains in the second step.

In the previous section (Section 3.2.7) we described the word sense disambiguation step using the disambiguation graph. At this point, we have already assigned a sense to

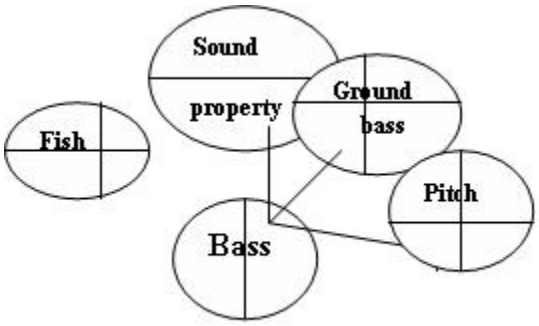

Figure 3.4: Lexical chain graph

each word. The next step is to build the actual lexical chains by processing the entire disambiguation graph. In this step, we remove all semantic links that connect words taken under wrong senses and keep only the links that connect the words with their right senses. We are left with the edges that are the actual lexical chains of our algorithm. For example, in Figure 3.3 the word bass's disambiguated sense is "music instrument" (4928349). If the final sense of the words *pitch, ground bass* and *sound property* is the sense by which it is connected with *bass* then we get the *chain graph* as Figure 3.4 which represents the lexical chain: (bass, pitch, ground bass, sound property).

In our QA system, we used this to extract the number of word associations between the question and the passage with candidate answer and use this in order to rank the candidate answers. We extracted the lexical chains from the documents for the text summarization system with which we participated in DUC 2007 (Chali and Joty, 2007a). We used lexical chain as one of the features for ranking a sentence to be included in the summary.

3.2.9 BE Extraction

Basic Elements (BEs) are defined as follows (Hovy et al., 2006):

- the head of a major syntactic constituent (noun, verb, adjective or adverbial phrases), expressed as a single item, or
- a relation between a head-BE and a single dependent, expressed as a triple: (head|modifier|relation).

With BE represented as a head-modifier-relation triple, one can quite easily decide whether any two units match or not- considerably more easily than with longer units. For instance, "United Nations", "UN", and "UNO" can be matched at this level (Hovy et al., 2006). Example BEs for the sentence "The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings" are as follows:

body|frankfurt-based|nn
said|body|subj
said|in|guest
said|its|in
report|annual|mod
said|released|fc
released|report|subj
released|today|mod
decided|it|subj
decided|on|guest
decided|two|on
decided|themes|obj
currency|new|mod
themes|currency|for

civilization|european|mod

history|civilization|of

civilization|abstract|conj

paintings|concrete|mod

abstract|paintings|conj

In order to extract the BEs, a syntactic parser (Minipar) is used to produce a parse tree and then a set of "cutting rules" are used to extract just the valid BEs from the tree. The BEs shown above are generated by BE package 1.0 distributed by ISI[4]. We used the standard BE-F breaker included in the BE package.

We might be hitting a limit imposed by the representation of sentences and queries which ignores syntax or semantics, so including these two features might be helpful. Moreover, the task like *query-based summarization* that requires the use of more complex syntactic and semantics, the approaches with only Bag of Words (BOW) are often inadequate to perform fine-level textual analysis. In the next two subsections, we discuss the syntactic and semantic parsing.

3.2.10 Syntactic Parsing: Word Dependencies Tagging

To discover word dependencies, a syntactic parse is used. Syntactic parsing is analyzing a sentence using the grammar rules. One method to tag word dependencies is by using the Charniak parser [5] to get a statistical syntactic parse of the passages. These probabilities are found using supervised machine learning. The probability is the chance that two words are dependent, given certain variables including part of speech and distance.

[4]BE website:http://www.isi.edu/ cyl/BE

[5]available at ftp://ftp.cs.brown.edu/pub/nlparser/

The following is an example of a sentence parsed with the Charniak parser:

```
(S1 (S (NP (NNP Angelina) (NNP Jolie))
     (VP (AUX is)
      (NP (NP (DT the) (NN woman))
       (SBAR (WHNP (WP who))
       (S (VP (ADVP (RB literally))
            (VBD jumped)
            (PP (IN in)
              (NP (NP (DT a) (VBG swimming) (NN pool))
                (PP (IN in) (NP (PRP$ her) (NN ballgown)))))
            (PP (IN after)
              (S (VP (VBG winning)
                  (NP (DT a) (NNP Golden) (NNP Globe) (NN Award))
                  (PP (IN for)
                    (NP (NP (`` ``) (NNP Gia\) ('' '') (NNP \) (POS '))
                      (PRN (-LRB- -LRB-) (NP (CD 1998)) (-RRB- -RRB-)))))
                       )))))
     (. .)))
```

(Pasca and Harabagiu, 2001) demonstrated that with the syntactic form one can see which words depend on other words. There should be a similarity between the words that are dependent in the question and the dependency between words of the passage containing the answer. The importance of syntactic feature in *question answering* is described by (Zhang and Lee, 2003), (Moschitti et al., 2007) and (Moschitti and Basili, 2006).

In our text summarizer system, we used the Charniak parser to parse the sentence as well as the query into syntactic trees before finding the syntactic similarity between the

query and the sentence.

3.2.11 Semantic Parsing: Semantic Role Labeling

The study of shallow semantic information such as predicate argument structures annotated in the PropBank(PB) project (Kingsbury and Palmer, 2002) is a promising research direction (Moschitti et al., 2007). To experiment with semantic structures, we parse the sentences as well as the questions semantically using a Semantic Role Labeling (SRL) system like ASSERT [6].

ASSERT is an automatic statistical semantic role tagger, that can annotate naturally occuring text with semantic arguments. When presented with a sentence, it performs a full syntactic analysis of the sentence, automatically identifies all the verb predicates in that sentence, extracts features for all constituents in the parse tree relative to the predicate, and identifies and tags the constituents with the appropriate semantic arguments.

For example, the output of the SRL system for the sentence "The report also said Duisenberg expects the future relationship between the dollar and the euro, which officially goes into effect on Jan. 12, to be stable.":

```
1) [ARG0 the report] [ARGM-DIS also] [TARGET said] [ARG1 Duisenberg
expects the future relationship between the dollar and the euro which
officially goes into effect on Jan. 12, to be stable]

2) the report also said [ARG0 Duisenberg] [TARGET expects] [ARG1 the
future relationship between the dollar and the euro which officially
goes into effect on Jan. 12, to be stable]
```

[6] available at http://cemantix.org/assert

The example sentence contains two verbs (predicates) with their arguments. The main predicate is “said”, followed by a subordinate predicate “expects”. The subordinate predicate is always a part of one of the arguments of the main predicate.

3.3 Chapter Summary

In this chapter, we discussed the tags that we need in our QA systems and the tools that were used to perform the required tagging. We used these tags to extract useful information from the documents and also to find the similarity between a document-sentence and a query-sentence. In the next chapter, we discuss our method for answering simple questions.

Chapter 4

Answering Simple Questions

4.1 Introduction

4.1.1 Problem Definition

When a user is served with a ranked list of relevant documents by the standard document retrieval systems (i.e. search engines), his search task is usually not over. The next step for him is to look into the documents in search for the precise piece of information he was looking for. This method is time consuming, and a correct answer could easily be missed, by either an incorrect query resulting in missing documents or by careless reading. *Question answering* tries to remove the onus on the end-user, by providing more direct access to the relevant information.

Given a collection of documents (such as the World Wide Web or a local collection) the QA system should be able to retrieve answers to questions posed in natural language. QA is regarded as requiring more complex natural language processing techniques than other types of information retrieval such as document retrieval, and it is sometimes regarded as the next step beyond search engines.

While information retrieval is workable, users now demand better tools. It is because, they want to cut time and effort involved in formulating effective queries and then they want their results to be real answers—not just the list of relevant links. They want to spend less time in searching and more time in 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. For example, given the

question “Who won the Nobel prize in peace in 2006?” what a user really wants is the answer “Dr. Muhammad Yunus”, instead of reading through lots of documents that contain the words “win”, “nobel”,“prize”, “peace” and “2006” etc.

As introduced in Chapter 1, QA research attempts to deal with a wide range of question types including: fact, list, definition, how, why, hypothetical, semantically-constrained and cross-lingual questions. Some questions are easier to answer which we call *simple questions*. For example, “Who killed Abraham Lincoln?”. This type of questions (i.e. factoid) require small snippets of text as the answers. Again, the question “What are the titles of the books written by Paul Krugman?” asks for a list of small snippets (i.e. list questions) of text.

Modern automatic QA systems normally consist of three separate components: a) a question processing module, b) a document processing module and c) 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 information retrieval 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.

The Text REtrieval Conference (TREC) has been running a QA track to support the competitive research on question answering, since 1999 (TREC8). The focus of the QA track is to build a fully automatic open-domain question answering system, which can answer different types of questions based on very large document sets. Today, the TREC QA track is the major large-scale evaluation environment for open-domain question answering systems. With our QA system, we participated in TREC 2007. The TREC question answering track currently has three types of questions:

Factoid questions that require only one answer. Example[1]: *For which newspaper does Krugman write?*

List questions that require a non-redundant list of answers. Example[2]: *What are the titles of the books written by Krugman?*

Other questions that require a non redundant list of facts about the target that has not already been discovered by a previous answer. For example, the fact that "Krugman criticizes Bush administration" could be an answer of the "other" question for the target "Paul Krugman".

Our QA system for answering simple questions is based on document tagging and question classification. *Question classification* extracts useful information (i.e. answer type) from the question about how to answer the question. *Document tagging* extracts useful information from the documents, which will be used in finding the answer to the question. The global architecture of our QA system is given in Figure 4.1. We used different available tools to tag the documents. Our system classifies the questions using manually developed rules.

4.1.2 Chapter Outline

The remaining sections of this chapter are organized as follows:

- Section 4.2 includes descriptions of question normalization, question categories and how our system goes about classifying questions into those categories.
- Section 4.3 summarizes the querying technique to information retrieval systems.

[1] target id 216, TREC 2007
[2] target id 216, TREC 2007

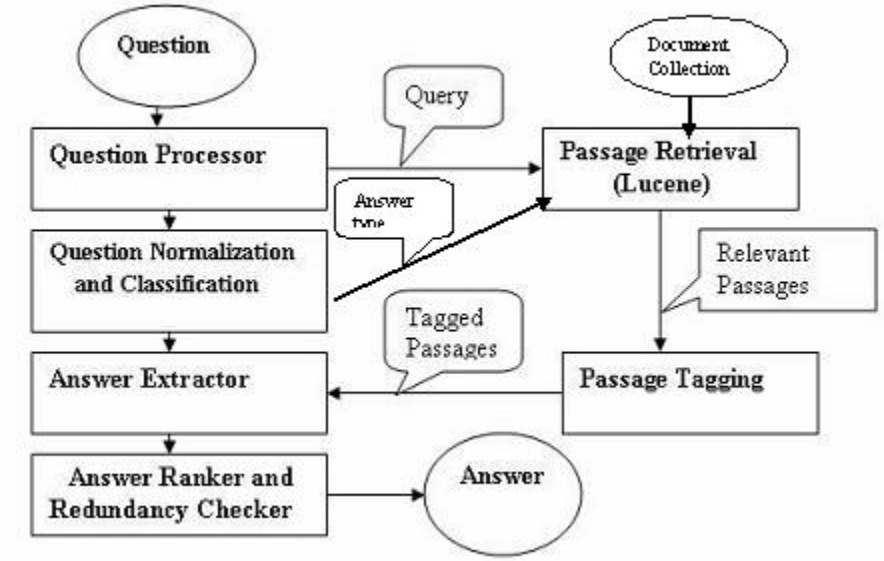

Figure 4.1: Model of our QA system

- Section 4.4 is a discussion of various methods of tagging useful information in the documents and how we used them in our system.
- Section 4.5 describes the methods in which our system goes about extracting candidate answers from the documents.
- Section 4.6 outlines how our system ranks those candidate answers and returns the answer that is considered the most probable.
- Section 4.7 is the evaluation based on TREC 2007.
- Section 4.8 consists of concluding remarks about our findings and views on the future of question answering systems.

4.2 Question Normalization and Classification

Classifying questions is a way to handle a group of similar questions in a similar way, rather than handling each question individually. Extracting the answer type from the question is crucial in finding the answer because the question contains the only clues available to find the answer. If information is incorrectly extracted from the question there is almost no chance to get a correct answer, but, if information is incorrectly extracted from a passage containing the correct answer, there is still a chance of finding the correct answer (Hermjakob, 2001).

Classifying a question based on answer type can narrow down the entities that could be possible answers. Answer types are, for the most part, named entities. A Named Entity (NE) is a term for a specific type of entity. Examples of NEs are; people's names, organizations' names, dates, names of places and quantities. Knowing which type of information the question is asking for is a large step toward answering the question. For instance, in the question, "Who is the CEO of 3M company?" the answer type is NE person. The candidate answers can be narrowed down to only the person names found in the retrieved passages.

4.2.1 TREC Questions

In order to derive a method to extract answer types from questions, we first need a large set of questions. The National Institute of Standards in Technology (NIST) has been running a QA track in TREC since 1999, and has included almost 3,500 questions since then. We used these questions for the purpose of finding different categories and ways to classify them as well as ways for our system to go about answering them.

The format of the question answering track changes somewhat from one year to the next. These changes provided us with a variety of questions. TREC 8 (Voorhees, 1999)

and TREC 9 (Voorhees, 2000) had a simple format. Systems were given fact questions that either had terms or a short statement as answers. These questions are called *factoid* questions. Examples of factoid questions are:

How hot is the core of the Earth?

How long would it take to get from Earth to Mars?

In TREC 10 (Voorhees, 2001) and TREC 11 (Voorhees, 2002), NIST added a *list* task, with questions that required a fixed number of answers. Factoid questions were also included in these years. Examples of list questions with a set number of answers are:

What are 3 residences of Queen Elizabeth II of the United Kingdom?

List 10 countries where motorcycles are produced.

In TREC 12 (Voorhees, 2003), the *definition* question type was introduced. Definition questions require as much unique and important information about a topic as possible. The importance of information was judged by NIST. In TREC 12, it was specified if a question was a list, definition or factoid question. Examples of definition questions are:

Who is Aaron Copland?

What is a golden parachute?

From TREC 13 (Voorhees, 2004), the format of how the questions were presented was changed. Instead of giving a basic list of questions, questions were grouped by their targets. The target of the group of questions is the theme of the questions. The target is given first, and then the questions that are about the target are listed after it. In the questions, the target is often referred to by a pronoun. The format of definition questions are also changed, and they are now referred to as *"Other"* questions. Other questions require a list of unique facts about the target that have not already been given as answers to previous questions of the target, and are just stated as *"Other."* For example, a portion of a target description from TREC 2007 is:

```
<target id = "222" text = "3M">
    <qa>
        <q id = "222.1" type="FACTOID">
            When was the 3M founded?
        </q>
    </qa>
    <qa>
        <q id = "222.2" type="FACTOID">
            Where is the company based?
        </q>
    </qa>
    <qa>
        <q id = "222.7" type="LIST">
            What brand name products does 3M manufacture?
        </q>
    </qa>
    <qa>
        <q id = "222.8" type="OTHER">
            Other
        </q>
    </qa>
</target>
```

4.2.2 Question Normalization

Before classifying the questions, our system changes some of the questions into a standard form that will be easier to classify.

For any question starting with a preposition followed by the word *what* or *which*, the preposition is extracted. Another quick question reformulation is extracting *the name of* from questions that start with *What was the name of* or *What is the name of*. Some other major normalizations involve:

Pronoun Resolution

The questions addressing a target often refer to the target by pronouns (i.e. he, she, it, they, his, her, its, their). We replace the personal pronouns (i.e. he, she, it, they) by the *target* and the possessive pronouns (i.e. his, her, its, their) by *target's*. For example, given the target *"Paul Krugman"*, the question: *"From which university did he receive his doctorate?"* is normalized to *"From which university did Paul Krugman receive Paul Krugman's doctorate?"*.

Apostrophe Resolution

Question with *'s* after the question word means *is*. For example, the question *Where's Montenegro?* is changed to *"Where is Montenegro?"*

"What" Factoid Normalization

There are different ways to say the same thing. For example, the questions: *"Name a film in which Jude Law acted."*, *"Jude Law was in what movie?"*, *"Jude Law acted in which film?"*, *"What is a film starring Jude Law?"*, *"What film was Jude Law in?"* and *"What film has Jude Law appeared in?"*, ask for the same information. So, they should all be classified as *What* questions, and more specifically, as *What-Film* questions. Our system will handle questions that involve the word *which* as a *what* question. Questions, like "*Jude Law acted in which film?*" and "*Jude Law was in what movie?*", that do not start with a question word are changed to *"What film was Jude Law acted in?"* and *"What movie was Jude Law in?"*, which are not in proper English, but our system will classify this form better. For this method, our system will first put a *What* at the beginning of the question,

followed by the second half of the question, forming *What movie* in these examples. Then, *was* is added followed by the first half of the question, resulting in *What movie was Jude Law acted in?*.

"What" Definition Normalization

There are questions that ask for why someone was famous, and questions requiring a definition of what something was. We considered these questions to be definition questions for our system, and normalize them to the form "*What is X?*" or "*Who is X?*", where X is the target. For example, the questions: "*What is Francis Scott Key famous for?*", "*What is Colin Powell best known for?*", "*Define thalassemia.*" and "*What does ciao mean?*" are changed to "*Who is Francis Scott Key?*", "*Who is Colin Powell?*", "*What is thalessemia?*" and "*What is ciao?*" respectively.

"What" List Normalization

Some list questions include the number of entities to be included in the list of answers which is helpful in returning answers for these questions, we extract the number, and reformat these questions to a general format. For each of these questions, the number is passed on to be used when the answer list is returned. Examples of normalization of these questions are:

- What are the titles of books written by Krugman? *is normalized to* What titles of books written by Krugman?
- Name 10 auto immune diseases. *is normalized to* What auto immune diseases?

- What are the 3 currencies Brazil has used since 1980? *is normalized to* What currencies Brazil has used since 1980?

4.2.3 Question Classification

Our system classifies questions into one of the categories: *who, when, why, how, where,* and *what*. Any question that does not include one of the first five stems (Whom is considered as Who) is considered as *what question*. Once the questions are categorized into one of the categories, a finer classification is done to extract the answer type of the question. Classifying questions aids our system in finding the answer type of the question, as well as helping to create the query in which the passages will be retrieved from our information retrieval system. Our system classifies questions using rules that we derived through observation from a test bed of questions. Supervised machine learning can also be used to classify questions, using methods similar to those outlined by (Hermjakob, 2001) and (Li and Roth, 2002). There are some other methods of finding answer types. Pinchak and Lin (2006) proposes a probabilistic model to find the answer types of "what", "where" and "who" questions. Pinchak and Bergsma (2007) addresses a method based on adjective expansion and web co-occurrence count to find the answer types for the "how-adjective" questions.

When Questions

When questions (Table 4.1) always ask for a *date*, but there are a few that are only asking for a day, and not a full year. The ones asking for a day will be asking for a special day that might appear on the same day each year, like a birthday, or a day that has a rule for when it is, such as labour day.

Question Type	Answer Type	Pattern	Example
When-Day	DATE without YEAR	"When is"	When is Father's day?
When-Year	DATE with YEAR	"All the rest"	When was Nimitz born?

Table 4.1: When question categories

Who Questions

There are three types of who questions outlined in Table 4.2. We pass the target directly to the answer extractor for the *who definition* questions. The answer type for *who list* is person, which is passed to the answer extractor along with the number of persons (if present in the question). For *who factoid* the answer extractor is given the information that the answer type is either person or organization.

Question Type	Answer Type	Pattern	Example
Who-Definition	FACTS	"Who [is or was] [NAME]	Who is Peter Weir?
Who-List	PERSONS	"Who [are or were or have]	Who are professional female boxers?
Who-Factoid	PERSON or ORG.	All the rest	Who invented paper?

Table 4.2: Who question categories

Where Questions

Most *where questions* require a type of location. There are other *where questions* that ask for a certain college or university someone went to. They are outlined in Table 4.3. For questions asking for schools, the answer extractor extracts all the schools. Other where questions ask for some kind of location more specifically the NEs: CITY, PROVINCE, COUNTRY, GEOGRAPHICAL_LOCATION. For these questions, the answer extractor

will extract these entities.

Question Type	Answer Type	Pattern	Example
Where-School	UNIVERSITY	college or university or school or degree	Where did Bill Gates go to college?
Where-Location	LOCATION	all the rest	Where is Las Vegas?

Table 4.3: Where question categories

Why Questions

These questions require reasons, which are not entities that our system currently recognizes.

How Questions

Most how questions are easy to classify and are of the form, "*How X*", where *X* (adjective) gives clues about what kind of entity the question is asking for. Table 4.4 gives the examples of "how-adjective" questions. The answer type for most of the *how questions* is straightforward. The exception is the *how many questions* (not shown in table) because they require a count of a certain entity. These questions start with the phrase *How many* and after it comes the entity needs to be counted. Examples of *"How many"* questions are: *How many rooms does the Las Vegas MGM Grand Hotel have?*, *How many times a day do observant Muslims pray?*, *How many layers of skin do we have?* These questions have a pattern of where the answer is found. The entities to be counted from these questions are *rooms*, *times a day*, and *layers*. For these questions this entity needs to be found and kept for the answer extraction module.

To extract the entity, first we extract *how many* from the start of each question, then tag

the question with a shallow parse. The reason we take off the *how many* first is because most taggers and parsers are trained on documents without question words and because of that, they do not do a very good job of tagging questions (Hermjakob, 2001). The last noun of the first noun phrase of each tagged question will be the entity that needs to be counted. In the above example the entities will be: *rooms, times and layers* respectively.

What Questions

Some what questions are classified by pattern matching and, for the rest of questions, the question focus is used to classify the questions.

What questions classified by pattern matching:

The *what questions* that are classified by patterns are in Table 4.5. *"What definition"* questions are handled similarly to the *"who definition"* questions, where the target is given to the answer extractor. *"What acro"* questions are looking for either the acronym for a certain entity, or what an acronym stands for. For the *"what verb"* questions, the answer type is ambiguous. For example, there are many types of things that can be invented, discovered, and eaten. Our system does not answer these questions. These questions can be answered using patterns in the syntactic parse of the documents. Once we add a syntactic parse of the documents to our system, We will attempt to handle these questions.

What questions classified by focus:

The what questions that are not classified by pattern matching can be classified by discovering the focus of the question. In the question, "*What country is the leading exporter of goats?*", the question focus is *country*. The focus of the question is a clue about what type of entity the answer will be. The importance of question focus when classifying questions

Question Type	Answer Type	Pattern	Example
How-Large	AREA or VOLUME	How [big or large]	How big is Mars?
How-Late	TIME or TIME PERIOD	How [late]	How late is Disneyland open?
How-Accurate	PERCENTAGE	How [accurate]	How accurate is HIV tests?
How-Distance	DISTANCE	How [far or tall or wide or short or high or close or deep]	How tall is Tom Cruise?
How-Often	TIME PERIOD	How [often or frequent]	How often is someone murdered in USA?
How-Long	DISTANCE or TIME PERIOD	How long	How long is Columbia River?
How-Much	MONEY or VOLUME	How much	How much did the first Barbie cost?
How-Temp	TEMPERATURE	How [warm or cold or hot]	How hot is the sun?
How-Fast	SPEED or TIME PERIOD	How fast	How fast is sound?
How-Old	AGE	How old	How old is the universe?
How-Death	METHOD OF DEATH	How did [NAME] die?	How did Anne Frank die?
How-Method	Method	No pattern	How did Hawaii become a state?

Table 4.4: How question categories

Question Type	Answer Type	Pattern	Example
What-Definition	FACTS	What [is or are]	What is caffeine?
What-Acro	INSTITUTE or ACRO or PHRASE	" stand(s) for" or "acronym for" or "abbreviation for" or the ((acronym) or (abbreviation))	What does CNN stand for?
What-Verb	THING	question ends in a verb	What do manatees eat?

Table 4.5: What simple question categories

is discussed in (Ferret et al., 2001). Once the questions are all tagged with chunked parts of speech, patterns are used to extract the question focus. A complete list of focus extracting patterns, where X represents the focus of the question, is:

- [NP What (type or kind or breed)] [PP of] [NP *X*]
- [NP What *X*]
- [NP What] [NP *X*]
- ('s or ') *X*]
- [NP What] [VP (is or was)] [NP *X*]
- [NP Name *X*]
- [NP Name] [NP *X*]

These focuses are then matched to a named entity, and that will be considered the answer type of the question. For each of these questions, the answer type is passed to the answer extractor.

4.3 Query Creation and Passage Retrieval

4.3.1 Query for Factoid and List Questions

We are using Java based Lucene [3] as our information retrieval system. (Moldovan et al., 2004) notes that because of the Lucene system's ability to have a greater understanding of natural language, the passages retrieved can be more relevant to the query. We index

[3] http://jkarta.apache.org/lucene/

all paragraphs in the document collection in *case folded* and *stemmed* fashion. Therefore, "bank" will be indexed as the same as the "banking". The query in Lucene can be thought as "vector-space on top of boolean", that means it will satisfy the boolean true as well as it will rank documents. We query the documents using three methods:

- Expanded boolean query
- Important word boolean query
- Expanded vector-space query

Expanded Boolean Query

We take all the nouns, verbs and adjectives from the question that are not stop words, and find synonyms for each of them, using WordNet. We use these synonyms to form a query, where each word in the synonym set for a word is separated by *OR*, and each synonym set is separated by *AND*. Our system does not include the words contained in the question focus when creating the expanded query as, (Moldovan et al., 1999) shows that the question focus is often not found in a document with the answer. When the focus is a *state, city, country, nationality* or *date*, the answer will appear as that specific entity, and those focus words will not necessarily appear in the documents.

For example, if a question has two words that are extracted, say *Word1* and *Word2*, and the synonym set for *Word1* is *S11* and *S12* and the synonym set for *Word2* is *S21*, *S22* and *S23*, the boolean query will be: "(S11 OR S12) AND (S21 OR S22 OR S23)". This query ensures that all documents retrieved should be somehow related to the question. These queries are often too restrictive and retrieve no documents. We did not perform any word sense disambiguation for the query-words as the context for these words is not large enough

to apply our WSD system. We took the first sense synonym set from the WordNet for the query-words.

Important Word Boolean Query

If the first method retrieves no documents, then our system forms a boolean query with the proper nouns and dates from the question. For example, for the question: "What is the population of Canada in 2006?", the important word boolean query will be: *Canada OR 2006*. This will usually retrieve the document with the answer to the question, but will also retrieve many documents unrelated to the question.

Expanded Vector-Space Query

If the previous two methods retrieve no documents, then a vector-space search is performed taking all the important words in the query along with their synonyms. Vector-space query is described in Section 2.4.2.

4.3.2 Query for Definition Questions

For definition questions we do a query just on the subject (focus) to be defined.

4.4 Document Tagging

The document tagging module of our system tags useful information from the passages retrieved by Lucene. The detailed description of the tags and the tools utilized is given in Chapter 3. So, in this section we just list them. The tags that we used are:

1. Tokenization and sentence splitting using OAK systems (Section 3.2.1)
2. Co-reference resolution using Lingpipe (Section 3.2.2)
3. Part of speech tagging using OAK systems (Section 3.2.4)
4. Stemming using OAK systems (Section 3.2.3)
5. Chunked part of speech tagging using OAK systems (Section 3.2.5)
6. Name entity tagging using OAK systems and WordNet (Section 3.2.6)
7. Word sense tagging using our developed WSD system (Section 3.2.7)

4.5 Answer Extraction

Our method of extracting answers is different for the list and factoid questions (which require extracting named entities) from the definition questions (that are extracted using patterns).

4.5.1 List and Factoid Questions

Since the answer types and question types are known from the question classifier, and the documents are tagged with both shallow parse and named entities, extracting the information from the documents will use patterns to extract the named entities associated with the answer type. For some questions, the answer is not tagged and will be extracted with just patterns from the documents.

The answers for certain types of questions can also be extracted with patterns rather than our approach of extracting answers using the answer type. These patterns can be learned by

using machine learning techniques (Ravichandran and Hovy, 2002). Many groups (Roussinov, Ding, and Robles-Flores, 2004), (Tanev, Kouylekov, and Magnini, 2004), (Echihabi et al., 2003), (Nyberg et al., 2003), (Wu et al., 2003) and (Prager et al., 2003) have used a similar technique in their systems to discover patterns for finding answers.

"How Many" Questions

How many questions are answered using patterns and not named entities since these questions are looking for a count of a certain entity. We discovered that answers will appear frequently in the tagged documents in the pattern "*[NP NUMBER ENTITY]*". For the question, *"How many hexagons are on a soccer ball?"*, the entity to be counted is *hexagons*, so the pattern it is trying to match is *[NP NUMBER hexagons]*. This pattern is found in the following passage:

> [PP After/IN] [NP all/DT] ,/, [NP a/DT buckyball/NN] [NP 's/POS structure/NN] [PP of/IN] [NP 12/CD pentagons/NNS] and/CC *[NP 20/CD hexagons/NNS]* [VP is/VBZ] [ADVP just/RB] [PP like/IN] [NP that/DT] [PP of/IN] [NP a/DT classic/JJ soccer/NN ball/NN] ./.

This module will extract *20 hexagons* and send that as a possible answer, along with the passage it is found in, to the answer ranker.

WordNet is also used to get synonyms for the entity that is being counted. For example, given the question, *"How many floors are in the Empire State building?"* the synset (*floor, level, storey* and *story*) of the word *floor* can be used to extract answers from the passage: *"The Empire State building climbed to an unthinkable height of* 102 stories *in that city four years later."*

Names of People

The OAK system tags people in four ways: 1. person, 2. last name, 3. female firstname and 4. male firstname. This presents a problem because the OAK tagger tags everything at once from a list of names. Some *last names* are missing from the list of tags, and other names are also names of cities, such as the name *Paris*. The answer extracting module needs to extract the full name of the person when it is available. Some examples of problems and their patterns for solutions are:

> [NP ⟨CITY DETROIT/NNP ⟩] :/: **[NP ⟨MALE_FIRSTNAME Juan/NNP ⟩ Gonzalez/NNP]** [VP was/VBD] [ADVP back/RB] [PP in/IN] [NP the/DT starting/VBG lineup/NN] [PP after/IN] [VP missing/VBG] [NP three/CD games/NNS] [PP because/IN of/IN] [NP a/DT sore/JJ foot/NN] but/CC [VP may/MD be/VB sidelined/VBN] [PP after/IN] [VP aggravating/VBG] [NP it/PRP] [SBAR while/IN] [VP running/VBG] [NP the/DT bases/NNS] [ADVP when/WRB] [NP he/PRP] [VP hit/VBD] [NP a/DT triple/JJ] ./.

In this example, the name *Juan* is followed by the untagged last name *Gonzalez*. This can be fixed by the pattern "⟨*[A-Z_]NAME [A-Za-z]NNP* ⟩ *[A-Za-z]/NNP*". Notice that it will also get *last name*, as well as the other two types.

Some people have last names that are usually considered to be first names:

> **[NP P.S./NNP ⟨MALE_FIRSTNAME Kevin/NNP ⟩ ⟨MALE_FIRSTNAME Ryan/NNP ⟩]** [NP forwards/RB a/DT ⟨CITY Sacramento/NNP Bee/NNP ⟩] [VP clipping/VBG describing/VBG] [NP a/DT paper/NN sign/NN] [PP on/IN] [NP the/DT men/NNS] [NP 's/POS room/NN wall/NN] [PP in/IN] [NP the/DT state/NN] [NP Capitol/NNP] [NP 's/POS ⟨FACILITY Legislative/NNP Office/NNP ⟩ Building/NNP] :/: "/" [VP Please/VB wash/VB] [NP

your/PRP$ hands/NNS] [PP before/IN] [VP touching/VBG] [NP legislation/NN] ./. ”/”

This will present the pattern of “⟨ *NAME_TYPE NAME/NNP* ⟩ ⟨*NAME_TYPE NAME/NNP* ⟩”.

Dates

There are two types of date questions, one looking for a certain day that happens every year, and ones that are looking for a particular day. Some dates that are tagged by OAK do not fit into either of these categories and are considered relative dates. These include *today, this year, this month, next week* and *midnight*. These dates are not helpful in answering questions, and are eliminated right away. Also, answers to questions that are looking for a particular date should have a four digit year in them.

Quantities

OAK tags each quantity it sees as a quantity, but does not tag quantities together that are of the same measurement if the quantities are of different units. For instance, it will tag the measurement *4 foot 2 inches* as ⟨ *PHYSICAL_EXTENT 4 foot* ⟩ ⟨ *PHYSICAL_EXTENT 2 inches* ⟩. We can extract the full quantity if we use a pattern that extracts more than one quantity when two quantities of the same measurement are together.

Other Types of Questions

For the rest of the questions, possible answers are extracted by extracting the named entities associated with the answer type of the question. This is done by pattern matching with "$\langle NE\ X\ \rangle$" where X is a possible answer if NE is a named entity tag corresponding the answer type of the question. These entities are listed in Appendix A.

4.5.2 Definition Questions

In fact finding questions, there is always a topic that is being sought. If the question is "*Who is X?*" (X being a name of a person), there will be different methods for finding facts for it, compared to a non-person entity that will be phrased as "*What is a Y?*" We have chosen to implement a method of pattern finding to answer fact based questions.

Pattern Finding

The fact finding patterns were determined by manually examining definition questions from the TREC question test bed, which includes examples of facts for each question. We used the following steps to determine the patterns:

Step 1: Manually Finding Facts

The fact sentences are found by forming a query from the topic, and retrieving relevant passages from our information retrieval system. NIST provides answers to past definition and other questions that can be used to form a query to try to get a specific fact.

For the question "*Who is Aaron Copland?*"

The following sentence contains the fact that *Aaron Copland* is a *composer*.

His circle of friends included Picasso and Piet Mondrian, composer Aaron Copland,

actors Charlie Chaplin and Paulette Goddard, and titans of U.S. industry such as the Fords and Rockefellers.

Step 2: Chunking Fact Sentences

The fact sentences are then chunked and patterns are observed. For example:

[NP His/PRP$ circle/NN] [PP of/IN] [NP friends/NNS] [VP included/VBD] [NP Picasso/NNP and/CC Piet/NNP Mondrian/NNP] ,/, [NP *composer/NN* Aaron/NNP Copland/NNP] ,/, [NP actors/NNS Charlie/NNP Chaplin/NNP and/CC Paulette/NNP Goddard/NNP] ,/, and/CC [NP titans/NNS] [PP of/IN] [NP U.S./NNP industry/NN such/JJ] [PP as/IN] [NP the/DT Fords/NNP and/CC Rockefellers/NNP] ./.

Step 3: Pattern Creation

Patterns are formulated from manual observations of the tagged sentences. The passage shows that information contained before the target, which is in the noun phrase, is a pattern to find out a fact about this target. Our current definition patterns include, with *X* representing facts and *TARGET* representing the subject of the fact:

- [NP *X TARGET*]
- *[TARGET]* , *X* (, or . or ;)
- *[TARGET]* (is or are) *X* (. or ;)
- *X* called *[TARGET]*
- *[BEGINNING OF SENTENCE] [TARGET]*, *X*, is *X*
- *[TARGET]* and other [NP *X*]

4.6 Answer Ranking

This module gives a score to each possible answer, and returns the one with the highest score, or the answers with the highest scores if a list of answers is required. It is possible that the corpus will not contain the answer to the question. Because of this, answers should only be given if the system is sure that the answer is correct. If a list of answers is required by the question, this module will only pass on answers that are over a certain rank.

4.6.1 Answer Patterns

As stated in the previous section, many systems use lexical patterns to extract answers from the documents, in contrast to our approach of extracting named entities. Our system used the following patterns to rank answers.

Date of Birth and Date of Death

The date of birth and date of death of a person are sometimes put in brackets, after a person's name, e.g. "PERSON (DATE - DATE)". For example, the sentence "*Elvis Presley (1935-1977), James Dean (1931-1955) are the new men.*" contains such patterns.

"What" Location Questions

When a location is the answer type of the question, and the question contains a preposition like *in, on, from* or *near*, those words will frequently appear before the location entity that is the answer. For questions that contain *on* and *from*, *in* can also appear before the answer. Examples of these types of questions are:

- What continent is Togo on?
- What continent is India on?

An example of a sentence with this pattern, for the question "*What continent is Togo on?*" is:

> The president, who returned home early on Tuesday after a three-day visit to Mali and Togo *in West Africa*, agreed to undergo the most intensive medical check-up to date as a bid by the government and the African National Congress to refute damaging rumours that his health was deteriorating.

"When" Questions Ending in a Verb

Some questions end with a verb that represents the particular action that is being asked about. Examples of these questions are:

- When was Microsoft established?
- When was the first Wal-Mart store opened?
- When was Hiroshima bombed?

For these questions, an answer is usually found in the following pattern, "*VERB in* ⟨*DATE*⟩", where *VERB* represents a verb with a stem that is a synonym of the last verb from the question. An example of a sentence with this type of pattern for the question, "*When was the first Wal-Mart store opened?*", is:

> Supercenters, the first of which *opened in 1988*, had already transformed the $420 billion-a-year grocery business.

Who Action

Who questions often contain the action of *being*. For example: *who was Khmer Rouge's first leader?* There are some questions that contain a physical action that are not being: *who wrote "Dubliners"?* These actions are represented in the passages that the answers are in, but might take different forms. For these types of questions, our system takes the pattern of the whole action. For example, the question "*who wrote"Dubliners"?*", has the action "*wrote "Dubliners"*", and is the pattern our system uses to rank.

Word Association

Lexical chains (Morris and Hirst, 1991) are created when we associate words together that have a similar theme. (Moldovan and Novischi, 2002) discussed using WordNet to help with the associations between words. Besides synonym, hypernym, hyponym and gloss each word in WordNet has derivationally related forms as well. The derived form for a noun is a verb that is associated with the noun. With these tools there can be a path formed from the question words to the words of the passage of a possible answer.

In order to find this association, we extracted the important words (i.e. nouns, verbs, adjectives and adverbs) from the passage with candidate answer as well as from the question. Then, we perform our WSD algorithm on these words (where the passage words preceeds the question words) and form lexical chains. Now, we check for each of the question words whether we have got any chain or not. A chain indicates there is association between the question word and the passage words. In this way we count the number of association.

For example,

Question: For which newspaper does Krugman write?

Passage: Paul Krugman is also an author and a columnist for the New York Times.

the passage words *author* and *columnist* are associated with the question word *write* by their semantic relation (gloss of *author* and *columnist* in this case). So, the number of association is two: one is for Krugman (repetition) and one is for write (gloss).

WordNet Glossary

For most words, WordNet has a glossary entry which contains a definition of the word. Our system extracts the glossary entry for the proper nouns in the question. For definition questions, the glossary entry for the target is used. For factoid questions, our system tags the WordNet glossary entry with named entities, and if it contains the answer type, that answer is given a higher rank. For definition questions, the WordNet glossary entry is passed to the redundancy checker to rank facts. For the question, *"where is Belize located?"*, *Belize* is a proper noun and its WordNet gloss is:

> a country on the northeastern coast of Central America on the Caribbean; formerly under British control.

The gloss tagged with POS and NE is:

> a/DT country/NN on/IN the/DT northeastern/JJ coast/NN of/IN ⟨ GEOLOGICAL_REGION Central/NNP America/NNP ⟩ on/IN the/DT ⟨ GEOLOGICAL_REGION Caribbean/NNP ⟩ ;/: formerly/RB under/IN ⟨ NATIONALITY British/JJ ⟩ control/NN.

For *where questions*, *GEOLOGICAL_REGION* is an accepted answer type, and both *Central America* and *Caribbean* are acceptable answers for this question.

Target distance

One other method that is commonly used to rank answers is the distance between a possible answer and keywords from the question (Kwok, Etzioni, and Weld, 2001) (Chen et al., 2004). Our system calculates distance as the count of words and punctuation between the important words from the question and the possible answer. For example, the question "*What does Kurt Vonnegut do for a living?*", has a target of *Kurt Vonnegut*. The following passage contains the answer to this question:

> Trust a crowd, says *author Kurt Vonnegut*, to look at the wrong end of a miracle every time.

The answer *author* is considered, by our system, to be one word away from the target of the question.

Redundancy Checking

Answers that appear frequently should be given a higher rank if the retrieved documents are related to the question.

If the question requires more than one answer, each answer returned by the system should be unique. When this module returns an answer, it performs a check to see if the answer has already been included in the final answer list. This check is performed by checking whether the answer contains nouns that have already appeared in a previous answer.

For definition questions, an answer can contain more than one fact. For example for the target *Jane Goodall*, the phrase, *the British primatologist*, gets extracted. This contains the fact that she is *British*, and the fact that she is a *primatologist*. If this is the case, then

the phrase that contains two or more facts is added to the list of answers, and if either of the contained facts were found in the list, their score will be added to the score of the combination fact.

Our system counts each redundancy as an occurrence, which will be used in ranking answers.

4.6.2 Answer Ranking Formula

The ranking formula is developed by reviewing which weights for the methods above yielded the highest accuracy on the corpus of questions and their answers.

Formula for Factoid and List Questions

For *factoid and list questions* our answer ranking formula is:

score of an occurance of a candidate answer = $(6 \times w_1) + (w_2) + (3 \times w_3) + \frac{1}{w_4}$

where,

w_1 denotes whether the answer is found in a pattern associated with the question type. The value will be 1 if it was found in such a pattern, and 0 if it was not.

w_2 denotes how many words from the question are represented in the passage with the answer, plus 3 more points for each word that is represented by a disambiguated word.

w_3 denotes if the answer appears in the WordNet glossary for important words from the question. (value of 0 if it does not and 1 if it does)

w_4 denotes the distance between the important words from the question and the answer.

Each answer's final rank is the sum of the ranks of the occurrences of that answer. This method of giving a higher score to answers that appear more than once is discussed in (Clarke, Cormack, and Lynam, 2001).

(Dubien, 2005) used this formula to rank the possible answers. This formula was derived using a test bed of 300 questions from TREC 1999 to TREC 2004. I created a test bed of 100 (50 factoid and 50 list) questions from TREC 2005 and TREC 2006 and test the performance of this formula. These questions from were chosen because our system extracts the answers to them. The average chance of our system, at random, picking a correct answer to a question in this test bed is about 10%. This means approximately one in ten extracted answers is correct. With this ranking formula, our system is able to answer this test bed correctly 64% of the time. This shows that this method of ranking is improving the chance of picking a correct answer.

Formula for Definition Questions

Of all our methods of ranking answers, only answer redundancy is applicable to definition questions. We rank answers to definition questions, giving them one point for repetition and four points if found in the WordNet gloss.

4.6.3 Answer Thresholds

When an answer is given for a question, the confidence that the answer is correct should be high. Even if the question answering system could return more than one candidate answer to the user, only the ones that have a chance of being correct should be shown (Xu, Licuanan, and Weischedel, 2003). For questions where a list of answers is required, all the answers should be correct, so the threshold should be a bit greater than that of an answer

candidate.

Our system does not currently use an answer checking threshold because our method for ranking answers is different for different types of questions. The threshold needs to be set as a relative value or a constant value, and we have not yet determined an appropriate relative score for which an answer is considered to be correct or incorrect. This is a direction that requires further consideration.

4.7 Evaluation

4.7.1 Factoid Questions

The system response to a factoid question was either exactly one [doc-id, answer-string] pair or the literal string NIL. Since there was no guarantee that a factoid question had an answer in the document collection, NIL was returned by the system when it believed there was no answer. Otherwise, answer-string was a string containing precisely an answer to the question, and doc-id was the id of a document in the collection that supported answer-string as an answer.

Each response was independently judged by two human assessors. When the two assessors disagreed in their judgments, a third adjudicator made the final determination. Each response was assigned exactly one of the following five judgments:

Incorrect the answer string does not contain a correct answer or the answer is not responsive;

Not supported the answer string contains a correct answer but the document returned does not support that answer;

Inexact the answer string contains a correct answer and the document supports that an-

swer, but the string contains more than just the answer or is missing bits of the answer;

Locally correct the answer string consists of exactly a correct answer that is supported by the document returned, but the document collection contains a contradictory answer that the assessor believes is better;

Globally correct the answer string consists of exactly the correct answer, that answer is supported by the document returned, and the document collection does not contain a contradictory answer that the assessor believes is better.

For example, Krugman taught in Yale university and now he is teaching in Princeton university. So, the globally correct answer to the question, " At which university does Krugman teach?" will be Princeton and locally correct answer is Yale university.

Out of 360 *FACTOID* questions in TREC 2007, 16 had no known correct response in the document collection, hence "NIL" is the correct answer for these questions. Table 4.6 shows our scores for FACTOID questions.

Total	Globally Correct	Locally Correct	Incorrect	Unsupported	Inexact	Accuracy	"NIL" Precision	"NIL" Recall
360	93	5	231	12	19	0.258	$8/153 =$ .052	$8/16 =$ .5

Table 4.6: UofL score for factoid questions

It can be seen that, our system (Chali and Joty, 2007b) could give 93 globally correct answers and 5 locally correct answers. The accuracy is computed as the ratio of the number of globally correct to the number of factoid questions. Table 4.7 shows the most accurate run for the factoid component for each of the top 10 groups out of 51 groups (Dang, Kelly, and Lin, 2007). Also reported are the recall and precision of recognizing when no answer

exists in the document collection. NIL precision is the ratio of the number of times NIL was returned and correct to the number of times it was returned; NIL recall is the ratio of the number of times NIL was returned and correct to the number of times it was correct in the entire test set (16). If NIL was never returned, NIL precision is undefined and NIL recall is zero.

Our accuracy is almost 26% which is fourth in the ranking.

Run Tag	Submitter	Accuracy	"NIL" Precision	"NIL" Recall
LymbaPA07	Lymba Corporation	0.706	0.000	0.000
LCCFerret	Language Computer Corporation	0.494	0.000	0.000
lsv2007c	Saarland University	0.289	-	0.000
UofL	**University of Lethbridge**	**0.258**	**0.052**	**0.500**
QASCU1	Concordia University	0.256	0.000	0.000
FDUQAT16A	Fudan University	0.236	0.053	0.312
pronto07run3	Universita di Roma "La Sapienza"	0.222	0.000	0.000
ILQUA1	State University of New York (SUNY) at Albany	0.222	0.000	0.000
Ephyra3	Carnegie Mellon University and Universitaet Karlsruhe	0.208	0.048	0.062
QUANTA	Tsinghua University (State Key Lab)	0.206	0.091	0.062

Table 4.7: Scores for the factoid component

4.7.2 List Questions

A system's response to a list question consists of an unordered set of [doc-id, answer-string] pairs such that each answer-string represents a correct answer instance. Each instance was evaluated in the same manner as the factoid questions, i.e., assigned one of the following judgments: incorrect, not supported, not exact, locally correct, and globally cor-

rect. Instances that were judged to be globally correct were then manually grouped into equivalence classes, where each equivalence class was considered a distinct answer. Thus, systems were not rewarded (and were in fact penalized) for returning equivalent answers multiple times.

The final set of known globally correct answers for a list question was compiled from the union of distinct globally correct answers across all runs plus additional distinct answers the assessor found during question development. For the 85 list questions in the test set, the median number of known distinct globally correct answers per question was 7, with a minimum of 2 and a maximum of 64. A systems response to a list question was scored using instance precision (IP) and instance recall (IR) based on the complete list of known distinct globally correct answers. Let S be the number of such answers, D be the number of distinct globally correct answers returned by the system, and N be the total number of instances returned by the system. Then IP = D/N and IR = D/S. Precision and recall were then combined to produce an F-score with equal weight given to recall and precision:

$$F = \frac{2 \times IP \times IR}{IP + IR}$$

The score for the list component of a run was the average F-score over the 85 questions. Our system achieved the "average F-measure" score of 0.132 with recall and precision weighted equally ($\alpha = 0.5$) which is ranked sixth. Table 4.8 gives the average F-score of the run with the best list component score for each of the top 10 groups.

Run Tag	Submitter	F-score
LymbaPA07	Lymba Corporation	0.479
LCCFerret	Language Computer Corporation	0.324
ILQUA1	State University of New York (SUNY) at Albany	0.147
QASCU1	Concordia University	0.145
Ephyra3	Carnegie Mellon University and Universitaet Karlsruhe	0.144
UofL	**University of Lethbridge**	**0.132**
FDUQAT16B	Fudan University	0.131
IITDIBM2007T	Indian Institute of Technology, Delhi	0.125
FDUQAT16A	Fudan University	0.107
pronto07run3	Universita di Roma "La Sapienza"	0.103

Table 4.8: Scores for the list component

4.7.3 Other Question

The method for evaluating "Other" questions is outlined in (Dang, Kelly, and Lin, 2007). Our system achieved the "average pyramid F-score" (with $\beta = 3$) of 0.030 over 70 *other* questions which is not a good score.

4.7.4 Per-series Combined Scores

The three component scores measure a system's ability to process each type of question, but may not reflect the system's overall usefulness to a user. Since each series is a mixture of different question types, we can compute a weighted average of the scores of the three question types on a per-series basis, and take the average of the per-series weighted scores as the final score for the run (Voorhees, 2005). In 2007, the weighted score for an individual series was computed as:

$$WeightedScore = \frac{1}{3} \times Factoid + \frac{1}{3} \times List + \frac{1}{3} \times Other$$

Table 4.9 shows the results of the multiple comparison for the 10 groups with the highest final per-series score (Dang, Kelly, and Lin, 2007).

Run Tag	Submitter	score
LymbaPA07	Lymba Corporation	0.4839
LCCFerret	Language Computer Corporation	0.3575
FDUQAT16B	Fudan University	0.2310
lsv2007c	Saarland University	0.2296
QASCU1	Concordia University	0.2216
ILQUA1	State University of New York (SUNY) at Albany	0.2023
Ephyra3	Carnegie Mellon University and Universitaet Karlsruhe	0.1804
IITDIBM2007T	Indian Institute of Technology, Delhi	0.1735
QUANTA	Tsinghua University (State Key Lab)	0.1592
csail3	Massachusetts Institute of Technology (MIT)	0.1415

Table 4.9: Multiple comparison based on ANOVA of per-series score.

Our system achieved average per-series score of 0.1410 which is almost the same as the 10-th system in TREC 2007.

4.8 Discussion and Concluding Remarks

In 2007, we have improved (compared to (Chali and Dubien, 2004)) our ranks in all three types of questions though our score for "Other" questions is very poor. This overall improvement was primarily because of our expanded classification and the addition of word dependencies to answer ranking. We will have to concentrate on answering "Other" questions. Our system extracted the answers from the top 100 (50 AQUAINT-2 + 50 BLOG06) relevant documents provided by TREC instead of using the whole collection. Our system did not find answers (i.e. NIL) for 153 factoid questions. According to the TREC 2007 evaluation there were 16 questions with "NIL" answers. Our system could find 8 correct "NIL" answers. This clearly indicates that if we would use the whole document collection, our system could have found more answers (i.e. less number of "NIL" answers) hence, we could have scored better in TREC 2007.

Chapter 5

Answering Complex Questions

5.1 Introduction

5.1.1 Problem Definition

As introduced in chapter 1, QA systems attempt to deal with both simple and complex questions. In the previous chapter, we described our method for answering simple questions that are generally of type factoid and list. After having made substantial headway in factoid and list questions (such as "Who won the nobel prize in peace in 2006?" or "Name the books written by Dr. Muhammad Yunus"), researchers have turned their attention to more complex information needs that cannot be answered by simply extracting named entities (persons, organization, locations, dates, etc.) from documents. For example, the questions *"Describe steps taken and worldwide reaction prior to the introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press."* require inferencing and synthesizing information from multiple documents. This information synthesis in NLP can be seen as a kind of topic-oriented, informative multi-document summarization, where the goal is to produce a single text as a compressed version of a set of documents with a minimum loss of relevant information. Unlike indicative summaries (which help to determine whether a document is relevant to a particular topic), informative summaries must be helpful to answer, for instance, factual questions about the topic (Amigo et al., 2004).

We believe that complex questions cannot be answered using the same techniques that have so successfully been applied to the answering of "factoid" questions. Unlike informationally-simple factoid questions, complex questions often seek multiple different

types of information simultaneously and do not presuppose that one single answer could meet all of its information needs. For example, with a factoid question like "How accurate are HIV tests?", it can be safely assumed that the submitter of the question is looking for a number or a range of numbers. However, with complex questions like "What are the causes of AIDS?", the wider focus of this question suggests that the submitter may not have a single or well-defined information need and therefore may be amenable to receiving additional supporting information that is relevant to some (as yet) undefined informational goal.

Over the past three years, complex questions have been the focus of much attention in both the automatic question-answering and Multi Document Summarization (MDS) communities. While most current complex QA evaluations (including the 2004 AQUAINT Relationship QA Pilot, the 2005 TREC Relationship QA Task, the TREC definition (and other) questions and the 2006 GALE Distillation Effort) require systems to return unstructured lists of candidate answers in response to a complex question, recent MDS evaluations (including the 2005, 2006 and 2007 Document Understanding Conferences (DUC)) have tasked systems with returning paragraph-length answers to complex questions that are responsive, relevant, and coherent.

The DUC conference series is run by the National Institute of Standards and Technology (NIST) to further progress in summarization and enable researchers to participate in large-scale experiments. We used the main task of DUC 2007 for evaluation. The task was:

"Given a complex question (topic description) and a collection of relevant documents, the task is to synthesize a fluent, well-organized 250-word summary of the documents that answers the question(s) in the topic".

For example, given the topic description (from DUC 2007):

```
<topic>
```

```
    <num>D0703A</num>
       <title> steps toward introduction of the Euro </title>
        <narr>
        Describe steps taken and worldwide reaction prior to
        the introduction of the Euro on January 1, 1999.Include
        predictions and expectations reported in the press.
        </narr>
</topic>
```

and a collection of relevant documents, the task of the summarizer is to build a summary that answers the question(s) in the topic description.

5.1.2 Our Approaches

We experimented with both empirical approach and machine learning approach. Our system (Chali and Joty, 2007a) that participated in DUC 2007 was based on the empirical approach. We then experimented with several machine learning techniques for this particular problem and evaluated their results. Our systems involve the following major steps:

- *Document processing*
- *Query processing*
- *Feature extraction*
- *Sentence ranking*
- *Redundancy removing* and
- *Summary generating*

Our systems (i.e. empirical approach and machine learning approach) differ mainly in *feature extraction* and *sentence ranking* phases. Our empirical approach used six features in order to rank a sentence while our machine learning techniques used eighteen features to rank a sentence.

5.1.3 Chapter Outline

The remaining sections of this chapter are organized as follows:

- Section 5.2 includes brief information about document processing for our summarizer systems.
- Section 5.3 summarizes the processing done to extract useful information from the query and each of the sentences in the document collection.
- Section 5.4 is a discussion of various features and the way we can extract those from the document collection.
- Section 5.5 describes the methods by which our systems rank the sentences. It includes our empirical approach and machine learning approach.
- Section 5.6 outlines how our systems remove the redundant sentences and generate summaries of fixed length.
- Section 5.7 is the evaluation of our systems.
- Section 5.8 consists of concluding remarks about our findings and our views on the future of text summarization.

5.2 Document Processing

Document processing for text summarization involves:

1. Tokenization and sentence splitting using OAK systems (Section 3.2.1)
2. Co-reference resolution using Lingpipe (Section 3.2.2)
3. Part of speech tagging using OAK systems (Section 3.2.4)
4. Stemming using OAK systems (Section 3.2.3)
5. Chunked part of speech tagging using OAK systems (Section 3.2.5)
6. BE extraction using BE package provided by ISI (Section 3.2.9)
7. Syntactic tree extraction using Charniak parser (Section 3.2.10)
8. Semantic tree extraction using semantic role labeler system ASSERT (Section 3.2.11)

All of these tasks and the tools used are described in detail in Chapter 3.

5.3 Query and Sentence Processor

Query processor module performs the chunked POS tagging and extracts the important words (i.e. nouns, verbs, adjectives and adverbs) from the query. It takes the first synonym list for each of these words and hypernym, hyponym list and important words from the gloss definitions of each of the nouns. The hierarchy level is restricted to 2 and 3 for hypernym and hyponym respectively. It creates a list of these related words which we call *query related words*. When it extracts these words it excludes the stop words. For example, given the query:

"Include predictions and expectations reported in the press."

This module extracts the following lists:

- **Important words:** report, include, prediction, press, expectation
- **Synonym words:** study, fourth estate, anticipation, prevision, outlook and prospect
- **Hypernym/hyponym words:** document, essay, case study, white book, blue book, green paper, position paper, estate, press corps, reasoning, projection, prophecy, prefiguration, belief, promise, foretaste, possibility, anticipation and apprehension.
- **Gloss words:** write, document, describe, finding, individual, group, accord, recent, study, hill, dale, newspaper, writer, photographer, act, predict, reasoning, future, belief, mental, picture, and future.

Sentence processor module is invoked for each sentence in the document collection. It extracts the important words (i.e. nouns, verbs, adjectives and adverbs), synonyms, hypernyms, hyponyms and gloss words for the sentence as we did for the query.

5.4 Feature Extraction

We have used 18 different features in total that can be divided into several categories:

5.4.1 Lexical Features

N-gram Overlap

N-gram overlap is the recall between the query and the candidate sentence. This can be computed as follows:

$$N-gram\ overlap\ score = \frac{\sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{gram_n \in S} Count(gram_n)}$$

Where n stands for the length of the $n-gram$, S is the candidate sentence, and $Count_{match}$ $(gram_n)$ is the maximum number of n-grams co-occurring in the query and the candidate sentence. $Count(gram_n)$ is the number of $n-gram$.

1-gram Overlap Measure

It measures the number of words common in the sentence in hand and the *query related words*. This can be computed as follows:

$$1-gram\ Overlap\ Score = \frac{\sum_{w_1 \in S} Count_{match}(w_1)}{\sum_{w_1 \in S} Count(w_1)}$$

Where S is the set of important words (i.e. nouns, verbs, adjectives and adverbs) in the candidate sentence and $Count_{match}$ is the number of matches between the *sentence important words* and *query related words*. $Count(gram_n)$ is the number of w_1.

Note that, in order to measure the 1-gram score, we took the *query related words* instead of the exact query words. The motivation behind this is, the sentence which has word(s) that are not exactly the query words but their synonyms, hypernyms, hyponym or gloss words, will get counted.

Example:

Query Describe steps taken and worldwide reaction prior to introduction of the Euro on

January 1, 1999. Include predictions and expectations reported in the press.

Sentence The Frankfurt-based body said in its annual *study* released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.

1-gram Score 0.06666

Note that, the above sentence got 1-gram overlap score of 0.06666 even though it has no exact word common with the query words. It got this score because the sentence word *study* is the synonym of the query word *report*.

Other N-gram Overlap Measures

With the view to measure other N-gram (N=2,3,4) overlap scores a *query pool* and a *sentence pool* is created. In order to create the query pool, we took the query sentences. For each query sentence, we create a set of related sentences by replacing an important word (i.e. nouns, verbs, adjectives and adverbs) by its synonym(s). we created a sentence pool in the same way.

For example, we have a sentence pool for the following sentence:

- *The Frankfurt-based body say in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.*
- The Frankfurt-based *organic structure* say in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.
- The Frankfurt-based body say in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete *picture*.

- The Frankfurt-based body say in its annual report released today that it has decided on two *subjects* for the new currency: history of European civilization and abstract or concrete paintings.

- The Frankfurt-based body say in its annual report released today that it has decided on two *topics* for the new currency: history of European civilization and abstract or concrete paintings.

- The Frankfurt-based body say in its annual *study* released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.

- The Frankfurt-based body say in its annual study released today that it has *determined* on two themes for the new currency: history of European civilization and abstract or paintings.

- The Frankfurt-based body say in its annual study released today that it has *made_up* on two themes for the new currency: history of European civilization and abstract or concrete paintings.

In the same way, we will have a query pool for each of the query sentences. We measure the recall based n-gram (n=2, 3, 4) using the following formula:

$$N-gram\ overlap\ score(S,Q) = \frac{\sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{gram_n \in S} Count(gram_n)} \tag{5.1}$$

$$n-gram\ score = argmax_i(argmax_j\ N-gram\ overlap\ score(s_i,q_j)) \tag{5.2}$$

Where, n stands for the length of the $n-gram(n=2,3,4)$ *and* $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in the query and candidate sentence. q_j is

the j^{th} sentence in the query pool and s_i is the i^{th} sentence in the sentence pool.

Example

Query Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence Despite skepticism about the actual realization of a single European currency as scheduled on *January 1, 1999*, preparations for the design of the Euro note have already begun.

2-gram: 0.14815

3-gram: 0.0800

4-gram: 0

Longest Common Subsequence

A sequence $W = [w_1, w_2, ..., w_n]$ is a subsequence of another sequence $X = [x_1, x_2, ..., x_m]$, if there exists a strict increasing sequence $[i_1, i_2, ..., i_n]$ of indices of X such that for all $j = 1, 2, ..., n$ we have $x_{ij} = w_j$ (Cormen, Leiserson, and Rivest, 1989). Given two sequences S_1 *and* S_2, the longest common subsequence (LCS) of S_1 *and* S_2 is a common subsequence with maximum length (Lin, 2004).

The longer the LCS of two sentences is, the more similar the two sentences are. We used LCS-based F-measure to estimate the similarity between the document sentence S of length m and the query sentence Q of length n as follows:

$$R_{lcs}(S,Q) = \frac{LCS(S,Q)}{m} \tag{5.3}$$

$$P_{lcs}(S,Q) = \frac{LCS(S,Q)}{n} \tag{5.4}$$

$$F_{lcs}(S,Q) = (1-\alpha) \times P_{lcs}(S,Q) + \alpha \times R_{lcs}(S,Q) \tag{5.5}$$

Where, LCS(S,Q) is the length of a longest common subsequence of S and Q and α is a constant that determines the importance of precision and recall. we set the value of α as 0.5 that means the equal importance to precision and recall. We call the Equation 5.5, LCS-based F-measure. Notice that, F_{lcs} is 1 when, S=Q; and F_{lcs} is 0 when there is nothing in common between S and Q.

One advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order as n-grams. The other advantage is that it automatically includes longest in-sequence common n-grams, therefore no predefined n-gram length is necessary. Moreover it has the property that its value is less than or equal to the minimum of unigram (i.e. 1-gram) F-measure of S and Q. Unigram recall reflects the proportion of words in S that are also present in Q; while unigram precision is the proportion of words in Q that are also in S. Unigram recall and precision count all co-occuring words regardless of their orders; while LCS counts in-sequence co-occurrences.

By only awarding credit to in-sequence unigram matches, LCS measure also captures sentence level structure in a natural way. Consider the following example:

S1 *John shot the thief*

S2 John shot the thief

S3 the thief shot John

Using S1 as reference sentence and S2 and S3 as the sentences under consideration, S2 and S3 would have the same 2-gram score, since they both have one bigram (i.e. "the thief") in common with S1. However, S2 and S3 have very different meanings. In case of LCS, S2 has a score of 3/4=0.75 and S3 has a score of 2/4=0.5, with $\alpha = 0.5$. Therefore, S2 is better than S3 according to LCS.

However, LCS suffers one disadvantage that it only counts the main in-sequence words; therefore, other alternative LCSes and shorter sequences are not reflected in the final score. For example, given the following candidate sentence:

S4 the thief John shot

Using S1 as its reference, LCS counts either "the thief" or "John shot", but not both; therefore, S4 has the same LCS score as S3 while, 2-gram would prefer S4 than S3.

In order to measure LCS score for a sentence we took a similar approach as the previous section (i.e. sentence pool and query pool). we calculated the LCS score using the following formula:

$$LCS\ score = argmax_i(argmax_j\ F_{lcs}(s_i, q_j))$$

Where, q_j is the j^{th} sentence in the query pool and s_i is the i^{th} sentence in the sentence pool.

Example:

Query Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence Despite skepticism about the actual realization of a single European currency as scheduled on *January 1, 1999*, preparations for the design of the Euro note have already begun.

LCS Score: 0.27586

Weighted Longest Common Subsequence

LCS has many nice properties as we described in the previous sections. Unfortunately, the basic LCS also has a problem that it does not differentiate LCSes of different spatial relations within their embedding sequences (Lin, 2004). For example, given a reference sequence S and two candidate sequences Y_1 *and* Y_2 as follows:

S: A B C D E F G

Y_1 : A B C D H I K

Y_2 : A H B K C I D

Y_1 *and* Y_2 have the same LCS score. However, Y_1 should be better choice than Y_2 because Y_1 has consecutive matches. To improve the basic LCS method, we can remember the length of consecutive matches encountered so far to a regular two dimensional dynamic program table computing LCS. We call it weighted LCS (WLCS) and use k to indicate the length of the current consecutive matches ending at words x_i and y_j. Given two sentences X and Y, the WLCS score of X and Y can be computed using the similar dynamic programming procedure as stated in (Lin, 2004). We computed the WLCS-based F-measure in the same way as before, using both the question pool and sentence pool.

$$WLCS\ score = argmax_i(argmax_j\ F_{wlcs}(s_i, q_j))$$

Example:

Query Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence Despite skepticism about the actual realization of a single European currency as scheduled on *January 1, 1999*, preparations for the design of the Euro note have already begun.

WLCS Score: 0.15961

Skip-Bigram Measure

Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps. Skip-bigram measures the overlap of skip-bigrams between a candidate sentence and a query sentence (Lin, 2004). Using the example given earlier:

S1 *John shot the thief*

S2 John shoot the thief

S3 the thief shoot John

S4 the thief John shot

each sentence has C(4,2)=6 skip-bigrams[1]. For example, S1 has the following skip-bigrams: ("John shot", "John the", "John thief", "shot the", "shot thief" and "the thief") S2 has three skip bi-gram matches with S1 ("John the", "John thief", "the thief"), S3 has one skip bi-gram match with S1 ("the thief"), and S4 has two skip bi-gram matches with S1 ("John shot", "the thief").

The skip bi-gram score between the document sentence S of length m and the query sentence Q of length n can be computed as follows:

[1] $C(n,r) = \frac{n!}{r! \times (n-r)!}$

$$R_{skip_2}(S,Q) = \frac{SKIP_2(S,Q)}{C(m,2)} \tag{5.6}$$

$$P_{skip_2}(S,Q) = \frac{SKIP_2(S,Q)}{C(n,2)} \tag{5.7}$$

$$F_{skip_2}(S,Q) = (1-\alpha) \times P_{skip_2}(S,Q) + \alpha \times R_{skip_2}(S,Q) \tag{5.8}$$

Where, $SKIP_2(S,Q)$ is the number of skip bi-gram matches between S and Q and α is a constant that determines the importance of precision and recall. We set the value of α as 0.5 that means the equal importance to precision and recall. C is the combination function. We call the equation 5.8 skip bigram-based F-measure. We computed the skip bigram-based F-measure using the formula:

$$SKIP\ BI-GRAM = argmax_i(argmax_j\ F_{skip2}(s_i, q_j))$$

For example, given the following query, the following sentence got skip bi-gram score of 0.05218.

Query Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence Despite skepticism about the actual realization of a single European currency as scheduled on *January 1, 1999*, preparations for the design of the Euro note have already begun.

Note that, skip bi-gram counts all in-order matching word pairs while LCS only counts one longest common subsequence.

We can put the constraint on the maximum skip distance, d_{skip}, between two in-order words that is allowed to form a skip bi-gram to avoid the spurious matches like “the the” or

"of from". For example, if we set d_{skip} to 0 then it is equivalent to bi-gram overlap measure. If we set d_{skip} to 4 then only word pairs of at most 4 words apart can form skip bi-grams. In our experiment, we set $d_{skip} = 4$.

Modifying the equations: 5.6, 5.7 and 5.8 to allow the maximum skip distance limit is straightforward: We count the skip bi-gram matches, $SKIP_2(S,Q)$, within the maximum skip distance and replace the denominators of the equations with the actual numbers of within distance skip bi-grams from the reference sentence and the candidate sentence respectively.

Head Overlap

We can parse a sentence by minipar[2] to get its dependency tree. Figure 5.1 shows an example dependency tree (a portion of the large tree) for the first part of the sentence "*Frankfurt-based body said in its annual report released today* that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings." The links in the diagram represent dependency relationships. The direction of a link is from the head to the modifier in the relationship. Labels associated with the links represent types of dependency relations (i.e. subj, mod etc.). Table 5.1 lists a subset of the relations in minipar outputs.

Relation	Description	Example
subj	subject of a verb	*John* eats rice
mod	adjunct modifier of any type of head	*annual* report
det	determiner of a noun	*the* man
nn	prenomial modifier of a noun	*strong* contestant
pcomp	complement of a preposition	*in* its

Table 5.1: Subset of dependency relations

[2]Available at http://www.cs.ualberta.ca/ lindek/minipar.htm

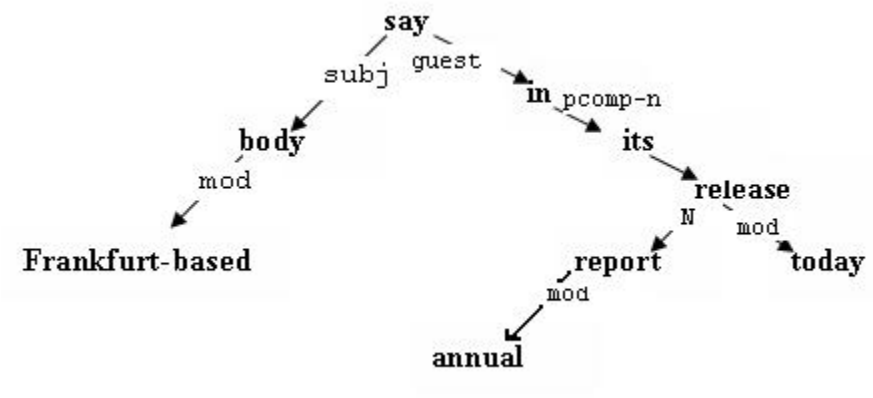

Figure 5.1: Example of a dependency tree

Exact Head Overlap:

Heads in sentences are important words that should be counted when we are measuring the relevancy between two sentences. The number of heads common in between two sentences can indicate how much they are relevant to each other. In order to extract the heads from both query and sentence, the query and the sentence are parsed by minipar. From the parse trees we extract the heads and measure the overlap between them as follows:

$$Exact\ head\ overlap\ score = \frac{\sum_{w_1 \in HeadSet} Count_{match}(w_1)}{\sum_{w_1 \in HeadSet} Count(w_1)}$$

Where HeadSet is the set of head words in the sentence and $Count_{match}$ is the number of matches between the HeadSet of query and sentence.

Head Related-words Overlap:

We take the synonyms, hyponyms and hypernyms of both the query-head words and sentence-head words and measure the overlap similarly using the formula:

$$Head\ related\ words\ score = \frac{\sum_{w_1 \in HeadRelSet} Count_{match}(w_1)}{\sum_{w_1 \in HeadRelSet} Count(w_1)}$$

Where, HeadRelSet is the set of synonyms, hyponyms and hypernyms of head words in the sentence and $Count_{match}$ is the number of matches between the head related sets of the query and the sentence.

Example:

For example, below we list the head words for a query and a sentence and their measures:

Query: Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Heads for Query: include, reaction, take, describe, report, Euro, introduction, press, prediction, 1999, expectation

Sentence: The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.

Heads for Sentence: history, release, currency, body, report,painting, say, abstract, civilization, theme, decide.

Exact Head Score: 0.1

Head Related Score: 0

Basic Element Overlap Measure

We extracted BEs for the sentences in the document collection (see Section 3.2.9 for details). Once we get the BEs for a sentence, we computed the Likelihood Ratio (LR) for each BE (Zhou, Lin, and Hovy, 2005). The LR score of each BE is an information theoretic measure that represents the relative importance in the BE list from the document set

that contains all the texts to be summarized. Sorting the BEs according to their LR scores produced a BE-ranked list.

Our goal is to generate a summary that will answer the questions about a certain topic. The ranked list of BEs in this way contains important BEs at the top which may or may not be relevant to the topic questions. We filter those BEs by checking whether they contain any word which is a *query word* or a *query related word* (i.e. synonyms, hypernyms, hyponyms and gloss words).

For example, for the following sentence:

Query: Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence: The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.

BE "decided|themes|obj" is not considered as it does not contain any word from the query words or query relevant words but BE "report|annual|mod" is taken as it contains a query word *"report"*. In this way, we filter out the BEs those are related to the query.

The score of a sentence is the sum of its BE scores divided by the number of BEs in the sentence. By limiting the number of the top BEs that contribute to the calculation of the sentence scores, we can remove the BEs with little importance and the sentences with many less important BEs. If we set the threshold to 100, it means that only the topmost 100 BEs in the ranked list can contribute to the normalized sentence BE score computation. For this thesis, we did not set any threshold— that means we took all the BEs counted when calculating the BE scores for the sentences.

5.4.2 Lexical Semantic Features

Synonym Overlap

Synonym overlap measure is the overlap between the list of synonyms of the important words extracted from the candidate sentence and *query related words*. This can be computed as follows:

$$Synonym\ Overlap\ Score = \frac{\sum_{w_1 \in SynSet} Count_{match}(w_1)}{\sum_{w_1 \in SynSet} Count(w_1)}$$

Where SynSet is the synonym set of the important words in the sentence and $Count_{match}$ is the number of matches between the SynSet and *query related words*.

Hypernym/Hyponym Overlap

Hypernym/hyponym overlap measure is the overlap between the list of hypernyms (level 2) and hyponyms (level 3) of the nouns extracted from the sentence in consideration and *query related words*. This can be computed as follows:

$$Hypernym/hyponym\ overlap\ score = \frac{\sum_{h_1 \in HypSet} Count_{match}(h_1)}{\sum_{h_1 \in HypSet} Count(h_1)}$$

Where HypSet is the hyponym/hyponym set of the nouns in the sentence and $Count_{match}$ is the number of matches between the HypSet and *query related words*.

Gloss Overlap

Gloss overlap measure is the overlap between the list of important words that are extracted from the gloss definition of the nouns in the sentence in consideration and *query related words*. This can be computed as follows:

$$Gloss\ Overlap\ Score = \frac{\sum_{g_1 \in GlossSet} Count_{match}(g_1)}{\sum_{g_1 \in GlossSet} Count(g_1)}$$

Where GlossSet is the set of important words (i.e. nouns, verbs and adjectives) taken from the gloss definition of the nouns in the sentence and $Count_{match}$ is the number of matches between the GlossSet and query related words.

Example: For example, given the following query, the following sentence got synonym overlap score of 0.33333, hypernym/hyponym overlap score of 0.1860465 and gloss overlap score of 0.1359223.

Query Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.

Statistical Similarity Measures

Statistical similarity measures are based on the co-occurance of similar words in a corpus. We have used two statistical similarity measures:

Dependency-based similarity measure

This method uses the dependency relations among words in order to measure the similarity (Lin, 1998b). It extracts the dependency triples and then uses a statistical approach to measure the similarity. We used the thesaurus provided by (http://www.cs.ualberta.ca/ lindek/downloads.htm). Using the data, one can retrieve most similar words for a given word. The similar words are grouped into clusters.

Note that, for a word there can be more than one cluster. Each cluster represents the sense of the word and its similar words for that sense. So, selecting the right cluster for a word is itself a problem. Our goals are : i) to create a bag of similar words to the query words and ii) once we get the bag of similar words (dependency based) for the query words, we have to measure the overlap score between the sentence words and this bag of words.

Creating Bag of Similar Words:

For each query-word we extract all of its clusters from the thesaurus. Now, in order to determine the right cluster for a query word we measure the overlap score between the *query related words* (i.e. exact words, synonyms, hypernyms/hyponyms and gloss) and the *clusters*. The hypothesis is that, the cluster that has more words common with the query related words is the right cluster. We choose the cluster for a word which has the highest overlap score.

$$Overlap\ score_i = \frac{\sum_{w_1 \in QueryRelatedWords} Count_{match}(w_1)}{\sum_{w_1 \in QueryRelatedWords} Count(w_1)} \tag{5.9}$$

$$Cluster = argmax_i(Overlap\ Score_i) \tag{5.10}$$

where QueryRelatedWords is the set of exact words, synonyms, hyponyms/hypernyms, and gloss words for the words in the query (i.e query words) and $Count_{match}$ is the number of matches between the query related words and the i^{th} cluster of similar words.

Measuring Overlap Score:

Once we get the clusters for the query words, we measured the overlap between the cluster words and the sentence words which we call dependency based similarity measure:

$$DependencyMeasure = \frac{\sum_{w_1 \in SenWords} Count_{match}(w_1)}{\sum_{w_1 \in SenWords} Count(w_1)}$$

Where, SenWords is the set of words for the sentence and $Count_{match}$ is the number of matches between the sentence words and the cluster of similar words.

Proximity-based similarity measure

This similarity is computed based on the linear proximity relationship between words only (Lin, 1998a). It uses the information theoretic definition of similarity to measure the similarity. We used the thesaurus provided by (http://www.cs.ualberta.ca/ lindek/downloads.htm) to measure this feature. The similar words are also grouped into clusters.

We took the similar approach to measure this feature as the previous section except that we used a different thesaurus.

Example: For example, For the following query and sentence we got the following measures:

Query: Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence: The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings.

Dependency-based Similarity Score: 0.0143678

Proximity-based Similarity Score: 0.04054054

So far, we have included the lexical features in other words Bag of Words (BOW) (i.e. without any structural information) features. The task like *query-based summarization* that requires the use of more complex syntactic and semantics, the approaches with only BOW are often inadequate to perform fine-level textual analysis. The importance of syntactic and semantic features in this context is described by (Zhang and Lee, 2003), (Moschitti et al., 2007) and (Moschitti and Basili, 2006).

An effective way to integrate syntactic and semantic structures in machine learning algorithms is the use of *tree kernel* functions (Collins and Duffy, 2001) which has been successfully applied to question classification (Zhang and Lee, 2003), (Moschitti and Basili, 2006). In more complex tasks such as computing the relatedness between the query sentences and the document sentences, to our knowledge no study uses kernel functions to encode syntactic/semantic information.

In the following two subsections, we describe the syntactic and semantic features and how we computed these two similarity measures.

5.4.3 Syntactic Features

In order to calculate the syntactic similarity between the *query* and the *sentence*, we first parse the sentence as well as the query into a syntactic tree using a parser like (Charniak, 1999) and then we calculate the similarity between the two trees using the *tree kernel* (Collins and Duffy, 2001).

Syntactic parsing and building trees

As introduced in Section 3.2.10, syntactic parsing is analyzing a sentence using the grammar rules. Following is an example of a sentence from DUC 2007 parsed with the Charniak

parser:

```
(S1 (S (NP (NNP Anjelina) (NNP Jolie))
  (VP (AUX is)
    (NP (NP (DT the) (NN woman))
      (SBAR (WHNP (WP who))
      (S (VP (ADVP (RB literally))
        (VBD jumped)
        (PP (IN in)
         (NP (NP (DT a) (VBG swimming) (NN pool))
           (PP (IN in) (NP (PRP$ her) (NN ballgown)))))
           (PP (IN after)
             (S (VP (VBG winning)
               (NP (DT a) (NNP Golden) (NNP Globe) (NN Award))
               (PP (IN for)
                (NP (NP (`` ``) (NNP Gia\) ('' ') (NNP \) (POS '))
                (PRN (-LRB- -LRB-) (NP (CD 1998)) (-RRB- -RRB-)))))))))))
     (. .)))
```

In order to use the tree kernel functions for measuring the syntactic similarity, we need to convert the representation above to its corresponding tree. We designed an algorithm to build tree from this representation. The algorithm extracts the substrings based on the number of "left" and "right" parenthesis and creates nodes in the tree for these substrings. We process the substring in the node until we get a single element (terminal or nonterminal). The algorithm is as follows:

Input: The String S that represents the tree
Output: The Tree T
Data: Array *Substrings*[]
Remove the First "(" and the Last ")" from S
Extract the first constituent (node) from S and make it as root (R) of the tree T
Mark R as processed
Substrings[] = extractSubstrings(S)
createChildren(Substrings,R)
while *not all the nodes in the tree are marked as processed* **do**
    Traverse the tree to look for a node N which is not processed
    Str = getSubstring(N) // "Str" is the string represented by the unprocessed node N
    Remove the First "(" from Str
    Remove the Last ")" from Str
    Extract the first constituent (node) from Str and make it as Node (N) of the tree T.
    Mark N as processed.
    Substrings[] = extractSubstrings(Str)
    createChildren(Substrings[],N)
end
return T;

Algorithm 1: Building tree from parenthesis representation

Input: The String S
Output: The Array of Substrings, SubStrs[]
Data: Scalar left, right, begin and end
begin = 0
for $(i = 0; i < length(S); i++)$ **do**
    if $S[i] == '('$ **then**
        left++
    end
    if $S[i] == ')'$ **then**
        right++
    end
    if $left == right$ **then**
        end = i
        push $S[begin, \cdot, end]$ into SubStrs
        begin = i+1
    end
end
return *SubStrs*

Algorithm 2: Extracting substrings from a string

Input: Array SubStrings[]
Input: Scalar P
Output: A subtree T with P as root
Data: Scalar parentNodeNo
foreach $str \in SubStrings$ **do**
    create a new node N with data str
    set P as N's parent in T
    set N as a child of P in T
end
Set the sibling links for the new nodes (N)
return *T*

Algorithm 3: Building subtrees for substrings

Applying the algorithm for example, to the following output of Charniak parser we get the parse tree as in Figure 5.2.

Query: *Include predictions and expectations reported in the press.*

Parser output:

```
(S1 H:0 (S H:0 (VP H:0 (VB H:0 Include)
 (NP H:0 (NP H:2 (NNS H:0 predictions) (CC H:0 and) (NNS H:0 expectations))
  (VP H:0 (VBN H:0 reported) (PP H:0 (IN H:0 in) (NP H:1 (DT H:0 the) (NN
   H:0 press))))))
 (. H:0 .)))
```

Syntactic Similarity Measure using Tree Kernel Function

Above we built the syntactic tree from the parenthesis representation. Every syntactic tree T is represented by an m-dimensional vector $v(T) = (v_1(T), v_2(T), \cdots v_m(T))$, where the i^{th} element $v_i(T)$ is the number of occurrences of the i^{th} tree fragment in tree T. The tree fragments of a syntactic tree are all of its sub-trees which include at least one production with the restriction that no production rules can be broken into incomplete parts. Figure 5.3 shows a portion of the above example tree and its subtrees.

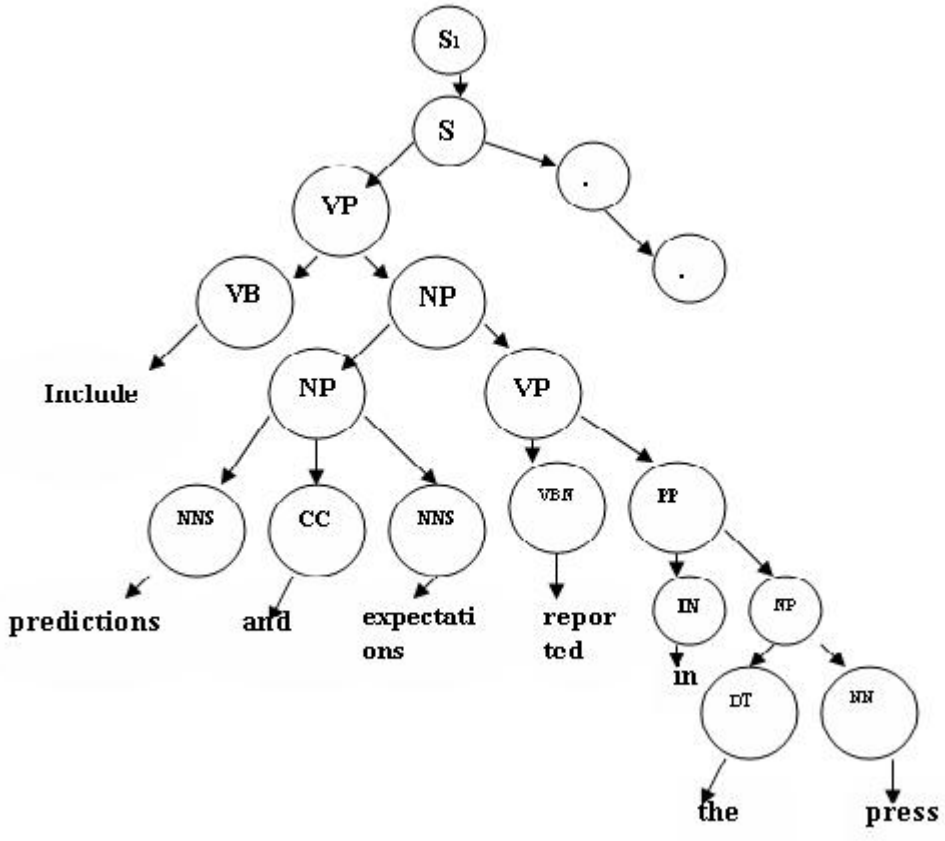

Figure 5.2: Example of syntactic tree

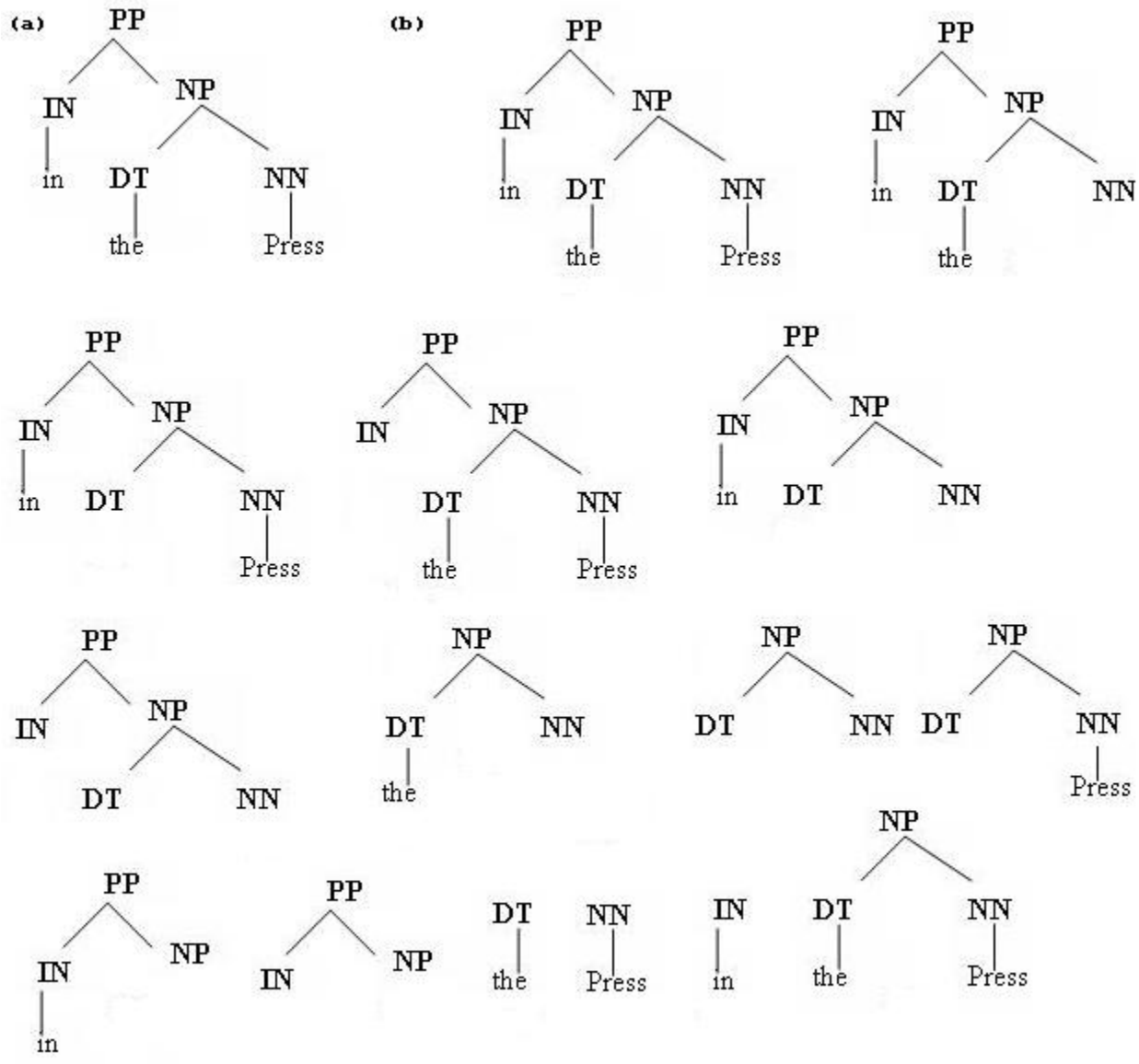

Figure 5.3: (a) Syntactic tree (b) subtrees

Implicitly we enumerate all the possible tree fragments $1, 2, \cdots, m$. These fragments are the axis of this m-dimensional space. Note that this could be done only implicitly, since the number m is extremely large. Because of this, (Collins and Duffy, 2001) defines the tree kernel algorithm whose computational complexity does not depend on m.

The tree kernel of two syntactic trees T_1 and T_2 is actually the inner product of $v(T_1)$ and $v(T_2)$:

$$TK(T_1, T_2) = v(T_1).v(T_2) \tag{5.11}$$

We define the indicator function $I_i(n)$ to be 1 if the sub-tree i is seen rooted at node n and 0 otherwise. It follows:

$$\begin{aligned} v_i(T_1) &= \sum_{n_1 \in N_1} I_i(n_1) \\ v_i(T_2) &= \sum_{n_2 \in N_2} I_i(n_2) \end{aligned}$$

Where N_1 and N_2 are the set of nodes in T_1 and T_2 respectively. So, we can derive:

$$\begin{aligned} TK(T_1, T_2) &= v(T_1).v(T_2) \\ &= \sum_i v_i(T_1) v_i(T_2) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i I_i(n_1) I_i(n_2) \\ &= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} C(n_1, n_2) \end{aligned} \tag{5.12}$$

where we define $C(n_1, n_2) = \sum_i I_i(n_1) I_i(n_2)$. Next, we note that $C(n_1, n_2)$ can be computed in polynomial time, due to the following recursive definition:

1. If the productions at n_1 and n_2 are different then $C(n_1, n_2) = 0$

2. If the productions at n_1 and n_2 are the same, and n_1 and n_2 are pre-terminals, then $C(n_1, n_2) = 1$

3. Else if the productions at n_1 and n_2 are not pre-terminals,

$$C(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j))) \tag{5.13}$$

where, $nc(n_1)$ is the number of children of n_1 in the tree; because the productions at n_1 and n_2 are the same, we have $nc(n_1) = nc(n_2)$. The i^{th} child-node of n_1 is $ch(n_1, i)$.

In cases where the query is composed of two or more sentences, we compute the similarity between the document sentence (s) and each of the query-sentences (q_i) then we take the average of the scores as the syntactic feature value.

$$Syntactic\ similarity\ value = \frac{\sum_{i=1}^{n} TK(q_i, s)}{n}$$

Where n is the number of sentences in the query q and s is the sentence under consideration. *TK* is the similarity value (tree kernel) between the sentence s and the query sentence q based on the syntactic structure. For example, for the following sentence s and query q we get the score:

Query (q): Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence (s): Europe's new currency, the euro, will rival the U.S. dollar as an international currency over the long term, Der Spiegel magazine reported Sunday.

Scores: 90, 41

Average Score: 65.5

5.4.4 Semantic Features

Semantic Role Labeling and Shallow Semantic Parsing

In the previous section, we have given an improvement on BOW by the use of syntactic parses, but these, too are not adequate when dealing with complex questions whose answers are expressed by long and articulated sentences or even paragraphs. Shallow semantic representations, bearing a more compact information, could prevent the sparseness of deep structural approaches and the weakness of BOW models (Moschitti et al., 2007).

Initiatives such as PropBank (PB) (Kingsbury and Palmer, 2002) have made possible the design of accurate automatic Semantic Role Labeling (SRL) systems (Hacioglu et al., 2003). Attempting an application of SRL to QA hence seems natural, as pinpointing the answer to a question relies on a deep understanding of the semantics of both.

For example, consider the PB annotation:

```
[ARG0 all] [TARGET use] [ARG1 the french franc] [ARG2 as their currency]
```

Such annotation can be used to design a shallow semantic representation that can be matched against other semantically similar sentences, e.g.

```
[ARG0 the Vatican] [TARGET uses] [ARG1 the Italian lira]
 [ARG2 as their currency]
```

In order to calculate the semantic similarity between the *query* and the *sentence*, we first represent the annotated sentence/query using the tree structures like Figure 5.4 which we call semantic tree (ST). In the semantic tree, arguments are replaced with the most

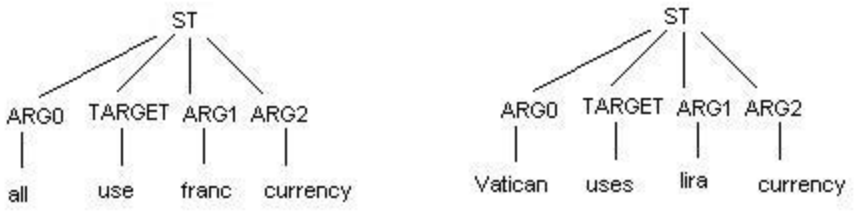

Figure 5.4: Example of semantic trees

important word—often referred to as the semantic head. We look for noun, then verb, then adjective, then adverb to find the semantic head in the argument. If none of these is present, we take the first word of the argument as the semantic head. This reduces the data sparseness with respect to a typical BOW representation.

However, sentences rarely contain a single predicate: it happens more generally that propositions contain one or more subordinate clauses. For instance let us consider a slight modification of the second sentence: "the Vatican, located wholly within Italy uses the Italian lira as their currency." Here, the main predicate is "uses" and the subordinate predicate is "located". The SRL system outputs the following two annotations:

```
(1) [ARG0 the Vatican located wholly within Italy] [TARGET uses]
    [ARG1 the Italian lira] [ARG2 as their currency]
(2) [ARG0 the Vatican] [TARGET located] [ARGM-LOC wholly]
  [ARGM-LOC within Italy] uses the Italian lira as their currency
```

giving the STs in Figure 5.5. As we can see in Figure 5.5(A), when an argument node corresponds to an entire subordinate clause, we label its leaf with ST, e.g. the leaf of ARG0. Such ST node is actually the root of the subordinate clause in Figure 5.5(B). If taken separately, such STs do not express the whole meaning of the sentence, hence it

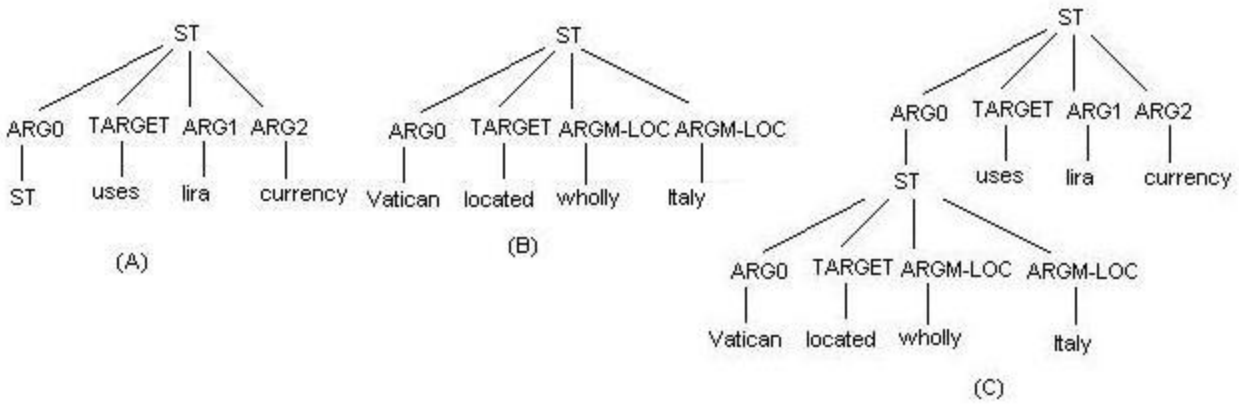

Figure 5.5: Two STs composing a STN

is more accurate to define a single structure encoding the dependency between the two predicates as in Figure 5.5(C). We refer to this kind of nested STs as STNs.

The algorithm to build the semantic tree is as follows:

Input: The Set of Strings S that represent the tree
Output: The Tree T
Build a separate tree for each of the sentences in S.
Assign the "part of" relation by comparing the target with the leaves of other trees.
Assign the "parent-child" relation for the trees.
Find the root tree and denote it as the current tree(CT).
if CT has child(ren) call: addChild(CT,CT).
assign CT as T.
return T.

Algorithm 4: Building semantic tree

Input: CurrentTree,WorkingTree
Output: The Tree T
Initialize one empty queue Q and one empty stack S.
Push the child(ren) of WorkingTree into Q and S.
while *Q is not empty* **do**
    remove the first element (p) from queue Q.
    merge p with the CurrentTree.
end
while *S is not empty* **do**
    pop the element (p) from stack S.
    denote p as the WorkingTree.
    addChild(CurrentTree, WorkingTree)
end
assign CurrentTree as T.
return T.

Algorithm 5: Adding child(ren)

Input: ParTree,TarTree
Output: The Marged ParTree
Search the TARGET of the TarTree in the arguments of ParTree.
Merge the TarTree under that argument.
return ParTree.

Algorithm 6: Merging two trees

Shallow Semantic Tree Kernel (SSTK)

Note that, the tree kernel (TK) function described in Section 5.4.3 computes the number of common subtrees between two trees. Such subtrees are subject to the constraint that their nodes are taken with all or none of the children they have in the original tree. This definition of subtrees makes the TK function appropriate for syntactic trees but for the semantic trees this definition is not well suited. For instance, although the two STs of Figure 5.4 share most of the subtrees rooted in the *ST* node, the kernel defined in Section 5.4.3 computes no match.

The critical aspect of steps: (1), (2) and (3) of the TK function (Section 5.4.3) is that the productions of two evaluated nodes have to be identical to allow the match of further

descendants. This means that common substructures cannot be composed by a node with only some of its children as an effective ST representation would require. (Moschitti et al., 2007) solve this problem by designing the Shallow Semantic Tree Kernel (SSTK) which allows to match portions of a ST.

The SSTK is based on two ideas: first, it changes the ST, as shown in Figure 5.6 by adding *SLOT* nodes. These accommodate argument labels in a specific order i.e. it provides a fixed number of slots, possibly filled with *null* arguments, that encode all possible predicate arguments. Leaf nodes are filled with the wildcard character * but they may alternatively accommodate additional information. The slot nodes are used in such a way that the adopted TK function can generate fragments containing one or more children like for example those shown in Figure 5.6 (b) and (c). As previously pointed out, if the arguments were directly attached to the root node, the kernel function would only generate the structure with all children (or the structure with no children, i.e. empty).

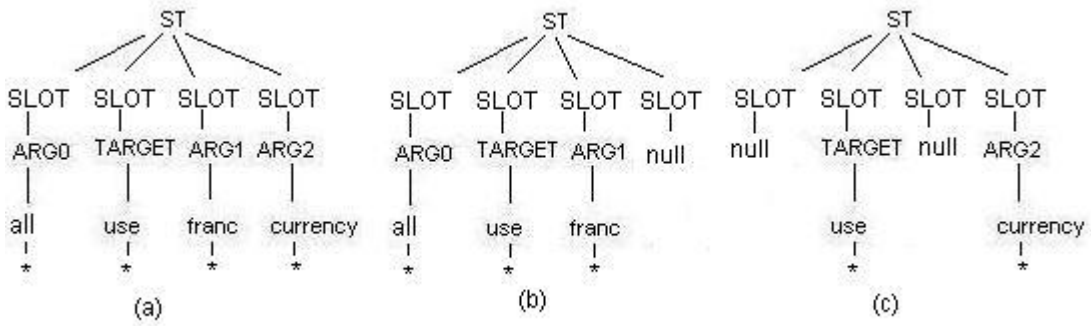

Figure 5.6: Semantic tree with some of its fragments

Second, as the original tree kernel would generate many matches with slots filled with the null label, we have set a new step 0 in the TK calculation:
(0) if n_1 (or n_2) is a pre-terminal node and its child label is *null*, $C(n_1, n_2) = 0$;

and subtract one unit to $C(n_1, n_2)$, in step 3:

$$(3)\ C(n_1, n_2) = \prod_{j=1}^{nc(n_1)} (1 + C(ch(n_1, j), ch(n_2, j))) - 1$$

The above changes generate a new C which, when substituted (in place of original C) in Eq. 5.12, gives the new SSTK.

For example, for the following sentence s and query q we get the semantic score:

Query (q): Describe steps taken and worldwide reaction prior to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press.

Sentence (s): The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency history of European civilization and abstract or concrete paintings.

Scores: 6, 12

Average Score: 9

5.4.5 Graph-based Similarity Measure

In (Erkan and Radev, 2004), the concept of graph-based centrality is used to rank a set of sentences, in producing generic multi-document summaries. A similarity graph is produced for the sentences in the document collection. In the graph, each node represents a sentence. The edges between nodes measure the cosine similarity between the respective pair of sentences. The degree of a given node is an indication of how much important the sentence is. Figure 5.7 shows an example of a similarity graph for 4 sentences.

Once the similarity graph is constructed, the sentences are then ranked according to their eigenvector centrality. The LexRank performed well in the context of generic summa-

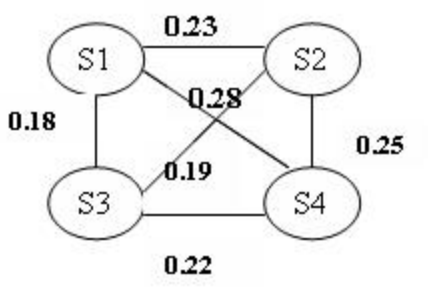

Figure 5.7: LexRank similarity

rization. To apply LexRank to query-focused context, a topic-sensitive version of LexRank is proposed in (Otterbacher, Erkan, and Radev, 2005). We followed a similar approach in order to calculate this feature. The score of a sentence is determined by a mixture model of the relevance of the sentence to the query and the similarity of the sentence to other high-scoring sentences.

Relevance to the question

We first stem out all the sentences in the collection and compute the word IDFs (Inverse Document Frequency) using the following formula:

$$idf_w = log\left(\frac{N+1}{0.5+sf_w}\right)$$

Where, N is the total number of sentences in the cluster, and sf_w is the number of sentences that the word w appears in.

We also stem out the questions and remove the stop words. The relevance of a sentence s to the question q is computed by:

$$rel(s|q) = \sum_{w \in q} log\left(tf_{w,s}+1\right) \times log\left(tf_{w,q}+1\right) \times idf_w$$

Where, $tf_{w,s}$ *and* $tf_{w,q}$ are the number of times w appears in s and q, respectively.

Mixture Model

In the previous section, we measured the relevance of a sentence to the question but a sentence that is similar to the high scoring sentences in the cluster should also have a high score. For instance, if a sentence that gets high score based on the question relevance model is likely to contain an answer to the question, then a related sentence, which may not be similar to the question itself, is also likely to contain an answer (Otterbacher, Erkan, and Radev, 2005).

We capture this idea by the following mixture model:

$$p(s|q) = d \times \frac{rel(s|q)}{\sum_{z \in C} rel(z|q)} + (1-d) \times \sum_{v \in C} \frac{sim(s,v)}{\sum_{z \in C} sim(z,v)} \times p(v|q) \tag{5.14}$$

Where, $p(s|q)$ is the score of a sentence s given a question q, is determined as the sum of its relevance to the question and the similarity to the other sentences in the collection. C is the set of all sentences in the collection. The value of the parameter d which we call "bias", is a trade-off between two terms in the equation and is set empirically. For higher values of d, we prefer the relevance to the question to the similarity to other sentences. The denominators in both terms are for normalization. We measure the cosine similarity weighted by word IDFs as the similarity between two sentences in a cluster:

$$sim(x,y) = \frac{\sum_{w \in x,y} tf_{w,x} \times tf_{w,y} \times (idf_w)^2}{\sqrt{\sum_{x_i \in x} (tf_{x_i,x} \times idf_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (tf_{y_i,y} \times idf_{y_i})^2}}$$

Equation 5.14 can be written in matrix notation as follows:

$$\mathbf{p} = [d\mathbf{A} + (1-d)\mathbf{B}]^T \mathbf{p} \tag{5.15}$$

$\mathbf{A}$ is the square matrix such that for a given index i, all the elements in the i^{th} column

are proportional to $rel(i|q)$. **B** is also a square matrix such that each entry **B***(i,j)* is proportional to *sim(i,j)*. Both matrices are normalized so that row sums add up to 1. Note that as a result of this normalization, all rows of the resulting square matrix $Q = [dA + (1-d)B]$ also add up to 1. Such a matrix is called *stochastic* and defines a Markov chain. If we view each sentence as a state in a Markov chain, then Q(i,j) specifies the transition probability from state *i* to state *j* in the corresponding Markov chain. The vector **p** we are looking for in Eq. 5.15 is the stationary distribution of the Markov chain. An intuitive interpretation of the stationary distribution can be understood by the concept of a random walk on the graph representation of the Markov chain. With probability *d*, a transition is made from the current node to the nodes that are similar to the query. With probability *(1-d)*, a transition is made to the nodes that are lexically similar to the current node. Every transition is weighted according to the similarity distributions. Each element of the vector **p** gives the asymtotic probability of ending up at the corresponding state in the long run regardless of the starting state. The stationary distribution of a markov chain can be computed by a simple iterative algorithm, called *power method* (Erkan and Radev, 2004). The power method algorithm is shown in Algorithm 7. Its starts with a uniform distribution. At each iteration, the eigenvector is updated by multiplying with the transpose of the stochastic matrix. Since the Markov chain is irreducible and aperiodic, the algorithm is guaranteed to terminate. The algorithm for computing graph-based similarity measure is shown in Algorithm 8.

Input: A stochastic, irreducible and aperiodic matrix $\boldsymbol{M}$
Input: matrix size N, error tolerance ε
Output: eigen-vector $\boldsymbol{p}$
$\boldsymbol{p}_0 = \frac{1}{N}1$;
$t = 0$;
repeat
  $t = t + 1$;
  $\boldsymbol{p}_t = \boldsymbol{M}^T \boldsymbol{p}_{t-1}$;
  $\delta = ||\boldsymbol{p}_t - \boldsymbol{p}_{t-1}||$;
until $\delta < \varepsilon$;
return $\boldsymbol{p}_t$;

Algorithm 7: Power method

Input: An array $\boldsymbol{S}$ of n sentences,d,ε
Output: An array $\boldsymbol{L}$ of Graph-based Scores
Data: Array $CosineMatrix[n][n]$, $\boldsymbol{M}[n][n]$, $QuesRelevancyMatrix[n]$, $L[n]$
Data: Scalar $QuesRelScore$, $RowSum$, d, ε
for $i \leftarrow 1$ *to* n **do**
  $QuesRelevancyMatrix[i] = getQuesRelScore(S[i])$;
  $QuesRelScore = QuesRelScore + QuesRelevancyMatrix[i]$;
end
for $i \leftarrow 1$ *to* n **do**
  $QuesRelevancyMatrix[i] = QuesRelevancyMatrix[i]/QuesRelScore$;
end
for $i \leftarrow 1$ *to* n **do**
  for $i \leftarrow 1$ *to* n **do**
    $CosineMatrix[i][j] = idf - modified - cosine(S[i], S[j])$;
  end
end
for $i \leftarrow 1$ *to* n **do**
  $RowSum = sum(CosineMatrix[i])$;
  for $i \leftarrow 1$ *to* n **do**
    $CosineMatrix[i][j] = CosineMatrix[i][j]/RowSum$;
  end
end
$\boldsymbol{M} = d \times QuesRelevancyMatrix + (1-d) \times CosineMatrix$;
$\boldsymbol{L} = PowerMethod(\boldsymbol{M}, n, \varepsilon)$;
return $\boldsymbol{L}$;

Algorithm 8: Computing graph-based similarity measure

5.5 Ranking Sentences

As mentioned earlier in this chapter, we are using several methods in order to rank sentences to generate summaries. In this section, we will describe the systems in detail. The Section 5.5.1 contains the brief description of the system with which we participated in DUC 2007. Section 5.5.2 and Section 5.5.3 contain the description of our machine learning approaches.

5.5.1 DUC 2007 Summarizer: An Experiment with Empirical Approach

In the early stage of my thesis, we developed DUC 2007 summarizer system (Chali and Joty, 2007a) to participate in DUC 2007 competition and gain experience. It was an empirical approach for generating summaries. The summarizer was based on two distinct but complementary concepts:

- how much the sentence is related to the user query and
- how much the sentence is salient to the overall concept.

Keeping these in focus we considered 6 important features:

1. Cosine similarity,
2. Lexical chain,
3. BE overlaps,
4. Question focus overlap,

5. Previous sentence overlaps and

6. Document overlap

We considered *cosine similarity* measure, for computing sentence importance based on the concept of eigenvector centrality in a graph representation of sentences (Erkan and Radev, 2004). *Lexical chains* efficiently identify the theme of the document. An additional argument for considering the chain representation, as opposed to a simple word frequency model, is the case when a single concept is represented by a number of words, each with relatively low frequency. Because the chain combines the number of occurrences of all its members, it can overcome the weight of the single word (Chali and Kolla, 2004). With *BE* represented as a head-modifier-relation triple, one can quite easily decide whether any two units match (express the same meaning) or not considerably more easily than with longer units (Hovy et al., 2006). We considered *question focus overlap* feature to extract the sentences, which are relevant to the topic and narration. We considered the other two features: *previous sentence* overlaps and *document overlap* in order to increase the coherence among the sentences in the summary.

Sentences were ranked by assigning feature-weights empirically. As our DUC 2007 system could not achieve one of the best results, we proposed to fine tune the weights. The next section describes the summarizer system based on our weight learning methodology.

5.5.2 Learning Feature-weights: A Local Search Strategy

This system is an enhanced version of our DUC 2007 summarizer system. Instead of using the six features described above we utilized 18 features extracted in Section 5.4 for each of the sentences in the document collection. In order to fine-tune the weights of the features, we have used a local search technique technique.

Initially, we set all the feature-weights, $w_1, \cdots, w_{18}$, as equal values (i.e. 0.5). Based on the current weights we score the sentences and generate summaries accordingly. We evaluate the summaries using the automatic evaluation tool ROUGE (Lin, 2004) and the ROUGE value works as the feedback to our learning loop. Our learning system tries to maximize the ROUGE score in every step by changing the weights individually. That means, to learn weight w_i, we change the value of w_i keeping all other weight values ($w_j \forall j \neq i$) stagnant. For each weight w_i, the algorithm achieves the local maximum of ROUGE value.

Once we have learned the feature-weights, we compute the final scores for the sentences using the formula:

$$score_i = \vec{x_i}.\vec{w}$$

Where, $\vec{x_i}$ is the feature vector for i-th sentence, $\vec{w}$ is the weight vector and $score_i$ is the score of i^{th} sentence.

Input: A sample of n data-points ($\boldsymbol{x}$) each represented by a feature vector of length L
Input: Stepsize l, Weight Initial Value v
Output: An array $\boldsymbol{w}$ of learned weights
Data: Array of weight values: w_L, Scalar $rg1,\ rg2,\ k,\ flag,\ prev$
Initialize the weight values w_i to v.
for $i \leftarrow$ 1 *to* L **do**
    $rg1 = rg2 = prev = k = flag = 0$
    while *(true)* **do**
        /* Score Sentences and Generate Summaries based on the current weight values */
        scoreSentences($\boldsymbol{w}$)
        generateSummaries()
        /* Get ROUGE value */
        $rg2$ = evaluateROUGE() **if** $rg1 \neq rg2$ *AND* $k \neq 0$ **then**
            flag = 1
        end
        if $rg1 \leq rg2$ **then**
            prev = w_i
            $w_i += l$
            $rg1 = rg2$
        end
        if $rg2 \leq rg1$ *OR* $w_i \geq 1$ **then**
            if *flag = 1* **then**
                w_i=prev
            else
                $w_i = v$
            end
            break
        end
        k++
    end
end
return $\boldsymbol{w}$

Algorithm 9: Tuning weights using local search technique

5.5.3 Statistical Machine Learning Approaches

The method used for selecting sentences for query-based summary extraction, is of type *unsupervised learning*, that means, before the learning begins, it is not known how many subsets (clusters) there are or how they are distinguished from each other. The data points

correspond to the feature vectors of the sentences in the document collection that we extracted in Section 5.4. We experimented with two unsupervised learning techniques with the features extracted in the previous section for the sentence selection problem:

1. K-means learning
2. Expectation Maximization (EM) learning

The K-means Learning

As described in Section 2.3.2, *k-means* is a hard clustering algorithm that defines clusters by the center of mass of their members (Manning and Schutze, 2000). We start with a set of initial cluster centers and go through several iterations of assigning each object to the cluster whose center is closest. After all objects have been assigned, we recompute the center of each cluster as the centroid or mean ($\boldsymbol{\mu}$) of its members. The distance function we use is squared Euclidean distance instead of the true Euclidean distance. Since the square root is a monotonically growing function, squared Euclidean distance has the same result as the true Euclidean distance but the computation overload is smaller when the square root is dropped.

Input: A sample of n data-points ($\boldsymbol{x}$) each represented by a feature vector of length L
Input: Number of Clusters K
Output: An array $\boldsymbol{S}$ of K-means-based Scores
Data: Array $\boldsymbol{d}_{nK}$, $\boldsymbol{\mu}_K$, $\boldsymbol{\Sigma}_K$
Data: Array $\boldsymbol{C}_K$, $\boldsymbol{y}_{nK}$
Randomly choose K data-points as K initial means: $\boldsymbol{\mu}_k,\ k = 1, \cdots, K$.
repeat
  for $i \leftarrow$ 1 *to* n **do**
    for $j \leftarrow$ 1 *to* K **do**

$$\boldsymbol{d}_{ij} \ = \ \left\|\boldsymbol{x}_i - \boldsymbol{\mu}_j\right\|^2 = (\boldsymbol{x}_i - \boldsymbol{\mu}_j)^T (\boldsymbol{x}_i - \boldsymbol{\mu}_j)$$

    end
    if $\boldsymbol{d}_{ik} \ < \ \boldsymbol{d}_{il}, \forall l \neq k$ **then**
      assign $\boldsymbol{x}_i$ to $\boldsymbol{C}_k$.
    end
  end
  for $i \leftarrow$ 1 *to* K **do**
    $\boldsymbol{\mu}_i = \frac{\sum_{\boldsymbol{x}_j \in \boldsymbol{C}_i} \boldsymbol{x}_j}{|\boldsymbol{C}_i|}$
  end
until *no further change occurs* ;
`/* calculating the covariances for each cluster */`
for $i \leftarrow$ 1 *to* K **do**
  $m = |\boldsymbol{C}_i|$
  for $j \leftarrow$ 1 *to* m **do**
    $\boldsymbol{\Sigma}_i \mathrel{+}= \ (\boldsymbol{C}_{ij} - \boldsymbol{\mu}_i) \ * \ (\boldsymbol{C}_{ij} - \boldsymbol{\mu}_i)^T$
  end
  $\boldsymbol{\Sigma}_i \mathrel{*}= \ (1/(m-1))$
end
`/* calculating the scores for sentences */`
for $i \leftarrow$ 1 *to* n **do**
  for $j \leftarrow$ 1 *to* K **do**
    $y_{ij} = \frac{e^{\frac{-1}{2}\left(\boldsymbol{x}_i - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1} \left(\boldsymbol{x}_i - \boldsymbol{\mu}_j\right)}}{\sqrt{2\pi}^d \sqrt{det(\boldsymbol{\Sigma}_j)}}$
  end
  for $j \leftarrow$ 1 *to* K **do**
    $z_{ij} = (y_{ij} * w_j) / \sum_{j=1}^{K} y_{ij} * w_j$; `// where,` $w_j = 1/K$
  end
  $m = max(\boldsymbol{\mu}_k)\ \forall k$
  Push z_{im} *to* $\boldsymbol{S}$
end
return $\boldsymbol{S}$

Algorithm 10: Computing k-means based similarity measure

Ranking the Sentences

Once we have learned the means of the clusters using the k-means algorithm, our next task is to rank the sentences according to a probability model. We have used Bayesian model in order to rank the sentences. Bayes' law says:

$$\begin{aligned} P(q_k|\boldsymbol{x},\Theta) &= \frac{p(\boldsymbol{x}|q_k,\Theta)P(q_k|\Theta)}{p(\boldsymbol{x}|\Theta)} \\ &= \frac{p(\boldsymbol{x}|q_k,\Theta)P(q_k|\Theta)}{\sum_{k=1}^{K} p(\boldsymbol{x}|q_k,\Theta)p(q_k|\Theta)} \end{aligned}$$

where q_k is a class, $\mathbf{x}$ is a feature vector representing a sentence and Θ is the parameter set of all class models. We set the weights of the clusters as equiprobable (i.e. $P(q_k|\Theta) = 1/K$). We calculated $p(\boldsymbol{x}|q_k,\Theta)$ using the gaussian probability distribution. The gaussian probability density function (pdf) for the d-dimensional random variable $\boldsymbol{x}$ is given by:

$$p_{(\boldsymbol{\mu},\boldsymbol{\Sigma})}(\boldsymbol{x}) = \frac{e^{\frac{-1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{\sqrt{2\pi}^d\sqrt{det(\boldsymbol{\Sigma})}}$$

where $\boldsymbol{\mu}$, the mean vector and $\boldsymbol{\Sigma}$, the covariance matrix are the parameters of the gaussian distribution. We get the means ($\boldsymbol{\mu}$) from the k-means algorithm and we calculate the covariance matrix using the unbiased covariance estimation procedure:

$$\hat{\boldsymbol{\Sigma}_j} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^T$$

The EM Learning

The EM algorithm for gaussian mixture models is a well known method for cluster analysis. A useful outcome of this model is that it produces a likelihood value of the clustering model and the likelihood values can be used to select the best model from a number of different models providing that they have the same number of parameters (i.e. same number of clusters).

A significant problem with the EM algorithm is that it converges to a local maximum of the likelihood function and hence the quality of the result depends on the initialization. This problem along with a method for improving the initialization is discussed later in this section.

As described in Section 2.3.3, EM is a "soft" version of k-means algorithm described above (Manning and Schutze, 2000). As the k-means, we start with a set of random cluster centers, $c_1 \cdots c_k$. In each iteration we do a soft assignment of the data-points to every cluster by calculating their membership probabilities. EM is an iterative two step procedure: 1. Expectation-step and 2. Maximization-step. In the expectation step, we compute expected values for the hidden variables $h_{i,j}$ which are cluster membership probabilities. Given the current parameters, we compute how likely it is that an object belongs to any of the clusters. The maximization step computes the most likely parameters of the model given the cluster membership probabilities.

Since, the data-points are considered to be generated by a mixture model of k-gaussians of the form:

$$P(\mathbf{x}) = \sum_{i=1}^{k} P(C=i)P(\mathbf{x}|C=i) \tag{5.16}$$

$$= \sum_{i=1}^{k} P(C=i)P(\mathbf{x}|\boldsymbol{\mu}_i,\boldsymbol{\Sigma}_i) \tag{5.17}$$

where the total likelihood of model Θ with k components given the observed data points, $X = \boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ is:

$$L(\Theta|X) = \prod_{i=1}^{n}\sum_{j=1}^{k} P(C=j)P(\boldsymbol{x}_i|\Theta_j) \tag{5.18}$$

$$= \prod_{i=1}^{n}\sum_{j=1}^{k} w_j P(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j) \tag{5.19}$$

$$\Leftrightarrow \sum_{i=1}^{n} log \sum_{j=1}^{k} w_j P(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j) \text{ (} taking\ the\ log\ likelihood \text{)} \tag{5.20}$$

where P is the probability density function. $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ are the mean and covariance matrix of component j, respectively. Each component contributes a proportion, w_j, of the total population, such that: $\sum_{j=1}^{K} w_j = 1$.

Log likelihood can be used instead of likelihood as it turns the product into sum. The gaussian probability for d-dimensional data point $\boldsymbol{x}_i$ in mixture j is given by:

$$P(\boldsymbol{x}_i|\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j) = \frac{exp^{\frac{-1}{2}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)^T\boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}_i-\boldsymbol{\mu}_j)}}{\sqrt{2\pi}^d\sqrt{det(\boldsymbol{\Sigma}_j)}} \tag{5.21}$$

In the following, we describe the EM algorithm for estimating a gaussian mixture:

Input: A Sample of n data-points ($\boldsymbol{x}$) each represented by a feature vector of length L
Input: Number of Clusters K
Output: An array S of EM-based Scores
Start with K initial Gaussian models: $N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ $k = 1, \cdots, K$, with equal priors set to $P(q_k) = 1/K$.
repeat
`/* Estimation step: compute the probability` $P(q_k^{(i)}|\boldsymbol{x}_j, \Theta^{(i)})$ `for each data point` x_j, $j = 1, \cdots, n$, `to belong to the class` $q_k^{(i)}$ `*/`
for $j \leftarrow$ 1 *to* n **do**
for $k \leftarrow$ 1 *to* K **do**

$$\begin{aligned} P(q_k^{(i)}|\boldsymbol{x}_j, \Theta^{(i)}) &= \frac{P(q_k^{(i)}|\Theta^{(i)}) p(\boldsymbol{x}_j|q_k^{(i)}, \Theta^{(i)})}{p(\boldsymbol{x}_j|\Theta^{(i)})} \\ &= \frac{P(q_k^{(i)}|\Theta^{(i)}) p(\boldsymbol{x}_j|\boldsymbol{\mu}_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)})}{\sum_{k=1}^{K} P(q_k^{(i)}|\Theta^{(i)}) p(\boldsymbol{x}_j|\boldsymbol{\mu}_k^{(i)}, \boldsymbol{\Sigma}_k^{(i)})} \end{aligned}$$

end
end
`/* Maximization step:` `*/`
for $k \leftarrow$ 1 *to* K **do**
for $j \leftarrow$ 1 *to* n **do**
`// update the means:`

$$\boldsymbol{\mu}_k^{i+1} = \frac{\sum_{j=1}^{n} \boldsymbol{x}_j P(q_k^{(i)}|\boldsymbol{x}_j, \Theta^{(i)})}{\sum_{j=1}^{n} P(q_k^{(i)}|\boldsymbol{x}_j, \Theta^{(i)})}$$

`// update the variances:`

$$\boldsymbol{\Sigma}_k^{(i+1)} = \frac{\sum_{j=1}^{n} P(q_k^{(i)}|\boldsymbol{x}_j, \Theta^{(i)})(\boldsymbol{x}_j - \boldsymbol{\mu}_k^{(i+1)})(\boldsymbol{x}_j - \boldsymbol{\mu}_k^{(i+1)})^T}{\sum_{j=1}^{N} P(q_k^{(i)}|\boldsymbol{x}_j, \Theta^{(i)})}$$

`// update the priors:`

$$P(q_k(i+1)|\Theta^{(i+1)}) = \frac{1}{n} \sum_{j=1}^{n} P(q_k^{(i)}|\boldsymbol{x}_j, \Theta^{(i)})$$

end
end
until *the total likelihood increase falls under some desired threshold* ;
return S

Algorithm 11: Computing EM-based similarity measure

Singularities

The covariance matrix, $\mathbf{\Sigma}$ above must be non-singular or invertible. The EM algorithm may converge to a position where the covariance matrix becomes singular ($|\mathbf{\Sigma}| = 0$) or close to singular, that means it is not invertible anymore. If the covariance matrix becomes singular or close to singular then EM may result in wrong clusters. We restrict the covariance matrices to become singular by testing these cases at each iteration of the algorithm as follows:

$$if\ (\sqrt{|\mathbf{\Sigma}|} > 1e^{-9})\ then\ update\ \mathbf{\Sigma}$$
$$else\ do\ not\ update\ \mathbf{\Sigma}$$

Discussion: Starting values for the EM algorithm

The convergence rate and success of clustering using the EM algorithm can be degraded by a poor choice of starting values for the means, covariances and weights of the components. We experimented with one summary (for document number D0703A from DUC 2007). The cluster means are initialized with a heuristic that spreads them randomly around $Mean(DATA)$ with standard deviation $\sqrt{Cov(DATA) * 10}$. Their initial covariance is set to $Cov(DATA)$ and the initial values of the weights are $w_j = 1/K$, where K is the number of clusters. That is, for d-dimensional data-points the parameters of j^{th} component are as follows:

$$
\begin{aligned}
\vec{\mu}_j &= rand(1,\cdots,d) * \sqrt{\mathbf{\Sigma}(DATA) * 10} + \vec{\mu}(DATA) \\
\mathbf{\Sigma}_j &= \mathbf{\Sigma}(DATA) \\
w_j &= 1/K
\end{aligned}
$$

The highly variable nature of the results of the tests is reflected in the very inconsistent values for the total log likelihood (see Figure 5.8) and the results of repeated experiments indicated that using random starting values for initial estimates of the means frequently gave poor results. There are two possible solutions to this problem.

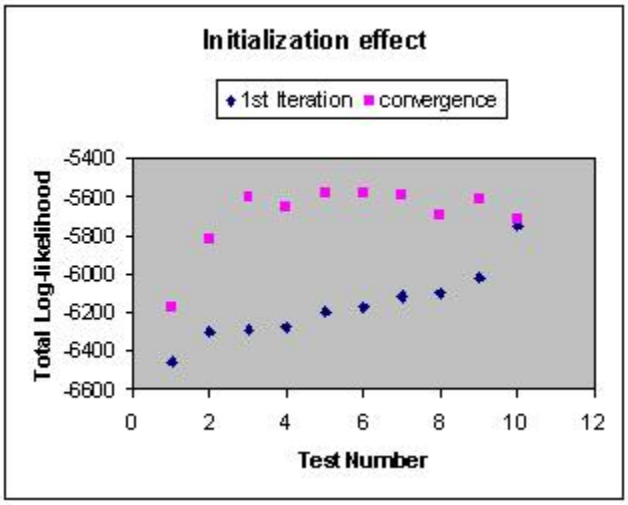

Figure 5.8: Log-likelihood values for random initial mean values

In order to get good results from using random starting values (as specified by the algorithm above) we will run the EM algorithm several times and choose the initial configuration for which we get the maximum log likelihood among all configurations. Choosing the best one among several runs is very computer intensive process. So, to improve the outcome of the EM algorithm on gaussian mixture models it is necessary to find a better method of estimating initial means for the components.

The best starting position for the EM algorithm, in regard to the estimates of the means, would be to have one estimated mean per cluster, which is closer to the true mean of that cluster.

To achieve this aim we explored the widely used "k-means" algorithm as a cluster (means) finding method. That is, the means found by the k-means clustering above will be utilized as the initial means for the EM and we calculate the initial covariance matrices using the unbiased covariance estimation procedure:

$$\hat{\mathbf{\Sigma}}_j = \frac{1}{N-1}\sum_{i=1}^{N}(\vec{x}_i - \vec{\mu}_j)(\vec{x}_i - \vec{\mu}_j)^T$$

Ranking the Sentences

Once the sentences are clustered by the EM algorithm, we filter out the sentences which are question-relevant by checking their probabilities, $P(q_r|\boldsymbol{x}_i, \Theta)$ where, q_r denotes the cluster "question-relevant". If for a sentence $\boldsymbol{x}_i$, $P(q_r|\boldsymbol{x}_i, \Theta) > 0.5$ then $\boldsymbol{x}_i$ is considered to be question-relevant. The cluster which has the mean values greater than the other one is considered as the question-relevant cluster.

Our next task is to rank the question-relevant sentences in order to include them in the summary. This can be done easily by defining a weight vector $\vec{w}$ for the features and multiplying it with the feature vector $\vec{x}_i$ representing the sentence. So, the rank of the sentence (or feature vector) $\vec{x}_i$ is given by:

$$score_i = \vec{x}_i.\vec{w}$$

5.6 Redundancy Checking and Generating Summary

Once the sentences are scored, the easiest way to create summaries is just to output the topmost N sentences until the required summary length is reached. In that case, we are ignoring other factors: such as redundancy and coherence.

As we know that text summarization clearly entails selecting the most salient information and putting it together in a coherent summary. The answer or summary consists of multiple separately extracted sentences from different documents. Obviously, each of the selected text snippets should individually be important. However, when many of the competing sentences are included in the summary, the issue of information overlap between parts of the output comes up, and a mechanism for addressing redundancy is needed. Therefore, our summarization systems employ two levels of analysis: first, a content level, where every sentence is scored according to the features or concepts it covers and second, a textual level, when, before being added to the final output, the sentences deemed to be important are compared to each other and only those that are not too similar to other candidates are included in the final answer or summary. (Goldstein et al., 1999) observed this in what the authors called "Maximum-Marginal-Relevancy (MMR)". Following (Hovy et al., 2006), we modeled this by BE overlap between an intermediate summary and a to-be-added candidate summary sentence.

We call this overlap ratio R, where R is between 0 and 1 inclusively. Setting $R = 0.7$ means that a candidate summary sentence, s, can be added to an intermediate summary, S, if the sentence has a BE overlap ratio less than or equal to 0.7.

5.7 Experimental Evaluation

This section describes the results of experiments conducted using DUC 2007 dataset provided by NIST [3]. Some of the questions these experiments address include:

- How do the different features affect the behavior of the summarizer system?
- Which one of the algorithms (K-means, EM and Local Search) performs better for this particular problem?

5.7.1 DUC Data

The documents of DUC 2007 came from the AQUAINT corpus, comprising newswire articles from the Associated Press and New York Times (1998-2000) and Xinhua News Agency (1996-2000). NIST assessors developed topics of interest to them and choose a set of 25 documents relevant (document cluster) to each topic.

Each topic and its document cluster were given to 4 different NIST assessors, including the developer of the topic. The assessor created a 250-word summary of the document cluster that satisfies the information need expressed in the topic statement. These multiple "reference summaries" are used in the evaluation of summary content.

5.7.2 Evaluation Measures

Evaluation methods determine the quality of the summaries based on the overlap with reference summaries. *Precision (P)* and *Recall (R)* are the widely used measures computed based on the number of units (i.e. sentences, words, etc) common to both system-generated and reference summaries. *Recall* is defined as the ratio of the number of units

[3] National Institute of Standards and Technology

(sentences/words) of the system-generated summaries in common with the reference summaries to the total number of units in the *reference* summary while *precision* is the ratio of the number of units of system-generated summaries in common with the reference summaries to the total number of units in the *system-generated* summaries.

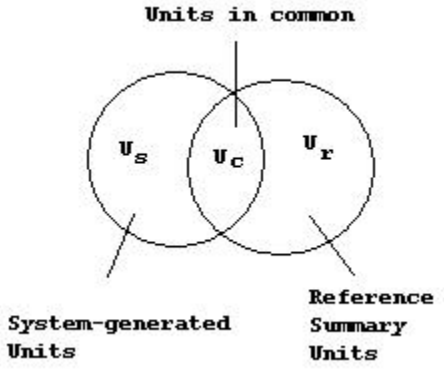

Figure 5.9: Precision and recall

In applications like IR, one can generally trade off precision and recall (one can select every document in the collection and get 100% recall but very low precision, etc.) but sometimes such a tradeoff does not make sense in many applications of NLP. For this reason, a widely used measure in NLP, F-measure combines precision and recall into a single measure of overall performance.

$$P = \frac{U_c}{U_s + U_c}$$
$$R = \frac{U_c}{U_r + U_c}$$
$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}}$$

Where P is the precision, R is recall and α is a factor which determines the weighting of precision and recall. A value of $\alpha = 0.5$ is often chosen for equal weighting of P and R.

With this α value, F measure simplifies to $2PR/(R+P)$.

We carried out automatic evaluation of our summaries using ROUGE (Lin, 2004).

5.7.3 ROUGE

ROUGE which stands for "Recall-Oriented Understudy for Gisting Evaluation" is a collection of measures that determines the quality of a summary by comparing it to reference summaries created by humans. The measures count the number of overlapping units such as n-gram, word-sequences, and word-pairs between the system-generated summary to be evaluated and the ideal summaries created by humans. ROUGE measures considered in the evaluation are: ROUGE-N (N=1,2,3,4), ROUGE-L, ROUGE-W and ROUGE-S.

ROUGE-N

ROUGE-N is n-gram recall between a candidate summary and a set of reference summaries which is computed as follows:

$$ROUGE-N = \frac{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in ReferenceSummaries} \sum_{gram_n \in S} Count(gram_n)}$$

where n is the length of $n-grams$ and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a set of reference summaries.

When multiple references are used, we compute pairwise summary-level ROUGE-N between a candidate summary s and every reference, r_i, in the reference set. Final ROUGE-N score is then obtained by taking the maximum of the summary-level ROUGE-N scores.

$$ROUGE-N_{multi} = argmax_i(ROUGE-N(r_i, s))$$

In the ROUGE evaluation package, the assessors use a jackknifing procedure. Given M references, they compute the best score over M sets of M-1 references. The final ROUGE-N score is the average of the M ROUGE-N scores using different M-1 references. For example, let us consider a document having 4 sentences, $A_1, \cdots A_4$ comprising 20 words, $w_1, \cdots, w_{20}$ as follows:

$A_1 :\ w_1\ w_2\ w_3\ w_4\ w_5$

$A_2 :\ w_6\ w_7\ w_8\ w_9\ w_{10}$

$A_3 :\ w_{11}\ w_{12}\ w_{13}\ w_{14}\ w_{15}$

$A_4 :\ w_{16}\ w_{17}\ w_{18}\ w_{19}\ w_{20}$

Given one peer summary S_1 and three reference summaries R_1, R_2 and R_3 as follows:

$S_1 :\ A_1\ A_2\ A_3$

$R_1 :\ A_1\ A_3\ A_4$

$R_2 :\ A_2\ A_4$

$R_3 :\ A_1\ A_2$

Let us denote $ROUGE(X \backslash R)$ as the ROUGE score of summary X without considering reference summary R then using jackknifing procedure $ROUGE-1$ score for peer summary S_1 can be computed as follows:

$$
\begin{aligned}
ROUGE-1(S_1 \backslash R_1) &= max(ROUGE-1(S_1,R_2), ROUGE-1(S_1,R_3)) \\
&= max(|A_2|/(|A_2|+|A_4|), (|A_1|+|A_2|)/(|A_1|+|A_2|)) \\
&= max(5/10, 10/10) \\
&= 1 \\
ROUGE-1(S_1 \backslash R_2) &= max(ROUGE-1(S_1,R_1), ROUGE-1(S_1,R_3)) \\
&= max((|A_1|+|A_3|/(|A_1|+|A_3|+|A_4|), (|A_1|+|A_2|)/(|A_1|+|A_2|)) \\
&= max(10/15, 10/10) \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
ROUGE-1(S_1 \backslash R_3) &= max(ROUGE-1(S_1,R_1), ROUGE-1(S_1,R_2)) \\
&= max((|A_1|+|A_3|/(|A_1|+|A_3|+|A_4|), |A_2|/(|A_2|+|A_4|)) \\
&= max(10/15, 5/10) \\
&= 0.66 \\
ROUGE-1(S_1)(Avg) &= 0.887
\end{aligned}
$$

ROUGE-L: Longest Common Subsequence

As described in Section 5.4.1, given two sequences $Z = [z_1, z_2, \cdots z_n]$ and $X = [x_1, x_2, \cdots x_m]$, Z is said to be the subsequence of X if there exists a strict increasing sequence $[i_1, i_2, \cdots, i_n]$ of indices of X such that for all $j = 1, 2, \cdots, n$, we have $x_{ij} = z_j$. Given two sequences A and B, the *LCS* is the common subsequence with maximum length.

Sentence-level LCS-based F-measure to estimate the similarity between two summary sentences r of length m and s of length n (assuming r is a reference summary sentence and s is a peer summary sentence) is as follows (Lin, 2004):

$$\begin{aligned} R_{lcs} &= \frac{LCS(r,s)}{m} \\ P_{lcs} &= \frac{LCS(r,s)}{n} \\ F_{lcs} &= \frac{P_{lcs}R_{lcs}}{\alpha R_{lcs} + (1-\alpha)P_{lcs}} \end{aligned}$$

Consider the following example (Lin, 2004):

S_1 : *Police killed the gunman*

S_2 : *Police kill the gunman*

S_3 : *The gunman kill police*

Using S_1 as reference, both S_2 and S_3 have the same ROUGE-2 score even when they differ in meaning. This can be differentiated using the ROUGE-L measure. S_2 has the ROUGE-L value as $3/4$ and S_3 has $1/2$ with $\alpha = 0.5$.

Summary-level LCS can be obtained by taking the union LCS matches between a reference summary, r_i and every candidate summary sentence, s_j. Given a reference summary of u sentences containing a total of m words and a candidate summary, S of v sentences containing a total of n words, the summary-level LCS-based F-measure can be computed as follows:

$$
\begin{aligned}
R_{lcs} &= \frac{\sum_{i=1}^{u} LCS \cup (r_i, S)}{m} \\
P_{lcs} &= \frac{\sum_{i=1}^{u} LCS \cup (r_i, S)}{n} \\
F_{lcs} &= \frac{P_{lcs} R_{lcs}}{\alpha R_{lcs} + (1-\alpha) P_{lcs}}
\end{aligned}
$$

For example, if $r_i = w_1, w_2, w_3, w_4, w_5$ and S contains two sentences s_1 and s_2 as follows:

$s_1 :\ w_1\ w_3\ w_8\ w_9\ w_5$

$s_2 :\ w_1\ w_2\ w_3\ w_8\ w_5$

We get the LCS-Score:

$LCS(r_i, s_1) :\ w_1\ w_3\ w_5$

$LCS(r_i, s_2) :\ w_1\ w_2\ w_3\ w_5$

$LCS - Score :\ 4/5$

ROUGE-W: Weighted Longest Common Subsequence(WLCS)

Consider the following example:

X: [A B C D E F G]

Y1: [A B C D K J L]

Y2: [A H B K C I D]

Both sentences, Y1 and Y2 have the same ROUGE-L score of 4/7 ($\alpha = 0.5$) with X as the reference. This would not reward the sentence Y1, which has consecutive sequence

of words, as compared with Y2. ROUGE-W provides an improvement to the basic LCS method of computation by using the function $f(n)$ to credit the sentences having the consecutive matches of words. F-measure based on WLCS can be calculated as follows:

$$
\begin{aligned}
R_{wlcs} &= f^{-1}\left(\frac{WLCS(X,Y)}{f(m)}\right) \\
P_{wlcs} &= f^{-1}\left(\frac{WLCS(X,Y)}{f(n)}\right) \\
F_{wlcs} &= \frac{P_{wlcs}R_{wlcs}}{\alpha R_{wlcs} + (1-\alpha)P_{wlcs}}
\end{aligned}
$$

By computing the ROUGE-W measure for the two candidate sentences in the above example using $f(k) = k^2$, we obtain scores of 0.571 and 0.286 for Y1 and Y2 respectively. This enables us to differentiate between the two sentences based on the spatial distance between the sequence of the words.

ROUGE-S: Skip-Bigram

Skip-bigram is any pair of words in their sentence order, allowing for arbitrary gaps (Lin, 2004). ROUGE-S measures the overlap of skip-bigrams between a candidate summary and a set of reference summaries. Using the example given in Section 5.7.3:

S_1 : *Police killed the gunman*

S_2 : Police kill the gunman

S_3 : The gunman kill police

S_4 : the gunman police killed

Each sentence has $C(4,2) = 6$ skip-bigrams. For example, S_1 has the following skip-bigrams: *(*"police killed", "police the", "police gunman", "killed the", "killed gunman", "the gunman").

S_2 has three skip-bigram matches with S_1 (*"police the", "police gunman", "the gunman"), S_3 has one skip-bigram match with S_1 ("the gunman"), and S_4 has two skip-bigram matches with S_1 ("police killed", "the gunman").

The skip bi-gram score between the candidate summary sentence S of length m and the reference summary sentence R of length n can be computed as follows:

$$R_{skip_2} = \frac{SKIP_2(S,R)}{C(m,2)} \tag{5.22}$$

$$P_{skip_2} = \frac{SKIP_2(S,R)}{C(n,2)} \tag{5.23}$$

$$F_{lcs} = \frac{P_{skip_2} R_{skip_2}}{\alpha R_{skip_2} + (1-\alpha) P_{skip_2}} \tag{5.24}$$

Where, $SKIP_2(S,Q)$ is the number of skip bi-gram matches between S and R and α is a constant that determines the importance of precision and recall. We set the value of α as 0.5 that means the equal importance to precision and recall. C is the combination function.

ROUGE-S is extended with the addition of unigram as counting unit which is called ROUGE-SU (Lin, 2004).

5.7.4 Experiments

Experimental Setup

We generated summaries for the 45 topics in DUC 2007 and experimented with the performance of the machine learning techniques: local search, EM, k-means for the task of query-based multi-document summary extraction. We also experimented with the effects of different types of features in generating good summaries for each of the algorithms. To assess the contribution of different features in generating quality summaries the features are grouped into seven classes. For each of the methods, we generated summaries for each of the systems as described below:

The **LEX** system generates summaries based on only lexical features: n-gram (n=1,2,3,4), LCS, WLCS, skip bi-gram, head, head synonym and BE overlap.

The **LEXSEM** system considers only lexical semantic features: synonym, hypernym/hyponym, gloss, dependency-based and proximity-based similarity.

The **SYN** system generates summary based on only syntactic feature.

The **COS** system generates summary based on the graph-based method.

The **SYS1** system considers all the features except the syntactic and semantic features.

The **SYS2** system considers all the features except the semantic feature and

the **ALL** system generates summaries taking all the features into account.

ROUGE parameters were set as the same as DUC 2007 evaluation setup. All the ROUGE measures were calculated by running ROUGE-1.5.5 with stemming but no removal of stopwords .

ROUGE run-time parameters:

ROUGE-1.5.5.pl -2 -1 -u -r 1000 -t 0 -n 4 -w 1.2 -m -l 250 -a

The K-means Learning

This section describes our experimental evaluation for the k-means algorithm.

ROUGE-N Measures

ROUGE-1

Table 5.2 shows the ROUGE-1 scores for different combinations of features in the k-means learning. It is noticable that the k-means performs best for the graph-based cosine similarity feature. Note that, including syntactic feature does not improve the score while including syntactic and semantic features increases the score but not a significant amount. Summaries based on only lexical features give us good ROUGE-1 evaluation.

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.366130	0.360584	0.346485	0.378852	0.376102	0.365689	0.366989
Precision	0.397352	0.393334	0.378840	0.408700	0.403340	0.415188	0.415395
F-score	0.381037	0.376133	0.361847	0.393097	0.389245	0.388695	0.389695

Table 5.2: ROUGE-1 measures in k-means learning

ROUGE-2

Table 5.3 shows the ROUGE-2 scores for different combinations of features in the k-means learning. Just like ROUGE-1 graph-based cosine similarity feature performs good here. We get a significant improvement in ROUGE-2 score when we include syntactic feature with all other features. Semantic feature does not affect the score much. Lexical Semantic features perform good here.

ROUGE-3 and ROUGE-4

Table 5.4 and Table 5.5 show the ROUGE-3 and ROUGE-4 scores. For the two measures, we get best scores when we consider all features except semantic (SYS2).

ROUGE-L, ROUGE-W and ROUGE-SU

As Table 5.6, Table 5.7 and Table 5.8 show: the ROUGE-L, ROUGE-W and ROUGE-

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.074488	0.076827	0.063012	0.085583	0.074625	0.077936	0.076737
Precision	0.080886	0.084017	0.069260	0.092514	0.080753	0.107239	0.109562
F-score	0.077543	0.080241	0.065973	0.088891	0.077568	0.090269	0.090258

Table 5.3: ROUGE-2 measures in k-means learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.021390	0.023441	0.015861	0.028721	0.023376	0.027165	0.026775
Precision	0.023241	0.025699	0.017619	0.031123	0.021238	0.059437	0.057156
F-score	0.022274	0.024513	0.016691	0.029867	0.022256	0.037288	0.036467

Table 5.4: ROUGE-3 measures in k-means learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.009388	0.010417	0.007346	0.013869	0.009837	0.012388	0.012388
Precision	0.010209	0.011449	0.008196	0.015055	0.010692	0.018749	0.015892
F-score	0.009780	0.010907	0.007747	0.014435	0.010247	0.014919	0.013923

Table 5.5: ROUGE-4 measures in k-means learning

SU scores are best for all features without syntactic and semantic, and including syntactic/semantic features with other features does not improve the scores rather degrades the scores. Summaries based on only lexical features achieve good scores.

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.337800	0.331936	0.313296	0.346491	0.348463	0.340751	0.340980
Precision	0.366650	0.362165	0.342599	0.373882	0.393244	0.401221	0.401664
F-score	0.351575	0.346288	0.327208	0.359562	0.369502	0.368522	0.368843

Table 5.6: ROUGE-L measures in k-means learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.097519	0.096689	0.090205	0.100854	0.099167	0.097279	0.097013
Precision	0.194786	0.194422	0.181533	0.200458	0.237490	0.241206	0.237345
F-score	0.129912	0.129085	0.120458	0.134129	0.139912	0.138643	0.137730

Table 5.7: ROUGE-W measures in k-means learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.131197	0.127442	0.116302	0.139002	0.135857	0.134871	0.134734
Precision	0.154859	0.152065	0.139711	0.162368	0.176021	0.174262	0.174370
F-score	0.141963	0.138527	0.126825	0.149630	0.153353	0.152057	0.152011

Table 5.8: ROUGE-SU measures in k-means learning

The EM learning

Table 5.9 to Table 5.15 show different ROUGE measures for the feature combinations in the context of the EM learning. It can be easily noticed that for all these measures we get significant amount of improvement in ROUGE scores when we include syntactic

and semantic features along with other features. We get 3-15% improvement over SYS1 in F-score when we include syntactic feature and 2-24% improvement when we include syntactic and semantic features. Cosine similarity measure does not perform that well as it did in the k-means experiments. Summaries considering only the lexical features achieve good results.

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.383635	0.357353	0.346485	0.375816	0.379908	0.399173	0.398786
Precision	0.415776	0.390724	0.378840	0.406517	0.411503	0.411127	0.399358
F-score	0.398973	0.373172	0.361847	0.390465	0.395075	0.405062	0.399072

Table 5.9: ROUGE-1 measures in EM learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.088564	0.079678	0.063012	0.087056	0.084658	0.089204	0.090564
Precision	0.095792	0.087032	0.069260	0.094192	0.091732	0.116060	0.138243
F-score	0.092018	0.083171	0.065973	0.090462	0.088053	0.100875	0.109436

Table 5.10: ROUGE-2 measures in EM learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.031811	0.022991	0.015861	0.029427	0.027791	0.030895	0.031344
Precision	0.034267	0.025108	0.017619	0.031910	0.031190	0.061676	0.061117
F-score	0.032987	0.023997	0.016691	0.030612	0.028908	0.041168	0.041437

Table 5.11: ROUGE-3 measures in EM learning

Local Search Technique

The ROUGE scores based on the feature combinations are given in Table 5.16 to Table 5.22. Summaries generated by including all features perform the best scores for all the measures.

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.016970	0.010058	0.007346	0.014910	0.013263	0.016564	0.016479
Precision	0.018238	0.010981	0.008196	0.016208	0.014413	0.017971	0.017529
F-score	0.017578	0.010497	0.007747	0.015528	0.013814	0.017239	0.016988

Table 5.12: ROUGE-4 measures in EM learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.354123	0.328675	0.313296	0.344994	0.348606	0.351070	0.340892
Precision	0.383792	0.359347	0.342599	0.373265	0.377634	0.399535	0.401865
F-score	0.368281	0.343217	0.327208	0.358481	0.362540	0.373738	0.368876

Table 5.13: ROUGE-L measures in EM learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.102680	0.095728	0.090205	0.100541	0.101989	0.101529	0.100874
Precision	0.205188	0.192882	0.181533	0.200486	0.203428	0.221940	0.222817
F-score	0.136794	0.127893	0.120458	0.133868	0.135863	0.139323	0.138876

Table 5.14: ROUGE-W measures in EM learning

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.145570	0.128454	0.116015	0.138369	0.143064	0.145246	0.143858
Precision	0.171446	0.153288	0.139414	0.162077	0.167845	0.186296	0.185299
F-score	0.157339	0.139623	0.126533	0.149153	0.154467	0.163230	0.161970

Table 5.15: ROUGE-SU measures in EM learning

We get 7-15% improvement over SYS1 in F-score when we include syntactic feature and 8-19% improvement over SYS1 in F-score when we include syntactic and semantic features. In this case also "lexical features" perform well but not better than "All features".

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.379911	0.358329	0.346485	0.375816	0.382146	0.388092	0.390038
Precision	0.411176	0.390928	0.378840	0.406517	0.414278	0.434648	0.438886
F-score	0.394821	0.373785	0.361847	0.390465	0.397564	0.410053	0.413023

Table 5.16: ROUGE-1 measures in local search technique

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.085592	0.079904	0.063012	0.087056	0.086746	0.095388	0.099076
Precision	0.092626	0.087123	0.069260	0.094192	0.093983	0.114568	0.116389
F-measure	0.088948	0.083334	0.065973	0.090462	0.090220	0.104102	0.107037

Table 5.17: ROUGE-2 measures in local search technique

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.028875	0.023277	0.015861	0.029427	0.029303	0.035392	0.037140
Precision	0.031203	0.025368	0.017619	0.031910	0.031706	0.050452	0.051903
F-score	0.029987	0.024271	0.016691	0.030612	0.030457	0.041601	0.043298

Table 5.18: ROUGE-3 measures in local search technique

5.7.5 Comparison

Table 5.23 shows the F-scores of all the ROUGE measures for one baseline system, the best system in DUC 2007 and our three learning algorithms taking all features into consideration. The baseline system generates summaries by returning all the leading sentences (up

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.014463	0.010268	0.007346	0.014910	0.014072	0.014889	0.015251
Precision	0.015654	0.011182	0.008196	0.016208	0.015256	0.027320	0.029803
F-score	0.015032	0.010703	0.007747	0.015528	0.014640	0.019274	0.020177

Table 5.19: ROUGE-4 measures in local search technique

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.349695	0.330177	0.313296	0.344994	0.350893	0.362318	0.363528
Precision	0.378526	0.360162	0.342599	0.373265	0.380452	0.415256	0.427125
F-score	0.363443	0.344396	0.327208	0.358481	0.365075	0.386985	0.392769

Table 5.20: ROUGE-L measures in local search technique

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.101698	0.095864	0.090205	0.100541	0.102449	0.104421	0.105361
Precision	0.202618	0.192696	0.181533	0.200486	0.204464	0.245776	0.247038
F-score	0.135353	0.127966	0.120458	0.133868	0.136502	0.146570	0.147720

Table 5.21: ROUGE-W measures in local search technique

Scores	LEX	LEXSEM	SYN	COS	SYS1	SYS2	ALL
Recall	0.143767	0.128871	0.116015	0.138369	0.145345	0.148896	0.150174
Precision	0.168582	0.153138	0.139414	0.162077	0.170838	0.195479	0.196615
F-score	0.155044	0.139792	0.126533	0.149153	0.157064	0.169037	0.170285

Table 5.22: ROUGE-SU measures in local search technique

to 250 words) in the $\langle TEXT \rangle$ field of the most recent document(s). It shows that the local search technique outperforms the other two and the EM algorithm performs better than the k-means algorithm. Comparing with the DUC 2007 participants our systems achieve top scores and for some ROUGE measures (ROUGE-3, ROUGE-4, ROUGE-SU) there is no statistically significant difference between our system and the best DUC 2007 system.

Algrothms	ROUGE 1	ROUGE 2	ROUGE 3	ROUGE 4	ROUGE L	ROUGE W	ROUGE SU
Baseline	0.334750	0.064900	0.018560	0.007940	0.310740	0.113810	0.112780
Best System	0.43889	0.122850	0.045450	0.021780	0.405610	0.153600	0.174700
K-means	0.389695	0.0890258	0.036467	0.013923	0.368843	0.137730	0.152011
EM	0.399072	0.109436	0.041437	0.016988	0.368876	0.138876	0.161970
Local Search	0.413023	0.107037	0.043298	0.020177	0.392769	0.147720	0.170285

Table 5.23: ROUGE F-scores for different systems

5.8 Chapter Summary

In this chapter, we presented our works on answering complex questions. We extracted eighteen important features for each of the sentences in the document collection to measure its relevancy to the query. Later we used a simple local search technique to fine-tune the feature weights. We used the statistical clustering algorithms: EM and k-means to select the relevant sentences for summary generation.

From our experiments, it is obvious that our systems achieve better results when we include the tree kernel based syntactic and semantic features though summaries based on only syntactic or semantic feature do not achieve good results. Graph-based cosine similarity and lexical semantic features are also important for selecting relevant sentences. As

our lexical features include the ROUGE measures, summaries based on only lexical features achieve good results also. Of the three learning algorithms, the local search technique ourperforms the other two and the EM performs better than the k-means based learning.

As for this research problem, our training data were not labeled so we could not use the supervised learning algorithms. Our future plan is to experiment with the supervised learning techniques (i.e. SVM, MAXENT) and see how they perform for this problem.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

6.1.1 Simple Question Answering

Our method for answering simple questions creates queries from the questions to extract most relevant passages using the information retrieval system. The most important part of our system is our question classifier, which uses rules we determined by manually observing a testbed of sample questions. Our system classifies questions by question type, and the type of the answer. The documents retrieved earlier are then tagged to enable our system to extract different information from the documents. Named entities are one of the types of information extracted from the documents. Some of these named entities will represent the answer type of the question. Our system then extracts possible answers that fit the type of answer, dictated by the type of question, from the document set. After the possible answers are extracted, our system uses an answer ranking formula to choose the answer ranked most probable by our system. Our system ranks the answers according to various criteria that we determined gave our system the best accuracy on the test set of questions as compared to other formulae we tried.

Our method of question answering does not include any deep processing of the documents, and the run time is dependent on how many documents are retrieved for the question being asked. In TREC 2007 evaluation, our system achieved almost 26% accuracy for the factoid questions which was ranked fourth among 51 participants and the average F-measure for list questions was 0.132 which was ranked sixth. Our system achieved the "average pyramid F-score" (with $\beta = 3$) of 0.030 for the "Other" questions which is a

poor score. In TREC 2007, we focused mainly on the factoid and the list questions. For the "Other" questions, we extracted the important facts about the target and gave the top-ranked fact as the answer while "Other" questions need a list of non-redundant facts about the target. This may be the reason that our system could not achieve good scores in "Other" questions.

Taking all three types of questions for a topic into consideration, our system achieved average per-series score of 0.1410 which is almost the same as the 10th system in TREC 2007. These results show that we have a relatively good accuracy compared to the other systems that participated in TREC 2007.

6.1.2 Complex Question Answering

We investigated several methods for answering complex questions. The problem can be rephrased as query-based summarization. In the early stage of my study, we participated in DUC 2007 competition with a system which is based on six features: a) cosine similarity, b) lexical chain, c) BE overlap, d) question-focus overlap, e) previous sentence overlap and f) document overlap. We assigned the feature-weights empirically for ranking the sentences. As the system could not achieve one of the best scores, we proposed an enhanced version based on eighteen features and a local search technique to fine tune the feature-weights. We also experimented with two unsupervised machine learning approaches for the same problem. Our extracted features include: 1) lexical , 2) lexical semantic, 3) cosine similarity, 4) syntactic, and 5) shallow-semantic features.

Our local search technique (i.e. hill climbing) tries to maximize the ROUGE score in every step by changing the weights individually. For each weight w_i, the algorithm achieves the local maximum of the ROUGE value. In this way, once we learn the weights we rank the sentences by multiplying the feature-vector with the weight-vector. We then generate

the summaries by taking the top *N* sentences.

We also experimented with two unsupervised learning techniques: 1) EM and 2) k-means with the features extracted.

We assume that we have two clusters of sentences: 1. query-relevant and 2. query-irrelevant. We learned the means of the clusters using the k-means algorithm then we used Bayesian model in order to rank the sentences and generate the summaries accordingly (i.e. by taking the top *N* sentences).

The learned means in the k-means algorithm are used as the initial means in the EM algorithm. We applied the EM algorithm to cluster the sentences into two classes : 1) query-relevant and 2) query-irrelevant. We take out the query-relevant sentences and rank them using the learned weights (i.e. in local search). We then filter out the non-redundant sentences using a redundancy checking module and generate summaries.

We also experimented with the effects of different kinds of features. Our evaluation measures show that our systems achieve better results when we include tree kernel based syntactic and semantic features. Our evaluations show that the local search technique outperforms the other two and EM performs better than the k-means learning. As the local search technique is a kind of supervised learning so it outperforms the other two unsupervised techniques. Comparing with the DUC-2007 participants, our systems achieve top scores and there is no statistically significant difference between our system and the top DUC 2007 system.

6.2 Future Work

6.2.1 Simple Question Answering

Though in this year we improved our rank in all three types of questions but our scores are not the best. We will have to improve the accuracy. We discovered that as the taggers and parsers were not trained on questions, they tagged incorrectly in many settings. Tagging questions correctly will lead to more accurate classifications.

We did not include the syntactic and semantic similarity measures between the possible answers and the questions in ranking answers. We have the plan to include them in future.

As explained earlier our system processed the top 100 (50 AQUAINT-2 + 50 BLOG06) relevant documents provided by TREC instead of using the whole collection. Our system did not find answers (i.e. NIL) for 153 factoid questions while there were only 16 "NIL" answers. Our system could find 8 correct "NIL" answers. So, we have the plan to extract answers from the whole collection in future.

With the knowledge we gained from classifying questions, we could experiment with machine learning techniques for classifying questions. We could develop patterns with machine learning to extract the question focus, and use it to classify the question. Machine learning could also be used to discover patterns in the document with which to extract answers. If we could collect a complete list of patterns we could extract possible answers using both patterns and named entities. Then our system would extract a list of possible answers that contain a better percentage of correct answers.

6.2.2 Complex Question Answering

As addressed earlier, we experimented with one hill climbing local search technique and two unsupervised learning techniques. As for this research problem, our training data were not labeled so we could not use the supervised learning algorithms. Our future plan is to label the data if possible and experiment with the supervised learning techniques (i.e. SVM, MAXENT) and see how do they perform for this particular problem.

While our QA systems used several features to find relevant answers we expect that by decomposing complex questions into the sets of subquestions that they entail, systems can improve the average quality of answers returned and achieve better coverage for the question as a whole. We have the plan to decompose the complex questions into several simple questions before we find the similarity measures between the document sentence and the query sentence.

The future starts now. 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.

References

Ageno, A., D. Ferres, E. Gonzalez, S. Kanaan, H. Rodriguez, M. Surdeanu, and J. Turmo. 2004. TALP-QA system at TREC-2004: Structural and Hierarchical Relaxation over Semantic Constraints. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Ahn, D., V. Jijkoun, J. Kamps, G. Mishne, K. Muller, M. de Rijke, and S. Schlobach. 2004. The University of Amsterdam at TREC2004. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Amigo, E., J. Gonzalo, V. Peinado, A. Peinado, and F. Verdejo. 2004. An Empirical Study of Information Synthesis Tasks. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 207–es, Barcelona, Spain.

Barzilay, R. and M. Elhadad. 1997. Using Lexical Chains for Text Summarization. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and the 8th European Chapter Meeting of the Association for Computational Linguistics, Workshop on Intelligent Scalable Test Summarization*, pages 10–17, Madrid.

Berger, A., 2001. *Statistical Machine Learning for Information Retrieval*. Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Bilmes, J. 1997. A Gentle Tutorial on the EM algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. In *Technical Report ICSI-TR-97-021*, University of Berkeley.

Brill, E. 1994. Some Advances in Transformation-based Part of Speech Tagging. In *National Conference on Artificial Intelligence*, pages 722–727, Seattle, Washington.

Brown, P. F., S. D. Pietra, D. Pietra, and R. L. Mercer. 1991. Word sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 264–270.

Cannataro, M. and D. Talia. 2003. The Knowledge Grid. *Communications of the ACM*, 46(1):89–93.

Chali, Y. and S. Dubien. 2004. University of Lethbridges participation in TREC-2004 QA track. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Chali, Y. and S. R. Joty. 2007a. University of Lethbridge's participation in DUC-2007 main task. In *Proceedings of the Document Understanding Conference*, Rochester. NIST.

Chali, Y. and S. R. Joty. 2007b. University of Lethbridge's Participation in TREC-2007 QA Track. In *Proceedings of the sixteenth Text REtrieval Conference*, Gaithersburg. NIST.

Chali, Y. and S. R. Joty. 2007c. Word Sense Disambiguation Using Lexical Cohesion. In *Proceedings of the 4th International Conference on Semantic Evaluations*, pages 476–479, Prague. ACL.

Charniak, E. 1999. A Maximum-Entropy-Inspired Parser. In *Technical Report CS-99-12*, Brown University, Computer Science Department.

Chen, J., G. He, Y. Wu, and S. Jiang. 2004. UNT at TREC 2004: Question Answering Combining Multiple Evidences. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Chu-Carroll, J., K. Czuba, J. Prager, A. Ittycheriah, and S. Blair-Goldensohn. 2004. IBM's PIQUANT II in TREC2004. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Clarke, C. L. A, G. V. Cormack, and T. R. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365, New Orleans, Louisiana.

Clifton, T. and W. Teahan. 2004. Bangor at TREC 2004: Question Answering Track. In *Proceedings of the Thirteenth Text REtreival Conference*, Gaithersburg, Maryland.

Collins, M. and N. Duffy. 2001. Convolution Kernels for Natural Language. In *Proceedings of Neural Information Processing Systems*, pages 625–632, Vancouver, Canada.

Cormen, T. R., C. E. Leiserson, and R. L. Rivest, 1989. *Introduction to Algorithms*. The MIT Press.

Dagan, I., Y. Karov, and D. Roth. 1997. Mistake-driven learning in text categorization. In *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, Brown University, Providence, Rhode Island.

Dang, H. T., D. Kelly, and J. Lin. 2007. Overview of the TREC 2007 Question Answering Track. In *proceedings of the 16th Text REtrieval Conference*, Gaithersburg, Maryland. NIST.

Diekema, A. R., O. Yilmazel, and E. D. Liddy. 2004. Evaluation of Restricted Domain Question-Answering Systems. In *Proceedings of EACL Workshop on Question Answering in Restricted Domains*, pages 2–7, Barcelona, Spain.

Dubien, S., 2005. *Question Answering Using Document Tagging and Question Classification*. M.Sc. Thesis, Department of Computer Science, University of Lethbridge, Canada.

Echihabi, A., U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. 2003. Multiple-Engine Question Answering in TextMap. In *Proceedings of the Twelfth Text REtreival Conference*, pages 772–781, Gaithersburg, Maryland.

Erkan, G. and D. R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Ferret, O., B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux, I. Robba, and A. Vilnat. 2001. Finding an answer based on the recognition of the question focus. In *Proceedings of the Tenth Text REtreival Conference*, Gaithersburg, Maryland.

Galley, M. and K. McKeown. 2003. Improving Word Sense Disambiguation in Lexical Chaining. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 1486–1488, Acapulco, Mexico.

Goldstein, J., M. Kantrowitz, V. Mittal, and J. Carbonell. 1999. Summarizing Text Documents: Sentence Selection and Evaluation Metrics. In *Proceedings of the 22nd International ACM Conference on Research and Development in Information Retrieval, SIGIR*, pages 121–128, Berkeley, CA.

Hacioglu, K., S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. 2003. Shallow Semantic Parsing Using Support Vector Machines. In *Technical Report TR-CSLR-2003-03*, University of Colorado.

Halliday, M. and R. Hasan, 1976. *Cohesion in English*. Longman, London.

Harabagiu, S., F. Lacatusu, and A. Hickl. 2006. Answering complex questions with random walk models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 220 – 227. ACM.

Harabagiu, S. M. and S. J. Maiorano. 1999. Finding Answers in Large Collections of Texts: Paragraph Indexing + Abductive Inference. In *AAAI Fall Symposium on Question Answering Systems*, pages 63–71, North Falmouth, Massachusetts.

Hartigan, J. A. and M. A. Wong. 1979. A K-means Clustering Algorithm. *Applied Statistics*, 28:100–108.

Haruno, M., S. Shirai, and Y. Ooyama. 1998. Using decision trees to contruct a practical parser. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 505–511, Montreal, Canada.

Hermjakob, U. 2001. Parsing and Question Classification for Question Answering. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, pages 17–22, Toulouse, France.

Hirst, G., C. DiMarco, E. Hovy, and K. Parsons. 1997. Authoring and generating health-education documents that are tailored to the needs of the individual patient. In *Proceedings of the Sixth International Conference on User Modeling*, pages 107–118, Sardinia, Italy.

Hirst, G. and D. St-Onge. 1997. Lexical chains as representation of context for the detection and correction of malapropisms. In *Christiane Fellbaum, editor, WordNet: An Electronic Lexical Database and Some of its Applications*, pages 305–332. MIT press.

Hovy, E., C. Y. Lin, L. Zhou, and J. Fukumoto. 2006. Automated Summarization Evaluation with Basic Elements. In *Proceedings of the Fifth Conference on Language Resources and Evaluation*, Genoa, Italy.

Hughes, J., 1994. *Automatically Acquiring a Classification of Words*. Ph.D thesis, School of Computing, University of Leeds.

Ibrahimov, O., I. Sethi, and N. Dimitrova. 2001. Clustering of imperfect transcripts using a novel similarity measure. In *Proceedings of the Special Interest Group on Information Retrieval Workshop on Information Retrieval Techniques for Speech Applications*, pages 23–35, New Orleans, LA.

Jijkoun, V. and M. de Rijke. 2004. Answer Selection in a multi-stream open domain question answering system. In *Proceedings 26th European Conference on Information Retrieval*, volume 2997 of LNCS, pages 99–111, Springer.

Jijkoun, V., G. Mishne, C. Monz, M. de Rijke, S. Schlobach, and O. Tsur. 2003. The University of Amsterdam at the TREC 2003 Question Answering Track. In *Proceedings of the Twelfth Text REtreival Conference*, pages 586–593, Gaithersburg, Maryland.

Jurafsky, D. and J. H. Martin, 2000. *Speech and Language Processing*. Prentice Hall.

Katz, B., J. Lin, D. Loreto, W. Hildebrandt, M. Bilotti, S. Felshin, A. Fernandes, G. Marton, and F. Mora. 2003. Integrating Web-Based and Corpus-Based Techniques for Question Answering. In *Proceedings of the Twelfth Text REtreival Conference*, pages 426–435, Gaithersburg, Maryland.

Kim, H., K. Kim, G. G. Lee, and J. Seo. 2001. MAYA: A Fast Question-answering System Based on A Predictive Answer Indexer. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, pages 9–16, Toulouse, France.

Kingsbury, P. and M. Palmer. 2002. From Treebank to PropBank. In *Proceedings of the international conference on Language Resources and Evaluation*, Las Palmas, Spain.

Kolla, M., 2004. *Automatic Text Summarization using Lexical Chains: Algorithms and Experiments*. M.Sc. Thesis, Department of Computer Science, University of Lethbridge, Canada.

Kwok, C., O. Etzioni, and D. Weld. 2001. Scaling question answering to the Web. In *World Wide Web*, pages 150–161, Hong-Kong.

Lewis, D., R. E. Schapire, J.P. Callan, , and R. Papka. 1996. Training algorithms for linear text classifiers. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval, SIGIR*, pages 298–306, Zurich, Switzerland.

Li, J., L. Sun, C. Kit, and J. Webster. 2007. A Query-Focused Multi-Document Summarizer Based on Lexical Chains. In *Proceedings of the Document Understanding Conference*, Rochester. NIST.

Li, X. and D. Roth. 2001. Exploring Evidence for Shallow Parsing. In *Proceedings of the Fifth Workshop on Computational Language Learning*, pages 1–7, Toulouse, France.

Li, X. and D. Roth. 2002. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, pages 1–7, Taipei,Taiwan.

Lin, C. Y. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of Workshop on Text Summarization Branches Out, Post-Conference Workshop of Association for Computational Linguistics*, pages 74–81, Barcelona, Spain.

Lin, D. 1998a. An Information-Theoretic Definition of Similarity. In *Proceedings of International Conference on Machine Learning*, pages 296–304, Madison, Wisconsin.

Lin, D. 1998b. Automatic Retrieval and Clustering of Similar Words. In *Proceedings of the International Conference on Computational Linguistics and Association for Computational Linguistics*, pages 768–774, Montreal, Canada.

Lin, J., A. Fernandes, B. Katz, G. Marton, and S. Tellex. 2003. Extracting Answers from the Web Using Knowledge Annotation and Knowledge Mining Techniques. In *Proceedings of the Eleventh Text REtreival Conference*, Gaithersburg, Maryland.

Magnini, B., M. Negri, R. Prevete, and H. Tanev. 2002. Is it the right answer? Exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 425–432, Philadelphia, PA.

Mani, I., 2001. *Automatic Summarization*. John Benjamins Co, Amsterdam/Philadelphia.

Mani, I. and M. Maybury, 1999. *Advances in Automatic Text Summarization*. MIT Press.

Manning, C. D. and H. Schutze, 2000. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Marcus, M. P., B. Santorini, and M. A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Maybury, M. and A. Merlino. 1997. Multimedia summaries of broadcast news. In *Proceedings of the IASTED International Conference on Intelligent Information Systems*, pages 442–449. IEEE Computer Society.

McKeown, K., R. Barzilay, J. Chen, D. Elson, D. Evans, J. Klavans, A. Nenkova, B. Schiffman, and S. Sigelman. 2003. Columbia's newsblaster: New features and future directions (demo). In *Proceedings of the Annual Meeting of the North American Association for Computational Linguistics*, Edmonton, Canada.

McKeown, K., R. Desmond, A. Jordan, and V. Hatzivassiloglou. 1998. Generating patient-specific summaries of online literature. In *AAAI 98 Spring Symposium on Intelligent Text Summarization*, pages 34–43, Stanford University.

Mirkin, B., 2005. *Clustering for Data Mining: A Data Recovery Approach*. Boca Raton F1, Chapman and Hall/CRC.

Moldovan, D., C. Clark, and M. Bowden. 2007. Lymbas PowerAnswer 4 in TREC 2007. In *Proceedings of the 16th Text REtreival Conference*, Gaithersburg, Maryland.

Moldovan, D., S. Harabagiu, C. Clark, M. Bowden, J. Lehmann, and J. Williams. 2004. Experiments and Analysis of LCC's two QA Systems over TREC2004. In *Proceedings of the 13th Text REtrevial Conference*, Gaithersburg, Maryland.

Moldovan, D., S. Harabagiu, R. Girju, P. Morarescu, F. Lactusu, A. Novischi, A. Badulescu, and O. Bolohan. 2002. LCC Tools for Question Answering. In *Proceedings of the Eleventh Text REtreival Conference*, Gaithersburg, Maryland.

Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. 2000. The Structure and Performance of an Open-Domain Question Answering System. In *38th Annual Meeting of the Association for Computational Linguistics*, pages 563–570, Hong Kong.

Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus. 1999. LASSO: A Tool for Surfing the Answer Net. In *Proceedings of the 8th Text REtreival Conference*, Gaithersburg, Maryland.

Moldovan, D. and A. Novischi. 2002. Lexical Chains for Question Answering. In *Proceedings of the International Conference on Computational Linguistics*, pages 674–680, Taipei, Taiwan.

Molla, D. and M. Gardiner. 2004. AnswerFinder at TREC 2004. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Molla, D. and S. Wan. 2006. Macquarie University at DUC 2006:Question Answering for Summarisation. In *Proceedings of the Document Understanding Conference*. NIST.

Morris, J. and G. Hirst. 1991. Lexical cohesion computed by thesaural relations as an indicator of structure of text. *Computational Linguistics*, 17(1):21–48.

Moschitti, A. and R. Basili. 2006. A Tree Kernel approach to Question and Answer Classification in Question Answering Systems. In *Proceedings of the 5th international conference on Language Resources and Evaluation*, Genoa, Italy.

Moschitti, A., S. Quarteroni, R. Basili, and S. Manandhar. 2007. Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classificaion. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 776–783, Prague, Czech Republic. ACL.

Munoz, M., V. Punyakanok, D. Roth, and D. Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of the joint SIGDAT Conference on Emperical Methods in Natural Language Processing and Very Large Corpora*, pages 168–178, Maryland.

Nyberg, E., T. Mitaamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupsc, L. V. Lita, V. Pedro, D. Svoboda, and B. Van Durme. 2003. The JAVELIN Question-Answering System at TREC 2003: A Multi-Strategy Approach with Dynamic Planning. In *Proceedings of the Twelfth Text REtreival Conference*, Gaithersburg, Maryland.

Otterbacher, J., G. Erkan, and D. R. Radev. 2005. Using Random Walks for Question-focused Sentence Retrieval. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 915–922, Vancouver, Canada.

Pasca, M. and S. M. Harabagiu. 2001. Answer Mining from On-Line Documents. In *Proceedings of the Association for Computational Linguistics 39th Annual Meeting and 10th Conference of the European Chapter Workshop on Open-Domain Question Answering*, pages 38–45, Toulouse, France.

Pinchak, C. and S. Bergsma. 2007. Automatic Answer Typing for How-Questions. In *Proceedings of Human Language Technologies and the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 516 – 523, Rochester, New York.

Pinchak, C. and D. Lin. 2006. A Probabilistic Answer Type Model. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 393 – 400, Trento, Italy.

Pingali, P., Rahul K., and V. Varma. 2007. IIIT Hyderabad at DUC 2007. In *Proceedings of the Document Understanding Conference*, Rochester. NIST.

Porter, M. F. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Prager, J., J. Chu-Carroll, K. Czuba, C. Wlty, A. Ittycheriah, and R. Mahindru. 2003. IBM's PIQUANT in TREC2003. In *Proceedings of the Twelfth Text REtreival Conference*, pages 283–292, Gaithersburg, Maryland.

Radev, D. R., S. Blair-Goldensohn, Z. Zhang, and R. S. Raghavan. 2001. Newsinessence: A system for domain-independent, real-time news clustering and multi-document summarization. In *Demo Presentation, Human Language Technology Conference*, San Diego, CA.

Ramshaw, L. and M. Marcus. 1995. Text Chunking Using Transformation-Based Learning. In D. Yarovsky and K. Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 82–94, Somerset, New Jersey. Association for Computational Linguistics.

Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Conference of Empirical Methods in Natural Language Processing*, pages 133–142, University of Pennsylvania.

Ravichandran, D. and E. Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual meeting of the association for Computational Linguistics*, pages 41–47, Philadelphia, PA.

Rohall, S. L., D. Gruen, P. Moody, M. Wattenberg, M. Stern, B. Kerr, B. Stachel, K. Dave, R. Armes, and E. Wilcox. 2004. Remail: a reinvented email prototype. In *Extended abstracts of the 2004 conference on Human factors and computing systems*, pages 791–792. ACM Press.

Roth, D. and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics*, pages 1136–1142, Montreal, Canada.

Roussinov, D., Y. Ding, and J. A. Robles-Flores. 2004. Experiments with Web QA System and TREC2004 questions. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Russel, S. and P. Norvig, 2003. *Artificial Intelligence A Modern Approach, 2nd Edition*. Prentice Hall.

Sekine, S. 2002. Proteus Project OAK System (English Sentence Analyzer), http://nlp.nyu.edu/oak.

Silber, H. G. and K. F. McCoy. 2002. Efficiently Computed Lexical Chains As an Intermediate Representation for Automatic Text Summarization. *Computational Linguistics*, 28(4):487–496.

Strzalkowski, T. and S. Harabagiu, 2008. *Advances in Open Domain Question Answering*. Springer.

Tanaka, H. 1996. Decision tree learning algorithm with structural attributes: Application to verbal case frame acquisition. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 943–948, Copenhagen, Denmark.

Tanev, H., M. Kouylekov, and B. Magnini. 2004. Combining Linguistic Processing and Web Mining for Question Answering: ITC-irst at TREC-2004. In *Proceedings of the Thirteenth Text REtreival Conference*, Gaithersburg, Maryland.

Teahan, W. J. 2003. Knowing about knowledge: Towards a framework for knowledgeable agents and knowledge grids. Technical report, Artificial Intelligence and Intelligent Agents Tech Report AIIA 03.2, School of Informatics, University of Wales, Bangor.

Toutanova, K., C. Brockett, M. Gamon, J. Jagarlamudi, H. Suzuki, and L. Vanderwende. 2007. The PYTHY Summarization System: Microsoft Research at DUC 2007 . In *proceedings of the Document Understanding Conference*, Rochester. NIST.

Voorhees, E. M. 1999. Overview of the TREC 1999 Question Answering Track. In *Proceedings of the 8th Text REtreival Conference*, Gaithersburg, Maryland.

Voorhees, E. M. 2000. Overview of the TREC 2000 Question Answering Track. In *Proceedings of the 9th Text REtreival Conference*, Gaithersburg, Maryland.

Voorhees, E. M. 2001. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text REtreival Conference*, Gaithersburg, Maryland.

Voorhees, E. M. 2002. Overview of the TREC 2002 Question Answering Track. In *Proceedings of the Eleventh Text REtreival Conference*, Gaithersburg, Maryland.

Voorhees, E. M. 2003. Overview of the TREC 2003 Question Answering Track. In *Proceedings of the 12th Text REtreival Conference*, pages 54–68, Gaithersburg, Maryland.

Voorhees, E. M. 2004. Overview of the TREC 2004 Question Answering Track. In *Proceedings of the 13th Text REtreival Conference*, pages 52–62, Gaithersburg, Maryland.

Voorhees, E. M. 2005. Using question series to evaluate question answering system effectiveness. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 299–306, Vancouver, Canada.

Waibel, A., M. Bett, M. Finke, and R. Stiefelhagen. 1998. Meeting browser: tracking and summarization meetings. In *Proceedings of the 1998 DARPA Broadcast News Workshop*, Lansdowne, Virginia.

Waltz, D. L. 1978. An English Language Question Answering System for a Large Relational Database. *Communications of the ACM*, 21(7):526–539.

Weiss, S.M., C. Apte, F.J. Damerau, D.E. Johnson, F.J. Oles, T. Goetz, and T. Hampp. 1999. Maximizing text-mining performance. In *IEEE Intelligent Systems*, pages 63–69.

Wu, L., X. Huang, L. You, Z. Zhang, X. Li, and Y. Zhou. 2004. FDUQA on TREC2004 QA Track. In *Proceedings of the 13th Text REtreival Conference*, Gaithersburg, Maryland.

Wu, M., X. Zheng, M. Duan, T. Liu, and T. Strzalkowski. 2003. Question Answering By Pattern Matching, Web-Proofing, Semantic Form Proofing. In *Proceedings of the Twelfth Text REtreival Conference*, pages 578–585, Gaithersburg, Maryland.

Xu, J., A. Licuanan, J. May, S. Miller, and R. Weischedel. 2002. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of the Eleventh Text REtreival Conference*, pages 96–101, Gaithersburg, Maryland.

Xu, J., A. Licuanan, and R. Weischedel. 2003. TREC 2003 QA at BBN: Answer Definitional Questions. In *Proceedings of the Twelfth Text REtreival Conference*, pages 98–108, Gaithersburg, Maryland.

Zhang, A. and W. Lee. 2003. Question Classification using Support Vector Machines. In *Proceedings of the Special Interest Group on Information Retrieval*, pages 26–32, Toronto, Canada. ACM.

Zhou, L., C. Y. Lin, and E. Hovy. 2005. A BE-based Multi-dccument Summarizer with Query Interpretation. In *Proceedings of Document Understanding Conference*, Vancouver, B.C. Canada.

Appendix-A

Question Type Examples

Example of Questions Types

When Questions

- *WHEN DAY* - Pattern "*When is*"
  - Examples:
    - When is hurricane season in the Caribbean?
    - When is the summer solstice?
- *WHEN YEAR* - No pattern. Anything not a *WHEN DAY*
  - Examples:
    - When was Florence Nightingale born?
    - When was the Nobel prize first given?

Who Questions

- *WHO DEFINITION* - Pattern "*Who [is or was] [NAME]?*"
  - Examples:
    - Who is William Wordsworth?
    - Who is Langston Hughes?
- *WHO LIST* - Pattern "*Who are*"
  - Examples:
    - Who are professional female boxers?
    - Who are 3 authors who have written books about near death experiences?
- *WHO FACTOID* - All the rest.
  - Examples:
    - Who is the prophet of the religion of Islam?
    - Who invented paper?

Where Questions

- *WHERE SCHOOL* - Pattern "*college, university, degree*"
  - Examples:
    - Where did David Ogden Stiers get his undergraduate degree?
    - Where did Bill Gates go to college?

- *WHERE LOCATION* - All the rest
  - Examples:
    * Where is Amsterdam?
    * Where is Tufts University?

How Questions

- *HOW LARGE* - Pattern "*How [big or large]*"
  - Examples:
    * How big is Mars?
    * How big does a pig get?
- *HOW LATE* - Pattern "*How late*"
  - Examples:
    * How late is Disneyland open?
    * How late in pregnancy will airlines let you fly?
- *HOW ACCURATE* - Pattern "*How accurate*"
  - Examples:
    * How accurate are HIV tests?
- *HOW DISTANCE* - Pattern "*How [far or tall or wide or short or high or close or deep]*"
  - Examples:
    * How wide is the Milky Way Galaxy?
    * How far away is the moon?
    * How tall is Tom Cruise?
- *HOW OFTEN* - Pattern "*How [often or frequent]*"
  - Examples:
    * How often does the men's soccer World Cup take place?
    * How often does Hale Bopp comet approach the Earth?
- *HOW LONG* - Pattern "*How long*"
  - Examples:
    * How long did the Charles Manson murder trial last?
    * How long does it take to travel from Tokyo to Niigata?

- *HOW MUCH* - Pattern "*How much*"
  - Examples:
    - How much vitamin C should you take in a day?
    - How much is the Sacajawea coin worth?
- *HOW TEMP* - Pattern "*How [warm or cold or hot]*"
  - Examples:
    - How hot is the core of the Earth?
    - How hot is the sun?
- *HOW FAST* - Pattern "*How fast*"
  - Examples:
    - How fast is the speed of light?
    - How fast is the world spinning?
- *HOW OLD* - Pattern "*How old*"
  - Examples:
    - How old was George Washington when he died?
    - How old must you be to become President of the United States?
    - How old was Elvis when he died?
- *HOW DEATH* - Pattern "*How did [NAME] die?*"
  - Examples:
    - How did Anne Frank die?
    - How did John Dillinger die?
- *HOW METHOD* - No pattern. Questions not classified yet are classified as this.
  - Examples:
    - How did Hawaii become a state?
    - How did Cincinnati get its name?

What questions

- *WHAT DEF* - Patterns "*What ((is) or (are)) [TERM TO BE DEFINED]*"
  - Examples:
    - What is an atom?
    - What are invertebrates?

- *WHAT ACRO* - Patterns "*stand for* or *stands for* or *an acronym for* or *is the abbreviation for* or *the ((acronym) or (abbreviation))*"
  - Examples:
    - What does EKG stand for?
    - What is the abbreviation for limited partnership?
- *WHAT VERB* - Pattern is question ends in a verb
  - Examples:
    - What did Alfred Noble invent?
    - What do manatees eat?
    - What did Vasco da Gama discover?
- *WHAT CITY* - Focus Pattern "*city or town or capital or village*"
  - Examples:
    - What city has the oldest relationship as a sister-city with Los Angeles?
    - What city is Disneyland in?
- *WHAT COUNTRY* - Focus Pattern "*country*"
  - Examples:
    - What country is Aswan High Dam located in?
    - What country made the Statue of Liberty?
- *WHAT PROVINCE* - Focus Pattern "*province*"
  - Examples:
    - What province in Canada is Niagara Falls located in?
    - What province is Calgary located in?
- *WHAT FLOWER* - Focus Pattern "*flower*"
  - Examples:
    - What is Hawaii's state flower?
    - What is the Illinois state flower?
- *WHAT BIRD* - Focus Pattern "*bird*"
  - Examples:
    - What is the Ohio state bird?
    - What is Maryland's state bird?

- *WHAT TREE* - Focus Pattern "*tree*"
  - Examples:
    - What is California's state tree?
- *WHAT DATE* - Focus Pattern "*date or day*"
  - Examples:
    - What date did the United States civil war start?
    - What date did the Lusitania sink?
- *WHAT YEAR* - Focus Pattern "*year*"
  - Examples:
    - What year did Nintendo 64 come out?
    - What year was Ebbets Field, home of Brooklyn Dodgers, built?
- *WHAT NAME* - Focus Pattern "*name*"
  - Examples:
    - What was Dr. Seuss' real name?
    - What is Tina Turners real name?
- *WHAT CONTINENT* - Focus Pattern "*continent*"
  - Examples:
    - What continent is Egypt on?
    - What continent is India on?
- *WHAT POPULATION* - Focus Pattern "*population*"
  - Examples:
    - What is the population of Ohio?
    - What is the population of the United States?
- *WHAT COMPANY* - Focus Pattern "*company*"
  - Examples:
    - What company created the Internet browser Mosaic?
    - What company manufactures Sinemet?
- *WHAT INSTRUMENT* - Focus Pattern "*instrument*"
  - Examples:
    - What instrument does the concertmaster of an orchestra usually play?
    - What instrument measures radioactivity?

- *WHAT COLOR* - Focus Pattern "*color*"
  - Examples:
    - What color is the top stripe on the United States flag?
    - What color belt is first in karate?
- *WHAT NATIONALITY* - Focus Pattern "*nationality*"
  - Examples:
    - What nationality is Pope John Paul II?
    - What is Al Jolson's nationality?

Appendix-B

Sample System Generated Summaries

Example of System Generated Summaries for EM Learning

Following are the example summaries generated by our system based on EM learning for the document collection of DUC 2007.

(1) Summary for the topic description (document set D0703A) : **"Describe steps taken and worldwide reaction to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press."**

The Euro will be launched on January 1, 1999. Local press reports today said that the building societies have produced blueprints for euro savings accounts, passbooks and cash-dispensers despite the government's increasingly Eurosceptical tone. Despite skepticism about the actual realization of a single European currency as scheduled on January 1, 1999, preparations for the design of the Euro note have already begun. The European single currency euro will go ahead on schedule on January 1, 1999 with a broad membership, according to a survey of some prominent British economists. The report also said Duisenberg expects the future relationship between the dollar and the euro, which officially goes into effect on Jan. 12, to be stable. Stressing that the introduction of a single currency will be a great contribution to the unity of an expanded European Union (EU), Juppe reiterated France's commitment to the timetable and criteria of the single currency system set in the Maastricht treaty, under which the single European currency, recently named Euro, will be realized by January 1, 1999. The Frankfurt-based body said in its annual report released today that it has decided on two themes for the new currency: history of European civilization and abstract or concrete paintings. Europe's new currency, the euro, will rival the U.S. dollar as an international currency over the long term, Der Spiegel magazine reported Sunday. The European Union member states are required to completely replace their own national currencies with the Euro from January 1, 2002.

(2) Summary for the topic description (document set D0708B) : **"What countries are having chronic potable water shortages and why?"**

The move came against a backdrop of a severe water shortage in the country. China is one of the many countries in the world facing water shortage, a situation plaguing more than 300 of its 660-odd cities. Although Nepal is rich in water resources, water shortage is pervasive in the country because of its inability to tap the resources and the lack of well-managed supply system. A Lebanese official has warned that his country is suffering an annual water shortage of more than 1 billion cubic meters, the Daily Star

reported Friday. Due to the current drought in the Horn of Africa, water shortage has reigned throughout the east African country and more than 2.2 million Kenyans are threatened by starvation. Fernandes was speaking to the press on the water shortage problem in Luanda. The water shortage has caused some 6,000 people in the province to move, the report added. In northwestern China, which has half of the country's land, arable land has become increasingly desertified and sandstorms have become more frequent because of improper use of water resources. It is widely believed here that water shortage would be eased to a great extent once these plants become fully operational. It was reported that water shortage brought by the El Nino weather El Nino will be very serious in the Philippines. The Addis Ababa Regional Water and Sewerage Authority announced that the shortage of potable water in the capital city of Ethiopia will be solved in the last quarter of this year.

Example of System Generated Summaries for K-means Learning

(1) Summary for the topic description (document set D0703A) : **"Describe steps taken and worldwide reaction to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press."**

About the future European currency, 20 percent feel "well informed ", 79 percent feel to be not well informed. The RBI has set up a working group to study the implications of the Euro launch. The Euro will be launched on January 1, 1999. The Council of Economic Ministers of Thailand has ordered the Finance Ministry and the Bank of Thailand to study the possibility as the single European currency is set to go into circulation on January 1 next year. Local press reports today said that the building societies have produced blueprints for euro savings accounts, passbooks and cash-dispensers despite the government's increasingly Eurosceptical tone. The report also said Duisenberg expects the future relationship between the dollar and the euro, which officially goes into effect on Jan. 12, to be stable. He pointed out that such a political plan concerns the future of France and Germany, as well as the future role which the European countries will play in the world. It has n't been decided whether the Euro should include country emblems. Despite skepticism about the actual realization of a single European currency as scheduled on January 1, 1999, preparations for the design of the Euro note have already begun. The dollar has been weakening against European currencies in recent months. Stressing that the introduction of a single currency will be a great contribution to the unity of an expanded European Union (EU), Juppe reiterated France 's commitment to the timetable and criteria

of the single currency system set in the Maastricht treaty, under which the single European currency, recently named Euro, will be realized by January 1, 1999.

(2) Summary for the topic description (document set D0708B) : **"What countries are having chronic potable water shortages and why?"**

The move came against a backdrop of a severe water shortage in the country. In dry seasons the shortage of water is particularly acute. Ningyang County Paper Mill, a large plant in east China 's Shandong Province, will take the lead early this year in trying a water pollution treatment technology approved by the State Science and Technology Commission (SSTC). The government is already implementing several programs to overcome the water problem and would take about five months to complete, Mahathir said in Jitra in the northern state of Kedah after a political party function, according to the Malaysian national news agency Bernama. It is widely believed here that water shortage would be eased to a great extent once these plants become fully operational. Petrides called on the public to make "painful saving" of water and said his ministry will launch an intensive campaign to increase public awareness of the problem. Although Nepal is rich in water resources, water shortage is pervasive in the country because of its inability to tap the resources and the lack of well-managed supply system. Fernandes was speaking to the press on the water shortage problem in Luanda. According to Bruno Mwanafunzi, Director of Water Resources in the Ministry of Energy, Water and Natural Resources, the situation in Umutara was exacerbated by an unusual drought that hit the disaster area because of a prolonged dry season. A Lebanese official has warned that his country is suffering an annual water shortage of more than 1 billion cubic meters, the Daily Star reported Friday.

Example of System Generated Summaries for Local Search Technique

(1) Summary for the topic description (document set D0703A) : **"Describe steps taken and worldwide reaction to introduction of the Euro on January 1, 1999. Include predictions and expectations reported in the press."**

This new code has been issued to allow progress with the technical preparations for the European single currency, scheduled to be launched on January 1, 1999, said a press release issued here today.

The Euro will be launched on January 1, 1999. Local press reports today said that the building societies have produced blueprints for euro savings accounts, passbooks and cash-dispensers despite the government's increasingly Eurosceptical tone. Despite skepticism about the actual realization of a single European currency as scheduled on January 1, 1999, preparations for the design of the Euro note have already begun. France has already struck one billion coins of the European single currency euro, which will be launched by January 1999, reported Agence France-Presse (AFP) on Thursday. The European single currency euro will go ahead on schedule on January 1, 1999 with a broad membership, according to a survey of some prominent British economists. The report also said Duisenberg expects the future relationship between the dollar and the euro, which officially goes into effect on Jan. 12, to be stable. The single currency will be also used in the reports issued by the General Directorate of Monetary and Hard Currency Policy, Vasilescu added. It hasn't been decided whether the Euro should include country emblems. Stressing that the introduction of a single currency will be a great contribution to the unity of an expanded European Union (EU), Juppe reiterated France's commitment to the timetable and criteria of the single currency system set in the Maastricht treaty, under which the single European currency, recently named Euro, will be realized by January 1, 1999.

(2) Summary for the topic description (document set D0708B) : **"What countries are having chronic potable water shortages and why?"**

The move came against a backdrop of a severe water shortage in the country. China is one of the many countries in the world facing water shortage, a situation plaguing more than 300 of its 660-odd cities. Although Nepal is rich in water resources, water shortage is pervasive in the country because of its inability to tap the resources and the lack of well-managed supply system. A Lebanese official has warned that his country is suffering an annual water shortage of more than 1 billion cubic meters, the Daily Star reported Friday. Due to the current drought in the Horn of Africa, water shortage has reigned throughout the east African country and more than 2.2 million Kenyans are threatened by starvation. In dry seasons the shortage of water is particularly acute. Fernandes was speaking to the press on the water shortage problem in Luanda. The water shortage has caused some 6,000 people in the province to move, the report added. Even in Kathmandu, the capital of the country, regular water supply can not be guaranteed. In northwestern China, which has half of the country's land, arable land has become increasingly desertified and sandstorms have become more frequent because of improper use of water resources. It is widely believed here that

water shortage would be eased to a great extent once these plants become fully operational. It was reported that water shortage brought by the El Nino weather El Nino will be very serious in the Philippines.

Appendix-C

OAK System 150 Named Entities

150 NEs Tagged by OAK system

Tag	Example
NAME	
PERSON	Bill Clinton, Satoshi Sekine
LASTNAME	Clinton, Sekine
MALE_FIRSTNAME	Bill, George
FEMALE_FIRSTNAME	Mary, Catherine
ORGANIZATION	United Nations, NATO
COMPANY	IBM, Microsoft
COMPANY_GROUP	Star Alliance, Tokyo-Mitsubishi Group
MILITARY	The U.S. Navy
INSTITUTE	the National Football League, ACL
MARKET	New York Exchange, NASDAQ
POLITICAL_ORGANIZATION	
GOVERNMENT	Department of Education, Ministry of Finance
POLITICAL_PARTY	Republican Party, Democratic Party, GOP
GROUP	The Beatles, Boston Symphony Orchestra
SPORTS_TEAM	the Chicago Bulls, New York Mets
ETHNIC_GROUP	Han race, Hispanic
NATIONALITY	American, Japanese, Spanish
LOCATION	Times Square, Ground Zero
GPE	Asia, Middle East, Palestine
CITY	New York City, Los Angeles
COUNTY	Westchester
PROVINCE	State (US), Province (Canada), Prefecture (Japan)
COUNTRY	the United States of America, Japan, England

Tag	Example
REGION	Scandinavia, North America, Asia, East coast
GEOLOGICAL_REGION	Altamira
LANDFORM	Rocky Mountains
WATER_FORM	Hudson River, Fletcher Pond
SEA	Pacific Ocean, Gulf of Mexico
ASTRAL_BODY	Halley's comet, the Moon
STAR	Sirius, Sun, Cassiopeia
PLANET	the Earth, Mars, Venus
ADDRESS	
POSTAL_ADDRESS	715 Broadway, New York, NY 10003
PHONE_NUMBER	222-123-4567
EMAIL	sekine@cs.nyu.edu
URL	http://www.cs.nyu/cs/projects/proteus
FACILITY	Empire State Building, Hunter Mountain Ski Resort
GOE	Pentagon, White House, NYU Hospital
SCHOOL	New York University, Edgewood Elementary School
MUSEUM	MOMA, the Metropolitan Museum of Art
AMUSEMENT_PARK	Walt Disney World, Oakland Zoo
WORSHIP_PLACE	Canterbury Cathedral, Westminster Abbey
STATION_TOP	
AIRPORT	JFK Airport, Narita Airport, Changi Airport
STATION	Grand Central Station, London Victoria Station
PORT	Port of New York, Sydney Harbour
CAR_STOP	Port Authority Bus Terminal, Sydney Bus Depot
LINE	Westchester Bicycle Road
RAILROAD	Metro-North Harlem Line, New Jersey Transit
ROAD	Lexington Avenue, 42nd Street
WATERWAY	Suez Canal, Bering Strait
TUNNEL	Euro Tunnel
BRIDGE	Golden Gate Bridge, Manhattan Bridge
PARK	Central Park, Hyde Park
MONUMENT	Statue of Liberty, Brandenburg Gate
PRODUCT	Windows 2000, Rosetta Stone
VEHICLE	Vespa ET2, Honda Elite 50s
CAR	Ford Escort, Audi 90, Saab 900, Civic, BMW 318i

Tag	Example
TRAIN	Acela, TGV, Bullet Train
AIRCRAFT	F-14 Tomcat, DC-10, B-747
SPACESHIP	Sputnik, Apollo 11, Space Shuttle Challenger, Mir
SHIP	Titanic, Queen Elizabeth II, U.S.S. Enterprise
DRUG	Pedialyte, Tylenol, Bufferin
WEAPON	Patriot Missile, Pulser P-138
STOCK	NABISCO stock
CURRENCY	Euro, yen, dollar, peso
AWARD	Nobel Peace Prize, Pulitzer Prize
THEORY	Newtons law, GB theory, Blum's Theory
RULE	Kyoto Global Warming Pact, The U.S. Constitution
SERVICE	Pan Am Flight 103, Acela Express 2190
CHARACTER	Pikachu, Mickey Mouse, Snoopy
METHOD_SYSTEM	New Deal Program, Federal Tax
ACTION_MOVEMENT	The U.N. Peace-keeping Operation
PLAN	Manhattan Project, Star Wars Plan
ACADEMIC	Sociology, Physics, Philosophy
CATEGORY	Bantam Weight, 48kg class
SPORTS	Men's 100 meter, Giant Slalom, ski, tennis
OFFENCE	first-degree murder
ART	Venus of Melos
PICTURE	Night Watch, Monariza, Guernica
BROADCAST_PROGRAM	Larry King Live, The Simpsons, ER, Friends
MOVIE	E.T., Batman Forever, Jurassic Park, Star Wars
SHOW	Les Miserables, Madam Butterfly
MUSIC	The Star Spangled Banner, My Life, Your Song
PRINTING	2001 Consumer Survey
BOOK	Master of the Game, 1001 Ways to Reward Employees
NEWSPAPER	The New York Times, Wall Street Journal
MAGAZINE	Newsweek, Time, National Business Employment Weekly
DISEASE	AIDS, cancer, leukemia

Tag	Example
EVENT	Hanover Expo, Edinburgh Festival
GAMES	Olympic, World Cup, PGA Championships
CONFERENCE	APEC, Naples Summit
PHENOMENA	El Nino
WAR	World War II, Vietnam War, the Gulf War
NATURAL_DISASTER	Kobe Earthquake
CRIME	Murder of Black Dahlia, the Oklahoma City bombing
TITLE	Mr., Ms., Miss., Mrs,
POSITION_TITLE	President, CEO, King, Prince, Prof., Dr.
LANGUAGE	English, Spanish, Chinese, Greek
RELIGION	Christianity, Islam, Buddhism
NATURAL_OBJECT	mitochondria, shiitake mushroom
ANIMAL	elephant, whale, pig, horse
VEGETABLE	spinach, rice, daffodil
MINERAL	Hydrogen, carbon monoxide
COLOR	black, white, red, blue
TIME_TOP TIMEX	
TIME	10 p.m., afternoon
DATE	August 10, 2001, 10 Aug. 2001
ERA	Glacial period, Victorian age
PERIODX	2 semesters, summer vacation period
TIME_PERIOD	10 minutes, 15 hours, 50 hours
DATE_PERIOD	10 days, 50 days
WEEK_PERIOD	10 weeks, 50 weeks
MONTH_PERIOD	10 months, 50 months
YEAR_PERIOD	10 years, 50 years
NUMEX	100 pikel, 10 bits
MONEY	$10, 100 yen, 20 marks
STOCK_INDEX	26 5/8,
POINT	10 points
PERCENT	10%, 10.5 percent
MULTIPLICATION	10 times
FREQUENCY	10 times a day
RANK	1st prize, booby prize
AGE	36, 77 years old

Tag	Example
MEASUREMENT	10 bytes, 10 Pa, 10 millibar
PHYSICAL_EXTENT	10 meters, 10 inches, 10 yards, 10 miles
SPACE	10 acres, 10 square feet
VOLUME	10 cubic feet, 10 cubic yards
WEIGHT	10 milligrams, 10 ounces, 10 tons
SPEED	10 miles per hour, Mach 10
INTENSITY	10 lumina, 10 decibel
TEMPERATURE	60 degrees
CALORIE	10 calories
SEISMIC_INTENSITY	6.8 (on Richter scale)
COUNTX N_PERSON	10 biologists, 10 workers, 10 terrorists
N_ORGANIZATION	10 industry groups, 10 credit unions
N_LOCATION	10 cities, 10 areas, 10 regions, 10 states
N_COUNTRY	10 countries
N_FACILITY	10 buildings, 10 schools, 10 airports
N_PRODUCT	10 systems, 20 paintings, 10 supercomputers
N_EVENT	5 accidents, 5 interviews, 5 bankruptcies
N_ANIMAL	10 animals, 10 horses, 10 pigs
N_VEGETABLE	10 flowers, 10 daffodils
N_MINERAL	10 diamonds