Lecture Notes in Artificial Intelligence 1299

Subseries of Lecture Notes in Computer Science

Maria Teresa Pazienza (Ed.)

Information Extraction

A Multidisciplinary Approach to an Emerging Information Technology

Springer

Lecture Notes in Artificial Intelligence 1299

Subseries of Lecture Notes in Computer Science
Edited by J. G. Carbonell and J. Siekmann

Lecture Notes in Computer Science
Edited by G. Goos, J. Hartmanis and J. van Leeuwen

Maria Teresa Pazienza (Ed.)

Information Extraction

A Multidisciplinary Approach to an Emerging Information Technology

International Summer School, SCIE-97
Frascati, Italy, July 14-18, 1997

Springer

Series Editors
Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editor

Maria Teresa Pazienza
Università degli Studi di Roma,Tor Vergata
Dipartimento di Informatica Sistemi e Produzione
Via della Ricerca Scientifica, I-00133 Roma, Italy
E-mail: pazienza@info.utovrm.it

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Information extraction : a multidisciplinary approach to an emerging information technology ; international summer school / SCIE-97, Frascati, Italy, July 14 - 18, 1997. Maria Teresa Pazienza (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ; Singapore ; Tokyo : Springer, 1997
(Lecture notes in computer science ; Vol. 1299 : Lecture notes in artificial intelligence)

CR Subject Classification (1991): I.2, H.3

ISBN 978-3-540-63438-6 ISBN 978-3-540-69548-6 (eBook)
DOI 10.1007/978-3-540-69548-6

Typesetting: Camera ready by author
SPIN 10546325 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

This book contains all the presentations given at the International Summer School on Information Extraction, SCIE '97, Frascati (Rome), Italy, July 14–18, 1997. The topics covered span various scientific areas. In fact, although Information Extraction is mainly based on language processing ability, contributions derive from several disciplines. This motivated the wide range of topics discussed in this book.

As the ability to access different kinds of information via the Internet is increasingly involving end-users with different skills, the demand for suitable tools for Information Extraction(IE), organization, and integration is becoming more and more pressing to filter relevance and sort the large number of retrieved documents. IE is very often compared with the more mature methodology of Information Retrieval (IR), which is currently attempting to use natural language processing (NLP) based technologies to improve the recall and precision of retrieved documents in response to users' queries. But information extraction is not information retrieval: Information Retrieval aims to select relevant documents, Information Extraction aims to extract facts from the documents. This activity is not easy, as there are many ways to express the same fact. Typically information is spread across several sentences in natural language. The ability to process textual documents is essential to IE. Therefore it should be considered a core language technology.

In an NLP system the lexicon conveys linguistic information relative to words. The lexicon, then, may be considered a comprehensive knowledge repository where both representation and content support many deductive processes. In many applications lexical information relates surface forms (words or word sequences) more directly to pattern matching-based procedures having some linguistic flavors but dependent on the underlying tasks: text mark-up, entity detection and classification, or template filling. In such a framework the amount of linguistic information involved (proper noun rules or grammatical relations related to subcategorization information) is very high.

The dynamic nature of language, as well as its relationship with the underlying knowledge domain, creates difficult problems for NLP systems. In the IR context, terminology identification tends to be an important supporting technology. In the NLP context the problem of terminology identification is more general, as it is essential to find all terms which are characteristic of the document. Hence they are representative of its content and/or domain. The identification of terms is strongly tied to discourse processing, seeking to select those terms which are highly representative and capable of providing broad characterization of the document content.

From a "global information" perspective, most of the lexical knowledge employed in a given application is at first unusable for newer tasks in newer domains. The crucial characteristic of any information is *what it is about*, i.e., the entities it refers to. It is this referential meaning that needs to be made explicit and or-

ganized, in order to extract and reuse relevant information. It is easy to see how ontological aspects play a fundamental role here. As in IE, a template is to be 'filled' by information conveyed by a natural language statement. The problem is to make explicit the *intended models* of the world used to convey information. Thus, the ontological assumptions implicit in the terms adopted for concepts, relations, and attributes are to be made explicit. Moreover, if a system exists and performs an extraction task in a language, it is not obvious how to perform the same task on texts in different languages. It is important to separate the task-specific conceptual knowledge the system uses, which may be assumed to be language independent, from the language-dependent lexical knowledge the system requires, which unavoidably must be extended for each new language.

It is also desirable to determine how well an IE system is performing a given task through the definition of objective measures. It is common opinion that even systems with a modest performance (which miss more events and include some errors) may be of interest. They may be useful to extract information for future manual verification. Incomplete extracted information is in fact better than nothing!

The international summer school on information extraction SCIE '97 was organized to stress all these different aspects of information extraction.

SCIE '97 was sponsored by:

AI*IA Italian Association for Artificial Intelligence
CNR National Research Council
EC European Community
ENEA National Institution for Alternative Forms of Energy
ESA European Space Agency
UTV University of Rome, *Tor Vergata*
FUB Ugo Bordoni Research Foundation

Thanks to all the people and institutions who contributed to the organization of this summer school. Special thanks to the staff of the ESRIN (Frascati) site of the European Space Agency for their valuable support in hosting the school.

I would like to express my personal gratitude to all colleagues of the NLP Group of the University of Rome, Tor Vergata, whose extraordinary effort over these past months made SCIE '97 possible. Thanks a lot.

Rome, July 1997 Maria Teresa Pazienza

SCIE-97 Organizing Committees

General Chairperson:

Maria Teresa Pazienza, University of Roma Tor Vergata (Italy)

Program Committee:

Maristella Agosti, University of Padua (Italy)
Paolo Atzeni, University of Rome III (Italy)
Luigia Carlucci Aiello, University of Rome La Sapienza(Italy)
Floriana Esposito, University of Bari (Italy)

Organizing Committee (University of Rome Tor Vergata - Italy):

Roberto Basili,
Massimo Di Nanni,
Giovanni Pedani,
Michele Vindigni.

Table of Contents

Information Extraction as a Core Language Technology

Yorick Wilks

Computer Science Department
Speech and Language Technology
University of Sheffield, UK

What is IE?

Information Extraction (IE) technology is now coming on to the market and is of great significance to information end-user industries of all kinds, especially finance companies, banks, publishers and governments. For instance, finance companies want to know facts of the following sort and on a large scale: what company take- overs happened in a given time span; they want widely scattered text information reduced to a simple data base. Lloyds of London need to know of daily ship sinkings throughout the world and pay large numbers of people to locate them in newspapers in a wide range of languages. All these are prima facie cases for what we are calling IE, and to which this book is devoted.

Computational linguistic techniques and theories are playing a strong role in this emerging technology (IE), not to be confused with the more mature technology of Information Retrieval (IR), which selects a relevant subset of documents from a larger set. IE extracts information from the actual text of documents, by computer and at high speed, and normally from publicly available electronic sources such as news wires. Any application of this technology is usually preceded by an IR phase, which selects a set of documents relevant to some query- normally a string of features or terms that appear in the documents. So, IE is interested in the structure of the texts, whereas one could say that, from an IR point of view, texts are just bags of unordered words.

You can contrast these two ways of envisaging text information and its usefulness by thinking about finding, from the World Wide Web, what TV programs you might want to watch in the next week: there is already a web site in operation with text descriptions of the programs on 25 or more British TV channels, more text than most people can survey easily at a single session. On this web site you can input the channels or genre (e.g. musicals, news etc.) that interest you and the periods when you are free to watch. You can also specify up to twelve words that can help locate programs for you, e.g. stars' or film directors' names. The web site has a conventional IR engine behind it, a standard boolean function of the words and genre/channel names you use. The results are already useful -and currently free- and treat the program descriptions as no more than "bags of words".

Templates and their limitations

Now suppose you also wanted to know what programs your favourite TV critic liked: and suppose the web site also had access to the texts of recent newspapers. An IR system cannot answer that question because it requires searching review texts for films and seeing which one are described in favourable terms. Such a task would require IE and some notion of text structure. In fact, such a search for program evaluations is not a best case for IE, and I mention it only because it is an example of the kind of leisure and entertainment application that will be so important in future informatics developments. To see that one only has to think of the contrast between the designed uses and the actual uses of the French Minitel system–designed for phone number information but actually used largely as an adult dating service.

Some domains push out the limits of templatability, particularly any area with an evaluative component: e.g. one can search movie reviews for directors and actors–even for films where an individual appears in the non-standard role, such as Mel Gibson as a director, and that is a difficult task for an IR system—those are potentially matchable to templates but a much harder task is to decide if a movie review is positive or not. It is said that US Congressmen, who receive vast amounts of email that almost certainly cannot read, would welcome any IE system that could tell them simply, of each email message. The result of such a component could clearly be expressed as a template–what is unclear is how one could fill it in a reliable manner.

It is worth considering, in this context of vaguer forms of information, the extent to which a full text IR system can provide some apparently IE features. If one types to a well known and venerable IR system that has a windowing capacity "fat Scots psychologist", where the system has access to a year of indexed London Financial Times text and one can specify that say those words occur in that order within a window of 6 words—i.e. so very few other adjectives or nouns could intervene to break up that string. One will obtain citations of television program reviews including two of the prize -winning British TV series Cracker–which is almost certainly what anyone in the UK would have wanted who used the quoted phrase. We must suppose a viewer who had a clear goal in mind but no access to the title or the star's name– many users of such systems will be in that kind of position! This is still clearly an IR task but one whose boundary with IE is blurred because of the importance of syntagmic (NLP if you prefer) features imposed by the very length of the window chosen, as opposed to searching for ANY text with those three words.

A Brief History

Information extraction is a new technology not a new idea: as long ago as 1964 can be found papers with titles like "Text searching with templates" [12], but these were ideas not backed by any computational power capable of carrying them out; Other work that was IE before it knew it was Cowie's 1983 extraction

of canonical structures [5] from field-guide descriptions of plants and animals. The earliest effective IE work was undoubtedly that of Sager [11] within a medical domain, and constituting a long- running project combining surface syntax analysis and the use of templates. The work depended on handcrafted structures and a narrow range of techniques, but was highly effective. More immediately striking was de Jong's FRUMP [7] an adaptation of Schank's high-level script structures, or what we would now call scenarios (e.g. for a terrorist event), which attempted to fill the slots in such structures from the AP newswires. This work was prescient in many ways– including the choice of terrorism as a topic–but it had the toy quality of its time and was never evaluated in any quantitative way, e.g. against standard information retrieval systems ability to select terrorism stories from the AP wire, since it was being used for routing as well as extraction. FRUMP later became the basis for an early commercial system TRANS.

The first IE system to result from a complex commercially posed problem was Hayes et al.'s JASPER system at Carnegie Group [1]. Like the earlier systems it relied on high degree of handcrafting, like them too it had no access to major external linguistic resources: corpora, lexica or gazetteers, nor did it incorporate any learning algorithms. However it was benchmarked and evaluated in a serious manner, even if not within the US MUC and TIPSTER regimes.

The key roles of evaluation regimes

IE shares a key feature with an older, more mature, discipline, machine translation: both are associated with a strong evaluation methodology. This means that one can determine, by objective standards, how well a system is doing at a chosen task and whether it is getting better with time. It is not possible when writing of the brief history of IE as a technology to separate it from the ARPA-sponsored MUC competitions and the TIPSTER IE project, for it is this competitive and objective environment that have created they field by a kind of forcing..

IE as a subject and the current standards of evaluation and success were first surveyed in [6], and broadly one can say that the field grew very rapidly when ARPA, the US defence agency, funded competing research groups to pursue IE, and based initially round scenarios like terrorism events as reported in newspapers, a task where the funders wanted to replace the government analysts who read the newspapers for terrorist events and then filled templates: when, where, a terrorist event took place, how many casualties etc.. It was this activity that IE was designed to automate.

Empirical linguistics: corpora, modularity and architectures

However, the IE movement has also grown by exploiting, and joining, the recent trend towards a more empirical and text based computational linguistics, that is to say by putting less emphasis on linguistic theory and trying to derive

structures and various levels of linguistic generalisation from the large volumes of text data that machines can now manipulate. This movement has many strands: the use not only of machine readable dictionaries, taken from publishers tapes and from which semantic information has been extracted on a large scale [9] but also resources like WordNet, a semantic thesaurus in network form, created explicitly as a basis for NLP. The empirical movement has also joined forces with a revival of machine learning [4] so that large scale linguistic facts are not only derived from these sorts of resources but algorithms are used (both classic and novel) to derive structures at whatever level and then adapt them automatically to more data.

Two structural features have played roles in the astonishing growth of IE: first, that of modularity, the notion that computational linguistic tasks could be separated out and evaluated quite separately, and that the separation between the modules need not correspond only to classic divisions of linguistic levels, such as syntax and semantics.

Secondly, came the notion of an an architecture for NLP, and environment in which the modules just referred to could be combined in various ways to perform large scale tasks (e.g. IE but also MT or QA) and different modules for the same task could be systematically compared, or the same module could have its performance compared over different text types. The ARPA IE movement has made this a priority and our own groups contribution can be found at [8].

A conspicuous success has been a module usually called a part-of-speech tagger: a system that assigns one and only one part-of-speech symbol (like Proper noun, or Auxiliary verb) to a word in a running text and do so on the basis (usually) of statistical generalisations across very large bodies of text. Recent research has shown that a number of quite independent modules of analysis of this kind can be built up independently from data, usually very large electronic texts, rather than coming from either intuition or some dependence on other parts of a linguistic theory.

That these modules, can be independent–in their construction and optimisation– is something of a break with much traditional computational linguistics which tended to suggest that linguistic modules all depended on each other and could not be separated out in this way. The knowledge-based approach to NLP in artificial intelligence tended to reinforce this view: that even the simplest tasks, like tagging, could and did require the application of the highest level resources and inferences. That this is not in general true has been amazingly refreshing, and the relative independence of such modules at quite high levels of performance has made the construction of whole systems much simpler.

Such modules would usually be taken to include, as well as part of speech taggers: morphology analysis modules; modules to align texts sentence-by-sentence in different languages; syntax analysis modules, modules attaching word sense tags to words in texts to disambiguate them in context and so on. That these tasks can be done relatively independently is very surprising to those who believed them all contextually dependent sub-tasks within a larger theory. These modules have been combined in various ways to perform tasks like IE as well as

more traditional ones like machine translation (MT). The modules can each be evaluated separately –but they are not in the end real human tasks that people actually do, as MT and IE are.

One can call the former "intermediate" tasks and the latter real or final tasks -and it is really only the latter that can be firmly evaluated against human needs -by people who know what a translation, say, is and what it is for. The intermediate tasks are evaluated internally to improve performance but are only, in the end, stages on the way to some larger goal. Moreover, it is not possible to have quite the same level of confidence in them since what is, or is not, a correct syntactic structure for a sentence is clearly more dependent on one's commitments to a linguistic theory of some sort, and such matters are in constant dispute. What constitutes proper extraction of people's names from texts, or a translation of it, can be assessed by many people with no such subjective commitments.

The MRD stream in the empirical movement: lexicon acquisition

The empirical movement, basing, as it does, linguistic claims on text data, has another stream, noted earlier: the use in language processing of large language dictionaries (of single languages and bilingual forms) that became available about ten years ago in electronic forms from publishers' tapes. These are not textual data in quite the sense above, since they are large sets of intuitions about meaning set out by teams of lexicographers or dictionary makers. Sometimes they are actually wrong, but they have nevertheless proved a useful resource for language processing by computer, and lexicons derived from them have played a role in actual working MT and IE systems [14] and [3].

What such lexicons lack is a dynamic view of a language; they are inevitably fossilised intuitions. To use a well known example: dictionaries of English normally tell us that "television" is as a technology or a TV set, although it is mainly used now to mean the medium itself. Modern texts are thus out of step with dictionaries–even modern ones. It is this kind of evidence that shows that, for tasks like IE, lexicons must be adapted or "tuned" to the texts being analysed which has led to a new, more creative wave, in IE research: the need not just to use large textual and lexical resources, but to adapt them as automatically as possible, to enable them to adapt to new domains and corpora, which will mean dealing with obsolescence and with the specialised vocabulary of a domain not encountered before(see [13] and [2].

Modularity and the stereotypical IE system

Most of the modules listed above can be found, in some form, in virtually all the major IE systems currently competing in MUC. This is because, to a perhaps surprising degree, they share not only modules but assumptions about how the

IE task with templates is to be done. In Cowie and Lehnert [6] these asumptions were identified as:

- the power of shallow parsing (as against full syntax analysis)
- the power of shallow knowledge (simple, low level knowledge such as gazetteer lists and what can be derived from the template fillers themselves)
- mining the templates–effectively using filled templates to derive more shallow knowledge
- use of the test corpora themselves for data,, including lexica, preferably with learning algorithms to tune at least some of the modules.

These assumptions are partially independent of the selection of modules, as can be seen from the Sheffield VIE system (see below) which has several conventional modules but does not embody these assumptions at all. However, the general structural similarity of the major systems is such that Hobbs [10] was able to describe, without satire, a Generic IE System which closely matches most major players. This genericity is in part to be expected from an intensively competitive environment like MUC in which data and modules are also shared: the systems tend to all tune to each other and innovation is penalised. The benefits, as in horse breeding, say, are well known but there are penalties: the gene pool, as it were, becomes smaller and this may explain why outside entrants (from outside the trained group or gene pool: exogametes in fact) have done better than expected.

The European IE scene

As part of its large scale Language Engineering R & D program, the European Commission supports a considerable range of IE projects, built into either specific industrial-backed applications or as research projects: these include TREE (IE and MT for job applications across national boundaries), AVENTINUS (a classic IE application to police and terrorism issues for EU police forces); FACILE, ECRAN and SPARKLE (research projects combining IE and IR in different ways and with "tuning" or automatic adaptation aspects). ECRAN, for example, searches movie and financial databases and exploits the notion we mentioned of tuning a lexicon so as to have the right contents, senses and so on to deal with new domains and relations unseen before.

The EC funders have, on the whole, been less sympathetic to the benefits of MUC- style competition. They argue that it is wasteful of resources, though this ignores the fact that much of the MUC effort is "voluntary" in the sense of not directly funded by ARPA. In any case, the EU, like everyone else gets the benefit of published MUC-related research. But a more sophisticated defence could be given of the EC position, which is that ARPA's needs are basically defence driven and the ECs are commerce driven. Since all EC projects need industrial partnership and exploitation plans, then the market will ideally decide all issues, and there is no need of explicit competitions, nor could there be sufficient cooperation between the sponsoring companies to allow such diversion

of effort and indeed openness. The paradox behind that view, of course, is that it is the US companies, driven by ARPAs competitive regime. which have also put the first IE products on the market.

So the EC has considered general NLP and IE assessment procedures and decided against them, but the French government is said to be actively considering some such competition in France, perhaps to compensate for the fact that French as a language is highly unlikely to interest ARPA in the forseeable future. The British government funds some IE programs (including VIE and the GATE architecture, see below) but has no incentive to promote competition since British teams can and do enter MUC, at their own cost.

Possible limitations of IE systems

We have described IE here and as a new and vital technology, though in the following section we will also review ways in which it, like all language processing technologies, are being merged in original combinations to meet particular needs. But let us here, as an initial conclusion, review the possible limits to what we have described, quite apart from quantitative limits which may be expected to appear in MUC competitions as they have in IR and MT.

The main problem with IE is the degree to which knowledge IS template like, in the way history was once taught (but is no longer) as factual templates of kings, presidents, battles and dates. A major research issue is seeing how far the boundaries of templatability can be pushed out.

A closely related question is how far the construction and adaptation of templates to new domains (along with the adaptation of the associated lexical structures and lexicons) can be made practical and cost effective.

Both these questions can be brought under the heading of the adaptation of IE, to new domains, and to the needs of new users: research is going on not only on a'semi-automatic techniques for deriving templates from corpora –by seeing if a corpus contains significant patterns of a template type–but doing this in conjunction with user modelling techniques, representing user needs, and uses of things like natural dialogue (as has been done in IR) to allow a user to express needs–such as particular patterns sought and so on–in a natural language.

IE as part of a cluster of information technologies

An important insight, even after accepting our argument that IE is a new, emergent technology, is that what may seem to be wholly separate information technologies are really not so: MT and IE, for example, are just two ways of producing information to meet people's needs and can be combined in differing ways: for example, one could translate a document and then information extract from the result or vice-versa, which would mean just translating the contents of the resulting templates. Which of these one chose to do might depend on the relative strengths of the translation systems available: a simpler one might only be adequate to translate the contents of templates, and so on. This last observation

emphasises that the product of an IE system–the filled templates– can be seen either as a compressed, or summarised, text itself, or as a form of data base (with the fillers of the template slots corresponding to conventional database fields). One can then imagine new, learning, techniques like data mining being done as a subsequent stage on the results of IE itself.

If we think along these lines we see that the first distinction of this paper, between traditional IR and the newer IE , is not totally clear everywhere but can itself become a question of degree. Suppose parsing systems that produce syntactic and logical representations were so good, as some now believe, that they could process huge corpora in an acceptably short time. One can then think of the traditional task of computer question answering in two quite different ways. The old way was to translate a question into a formalised language like SQL and use it to retrieve information from a database- as in "Tell me all the IBM executives over 40 earning under $50K a year". But with a full parser of large corpora one could now imagine transforming ing the query to form an IE template and searching the WHOLE TEXT (not a data base) for all examples of such employees—both methods should produce exactly the same result starting from different information sources — a text versus a formalised database.

What we have called an IE template can now be seen as a kind of frozen query that one can reuse many times on a corpus and is therefore only important when one wants stereotypical, repetitive, information back rather than the answer to one-off questions.

"Tell me the height of Everest?", as a question addressed to a formalised text corpus is then neither a IR nor IE but a perfectly reasonable single request for an answer. "Tell me about fungi", addressed to a text corpus with an IR system, will produce a set of relevant documents but no particular answer. Tell me what films my favourite movie critics likes, addressed to the right text corpus , is undoubtedly IE as we saw, and will produce an answer also. The needs and the resources available determine the techniques that are relevant, and those in turn determine what it is to answer a question as opposed to providing information in a broader sense.

A final issue is the future relationship of IE to text summarization, a functionality for which there is a clearly established need. In some sense the data base provided by an IE system can be sen either as pure data, for subsequent mining, or as a base from which a summary could be generated in a natural language, which could be the original language on which the IE operated or another one. This is IE as the basis of a text summarization system; however, there are quite other techniques already near market, (a typical one would be British Telecom's), which often rely on summaries derived from individual sentences pieced together from the original text, deemed significant sentences on broadly statistical grounds. This technique has the advantage of not needing templates, but the disadvantage that such sentences may or may not form a coherent text.

Moral

One moral from all this, and which is important to keep in mind with the advent of speech research products and the multimedia associated with the Web, is that most of our cultural, political and business patrimony is still bound up with texts, from manuals for machines, to entertainment news to newspapers themselves. The text world is vast and growing exponentially: one should never be seduced by multi-media fun into thinking that text and how to deal with it, how to extract its content, is going to go away.

References

1. P.M. Andersen, P.J. Hayes, A.K. Huettner, I.B. Nirenburg, L.M. Schmandt, and S.P. Weinstein. Automatic extraction of facts from press releases to generate news stories. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 170–177. Association for Computational Linguistics, 1992.
2. R. Basili, M. Pazienza, and P. Velardi. Aquisition of selectional patterns from sublanguages. *Machine Translation*, 8, 1993.
3. B. Boguraev and T. Briscoe, editors. *Computational Lexicography for Natural Language Processing.* Longman, London, 1989.
4. E. Charniak. *Statistical Language Learning.* MIT Press, 1993.
5. J. Cowie. Automatic analysis of descriptive texts. In *Proceedings of the Conference on Applied natural language Processing*, 1983.
6. J. Cowie and W. Lehnert. Information extraction. *Special NLP Issue of the Communications of the ACM*, 1996.
7. G DeJong. An overview of the frump system. In W. Lehnert and M. Ringle, editors, *Strategies for Natural Language Processing.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1982.
8. R. Gaizauskas, T. Wakao, K. Humphreys, H. Cunningham, and Y. Wilks. University of sheffield: Description of the LaSIE System as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, Maryland, USA, November 1995.
9. Pustejovsky J. Wilks Y. & Slator B Guthrie, L. The role of lexicons in natural language processing. *Communications of the ACM*, 39(1), 1996.
10. J.R. Hobbs. The generic information extraction system. In *Proceedings of the Fifth Message Understanding Conference (MUC-5).* Morgan Kaufman, 1993.
11. N. Sager. *Natural Language Information Processing.* Addison-Wesley, Reading, Massachusetts, 1981.
12. Y. Wilks. Text searching with templates. Technical Report ML 162, Cambridge Language Research Unit, 1987.
13. Y. A. Wilks. Making prefernces more active. *Artificial Intelligence, 11*, 1978.
14. Y. A. Wilks, B. M. Slator, and L. M. Guthrie. *Electric Words: Dictionaries, Computers and Meanings.* MIT Press, 1996.

Information Extraction: Techniques and Challenges

Ralph Grishman

Computer Science Department
New York University
New York, NY 10003, U.S.A.

1 Introduction

This volume takes a broad view of information extraction as any method for filtering information from large volumes of text. This includes the retrieval of documents from collections and the tagging of particular terms in text. In this paper we shall use a narrower definition: the identification of instances of a particular class of events or relationships in a natural language text, and the extraction of the relevant arguments of the event or relationship. Information extraction therefore involves the creation of a structured representation (such as a data base) of selected information drawn from the text.

The idea of reducing the information in a document to a tabular structure is not new. Its feasibility for sublanguage texts was suggested by Zellig Harris in the 1950's, and an early implementation for medical texts was done at New York University by Naomi Sager[20]. However, the specific notion of information extraction described here has received wide currency over the last decade through the series of Message Understanding Conferences [1, 2, 3, 4, 14]. We shall discuss these Conferences in more detail a bit later, and shall use simplified versions of extraction tasks from these Conferences as examples throughout this paper.

Figure 1 shows a simplified example from one of the earlier MUC's, involving terrorist events (MUC-3) [1]. For each terrorist event, the system had to determine the type of attack (bombing, arson, etc.), the date, location, perpetrator (if stated), targets, and effects on targets. Other examples of extraction tasks are international joint ventures (where the arguments included the partners, the new venture, its product or service, etc.) and executive succession (indicating who was hired or fired by which company for which position).

Information extraction is a more limited task than "full text understanding". In full text understanding, we aspire to represent in a explicit fashion *all* the information in a text. In contrast, in information extraction we delimit in advance, as part of the specification of the task, the semantic range of the output: the relations we will represent, and the allowable fillers in each slot of a relation.

2 Why the Interest in Information Extraction?

There has been a growing interest in developing systems for information extraction, of which this volume is just one indication. This interest represents a

19 March — A bomb went off this morning near a power tower in San Salvador leaving a large part of the city without energy, but no casualties have been reported. According to unofficial sources, the bomb — allegedly detonated by urban guerrilla commandos — blew up a power tower in the northwestern part of San Salvador at 0650 (1250 GMT).

INCIDENT TYPE	bombing
DATE	March 19
LOCATION	El Salvador: San Salvador (city)
PERPETRATOR	urban guerrilla commandos
PHYSICAL TARGET	power tower
HUMAN TARGET	-
EFFECT ON PHYSICAL TARGET	destroyed
EFFECT ON HUMAN TARGET	no injury or death
INSTRUMENT	bomb

Fig. 1. A terrorist report and a template of extracted information.

confluence of need and ability — observing what is possible with current natural language processing technology, and how the possible may indeed be useful.

An enormous amount of information exists only in natural language form. If this information is to be automatically manipulated and analyzed, it must first be distilled into a more structured form in which the individual "facts" are accessible. Most of the world's news, for example, is reported in newspapers, radio, and TV broadcasts. The best that current commercial technology has to offer is to (try to) retrieve relevant passages from a text archive. If we want to know who has signed contracts over the past year to deliver airplanes or natural gas, or which jurisdictions have enacted new restrictions on smoking, we must pour over reams of retrieved documents by hand. Information extraction offers the potential of extracting such data with much greater precision, producing lists of companies or cities rather than lists of documents. Equal benefits would accrue in processing more technical texts: extracting information from scientific journals, from legal decisions, or from hospital reports. Hospitals and medical researchers, in particular, are faced with a need to perform a wide range of retrospective analyses on reports collected primarily in natural language form.

Research groups in the 1950's and 1960's already recognized the potential value of automatically structuring natural language data, and projects were created for tasks such as the transformation of entire encyclopedia entries to structured form. Such projects, however, faced a broad range of problems in natural language processing and knowledge representation, many of which still lack effective solutions. More recently, it has been recognized that by setting a goal of *selective* information structuring — i.e., information extraction — we can define a range of tasks that appears within reach of current technology.

A mature information extraction technology would allow us to rapidly create extraction systems for new tasks whose performance was on a par with human

performance. We are not there yet, for reasons to be discussed later in this paper. However, systems with more modest performance (which miss some events and include some errors) can be of value. They can be used to extract information for later manual verification.[1] They can also be useful in circumstances where there would not be time to review all the source documents, and incomplete extracted information is better than no information.

Recent research (stimulated, in part, by the MUC conferences) has shown that such modest extraction systems can – in some cases – be implemented using relatively simple natural language analysis methods. Current methods will be successful if the information to be extracted is expressed directly (so that no complex inference is required), is predominantly expressed in a relatively small number of forms, and is expressed relatively locally within the text.[2]

The following section describes the structure and components of an information extraction system. Then, after a brief discussion of system evaluation, we examine some of the current issues in trying to advance the state of the art of information extraction.

3 The Basic Techniques

3.1 The Overall Flow

The process of information extraction has two major parts. First, the system extracts individual "facts" from the text of a document through local text analysis. Second, it integrates these facts, producing larger facts or new facts (through inference). As a final step after the facts are integrated, the pertinent facts are translated into the required output format.

The individual facts are extracted by creating a set of patterns to match the possible linguistic realizations of the facts. Because of the complexity of natural language (except in the most restricted of sublanguages), it is not practical to describe these patterns directly as word sequences. So, as in most natural language processing systems, we begin by structuring the input, identifying various levels of constituents and relations, and then state our patterns in terms of these constituents and relations. This process typically begins with lexical analysis (assigning parts-of-speech and features to words and idiomatic phrases through morphological analysis and dictionary lookup) and name recognition (identifying names and other special lexical structures such as dates, currency expressions, etc.). This is followed by a full syntactic analysis (parse), or – in most current systems – by some form of partial parsing to identify noun groups, verb groups, and possibly head-complement structures. After all this is done, we use task-specific patterns to identify the facts of interest.

[1] A process which may offer economic benefits when compared to purely manual extraction, in the same way that machine translation followed by post-editing may be more efficient than manual translation.

[2] Bagga and Biermann [7] discuss the relation between the success rate of extraction systems and the locality of the information in the text, measured in terms of the number of syntactic relations involved.

The integration phase examines and combines facts from the entire document or discourse. It resolves relations of coreference, which can range from the use of pronouns to multiple descriptions of the same event. It may also need to draw inferences from the explicitly stated facts in the document. The overall flow is shown in Figure 2.

Following the terminology established by the Message Understanding Conferences, we shall call the specification of the particular events or relations to be extracted a *scenario*. Thus, we distinguish between a general *domain*, such as financial news, and a particular scenario, such as international joint ventures or aircraft sales. We shall refer to the final, tabular output format of information extraction as a *template*.

3.2 Pattern Matching and Structure Building

In the remainder of this section, we shall look at each of the stages of processing. As we go through the stages, we shall focus on a simplified version of the MUC-6 scenario, involving executive succession, and shall follow the progress of the following brief document

> Sam Schwartz retired as executive vice president of the famous hot dog manufacturer, Hupplewhite Inc. He will be succeeded by Harry Himmelfarb.

as it is gradually transformed into the two templates shown in Figure 3. The details we present will be those of our own (New York University Proteus Project) extraction system[3] [11], but we will note some of the places where they differ significantly from those of other systems.

In our system, and many other current extraction systems, most of the text analysis is performed by matching the text against a set of regular expressions. If the expression matches a segment of text, the text segment (constituent) is assigned a label, and possibly one or more associated features. The patterns are organized in sets, and constituent labels assigned in one pattern set may be referenced in patterns in subsequent sets. In effect, we perform a limited form of deterministic, bottom-up parsing.[4]

[3] Although with many simplifications from our actual implementation.

[4] The overall control of the matching process differs substantially among pattern-matching extraction systems. In the NYU system, each pattern has an associated set of actions; the main action generally is the tagging of a text segment with a new label, but other actions may be performed. The pattern sets are applied one at a time. All the patterns in a set are matched starting at the first word of the sentence. If more than one pattern matches, the one matching the longest segment is selected; if more than one pattern matches the longest segment, the first is taken. The actions associated with that pattern are executed. If no pattern matched, the patterns are reapplied starting at the next word of the sentence; if a pattern matched and an action labeled a text segment, the patterns are reapplied past the end of that segment. This process continues until the end of the sentence is reached.

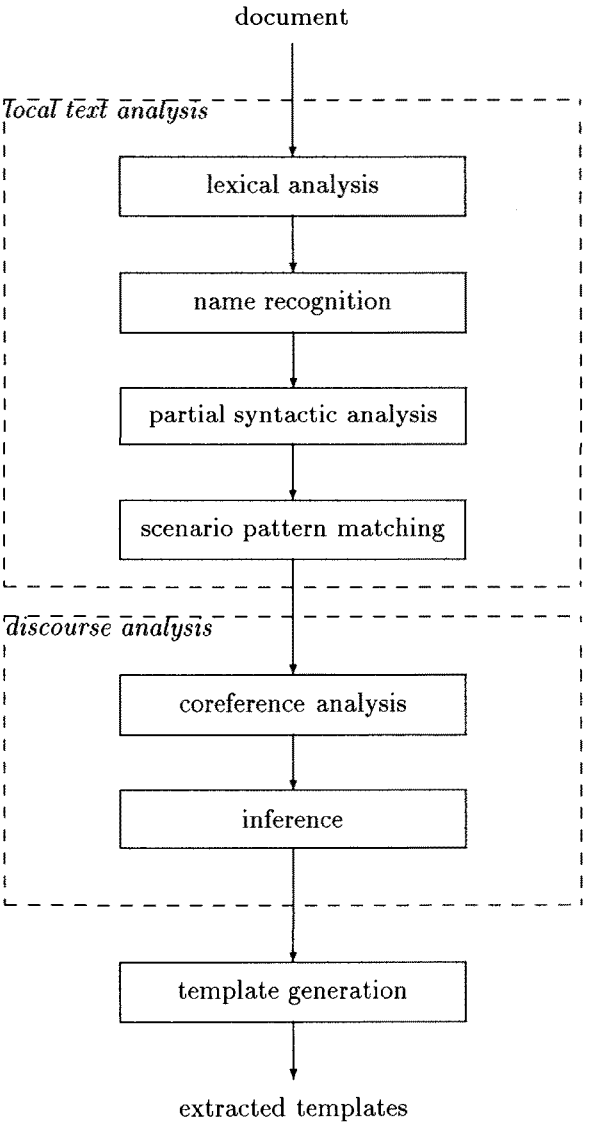


Fig. 2. Structure of an information extraction system.

Associated with some of the constituents in our system are semantic structures called *entities* and *events*. These structures will ultimately be used to construct the templates.

3.3 Lexical Analysis

The text is first divided into sentences and into tokens. Each token is looked up in the dictionary to determine its possible parts-of-speech and features. The Pro-

```
EVENT     leave job
PERSON    Sam Schwartz
POSITION  executive vice president
COMPANY   Hupplewhite Inc.
```

```
EVENT     start job
PERSON    Harry Himmelfarb
POSITION  executive vice president
COMPANY   Hupplewhite Inc.
```

Fig. 3. Events extracted from Hupplewhite text.

teus dictionary includes the Comlex Syntax dictionary (a large, general-purpose English dictionary [5]) [12] plus various special dictionaries, such as dictionaries of major place names, major companies, common (American) first names, and company suffixes (such as "Inc."). In our example "Sam" and "Harry" would be tagged as first names; Inc. will be tagged as a company suffix.[6]

3.4 Name Recognition

The next phase of processing identifies various types of proper names and other special forms, such as dates and currency amounts. Names appear frequently in many types of texts, and identifying and classifying them simplifies further processing; names, furthermore, are important as argument values for many extraction tasks.

Names are identified by a set of patterns (regular expressions) which are stated in terms of parts-of-speech, syntactic features, and orthographic features (e.g., capitalization). Personal names, for example, might be identified by a preceding title

Mr. Herrington Smith

by a common first name

Fred Smith

by a suffix

Snippety Smith Jr.

or by a middle initial

Humble T. Hopp

[5] Available through the Linguistic Data Consortium.

[6] In addition, our system uses a probabilistic part-of-speech tagger from BBN to exclude unlikely part-of-speech assignments; the system can operate without the tagger, but with slightly degraded performance.

Company names can usually be identified by their final token(s), such as

Hepplewhite Inc.
Hepplewhite Corporation
Hepplewhite Associates
First Hepplewhite Bank

However, some major company names (e.g., "General Motors") may be mentioned without any such overt clues, so it is important to also have a dictionary of major companies.

In our example passage, three names will be identified:

[*name type: person*Sam Schwartz] retired as executive vice president of the famous hot dog manufacturer, [*name type: company*Hupplewhite Inc.] He will be succeeded by [*name type: person*Harry Himmelfarb].

Name identification typically also includes the processing required to identify *aliases* of a name — in effect, name coreference. For example, a system would identify "Larry Liggett" with a subsequent mention of "Mr. Liggett", and "the Hewlett-Packard Corp." with "HP". Alias identification may also help name classification. Thus, if we read "Humble Hopp reported ..." we may not know if "Humble Hopp" is a person or company, but if there is a subsequent reference to "Mr. Hopp", the ambiguity is resolved.

Name identification has been worked on quite intensively for the past few years, and has been incorporated into several products which extract classified name lists from documents. The highest performing systems at present use large numbers of hand-coded patterns, but the performance of systems which learn rules from annotated corpora has been steadily improving, and is now only a few percent below that of the hand-coded systems [8].

3.5 Syntactic Structure

Identifying some aspects of syntactic structure simplifies the subsequent phase of fact extraction. After all, the arguments to be extracted often correspond to noun phrases in the text, and the relationships to be extracted often correspond to grammatical functional relations. On the other hand, the identification of the complete syntactic structure of a sentence is a difficult task. As a result, there is a great variation in the amount of syntactic structure which is explicitly identified. The trade-offs will be discussed further below (section 5.1).

Some systems don't have any separate phase of syntactic analysis. Others attempt to build a complete parse of a sentence. Most systems fall in between, and build a series of parse fragments. In general, they only build structures about which they can be quite certain, either from syntactic or semantic evidence. The current NYU Proteus system, following the lead of the SRI FASTUS system [6, 5], builds structures for noun groups (a noun plus its left modifiers) and for verb

groups (a verb with its auxilliaries); both of these can be built in most cases using just local syntactic information. In addition, it builds certain larger noun phrase structures (conjoined noun groups, noun groups with appositional modifiers) if it has semantic information to confirm the correctness of the structure. All of this is done using the same regular expression pattern matcher; unlike some other systems, no special procedures are used for parsing.

The first set of patterns labels all the basic noun groups as noun phrases (*np*); in our example, this includes the three names, the pronoun, and two larger noun groups. It is followed by a set to label the verb groups (*vg*). After these patterns have been applied, the text is labeled as follows:

> [*np entity: e1* Sam Schwartz] [*vg* retired] as [*np entity: e2* executive vice president] of [*np entity: e3* the famous hot dog manufacturer], [*np entity: e4* Hupplewhite Inc.] [*np entity: e5* He] [*vg* will be succeeded] by [*np entity: e6* Harry Himmelfarb].

Associated with each constituent are certain features which can be tested by patterns in subsequent stages. For vg's these include information on tense (past / present / future), voice (active / passive), and the root form of the verb; np's have information on the root form of the head (including whether or not it is a name) and syntactic number. In addition, for each np the system creates a "semantic" *entity*; for our example, following entities will be created:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president"
entity e3	type: manufacturer
entity e4	type: company name: "Hupplewhite Inc."
entity e5	type: person
entity e6	type: person name: "Harry Himmelfarb"

The sets of patterns which follow build up larger noun phrase structures by attaching right modifiers. Because of the syntactic ambiguity of right modifiers,[7] these patterns incorporate some semantic constraints and therefore, unlike the noun and verb group patterns, are domain specific. These patterns typically coalesce two noun phrases, and possible intervening words, into a larger noun phrase, and modify the entity associated with the head noun to incorporate information from the modifier.

For our example text, the two relevant patterns will recognize the appositive construction

> *company-description*, *company-name*,

and the prepositional phrase construction

> *position* of *company*

[7] For example, sequences which look like apposition might instead be part of conjoined structures; prepositional phrases to the right of a noun might attach to a preceding verb.

In the second pattern, *position* represents a pattern element which matches any np whose entity is of type "position" and *company* matches any np whose entity is of type "company". The system includes a small semantic type hierarchy (an *isa* hierarchy), and the pattern matching uses the *isa* relation, so any subtype of *company* (such as, in our example, *manufacturer*) will be matched. In the first pattern, *company-name* specifies an np of type *company* whose head is a name; *company-description* specifies an np of type *company* whose head is a common noun. These patterns produce the following labeling[8]

> [*np entity: e1* Sam Schwartz] [*vg* retired] as [*np entity: e2* executive vice president of the famous hot dog manufacturer, Hupplewhite Inc.] [*np entity: e5* He] [*vg* will be succeeded] by [*np entity: e6* Harry Himmelfarb].

and the entities are updated as follows:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e5	type: person
entity e6	type: person name: "Harry Himmelfarb"

3.6 Scenario Pattern Matching

All of the processing until now has been in a sense preparatory for the scenario pattern matching. The role of these patterns is to extract the events or relationships relevant to the scenario. For the example of executive succession we have been following, there will be two such patterns,

> *person* retires as *position*

and

> *person* is succeeded by *person*

where *person* and *position* are pattern elements which match np's with the associated type. "retires" and "is succeeded" are pattern elements which match active and passive verb groups, respectively. The result is a text labeled with two clauses, each pointing to an event structure; these event structures point in turn to the previously created entities:

> [*clause event: e7* Sam Schwartz retired as executive vice president of the famous hot dog manufacturer Hupplewhite Inc.] [*clause event: e8* He will be succeeded by Harry Himmelfarb].

and the entities / events are updated as follows:

[8] In our current system, if a new label subsumes earlier labels, the earlier labels are no longer visible for subsequent patterns, so we have removed these labels from the text, and removed entity e4.

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e5	type: person
entity e6	type: person name: "Harry Himmelfarb"
event e7	type: leave-job person: e1 position: e2
event e8	type: succeed person1: e6 person2: e5

3.7 Coreference Analysis

Coreference analysis has the task of resolving anaphoric references by pronouns and definite noun phrases. In our little text, the only example is the pronoun "he" (entity e5). Coreference analysis will look for the most recent previously mentioned entity of type *person*, and will find entity e1. It will then in effect change references to e5 to refer to e1 instead, leaving us with the following events and entities:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e6	type: person name: "Harry Himmelfarb"
event e7	type: leave-job person: e1 position: e2
event e8	type: succeed person1: e6 person2: e1

The coreference module also makes use of the *isa* hierarchy, so that if a reference to "the company" appeared in the text, it would be properly resolved to entity e3, the manufacturer.

3.8 Inferencing and Event Merging

In many situations, partial information about an event may be spread over several sentences; this information needs to be combined before a template can be generated. In other cases, some of the information is only implicit, and needs to be made explicit through an inference process.

In the executive succession domain, we need to determine what the "succeed" predicate implies, if we wish to ultimately produce templates specifying that particular individuals got or lost particular positions. For example, if we have

Sam was president. He was succeeded by Harry.

we infer that Harry will become president; conversely, if

Sam will be president; he succeeds Harry.

we infer that Harry was president. Such inferences can be implemented by production system rules,[9] such as

[9] In fact, these inferences were hard coded in our system for MUC-6, but production systems with similar rules were used in most of our prior MUC systems [13].

leave-job(X-person,Y-job) & succeed(Z-person,X-person)
⇒ start-job(Z-person,Y-job)

start-job(X-person,Y-job) & succeed(X-person,Z-person)
⇒ leave-job(Z-person,Y-job)

This will leave us with the following events:

entity e1	type: person name: "Sam Schwartz"
entity e2	type: position value: "executive vice president" company: e3
entity e3	type: manufacturer name: "Hupplewhite Inc."
entity e6	type: person name: "Harry Himmelfarb"
event e7	type: leave-job person: e1 position: e2
event e8	type: succeed person1: e6 person2: e1
event e9	type: start-job person: e6 position: e2

which can be translated into the templates shown earlier as Figure 3.

The simple scenario shown here did not require us to take account of the time of each event. For many scenarios, however, time is important: either explcit times must be reported, or the sequence of events is significant. In such cases, time information may be derived from many sources, including absolute dates and times ("on April 6, 1995"), relative dates and times ("last week"), verb tenses, and knowledge about the inherent sequence of events. Since time analysis may interact with other inferences about the discourse, it will normally be performed as part of this stage of processing.

4 Evaluation

As we noted at the beginning, one of the unusual aspects of information extraction is the degree to which its development over the past decade has been fostered and shaped by a series of evaluations, the Message Understanding Conferences (MUC). So, in order to understand the discussion which follows about the progress and problems of information extraction, we need to briefly describe the MUC evaluation process.

In a MUC evaluation, participants are initially given a detailed description of the scenario (the information to be extracted), along with a set of documents and the templates to be extracted from these documents (the "training corpus"). Systems developers then get some time (1 to 6 months) to adapt their system to the new scenario. After this time, each participant gets a new set of documents (the "test corpus"), uses their system to extract information from these documents, and returns the extracted templates to the conference organizer. Meanwhile, the organizer has manually filled a set of templates (the "answer key") from the test corpus.

Each system is assigned a variety of scores by comparing the system response to the answer key. The primary scores are *precision* and *recall*. Let N_{key} be the total number of filled slots in the answer key, $N_{response}$ be the total number of

filled slots in the system response, and $N_{correct}$ be the number of correctly filled slots in the system response (i.e., the number which match the answer key). Then

$$\text{precision} = \frac{N_{correct}}{N_{response}}$$

$$\text{recall} = \frac{N_{correct}}{N_{key}}$$

Sometimes an "F score" is also used as a combined recall-precision score; $F = (2 \times \text{precision} \times \text{recall})/(\text{precision} + \text{recall})$.

5 Design Issues

5.1 To Parse or not to Parse

As we noted above, one of the most evident differences among extraction systems involves the amount of syntactic analysis which is performed. The benefits of syntax analysis were noted above. In particular, the arguments we wish to collect during scenario pattern matching are usually connected by grammatical relations; for example, we want to extract the subject and object of verbs such as "hire", "fire", and "succeed". Thus, if syntactic relations were already correctly marked, we would expect the scenario patterns to be simpler and more accurate. These considerations motivated many early extraction systems to perform full syntactic analysis before looking for scenario patterns.

However, building a complete syntactic structure is not easy. Some decisions, such as conjunction scope and the attachment of modifiers, are particularly difficult. In principle, full sentence analyzers should be able to use global constraints to resolve local ambiguities. In fact, however, because of the inevitable gaps in grammatical and lexical coverage, full sentence parsers may end up making poor local decisions about structures in their quest to create a parse spanning the entire sentence. In effect, global constraints may make things worse. Furthermore, full sentence parsers are relatively expensive of computer time — not surprising, since they have a large space to search. Finally, we should note that for information extraction we are only interested in the grammatical relations relevant to the scenario; correctly determining the other relations may be a waste of time.

As a result, most current high performance extraction systems create only partial syntactic structures; syntactic structures which can be created with high confidence and using local information. Some systems, such as the BBN system [22], use separate parsing procedures. In our own work, the considerations just listed led us to shift from a full parsing approach (employed for earlier MUCs) to the partial parsing described here. Specifically, we followed the approach adopted by FASTUS [6, 5], which brackets noun and verb groups; both of these structures

are quite reliably identified by local information.[10] In addition, as illustrated in our earlier example, we identify some larger noun phrase constituents if there is semantic evidence to confirm the correctness of our attachment.

In general, we can say that most current extraction systems parse *conservatively* — only if there is strong evidence of the correctness of a reduction. This gains most (though not all) of the benefits of a full parse, while not introducing errors which would block applications of scenario patterns.

Traditional syntactic analysis has the role not only of *identifying* syntactic structure but also of *regularizing* syntactic structure. In particular, different clausal forms, such as active and passive forms, relative clauses, reduced relatives, etc. are mapped into essentially the same structure. This regularization simplifies the scenario pattern matching — it need deal with fewer forms of each scenario pattern.

If we employ a partial parsing approach which does not perform such regularization, we must have separate scenario patterns for each syntactic form, in effect multiplying the number of patterns to be written by a factor of 5 to 10. For example, we would need separate patterns for

IBM hired Harry
Harry was hired by IBM
IBM, which hired Harry, ...
Harry, who was hired by IBM, ...
Harry, hired by IBM, ...
etc.

To lessen this burden, the SRI FASTUS system [5] and the NYU Proteus system [11] have recently introduced *metarules* or *rule schemata*: methods for writing a single basic pattern and having it transformed into the patterns needed for the various syntactic forms of a clause. Thus we might write something like

subject=*company* verb=hired object=*person*

and the system would generate the patterns

company hired *person*
person was hired by *company*
company, which hired *person*
person, who was hired by *company*
person, hired by *company*
etc.

This approach bears some similarity to the metarules of GPSG [Generalized Phrase Structure Grammar], which expand a small set of productions into a larger set involving the different clause-level structures.

[10] The most common ambiguity regarding noun groups involves leading present participles, as in the classic example "They are flying planes." In such cases, we bracket the minimal noun group, excluding the present participle, and make allowances in subsequent patterns for the possibility that there may be an unattached present participle preceding the noun group.

Clausal patterns need also to account for modifiers which can intervene between the sentence elements of interest. For example, we may find between the subject and verb either sentence modifiers

IBM *yesterday* promoted Mr. Smith to executive vice president.

or noun modifiers which are not accounted for by the noun phrase patterns

GE, *which was founded in 1880*, promoted Mr. Smith to president.

With our partial parsing approach, we would need to include optional pattern elements separately in each clausal pattern to match such modifiers. With the metarule mechanism, we can automatically insert such pattern elements into the appropriate positions in each clausal pattern. In this way we can regain some of the benefits of syntactically-based clausal analysis.

The widespread use of partial parsing is a reflection of the current state of full-sentence parsing technology: experience shows that such parsers are likely to introduce more errors than they fix for information extraction tasks. However, this parsing technology is improving rapidly, thanks largely to corpus-based methods. A few years ago, good hand-coded grammars, operating on newspaper texts, got recall and precision scores in the range of 60 to 70%.[11] Recently, several parsers trained on large hand-bracketed corpora have gotten recall and precision scores above 80% [18, 9], and the performance is steadily improving. If the performance gets good enough, it may be worth revisiting the decision about using full-sentence parsing.

5.2 Portability

One of the barriers to making information extraction a practical technology is the cost of adapting an extraction system to a new scenario. In general, each application of extraction will involve a different scenario. If implementing this scenario requires a few months of effort and the skills of the extraction system designers, the market for extraction systems will remain limited indeed. We need to have tools which will allow potential users to adapt such a system, and create an initial system in days or weeks, not months.

The basic question in developing such a customization tool is the form and level of the information to be obtained from the user (assuming that the goal is to have the customization performed directly by the user rather than by a expert system developer). If we are using a "pattern matching" system, most work will probably be focused on the development of the set of patterns. However, changes will also be needed to the semantic hierarchy, to the set of inference rules, and to the rules for creating the output templates.

Unless users have considerable experience with writing patterns (regular expressions with associated actions) and some familiarity with formal syntactic structure, they may find it difficult and inconvenient to operate directly on the

[11] Using the Parseval metric, which compares the bracketing produced by a system to that of the University of Pennsylvania Tree Bank.

patterns. One possibility is to provide a graphical representation of the patterns (i.e., a finite state network), but this still exposes many of the details of the patterns. Instead, several groups are developing systems which obtain information primarily from *examples* of sentences of interest and the information to be extracted.

One of the first sites to experiment with this approach was the University of Massachusetts at Amherst. Their system, as developed for MUC-3, relied on a large number of small, lexically-triggered patterns which built individual concepts; these concepts were then consolidated to create the information needed for template filling [17]. For MUC-4, they developed a tool ("AutoSlog") to create such patterns semi-automatically from the development corpus with its templates [16, 19]. Given a template slot which is filled with words from the text, such as a name, their program would search for these words in the text and would hypothesize a pattern based on the immediate context of these words (for example, the governing verb of a noun phrase). These patterns would then be presented to a system developer, who could accept or reject the pattern. More recently, they have developed a system which uses machine learning techniques and does not rely on any human review [21, 10]. This system seeks to generalize and merge the patterns derived from individual examples, checking that the resulting patterns do not overgenerate (do not match corpus examples which are not marked as being relevant to the scenario).

Several of the earlier MUCs involved large training corpora, with over a thousand documents and their templates; such corpora encouraged such a corpus-based approach. However, the centralized preparation of large, consistent training corpora proved to be an expensive proposition; this suggested that such large corpora would not be available for most real tasks. Users may only be willing to prepare a few examples, or at best a few dozen examples, of filled templates. Experiments with smaller training collections, such as the 100 documents provided for MUC-6, suggest that fully automated learning techniques, when provided only with text examples and associated templates, and with minimal automatic syntactic generalization, may not be able to achieve sufficient coverage [10].

It is possible to compensate in part for the lack of training data by providing more information about *each* example. In HASTEN, the system developed by SRA for MUC-6 [15], the developer builds a structural description of each example, marking the type of constituent, the constraints on the constituent, and the semantic label of each constituent. This approach was able to achieve a good level of performance on the MUC-6 task, but requires some expertise on the part of the developer to mark up the examples.

At NYU, we are building an *interactive* tool for customizing an extraction system. We believe that this will provide the most efficient approach to acquisition in situations where the user does not have a large, pre-annotated corpus (and, in fact, may be refining the scenario in response to new examples), and does not have the expertise to create patterns unaided. The user begins by providing an example (normally drawn from the corpus) and the fact (template) to be extracted. The system responds by using the existing patterns to create

a structural description of the example. It then can interact with the user to extend and generalize the example, both syntactically and semantically. Syntactic generalizations can be produced through the metarule mechanism discussed above; semantic generalizations can be produced through the *isa* hierarchy. In this way, it is possible to quickly extend a single example into a pattern with broad coverage. In addition, by allowing the user to see other examples in the corpus which match the generated pattern, it is possible to insure that the pattern is not being overgeneralized.[12]

5.3 Improving Performance

The other major barrier to the widespread use of extraction systems is the limitation on performance. Any observer of the most recent MUC, MUC-6, would be struck by the relatively similar level of performance of the top-ranked systems. Five of the nine systems got F scores in the range of 51 to 56 (reflecting recall of 43 to 50% and precision of 59 to 70%) [4].

What can account for such a clustering of performance? Any response at present must be speculative. In part, it reflects a convergence of technologies; the top systems are fairly similar in their overall design. However, it probably also reflects characteristics of the task itself. It seems reasonable from what we know of other linguistic phenomena (e.g., the distribution of vocabulary or syntactic structures) that a large fraction of the relevant facts are encoded linguistically by a small number of forms (a small number of lexical items, a small number of syntactic structures, etc.). As a result, it is relatively easy (once a reasonable framework has been established) to reach some middling level of performance. Beyond that point, one is working on the "tail" of the distribution of linguistic phenomena, and further improvement becomes increasingly expensive.

How can we hope to move further along the tail — improve extraction performance — with a fixed investment of labor for each new scenario? Some of the shortcomings represent scenario-independent phenomena. These include more complex syntactic structures, such as different types of subordinate clauses, and more complex anaphoric phenomena, such as anaphors with split antecedents. Temporal information plays a significant role in many extraction tasks, and most temporal processing can be made independent of the specific scenario. Thus, as these aspects of the core extraction system are gradually enhanced, we can expect an improvement in performance across all scenarios.

Other shortcomings in performance reflect a lack of knowledge relative to a specific scenario, and will be more difficult to address. As tools for interacting with users to acquire and generalize patterns improve, we can expect to augment this knowledge more rapidly. For example, a tool which suggests related lexical items can broaden lexical coverage beyond what is seen in the training corpus. In addition, as more extraction scenarios are implemented, we can expect to see pattern sets which are applicable to families of related scenarios or to entire

[12] Such interactive analysis requires a fast extraction system; the current NYU system can process a typical newspaper article in about 2 seconds on a high-end PC.

domains. For example, patterns for basic actions such as the purchase and sale of goods may be applicable to many scenarios within the domain of business.

References

1. *Proceedings of the Third Message Understanding Conference (MUC-3).* Morgan Kaufmann, May 1991.
2. *Proceedings of the Fourth Message Understanding Conference (MUC-4).* Morgan Kaufmann, June 1992.
3. *Proceedings of the Fifth Message Understanding Conference (MUC-5)*, Baltimore, MD, August 1993. Morgan Kaufmann.
4. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
5. Douglas Appelt, Jerry Hobbs, John Bear, David Israel, Megumi Kameyama, Andy Kehler, David Martin, Karen Meyers, and Mabry Tyson. SRI International FASTUS system: MUC-6 test results and analysis. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
6. Douglas Appelt, Jerry Hobbs, John Bear, David Israel, and Mabry Tyson. FASTUS: A finite-state processor for information extraction from real-world text. In *Proc. 13th Int'l Joint Conf. Artificial Intelligence (IJCAI-93)*, pages 1172–1178, August 1993.
7. Amit Bagga and Alan Biermann. Analyzing the performance of message understanding systems. Technical Report CS-1997-01, Dept. of Computer Science, Duke University, 1997.
8. Daniel Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proc. Fifth Applied Natural Language Processing Conf.*, Washington, DC, April 1997. Assn. for Computational Linguistics.
9. Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proc. 34th Annual Meeting Assn. Computational Linguistics*, pages 184–191, Santa Cruz, CA, June 1996.
10. David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. Description of the UMass system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
11. Ralph Grishman. The NYU system for MUC-6 or where's the syntax? In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
12. Ralph Grishman, Catherine Macleod, and Adam Meyers. Comlex Syntax: Building a computational lexicon. In *Proc. 15th Int'l Conf. Computational Linguistics (COLING 94)*, pages 268–272, Kyoto, Japan, August 1994.
13. Ralph Grishman, Catherine Macleod, and John Sterling. New York University: Description of the Proteus System as used for MUC-4. In *Proc. Fourth Message Understanding Conf. (MUC-4)*, pages 233–241, McLean, VA, June 1992.
14. Ralph Grishman and Beth Sundheim. Message Understanding Conference - 6: A brief history. In *Proc. 16th Int'l Conf. on Computational Linguistics (COLING 96)*, Copenhagen, August 1996.

15. George Krupka. SRA: Description of the SRA system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.
16. W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. University of Massachusetts: MUC-4 test results and analysis. In *Proc. Fourth Message Understanding Conf.*, McLean, VA, June 1992. Morgan Kaufmann.
17. W. Lehnert, C. Cardie, D. Fisher, E. Riloff, and R. Williams. University of Massachusetts: Description of the CIRCUS system as used for MUC-3. In *Proc. Third Message Understanding Conf.*, San Diego, CA, May 1991. Morgan Kaufmann.
18. David Magerman. Statistical decision-tree models for parsing. In *Proc. 33rd Annual Meeting Assn. Computational Linguistics*, pages 276–283, Cambridge, MA, June 1995.
19. Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proc. 11th Annl. Conf. Artificial Intelligence*, pages 811–816, 1993.
20. Naomi Sager, Carol Friedman, and Margaret Lyman. *Medical Language Processing: Computer Management of Narrative Data.* Addison Wesley, 1987.
21. W. Soderland, D. Fisher, J. Aseltine, and W. Lenhert. CRYSTAL: Inducing a conceptual dictionary. In *Proc. Int'l Joint Conf. Artificial Intelligence (IJCAI-95)*, pages 1314–1319, Montreal, Canada, 1995.
22. Ralph Weischedel. BBN: Description of the PLUM system as used for MUC-6. In *Proc. Sixth Message Understanding Conf. (MUC-6)*, Columbia, MD, November 1995. Morgan Kaufmann.

Concepticons vs. Lexicons: An Architecture for Multilingual Information Extraction

Robert Gaizauskas, Kevin Humphreys,
Saliha Azzam, Yorick Wilks
{initial.surname}@dcs.shef.ac.uk

Department of Computer Science,
University of Sheffield

Abstract. Given an information extraction (IE) system that performs an extraction task against texts in one language, it is natural to consider how to modify the system to perform the same task against texts in a different language. More generally, there may be a requirement to do the extraction task against texts in an arbitrary number of different languages and to present results to a user who has no knowledge of the source language from which the information has been extracted. To minimise the language-specific alterations that need to be made in extending the system to a new language, it is important to separate the task-specific conceptual knowledge the system uses, which may be assumed to be language independent, from the language-dependent lexical knowledge the system requires, which unavoidably must be extended for each new language. In this paper we describe how the architecture of the LaSIE system, an IE system designed to do monolingual extraction from English texts, has been modified to support a clean separation between conceptual and lexical information. This separation allows hard-to-acquire, domain-specific conceptual knowledge to be represented only once, and hence to be reused in extracting information from texts in multiple languages, while standard lexical resources can be used to extend language coverage. Preliminary experiments with extending the system to French are described.

1 Introduction

Information extraction (IE) [4, 12] is the mapping of natural language texts (such as newswire reports, newspaper and journal articles, patents, electronic mail, World Wide Web pages, etc.) into predefined, structured representations, or *templates*, which, when filled, represent an extract of key information from the original text. The information pertains to entities of interest in the application domain (e.g. companies or persons), or to relations between such entities, usually in the form of events in which the entities take part (e.g. company takeovers, management successions). Once extracted, the information can then be stored in databases to be queried, data mined, summarised in natural language, *etc.*

To date most work on IE has been done in English, stimulated to a large degree by the DARPA Message Understanding Conferences [6, 13]. However,

in the last few years there has been a growing European presence in the field due to support from the Language Engineering programme of the European commission, and a number of IE-related projects have been or are being carried out which address extraction from texts in various European languages other than English, including French, German, Italian and Spanish [3, 8, 7, 19, 20]. Of course the DARPA MUC programme has not totally neglected multilingual issues either: MUC-5 contained a Japanese extraction task [1] and the MET evaluation exercise [17] focussed on named entity recognition and classification in Chinese, Japanese and Spanish.

Some of this work is simply the transfer of monolingual techniques and approaches from applications in one language to those in another. For example, the MUC-5 Japanese and English extraction exercises exactly parallel each other in specifying a task which requires the extraction of information about joint ventures from Japanese or English texts and the completion of identically structured templates in which the slot fills are strings from the respective source languages. Most of those who participated in both the English and Japanese tasks fielded parallel, distinct monolingual systems. But increasingly there is an interest in extracting information from texts in one language and presenting it to a user in another language, hence enhancing the information gathering power of a linguistically limited user. The impetus for this cross-language extraction is coming both from the need for multilingual co-operation in specific areas such as policing (addressed by the AVENTINUS project [5], which aims to construct a multilingual IE system for drug enforcement) and from the rapidly growing availability via the World Wide Web of electronic text in different languages.

Whether one is interested in building monolingual IE systems for new languages or in constructing cross-lingual IE systems, there are architectural issues to be addressed concerning how such systems may be constructed so as to maximise reuse of algorithmic and domain model components and minimise the requirement for adding language-specific mechanisms and data resources. We are aware of little work explicitly addressing these topics in the context of IE, with the exception of [16] which addresses these general issues and focusses on cross-linguistic named entity extraction. This paper investigates some of the issues involved in achieving multilingual IE, and proposes an approach and an architecture for an IE system which addresses the problems. A first prototype of the proposed system has been built for French and English, and Spanish, German are being added in the context of the AVENTINUS project mentioned above.

We discuss several alternative approaches to constructing a multilingual IE system, and then describe in detail the strategy we have chosen. Our choice is based on the assumption that it is possible, when dealing with a particular domain or application, to construct a language-independent representation of concepts relevant to the domain. We call this representation a domain model or *concepticon*. This unaesthetic neologism is advanced on rhetorical grounds: our proposal is that it is both possible and sensible to represent some (natural) language-independent conceptual content separately from language-specific information about particular lexical items. If the latter belongs in a lexicon (which

must, by any reasonable definition, be language-specific) then the former belongs in its own repository for which it is useful to employ an equally compelling term. Decoupling what many authors would refer to as "lexical semantics" from the lexicon, makes it possible to construct a system in which entries in multiple language-specific lexicons stand in a many-to-many relation with entries in a common concepticon. It is this architecture that makes multilingual IE possible. It also facilitates the extension of an IE system to perform monolingual extraction in a new language.

Representation of language-independent 'universal' concepts is a familiar, if uncomfortable, idea in theoretical NLP generally, and we wish to make it clear we are not advocating yet another *interlingua*. Nor do we wish to stir up the philosophical hornet's nest of issues in the Quinian tradition pertaining to the possibility of translation and very idea of language independent concepts. Rather we are advancing the modest, pragmatic claim that for the small well-defined domains typical of IE tasks it is possible to represent the key concepts of the domain in a language-independent fashion such that useful cross-linguistic extraction may take place. We believe the working prototype system which we describe below supports this claim. To what extent a more general-purpose concepticon can be created remains an open question: time and experimentation will tell.

2 IE and Multilinguality

Consider a simple multilingual IE task where we have texts in two languages, say French and English, and the user requires templates to be filled in English from texts in either language. To make the example more concrete, suppose the template definition specifies information concerning business meetings – organisations and individuals involved, location, date, etc. So both the sentences

Fr: *Gianluigi Ferrero a assisté à la réunion annuelle de Vercom Corp à Londres.*
En: *Gianluigi Ferrero attended the annual meeting of Vercom Corp in London.*

should give rise to the same template fill:

```
<MEETING-EVENT-01> :=
    ORGANISATION:   'Vercom Corp'
    LOCATION:       'London'
    TYPE:           'annual meeting'
    PRESENT:        <PERSON-01>

<PERSON-01> :=
    NAME:           'Gianluigi Ferrero'
    ORGANISATION:   UNCLEAR
```

We see three main ways of addressing the problem:

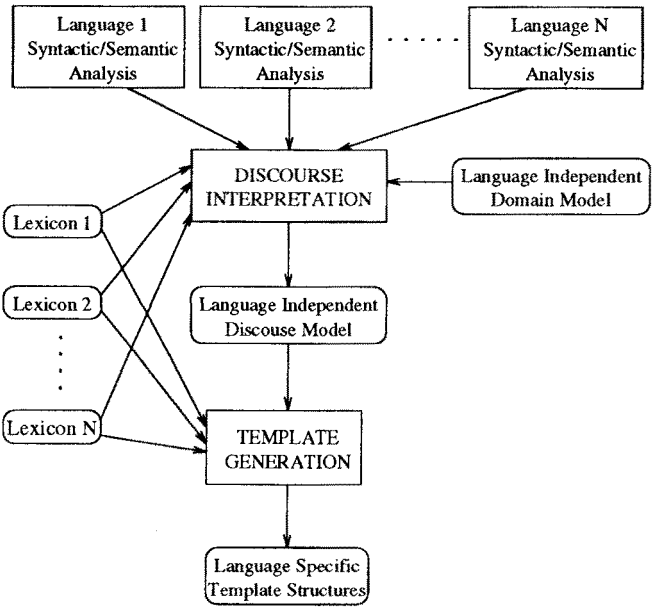


Fig. 1. An Archictecture for Multilingual IE

1. A full French-English machine translation (MT) system translates all the French texts to English. An English IE system then processes both the translated and the English texts to extract English template structures. This solution requires a separate full IE system for each language and a full MT system for each language pair.
2. Separate IE systems process the French and English texts, producing templates in the original source language. A 'mini' French-English MT system then translates the lexical items occurring in the French templates. This solution requires a separate full IE system for each language and a mini-MT system for each language pair.
3. A general IE system, with separate French and English front ends, uses a language-independent domain model in which 'concepts' are related via bi-directional mappings to lexical items in multiple language-specific lexicons. This domain model is used to produce a language-independent representation of the input text – a discourse model. The required information is then extracted from the discourse model and the mappings from concepts to the English lexicon are used to produce templates with English lexical items. This solution requires a separate syntactic/semantic analyser for each language, and the construction of mappings between the domain model and a lexicon for each language. (See figure 1)

Of the three alternative architectures we advocate the last, because, in principle, it allows new languages to be integrated independently, with no requirement to consider language pairs. The rest of the paper expands on the main issues involved in this approach.

3 The Multilingual Architecture

The prototype multilingual IE system M-LaSIE was derived from the English-only LaSIE system [11, 10]. LaSIE was designed as a general purpose IE research system, initially geared towards, but not solely restricted to, carrying out the English language tasks specified in MUC-6: named entity recognition, coreference resolution, template element filling, and scenario template filling (see [6] for further details of the tasks and for evaluations of participating systems). In addition, the system can generate a brief natural language summary of any scenario templates it has filled from the text.

The LaSIE system is a pipelined architecture which processes a text sentence by sentence. It consists of three principal processing stages: lexical preprocessing, parsing plus semantic interpretation, and discourse interpretation. The overall contributions of these stages may be briefly described as follows:

- Lexical preprocessing reads and tokenises the raw input text, tags the tokens with parts-of-speech, performs morphological analysis, and performs multi-word matching against lists of known proper names;
- Parsing does two pass chart parsing, pass one with a special named entity grammar, and pass two with a general phrasal grammar. A 'best parse' is then selected, which may be only a partial parse, and a predicate-argument representation or quasi-logical form (QLF) of each sentence is constructed compositionally.
- Discourse interpretation adds the QLF representation to a semantic net, which encodes the system's domain model as a hierarchy of concepts. Additional information presupposed by the input is also added to the model, then coreference resolution is performed between new and old instances. Information consequent upon the input is then added, resulting in an updated discourse model.

When an entire text has been processed, the result is a single, integrated discourse model. The template and summary are generated directly from this model.

The lexical preprocessing and parsing stages are necessarily language-specific, since separate languages are morphologically and syntactically distinct, though the same algorithms may be used for separate languages (e.g., trainable part-of-speech taggers). However, the QLF produced by each language-specific parser marks the point where a common representation begins to emerge. We discuss the QLF in the next section and then carry on to describe how the QLF from each language-specific front end is integrated into the language-independent domain model to form a language-independent discourse model for a specific text.

3.1 QLF

The QLF representation used in LaSIE is much cruder than that used in [2], but shares the characteristics of retaining various proximities to the surface form and of postponing some disambiguation, e.g. full analysis of quantifier scope and word sense disambiguation. Here is an example of the QLF as produced by the French and English parsers for the simple example from section 2, *i.e.*, *Gianluigi Ferrero a assisté à la réunion annuelle de Vercom Corp à Londres/ Gianluigi Ferrero attended the annual meeting of Vercom Corp in London*:

```
assister(e1),                    attend(e1),
tense(e1,past),                  tense(e1,past),
voice(e1,active),                voice(e1,active),
lsubj(e1,e2),                    lsubj(e1,e2),
lobj(e1,e3),                     lobj(e1,e3),
name(e2,'Gianluigi Ferrero'),    name(e2,'Gianluigi Ferrero')
reunion(e3),                     meeting(e3),
number(e3,sing),                 number(e3,sing),
name(e5,'Vercom Corp'),          name(e5,'Vercom Corp'),
organisation(e5),                organisation(e5),
compagnie(e5),                   company(e5),
name(e6,'Londres'),              name(e6,'Londres'),
lieu(e6),                        location(e6),
ville(e6)                        city(e6)
```

Syntactically, QLF expressions are simply conjunctions of first order logical terms. The predicates in the QLF representation are derived from the appropriate lexical morphological roots of head words (`assister`), from semantic classes assigned by the proper name taggers (`ville`), or from a fixed stock of relation predicates which express modification or semantic role relations (`lsubj`). Each noun and verb phrase recognised by the grammar leads to the introduction of a unique instance constant in the QLF representation (of the form `eN`) which serves as an identifier for the object or event referred to in the text. For example, the word *assisté* (*attended*) would be assigned the QLF representation `assister(e1), tense(e1,past)` (*assister* being the root form of *assisté*). Verb complements are represented via `lsubj` (logical subject) and `lobj` (logical object) relations between instance constants.

The stages required to produce the QLF representation for each language do not necessarily need to follow the current LaSIE architecture. In particular LaSIE's approach which replaces a conventional lexical lookup stage with part-of-speech tagging (to obtain word class information) and morphological analysis (to obtain the QLF predicate for head words) need not be adopted; a lexical lookup approach could be pursued instead. All that is required, by whatever technique or combination of techniques, is that the morphological and syntactic processing stages lead to a QLF representation in which the predicate terms are either from a fixed set which will be specially handled by the discourse interpreter

(`lsubj`, etc.) or are root forms that can be mapped via a lexeme-to-concept index onto language-independent concepts in the domain model.

3.2 Discourse Interpretation

Language-specific QLF expressions for individual sentences are passed on to the discourse interpreter. The principal task of the discourse processing module in LaSIE is to integrate the semantic representations of multiple sentences into a single model of the text from which the information required for filling a template may be derived.

The discourse interpretation stage of LaSIE relies on an underlying domain model, a declarative knowledge base that both contains general conceptual knowledge and serves as a frame upon which a discourse model for a multi-sentence text is built. This domain model is expressed in the XI knowledge representation language [9] which allows straightforward definition of cross-classification hierarchies, the association of arbitrary attributes with classes or individuals, and the inheritance of these attributes by individuals.

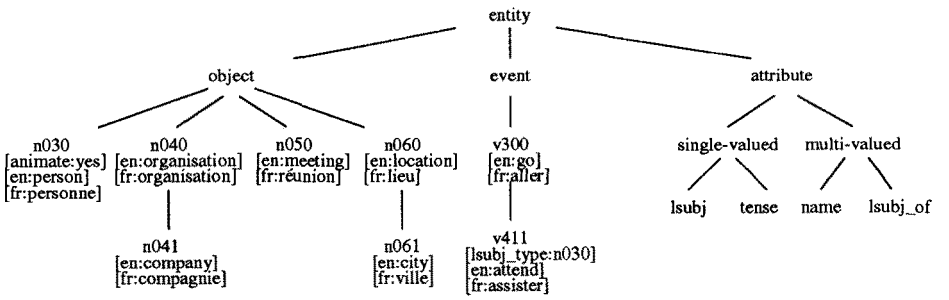


Fig. 2. Domain Model Fragment

The domain model consists of an *ontology* plus an associated *attribute knowledge base*. The ontology is a directed graph whose nodes represent language-independent classes or 'concepts' directly relevant to a specific template filling task. So, for example, for a business meeting scenario the ontology would be constructed to contain details about persons, places, and organisations, and also about events involving persons attending meetings.

Associated with each node in the ontology is an attribute-value structure. Attributes are simple `attribute:value` pairs where the value may either be fixed, as in the attribute `animate:yes` which is associated with the `person` node, or where the value may be dependent on various conditions, the evaluation of

which makes reference to other information in the model. The set of attribute-value structures associated with the whole ontology form the attribute knowledge base.

The higher levels of the ontology for the business meeting extraction task are illustrated in figure 2, along with some very simple attribute-value structures. Note how attributes can be used to store lexical realisations of the concept in various languages (which can be implemented as links into language-specific lexicons, rather than the actual strings).

The domain model described above can be regarded as an empty shell or frame to which the semantic representation of a particular text is added, populating it with the instances mentioned in the text. The model which results is then specialised for the world as described by the current text; we refer to this specialised model as the *discourse model.*

Figure 3 illustrates how instances are added to the world model, specialising it to convey the information supplied in a specific text. In the figure instances are indicated with the notation `e1`, `e2`, etc. and are shown connected by dashed lines to their classes.

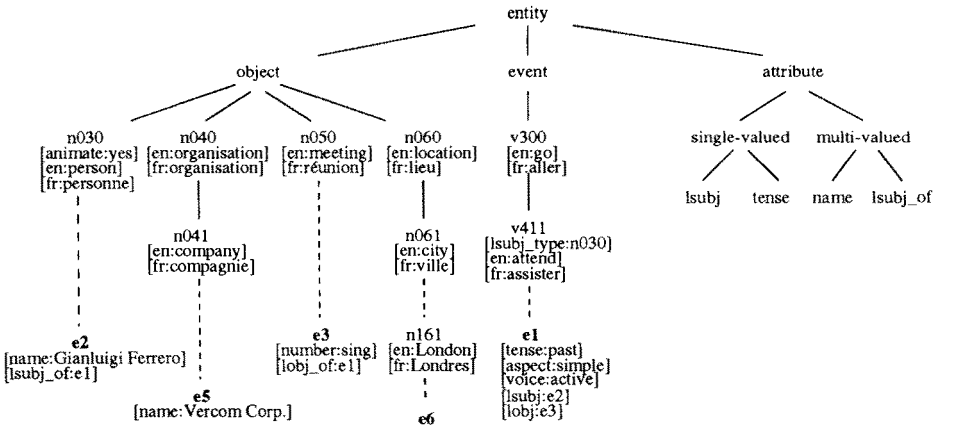


Fig. 3. Discourse Model fragment

Discourse processing proceeds in four substages for each new QLF representation passed on from the parser:

1. each instance in the QLF together with its attributes is added to the discourse model which has been constructed so far for the text;
2. presuppositions (stored as attributes of concepts in the domain model) are expanded, leading to further information being added to or removed from the model;

3. all instances introduced by the current sentence are compared with previously existing instances to determine whether any pair can be merged into a single instance, representing a coreference in the text;
4. consequences (also stored as attributes of concepts in the domain model) are expanded, leading to further information being added to or removed from the model, specifically information pertaining to template objects and slots.

The key stage for multilingual processing is stage 1. At this stage the language-specific predicate names used in the QLF are mapped to language-independent concept nodes in the domain model, via a language-specific lexeme-to-concept index (we are ignoring for the moment the problems of word sense ambiguity and synonymy which make the mapping between lexemes and concepts many-to-many, but shall come back to them in section 5). This index could simply be a pointer added to an existing lexicon to link it into the domain model, or, as in our current implementation, a standalone data structure.

Returning to our example, consider the French QLF term `compagnie(e5)`. Adding the instance `e5` below the appropriate class in the domain model requires searching the French lexeme-concept index to determine the corresponding concept node name in the domain model (`n041`).

Property names, such as `lobj` or `tense`, are expressed in English both in the QLF and in the domain model, so no mapping is required, but these labels are not significant and could be mapped to a language independent notation.

Some instances may be mentioned in the QLF representation without an explicit class, but a class may be inferrable from the instance's attributes. For instance, an attribute of the `title` attribute node in the domain model may specify that all instances with this attribute must be instances of the `person` class, so that, for example, `person(e2)` can be inferred from `title(e2, Mr.)`.

Further, in our example, `e2` is the `lsubj` of `e1`, and the `lsubj_type` attribute of `e1`'s class (`attend`) indicates that the subject must be of the class `n030` (`person`). This allows the placement of `e2` as an instance of `n030`, in the absence of any explicit class in the input.

The addition of `e6` is a special case. If the value of a `name` attribute, such as `London`, can be found as an entry in the lexicon, it will be possible to map the value, and therefore the instance, to a concept node in the domain model. The name attribute of `e6` then becomes redundant since a realisation can be inherited from the parent node for the required target language. Other named instances, such as `Gianluigi Ferrero`, will not be found in the source lexicon and will retain the name attribute for use as a realisation in all target languages. However, as the examples in [16] pertaining to Japanese and English name translation demonstrate, this simple solution will not always be acceptable.

Once the QLF representation has been added to the discourse model, the further stages of discourse processing take place within the same language-independent representation: presuppositions concerning role players in events are processed to allow missing verbal or prepositional complements to be identified; coreference is carried out to merge instances; inferences are performed to

attempt to derive template-specific information. Details of how these operations are carried out in LaSIE may be found in [10]. [1]

3.3 Template Generation and Summarisation

All results are produced by searching the final discourse model for predefined structures and formatting the resulting information as required. For template generation in a monolingual system this involves simply retrieving attribute values of certain classes of instances and writing them out directly. In the multilingual system, however, all output is via the lexicon of the target language, meaning that the template specification must be language independent, making use of concept nodes in the domain model. A multilingual template representation for the example used in Section 2 (repeated here) can be given as follows:

```
Heading: n050-e3                <MEETING-EVENT-01> :=
  Slot: n040, Value: e5           ORGANISATION:   'Vercom Corp'
  Slot: n060, Value: e6           LOCATION:       'London'
  Slot: a010, Value: e10          TYPE:           'annual meeting'
  Slot: a020, Value: n030-e2      PRESENT:        <PERSON-01>

Heading: n030-e2                <PERSON-01> :=
  Slot: a030, Value: e2           NAME:           'Gianluigi Ferrero'
  Slot: n040, Value:              ORGANISATION:   UNCLEAR
```

The template generation process converts this representation to the output form (on the right) by following pointers from the discourse model to a particular lexicon. For French output, for example, the pointers associated with the slot name `n060` and its value `e6` will produce: `LIEU: 'Londres'`.

The natural language summarisation operates in a similar method. Simple syntactic patterns, roughly corresponding to the template structures, are predefined for each output language. These are filled by instances from a text's final discourse model, and output involves the expansion of all complete patterns, retrieving lexical entries from the output lexicon, or using `name` attributes where no pointer is available. Some simple language-specific post-processing is then required, *e.g.* to produce tensed forms of verbs with appropriate particles, *etc.*

The summarisation is currently very simple but provides an effective multilingual output mechanism for the extracted information. Some examples are given in Section 4.

4 Real World Data

The current French/English system has been developed through experimentation with a small corpus (about 20 texts) of parallel French and English texts

[1] While discourse processing manipulates language-independent representations, some of the processing may need to be parametrised by language. For example, language-specific word order constraints may need to be taken into account in performing coreference.

from Canada NewsWire Ltd. Using the domain model developed for the MUC-6 management succession task, for the extraction of information regarding changes in company managerial positions, the prototype M-LaSIE system successfully processes these texts and carries out English-English, English-French, French-English, and French-French extraction (template filling and summarisation).

For example consider the pair of texts:

CHARLOTTE, N.C., Sept. 13 /CNW/ - United Dominion Industries Ltd. (NYSE, TSE: UDI), a manufacturer of diversified engineered products, today announced the appointment of John G. MacKay, 56, to the newly created position of executive vice president-Europe.
Mr. MacKay, who has been president of United Dominion's Industrial Products Segment, will be responsible for working with various operating units and the corporate staff to enlarge the company's presence in Europe through acquisitions, joint ventures and other strategic alliances. He will be based in Europe at a site to be determined.

CHARLOTTE, Caroline du Nord, 13 septembre /CNW/ - United Dominion Industries Ltd. (UDI aux bourses de Toronto et de New York), fabricant de produits usinés diversifiés, a annoncé aujourd'hui la nomination de John G. Mackay, qui est âgé de 56 ans, au poste nouvellement créé de vice-président directeur pour l'Europe.
M. MacKay, président de la section des produits industriels de United Dominion, travaillera avec diverses divisions d'exploitation et le personnel pour accroître la présence de la Société en Europe par des acquisitions, des coentreprises et d'autres alliances stratégiques. M. MacKay sera établi en Europe dans une ville qui n'a pas encore été choisie.

The system produces identical summaries from both these texts, in English:
United Dominion Industries Ltd. appoints John G. MacKay as executive vice president.
and in French:
United Dominion Industries Ltd. nomme John G. MacKay vice-président directeur.

MUC-style templates can also be produced in either language, from either input text:

```
<SUCCESSION_EVENT-c2097.french.txt-16> :=
        ORGANIZATION:       <ORGANIZATION-c2097.french.txt-43>
        POST:               "executive vice president"
        PERSON:             <PERSON-c2097.french.txt-48>
        VACANCY_REASON:     OTH_UNK
<ORGANIZATION-c2097.french.txt-43> :=
        NAME:               "United Dominion Industries Ltd."
        ALIAS:              "United Dominion"
                            "UDI"
        TYPE:               COMPANY
```

```
<PERSON-c2097.french.txt-48> :=
        NAME:                   "John G. MacKay"
        ALIAS:                  "MacKay"
        TITLE:                  "Mr."

<EVENEMENT_DE_SUCCESSION-c2097.english.txt-16> :=
        ORGANISATION:           <ORGANISATION-c2097.english.txt-43>
        POSTE:                  "vice-president directeur"
        PERSONNE:               <PERSONNE-c2097.english.txt-48>
        RAISON_DU_POSTE_VACANT: AUTRE_INCONNU
<ORGANISATION-c2097.english.txt-43> :=
        NOM:                    "United Dominion Industries Ltd."
        AUTRES_NOM:             "United Dominion"
                                "UDI"
        TYPE:                   COMPAGNIE
<PERSONNE-c2097.english.txt-48> :=
        NOM:                    "John G. MacKay"
        AUTRES_NOM:             "MacKay"
        TITRE:                  "M."
```

Separate syntactic/semantic analysis stages are used for French and English (though certain processing modules are shared) to produce QLF representations, and the discourse interpretation stage then requires the specification of an input and an output lexicon. Very few changes to the domain model developed for English texts were required to obtain reasonable performance on French texts. As soon as pointers between the domain model and a French lexicon were established, texts in the new language could be processed, demonstrating the low overhead involved in adding new languages to a system with the proposed architecture.

5 Discussion

The M-LaSIE system described above relies on a robust domain model that constitutes the central exchange through which all multilingual information circulates. The addition of a new language to the IE system consists mainly of mapping a new monolingual lexicon to the domain model and adding a new syntactic/semantic analysis front-end, with no interaction at all with other languages in the system.

The language independent domain model can be compared to the use of an *interlingua* representation in MT (see, *e.g.*, [15]). An IE system, however, does not require full generation capabilities from the intermediate representation, and the task will be well-specified by a limited 'domain model' rather than a full unrestricted 'world model'. This makes an *interlingua* representation feasible for IE, because it will not involve finding solutions to all the problems of such a representation, only those issues directly relevant to the current IE task. Despite this simplification, however, there are still a number of problematic issues related

to the mapping between the domain model and monolingual lexicons which need to be discussed.

5.1 Lexical Gaps

Lexical gaps occur between languages when a concept expressed by a single word in one language has no direct match in a single word in another, but rather requires several words or a phrase to express it. For example, Sicilians have no word to translate the English word *lawn*[2]. However, if, for a particular multilingual IE task, such a concept is included in a domain model, language-specific rules can be used to map phrases to single concepts where necessary.

It is tempting to presume that lexical gaps will be rare, since the domain model will generally define an area of interest which has much in common across all the languages involved, but they will occur. In the AVENTINUS drug domain, for example, the domain model may include a drug concept node for the English word *crack*, but pointers to the appropriate entries in the lexicons of non-English languages may not exist. To express the concept in a target language without a pointer will then require the selection of either the text's original language or some fixed default. (To attempt to generate a definition of the concept in the target language is beyond the scope of the current system, though relating it to its genus – *a type of cocaine* – would be possible, assuming a word for the genus exists in the target language).

5.2 Word Sense Disambiguation

The approach we have described and implemented ignores the problem of word sense ambiguity: one lexeme may map onto more than one concept (for example, *company* may be a business organisation or a unit of an army). The word sense problem becomes serious if the domain model is created automatically from a large-scale pre-existing resource, such as a machine-readable dictionary or thesaurus. If used without restriction such resources will provide us with a large number of concept nodes in the hierarchy for each word (e.g. *bank* has 9 nominal senses in WordNet [18]). This problem is minimised in the context of domain-specific IE applications because words will often have only a single sense relevant to a particular domain. However we must still consider potential solutions within the proposed architecture.

One solution is to utilise a word sense tagger, the results of which could be used to determine which concept node in the domain model each referring term should map to. While word sense tagging is still an open area for research, current techniques hold promise, *e.g.* [22, 21]. Another approach is to appeal to knowledge held in the domain model hierarchy, using, *e.g.*, selectional restrictions as a set of constraints which, when satisfied, will hopefully eliminate all but one sense for each semantically ambiguous word in the input. Other information, such as the frequency of sense occurrences, may also be brought to bear in the

[2] We are indebted to Roberto Garigliano for this example.

form of constraints. Further, the discourse model built so far can also be used to define a context in which certain senses are much more likely to occur.

5.3 Lexical Choice for Generation

The reverse problem to word sense disambiguation – lexical choice – is relevant particularly in the generation of natural language summaries. Several lexical entries may have pointers to the same node in a domain model, thus requiring a selection be made when producing a realisation for the node. Of course this is a significant problem for natural language generation systems in general ([14]), usually requiring reference to features of the discourse context and user model to resolve.

For example, in the domain model for the management succession task, the English lexical entries for 'chief executive officer' and 'CEO' both have pointers to the same node. The current system just selects the first pointer found, and for the simple summaries produced at present this is usually adequate. The problem will become more apparent with larger domain models but, as mentioned before, large domain models are not required for typical IE tasks.

5.4 Unknown Words

In general, if the root form of a word cannot be mapped to a concept node in the domain model, the word will simply be interpreted as irrelevant to the current domain. Of course this assumes that the domain model is complete and correct, which may not always be the case. To allow for this problem requires mechanisms for extending the domain model as new words are encountered. This is effectively a form of training to construct new domain models from a representative text corpus, and as such is beyond the current system. Suffice to say that the issue of creating multilingual mappings between lexicons and new concept nodes, derived either manually or automatically through training, is likely to require the use of existing bilingual dictionaries to find, say, the French lexical entry corresponding to a node created from the English word *crack*, as the name of a drug.

6 Conclusion

Multilingual information extraction is the extraction of templates in a target language from texts in one or more different source languages. As pointed out by Kameyama in [16], a key question in multilingual IE is how much monolingual IE infrastructure (*e.g.*, rules and data structures) can be either reused or shared when more languages are added.

In this paper we have proposed an approach to multilingual IE which we believe minimises the amount of work required when adding new languages. The system architecture we have presented is based on a clean separation of conceptual and lexical information. The multilingual capabilities are provided via bi-directional mappings between the concept nodes in a language-independent

domain model relevant to a particular IE task, and entries in a number of monolingual lexicons. These mappings allow a language-independent discourse model for a text to be constructed, based on the initial domain model. From this discourse model a language-independent representation of a template can then be constructed and, again via the concept/lexicon mappings, the template representation can be expressed in a particular target language.

A French-English prototype of this architecture has been implmented and successfully tested on a limited amount of data. The architecture is being further developed in the AVENTINUS project [5] which will extend the system to include Spanish and German.

References

1. Advanced Research Projects Agency. *Proceedings of the Fifth Message Understanding Conference (MUC-5).* Morgan Kaufmann, 1993.
2. H. Alshawi, editor. *The Core Language Engine.* MIT Press, Cambridge MA, 1992.
3. AVENTINUS: Advanced information system for multinational drug enforcement. http://www2.echo.lu/langeng/en/lel/aventinus/aventinus.html. Site visited 29/05/97.
4. J. Cowie and W. Lehnert. Information extraction. *Communications of the ACM,* 39(1):80–91, 1996.
5. H. Cunningham, S. Azzam, and Y. Wilks. Domain Modelling for AVENTINUS (WP 4.2). LE project LE1-2238 AVENTINUS internal technical report, University of Sheffield, UK, 1996.
6. Defense Advanced Research Projects Agency. *Proceedings of the Sixth Message Understanding Conference (MUC-6).* Morgan Kaufmann, 1995.
7. ECRAN: Extraction of Content: Research at Near-Market. http://www2.echo.lu/langeng/en/lel/ecran/ecran.html. Site visited 29/05/97.
8. FACILE: Fast and Accurate Categorisation of Information by Language Engineering. http://www2.echo.lu/langeng/en/lel/facile/facile.html. Site visited 29/05/97.
9. R. Gaizauskas. Xl: A Knowledge Representation Language Based on Cross-Classification and Inheritance. Technical Report CS-95-24, Department of Computer Science, University of Sheffield, 1995.
10. R. Gaizauskas and K. Humphreys. Using a semantic network for information extraction. *Journal of Natural Language Engineering,* 1997. In press.
11. R. Gaizauskas, T. Wakao, K Humphreys, H. Cunningham, and Y. Wilks. Description of the LaSIE system as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6).* Morgan Kaufmann, 1995.
12. R. Gaizauskas and Y. Wilks. Information Extraction: Beyond Document Retrieval. Submitted to *Journal of Documentation,* 1997.
13. R. Grishman and B. Sundheim. Message understanding conference - 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics,* Copenhagen, June 1996.
14. H. Horacek and M. Zock, editors. *New Concepts in Natural Language Generation: Planning, Realization and Systems.* Pinter Publishers, London, 1993.
15. W.J. Hutchins. *Machine Translation: past, present, future.* Chichester : Ellis Horwood, 1986.

16. M. Kameyama. Information Extraction across Linguistic Boundaries. In *AAAI Spring Symposium on Cross-Language Text and Speech Processing*, 1997.
17. R. Merchant, M.E. Okurowski, and N. Chinchor. The Multi-Lingual Entity Tast (MET) Overview. In *Advances in Text Processing – TIPSTER Programme Phase II*, pages 445–447. DARPA, Morgan Kaufman, 1996.
18. G. A. Miller (Ed.). WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.
19. SPARKLE: Shallow parsing and knowledge extraction for language engineering. http://www2.echo.lu/langeng/en/le1/sparkle/sparkle.html. Site visited 10/06/97.
20. TREE: Trans European Employment. http://www2.echo.lu/langeng/en/le1/tree/tree.html. Site visited 29/05/97.
21. Y. Wilks and M. Stevenson. Sense tagging: Semantic tagging with a lexicon. In *Proceedings of the ANLP97 Workshop on Tagging Text with Lexical Semantics*, 1997.
22. D. Yarowsky. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *COLING-92*, 1992.

Lexical Acquisition and Information Extraction

Roberto Basili, Maria Teresa Pazienza

Department of Computer Science, System and Production
University of Roma, *Tor Vergata*
Via Della Ricerca Scientifica s.n.c., 00133, Roma, ITALY
e-mail: {basili,pazienza}@info.utovrm.it

1 Introduction

There is a growing consesus in viewing the lexicon as the crucial component of precise and robust Natural Language Processing (NLP) systems for a variety of application tasks: Information Extraction (IE), Information Retrieval (IR), Machine Translation (MT), ... What is still controversial in the engineering of NL systems is the nature and the format of the information which we refer to as lexical.

Using a simple definition the lexicon is the component of a NLP system that expresses linguistic information about individual words (or word strings). Another valid and slightly more comprehensive definition describes the lexicon *as an association between surface forms and linguistic information* ([72]). This characterization is wide enough to capture linguistic principles and application oriented purposes, as opposite limits in a wide range of formalisms and lexical assumptions.

Linguistic theories (as HPSG, [65]), on one side, postulate lexical descriptions where surface forms are related to a complex structure of linguistic principles (from morphsyntactic features to semantic constraints), aimed to support a variety of still very complex inferences. Lexicon is here a comprehensive knowledge repository where representation and content support many deductive processes: inheritance, forward/backward reasoning, constraint propagation.

In many applications, lexical information links surface forms (words or simple words sequences) directly to pattern-matching procedures having some linguistic flavor but more dependent onto the underlying task: text mark-up, entity detection/classification or template filling. In these latter cases the variety of inferences triggered by the lexical knowledge is smaller but the amount of the involved linguistic information (e.g. proper nouns defining rules or grammatical relations related to subcategorization information) may still be very high.

The degree of lexicalization of application systems is thus typically very high either for theoretical aspects (types and expressiveness of supported linguistic inferences) or for functional requirements (types and variety of supported lexically-driven actions). On one side linguistic theories provide growing evidences of lexical dependencies throughout the different language recognition/generation

processes ([65]), on the other side efficiency, coverage and robustness push for more and more lexicalized approaches in application systems. It is not surprising that several successufull IE systems in the last MUC-6 Conference use very specific lexical rules for several tasks like Template Creation or Scenario Recognition ([7, 40]).

The dynamic nature of language as well as its critical dependence on the underlying knowledge domain create strong problems to portability and scalability of lexicons. Most of the lexical knowledge employed in a given application is at first unsuitable for newer tasks, in newer domains. As domain expectations are crucial for IE tasks, the portability of the lexicon is a relevant issue in the design of flexible and large scale IE systems. In the worst cases, lack of domain-specific information results in redundancy, overambiguity, inconsistency (i.e. incorrect lexical inferences), thus worsening the linguistic analysis, asking for some extra-linguistic control heuristics.

Several studies in the area of Lexical Acquisition (LA) have been proposed to provide NL systems with adaptive capabilities for:

- automatizing the construction of domain-specific lexicons from Machine Readable Dictionaries (MRDs), Lexical Knowledge Bases (LKBs) and application corpora;
- improving linguistic inferences according to domain-oriented knowledge, typically extracted from large samples of domain-specific texts (e.g. grammar or syntactic preference rules induction);
- augmenting NL system with semantic disambiguation capabilities aiming to support more precise recognition/look-up rules and/or linguistic inference (e.g. semantic interpretation) on a larger scale.

2 The Lexicon and IE

Although a reasonable agreement on a general linguistic architecture for IE tasks has been reached ([53]), the type and nature of the lexicalization of those tasks is still variegate and underspecified. Lexically driven modules are employed in different text analysis stages and the underlying models may be very different. The difference lies not only on the "parsing or not parsing" issue [48]. More or less (explicitly) lexicalized actions are also differently undertaken in other "classical" IE tasks. The PROTEUS system, for example, relies on explicit lexical predictions for early stages of the scenario recognition (i.e. the domain pattern recognizer module, ([48]). Other systems (i.e. LaSIE [47]) include implicit lexical information, but spread through different KBs, without access to a centralized lexicon. Furthermore, several *lexicons* are generally used for different linguistic processes.

POS tagging in fact refers to a given lexicon: HMM models store lexicalized probabilities, (i.e. n-grams frequencies [32, 33]), while transformational learning approaches [29] rely on lexical POS probabilities and lexicalized contextual rules. Even these shallow lexicons are specific to the domain and availability of training data is not always ensured.

Robust approaches to parsing are also dependent on (partial) lexical information: FAUSTUS, for example, relies on a subcategorization lexicon to deal with PP-attachment. Named Entity (NE) recognition grammars also rely on a variety of lexical knowledge: Gazzetteer entries and trigger words are typical lexicalized information. The lexicons used for these tasks are usually general, although NE catalogues or proper noun formation rules are not fully domain-independent. Furthermore, evidence from experimentation in application domains (e.g. [24]) reveals that a lot of specific senses of verbs, useful to design precise PP-attachment resolution (e.g. recurrent subcategorizations that are not strictly *argumental*), are absent from general lexicons.

Source lexical information is also widely adopted for word semantic classification (e.g. sense assignment). Selectional restrictions usually rely on semantic primitives to rule out ambiguities and extract correct interpretation for predicate argument relations. Accordingly, either domain ontologies (i.e. not strictly lexical sources) (LaSIE, [47]) or semantic dictionaries (LDOCE, WORDNET) can be used. WSD is also strictly related to complex IE inferences like interpretation for scenario recognition (either deductive or driven by some form of pattern-matching). More generally each scenario recognition process is based on some form of more or less lexicalized predicate argument interpretation. In systems like PROTEUS ([48]), specific semantic patterns are used in the early stages of the scenario recognition. Domain specific patterns like

$$< company\_name > \quad joint\ venture \quad < company\_name > \qquad (1)$$

are widely used to capture coarse relationships in the text, able to trigger the scenario recognition. This source of information is clearly lexical according to [72]. Besides a variety of considerations related to design, reusability and acquisition of such lexical information, it should be stressed here that a considerable improvement in performances (6 point increase of the MUC F's metrics) is reported, despite the relative coarse grain of the representation. Note that similar techniques are widely adopted in IE systems (SMES [62], MILK [28] or [34]). Even in not fully lexicalized approaches (LaSIE), the ontology includes a variety of patterns like (1) as relational links among (domain specific) concepts of the hierarchy.

Last, template filling processes are also lexicalized at a certain degree. In case a radical lexicalized view is adopted slot filling actions would be directly stored in the lexicon (e.g. Facile [6]). A less radical view would rely on sense definitions and lexical constraint satisfaction to combine the interpretation and the slot filling activity. In these cases the role of the lexicon is to store non redundant information in sense definitions and to support taxonomy traversing during the slot filling process.

2.1 Sources of Lexical Information

Lexical acquisition from MRDs, lexical knowledge bases and corpora can result in a significant advance in the development of robust and large scale linguistic

knowledge bases for IE systems. Adaptive capabilities for NLP systems require a set of well-established language samples to be used as source evidence for extracting and self-organizing lexical information. Traditionally, two *privileged* sources have been studied: Machine Readable Dictionaries (MRD) and Corpora.

Machine Readable Dictionaries (MRD) have been largely used to build computational dictionaries, better suited to be coupled with parsers, or to populate lexical knowledge bases: morphologic (part-of-speech, inflections, terminologic variants), syntactic (subcategorization frames and selectional restrictions) and semantic (sense distinctions, semantic classification and topics) information is generally extracted from the source MRD.

Many problems depend on the crude fact that heterogeneous formats of dictionaries (even of well established ones) pose hard problems to the induction process (see for example, Byrd et al in [27]). Complex extraction processes are required to map the dictionary content to specific lexical data structures (better suited for computational tasks). Such database oriented approaches aim to optimize the interface between a NLP system and the dictionary as *it is* (or as it has been filled in by humans).

Further analysis has also demonstrated that dictionaries are core repositories of a variety of information that is only implicit in the content itself: new information, not present (at least not explicit) in the dictionary, can also be derived. A typical example is the kind of taxonomic information expressed by textual definitions of word(sense)s. Definitions in dictionaries are synthetic ways of expressing taxonomic relations. Wilks et al., in [50], use as an example the following definition

"*knife - a blade fixed in a handle, used for cutting as a tool or weapon*"

where *blade* is a *genus* term, coupled with additional information that specifies inherent *differentiae* information: *knife* has thus *blade* as hypernim in a potential taxonomic representation. Textual definitions are also used to induce other semantic aspects, e.g. concept similarity [58] or selectional knowledge [61].

Nevertheless, criticism regarding the intensive use of MRD for lexical acquisition have outlined several drawbacks. The complexity of the extraction process is one issue: the format of dictionaries (even well established ones) is specifically designed to be used by humans. Special purpose parsers have been devised to support automatic approaches (e.g. PLNLP [55] or DEP [27]). Processing the dictionary content may be highly complex, too. Semantically ambiguous words are in fact widely used to disambiguate other words: in the example, the definition of *knife* refers to a specific sense of the ambiguous word *blade*. The circularity of this disambiguation process affects the extraction quality. Loss of information may thus affect the quality of the derived information and introduce inconsistency.

A further traditional limitation is due to the static nature of dictionaries, as snapshots of a given stage of development of a language, without support for adaptation to changes. On one side, coverage of linguistic phenomena (that are dynamic in nature) is inherently limited. Secondly, specialized languages,

with their own jargon, style and phenomena, and with a specific growth rate[1] are very difficult to characterize just relying on dictionary definitions. Several corpus-driven studies claim that relevant and specific lexical phenomena are usually absent in dictionaries ([17, 24]).

A complementary branch of research has thus been involved in studies and experiments in corpus-driven linguistics, with the aim of complementing, substituting or supporting MRD acquisition: data are extracted from real texts, as embodiment of language *in use*, to capture lexical regularities and to code them into operational rules.

Large scale corpora have been exploited ([4]) to study general phenomena in a language (by coupling several sublanguages in single samples) or narrower (i.e. domain specific) aspects (by selective sampling of given sublanguages). The focus in empiricist CL is on induction of computational NL models (quantitative or, at different degrees, rule-based). Pervasive ambiguity of language has thus been approached by inducing preference models (or rules) either from untagged or from previously disambiguated samples. Supervised and unsupervised approaches have often been proposed according to the availability of large scale controlled or raw data. However, large and balanced corpora are also problematic, as their construction and maintenance (extension/upgrade) is very costly.

A set of open issues still remains relevant to the activity of exploiting LA in an IE perspective:

- availability of controlled (i.e. disambiguated) corpora;
- cost effectiveness of the manual syntactic and semantic tagging process;
- scalability and robustness of induction methods with respect to noise in source data, uncompleteness of linguistic samples and rare phenomena;
- portability of induction methods and portability of extracted knowledge throughout domains/languages.

The use of corpora is not only of interest for Computational Linguistics as a basis to develop language processing tools, but it has also implications on previous research in lexical acquisition. Corpora have the relevant role of providing empirical evidence for guiding the MRD driven extraction. Corpora, in fact, are sources of knowledge for general studies on language, supporting and reinforcing linguistic intuitions. Moreover, they are repositories of functional information on word uses with respect to the more static information stored in MRDs. Hence, paradigmatic information stored in dictionaries may suitably be integrated with the syntagmatic evidence emerging from corpora:

Complex interactions between corpus evidence and MRDs are possible. *Parallelizing* MRDs and corpora may provide concurrent disambiguation models: combination of independent preference criteria (i.e. collocational data vs. definitory knowledge) may usefully deal with conflicting evidences (e.g. "*the astronomer*

[1] It is enough to consider differences in growth and transformation that characterize technical languages (e.g. computer science prose) with respect to more stable sublanguages (e.g. legal or administrative ones)

married the star" [76]). *Pipelining* approaches are also possible where MRDs or corpora are used in predefined order :

- MRDs first: A MRDs is taken as the general reference knowledge and corpus-driven tuning is then used to optimize domain specific inference ([12, 41]);
- CORPUS first: A corpus first helps in focusing on sets of target lexical phenomena and then the dictionary provides the suitable explanations of the extracted lexical evidences.

3 Corpus-driven LA: Methods and Tools

The empirical view on language processing that has currently attracted a wide consensus in computational linguistics (CL) has several implications on the type and use of lexical knowledge throughout both acquisition and application systems. It is our aim in this paper to outline issues related to expressivity and learnability of lexical knowledge, as they influence models and implementations of IE systems.

Since the early conference on Text based approaches to NLP ([54]), LA has captured a growing interest within the CL community. This also emerges from the high number of specific research initiatives of the last ten years: conferences ([1, 2, 3]) special interest groups (SIGCORP, SIGMLNL) and consortia for development of lexical resources (e.g. Linguistic Data Consortium (LDC)).

LA techniques have been applied to several linguistic tasks, as:

- POS tagging and lemmatization
- Terminology extraction
- Acquisition of grammatical information (e.g. verb subcategorization frames)
- Parsing (PCFGs, PP_disambiguation)
- Semantic disambiguation (WSD)
- Text Classification/Understanding
- Machine Translation

As a brief overview, a synthetic diagram reflecting the coarse methodological areas and some (not all, of course) successful results is reported in Fig. (1).

A non-exhaustive classificatory attempt may detect the methodological areas reported in Fig. (1):

1. **Statistical Induction** refers to probabilistic and stochastic approaches to LA, mainly characterized by the derivation of quantitative models. The main linguistic representation levels of these models are:
   - collocations (e.g. bigrams, trigrams, class-based n-grams)
   - syntactic features (e.g. grammatical collocations, noun-verbs)
   - lexemes (e.g. unigrams in word vectors)

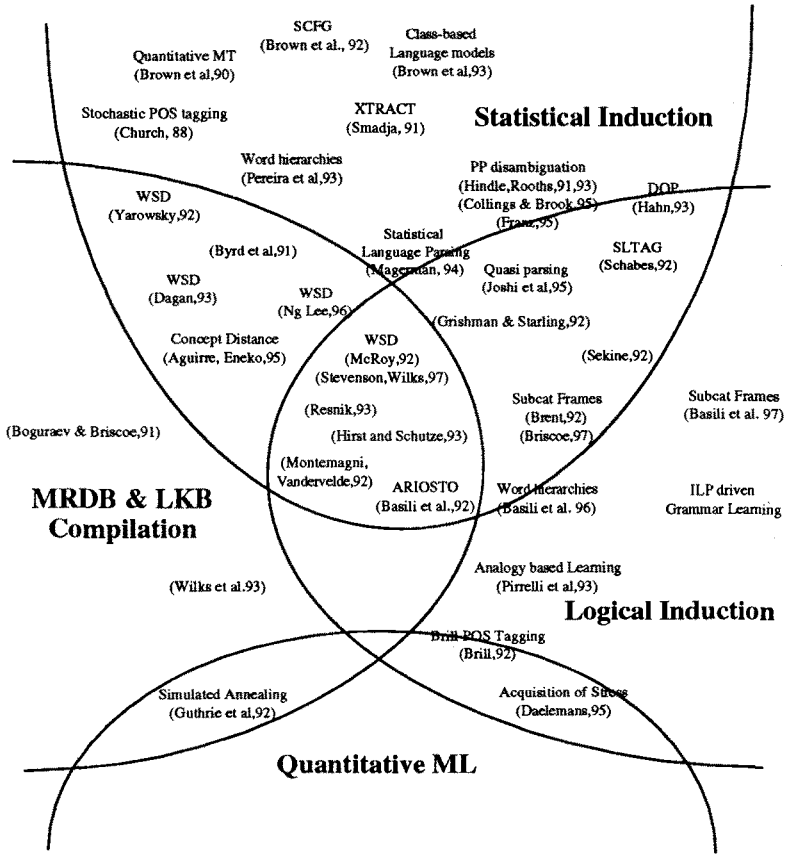


Fig. 1. Overview of LA areas and results

2. **Logical Induction** characterizes methods based on logical inferences usually drawn on symbolic representations, either at word, phrase or sentence level. In this area logical (or qualitative, [9] representations are used both as a source evidence (e.g. bracketed trees in annotated tree banks) and as the target information (e.g. lexical symbolic rules or word hierarchies). Methods in this area combine usually corpus evidences with complex linguistic representations: parse results (see DOP, [26] or [49]), or word semantic classes (see [70, 17, 23]) are typical examples. Usual inductive inferences are conceptual clustering ([22, 25]), word generalization [17] and collocation statistics. Conceptual clustering methods [35, 22], are included in this class as they map conceptual descriptions of positive examples into explicit logical descriptions of the derived classes (conjunctive membership conditions). The Brill learn-

ing method [29], based on transformation rules, is classified here according to the declarative nature of the acquired rules (i.e. lexical and contextual POS disambiguation rules)

3. **MRD & LKB Extraction** includes all methods that deals with some systematic source like dictionaries (e.g. LDOCE) or general purpose lexical knowledge bases (e.g. WORDNET). Methods of section 2.1 fall within this class. Note that, whatever the source is (MRDs or LKBs), target information of these methods may be very different: it varies from grammatical information (POS tags or surface syntactic rules, like subcat information) to semantic information like selectional constraints (e.g. [61]), sense definitions ([77, 39]) or taxonomic knowledge. Further classification mainly lies in differences in the inductive approaches. Numerical methods are widely adopted (as for example, conditioned probability, e.g. [79] or mutual information statistics, e.g [70]). Some methods are characterized by integration of several sources: parallel approaches (see previous section) are proposed, using corpora and dictionaries in a cooperative fashion [60, 78, 63].
4. **Quantitative Machine Learning** refers to other inductive methods that are not purely statistical (e.g. neural networks) although using numerical models for specific linguistic tasks (e.g. parsing). In particular not fully probabilistic methods (e.g. simulated annealing) are reported here (e.g. [39]).

The relevance of probabilistic approaches emerges by the higher density of the upper side of Fig. (1). These works are usually based on raw, untagged data (e.g. corpora) or on more complex training data sets (large collection of lexical resources (e.g. the bracketed tree collections, like PennTreeBank, [59])).

3.1 ARIOSTO: a large scale corpus-driven LA system

It will be hereafter described, the ARIOSTO LA system, as an instance of a corpus-driven approach to acquisition and modeling of lexical knowledge. ARIOSTO ([21, 19, 23]) is an integrated system for the acquisition of large-scale lexical information based on an hybrid probabilistic and symbolic approach. Several ideas on learnability and exploitation of shallow linguistic knowledge for complex NL tasks are embodied in the system, and have been tested over corpora for two languages: English and Italian.

In fig. (2) the current version of the ARIOSTO architecture is shown. Linguistic processing modules are white ellipsis while inductive (i.e. acquisition) modules are grey. As shown, feedback cycles are present in several phases:

1. *Terminologic analysis* is based on morphologic information produced by the lemmatizer and the POS tagger. A database of domain specific terminologic entries is acquired. It includes both relevant proper nouns (e.g. *Aereoporti di Roma*, *Agenzia Giornalistica Italia* in financial domains) and complex domain concept nominalizations (e.g. *imposte dirette* or *societa' a controllo publico*). A mixed syntactic and statistical approach, based on a grammar of legal terminologic expressions and on mutual information statistics, has been proposed. Details are in [11].

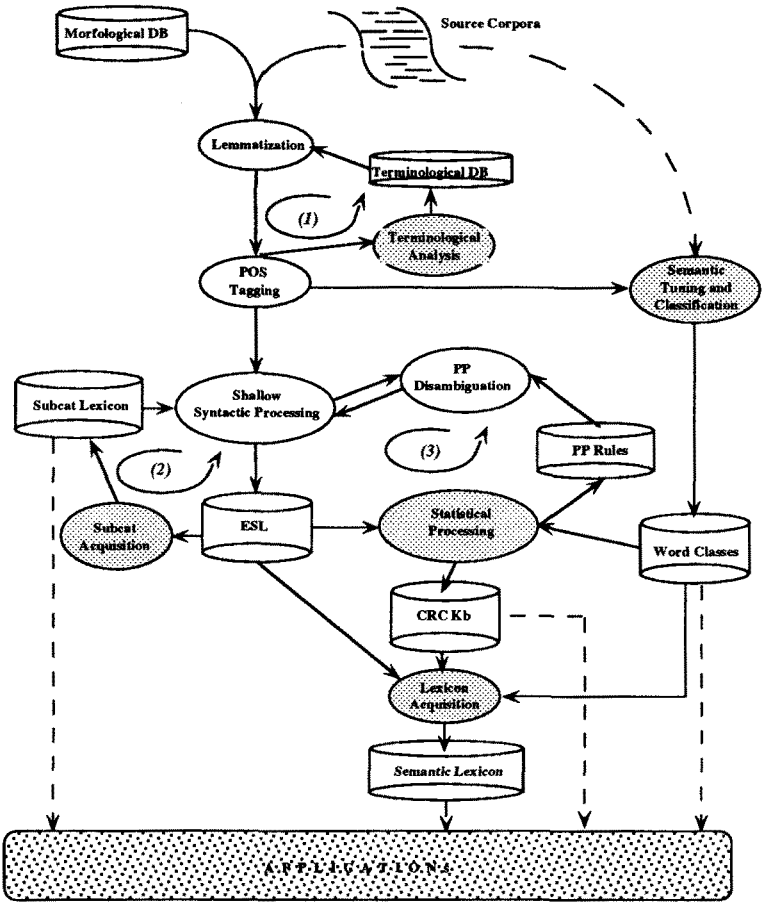


Fig. 2. The architecture of the ARIOSTO system

2. *Derivation of verb subcategorization frames* is triggered by elementary syntactic links extracted by SSA. A conceptual clustering algorithm is adopted to build a lattice of the potential surface subcategorization vectors, as they are instantiated by sentences including a given verb. Then an hybrid statistical and linguistic weight is assigned to each node in the lattice in order to

emphasize nodes that are statistically relevant in the corpus while expressing linguistically correct subcategorization patterns. Domain specific uses of verbs have been compared with a computational lexicon for Italian (LIFUV, [44]) and promising performances (i.e. correct subcategorization patterns) have been obtained for different corpora (about 75%-80% of precision and recall). Details are in [25]. The subcategorization patterns derived for frequent verbs in the domain are then used in ARIOSTO to control the SSA behavior on the corpus: several ambiguous verb phrases are disambiguated whenever enough argumental information is available. A second syntactic scan of the corpus provides less noisy data.

3. Finally, extracted esl are used to model prepositional disambiguation rules of structures like

   verb noun preposition noun2

   according to conditional probabilities P of trigrams

   verb preposition Class2
   noun preposition Class2

   where Class2 is the semantic type of lemma *noun2* (i.e. the modifier head). The semantic classification adopted in ARIOSTO is a flat taxonomy of (possibly overlapping) word sets: typical class labels are `Instrument`, `Possession`, `Abstraction`, `Human Entity`, ... Conflicting attachments are thus disambiguated with a "*disambiguate as late as possible*" strategy: global evidence (i.e. probabilities over the whole corpus) are used to carry out local disambiguation. Difficult cases (i.e. ambiguities with similar distributions) are retained. An iterative approach is proposed in [16] for incremental probability estimation of PP disambiguation rules.

The estimation of probabilistic preference for prepositional disambiguation as well as other acquisition modules (e.g. the Lexicon Acquisition module) crucially depend on naive semantic types . However, semantic ambiguity still exists even if high level tags (e.g. `Instrument` or `Human Entity`) are used. An average of 2.8 classes per noun has been evaluated in a 1.300.000 words corpus of financial news ([5]). In the architecture of Fig. (2) a specific module (i.e. "Semantic Tuning and Classification") is foreseen for acquisition of semantic classification rules in a domain. An unsupervised method for (1) tuning an existing taxonomy (i.e. Wordnet) by means of an application corpus and (2) local word sense disambiguation has been described in [13, 12]. Collective contexts as hints for sense assignment (in the narrow meaning of semantic classification) are used in a way similar to [79]. However, as opposed to this approach, a method to bias the probabilistic model of semantic classes (i.e. conditional probabilities of typical contexts) to the corpus is proposed: representative nouns/verbs in each semantic class are first selected according to the corpus and Wordnet. Only representative candidates in each class are thus used during the learning phase (see [12] for details). Performance evaluation on samples of verbs and nouns in

different domains has provided more than promising results. Manual seeding is used to compensate the lack of resources like Wordnet for the Italian language: probabilistic tuning and manual validation are thus interleaved, as incremental refinements. The resulting naive classification thus represents a good compromise between full corpus driven analysis and human validation. A similar approach has been recently proposed in [71]. In the architecture of fig. (2) the "Semantic Tuning and Classification" is applied since the early acquisition phases in order to support the class-based lexical induction stages, foreseen later.

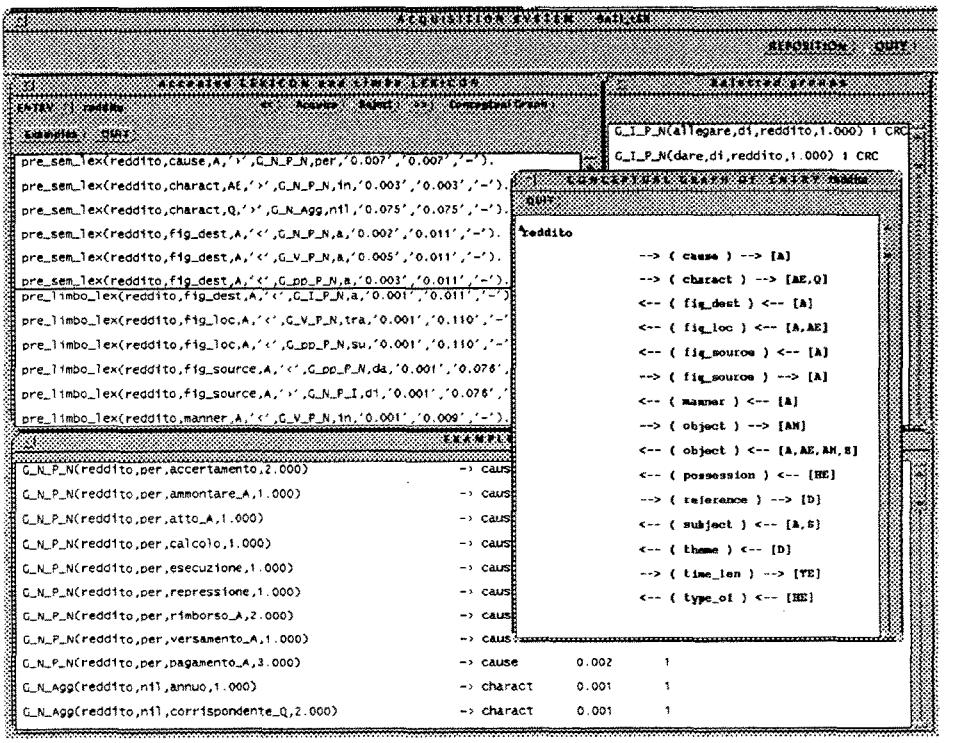


Fig. 3. ARIOSTO: Lexical Information for word reddito

As an example of the typical outcome of ARIOSTO, the derived lexical information for the word "*reddito*" (*income*) is shown in figure 3. The underlying corpus is a collection of legal texts on V.A.T. laws. The conceptual graph in linear form [75] expressing typical domain selectional rules is shown. Seman-

tic labels express high level classes: **A**,**AE**,**HE**,**D** denote **Acts**,**Abtract Entity**, **Human Entity** and **Documents**, respectively). Conceptual relations (usually specific to the domain, e.g. **cause**, **reference**, **possession**) are used as surface semantic interpretations of observed syntactic collocations. Domain specific catalogues are semiautomatically extracted from the corpus. Details of acquisition of this selectional information are in [17].

In synthesis, the main issues of ARIOSTO can be outlined:

- *Portability*: the system is specifically designed as a support for the development of computational lexicons for *real* NL applications in specific knowledge domains. The applicability of tools and methods throughout a variety of sublanguages has been a major inspiring factor. The aim is to support the knowledge engineer since the early stages of design and development of the required linguistic knowledge bases.

- *Corpus-driveness*: ARIOSTO makes extensive use of corpus evidence to extract significant components of the background lexicons necessary in a NL application. Semantic selectional patterns [17], lexicalized rules for prepositional phrase disambiguation [20], acquisition of semantic classes in word hierarchies [22] are few examples.

- *Shallow syntax*: a syntax-driven processing is adopted in ARIOSTO to extract linguistic evidences from the corpus. The syntactic approach is based on a robust discontinuous grammar [18] aiming to derive possibly ambiguous elementary relations (elementary syntactic structures, *esl*) in sentence fragments. Ambiguous texts are also processed, and conflicting readings (i.e. mutually exclusive *esl*) are retained. Uncertainty scores are then assigned to each *esl* aiming to maximize inferences drawn over unambiguous *esl*. Shallow here refers both to the adopted grammatical formalism (sentences are rewritten into sets of esl rather than in trees) and to the parsing activity that uses very local evidence to drive the grammatical recognition (Shallow Syntactic Analyzer, SSA [18, 15]).

- *Shallow semantics*: in ARIOSTO, as in several lexical acquisition methods (e.g. [70, 30]), semantic classes rather than purely numerical clusters (e.g. [64, 43, 31]) are used. As the early phases of corpus processing cannot rely on precise semantic classification, an high level semantic type system is adopted to assign a naive interpretation to most of the corpus derived observations. The source taxonomic information (i.e. the class catalogue) for the English language has been usually derived from high level nodes in the WordNet hierarchy (see [13, 12]). For the Italian language (where systematic sources of taxonomic information are still missing) manual encoding has been largely adopted even if a semiautomatic procedure is under experimentation.

- *Hybrid Symbolic and Probabilistic acquisition*: availability of large data samples as well as semantic information is combined in ARIOSTO to integrate probabilistic modeling (conditioned probabilities and/or mutual information

statistics) and logical induction (i.e. maximally specific generalization of words into classes). The combination of different learning techniques allow to overcome some well known problems:

- *data sparseness*, as class based information is less affected by the low frequencies of rare phenomena,
- *readability*, as syntactic and semantic information is described declaratively in the system,
- *coverage and scalability*, as numerical methods are better suited for large scale acquisition than purely symbolic ones

– *Incrementality*: several acquisition phases are designed to be incrementally refined. Early phases apply very constrained disambiguation rules, aiming to delay data pruning (e.g. syntactic disambiguation) as late as possible. Syntactic processing, for example, is first applied according to limited grammatical constraints (simple *esl* are extracted from very local contexts, collecting data even from conflicting sentence readings): disambiguation and pruning actions are undertaken later whenever larger data sets derived from the whole corpus are available. The same apply to induction of word classification rules: hand-validation of the first outcome may result in a re-estimation of the related probabilities. Incremental corpus processing provides more precise data for the later acquisition tasks in fig. (2).

All ARIOSTO modules have been fully implemented. Integration within complex software infrastructures (i.e. the GATE architecture [42]) has been recently accomplished for all linguistic preprocessing phases.

4 Integrating Lexical Acquisition and Information Extraction

With respect to a lexical phenomenon X, that seems useful to some IE tasks, the following questions should be answered:

- is this lexical information related to X learnable from the available (domain specific or not) source?
- what kind of lexical inferences are supported by information on X ?
- what kind of extra-linguistic inferences (e.g. domain specific NE recognition or template filling) can also be triggered ?
- how much effort is to be spent in the acquisition activity so that designed inferences based on X are enough precise and robust (i.e. what is the efficacy of the foreseen acquisition of X, within the overall IE process) ?

Answers to these questions require adequate models for shallow lexical knowledge acquisition and suitable integrated architectures. The idea is to clearly separate the shallow knowledge required by the IE process from the specific and more precise lexical information strictly dependent on the application.

4.1 Foreground and Background Lexical Knowledge

One of the main features of empiricist computational linguistics is the emphasis given to shallow information. Shallow lexical models, as opposed to explanatory knowledge about deeper linguistic phenomena, have played a crucial step forward in several LA methods. It is worth noticing that shallow knowledge is also important in several phases of text-processing task in NLP applications, in particular IE. This issue is very important to better understand the contribution of LA to IE.

Lexical information useful to IE tasks includes two main components ([57]): a *foreground* lexicon that refers to those specific words able to trigger certain IE actions, as opposed to the remaining words , i.e. the *background* lexicon.

Foreground lexical knowledge includes explanatory knowledge that supports complex inferences ([57]). Typical examples of foreground information in IE are lexicalized information like (1), where rules of scenario recognition are stored as selectional patterns. The dependency of this information on the target application is here expressed by specific sets of words, used as triggers for extraction actions (e.g the key term *joint venture* in (1)). These words are usually verbs denoting important domain events: in financial domains relevant expressions are for example *joint venture*, *resign* from a position, *announce* a new product, ... This kind of information has implications on the typical verb subcategorization frames. Specific senses of verbs are more relevant in a domain: in a financial domain *operare in un settore* (*to operate in a commercial field*), is more relevant than *operare una persona* (**to operate (surgically) someone* or *someone is operated*)). The selectional information related to those senses (i.e. the subcategorization *to operate in "a field"* or *"in an organization"*), should be explicitly coded in the target IE foreground lexicon. Similar subcategorization information can be used to trigger template filling actions: radical lexicalized approaches would store the corresponding rules directly in the lexicon. Sentence fragments that satisfy the lexical constraints (i.e. the suitable verb argument structure) are easily detected and used as source information for the template filling.

Background information is less dependent on the application and mostly relates to pure linguistic inferences. POS lexicons (e.g. the POS probabilities of lexical entries in [29]) are typical instances of background information. POS tagging is in fact carried out on a lexical basis. These lexicons are independent from the application, as there is no special constraint on specific sets of words. However, they are dependent on the underlying sublanguage, as they are sensible to changes in the domain. In an experiment carried out on a Remote Sensing domain (about 500,000 words) POS tagging has been carried out by the Brill POS tagger trained on the Wall Street Journal and 10% of words were unknown to system (i.e. they were absent in the POS lexicon). This resulted in a significant decrease in the performances (-7%).

Other typical background information relates to *word senses*: although word senses described in the foreground lexicon are very application specific, the semantic description of the remaining words is also important for IE. The satisfaction of lexical constraints, as they are stored in a foreground lexicon (e.g.

subcategorization frames and predicate argument relations), still relies on the semantics of the surrounding words: arguments must undergo a first interpretation to fill predicate positions. In the sentence

L'azienda opera in borsa dal 1994
The enterprise operates in the Stock Exchange since 1994

the argument structure of the verb *operare* selects the `Organization` sense of the italian word *borsa*, while neglecting the (usually more likely) sense `Artifact` (i.e. *bag*) of the same word. In order to activate the correct extraction, the foreground (i.e. the argument structure of the verb *operate*) and the background information (i.e. word senses of the word *borsa*) must interact. Note that the word sense definitions (e.g. verbs in the foreground lexicon relevant for the application like *operare*) require a more precise representation. The granularity of word senses in the background lexicon should be able to combine with foreground information, in order to optimize the trade-off between coverage and precision (i.e. minimize undergeneration and overgeneration). Domain specific classes are of course needed as different domains may require very different semantic descriptions. Furthermore, high level semantic description (i.e. coarse semantic classification rather than more precise sense definition) seems adequate for the encoding of the foreground selectional information. Hence, while the foreground component is tightly bound to the ontology, as it includes the words that map into the ontologic concepts of the domain, a more coarse level can be used in the background lexicon. The former can thus be used for semantic inference and template filling according to a detailed (ontologic) domain description. The latter is responsible for morphosyntactic disambiguation as well as for a shallow semantic interpretation.

As the size of background lexicons is considerable (hundreds of thousands of words), robust acquisition methods of such lexical information are crucial tools in the engineering of large scale IE systems.

Acquisition of background knowledge is carried out at different levels.

Collocational data have been suggested as a key information for tasks like:

- lexical encoding of morphological information (e.g. the *powerful* vs. *strong tea* example [74]),
- grammatical inference (e.g. stochastic approaches to POS tagging [37] and parsing [33]),
- syntactic disambiguation (e.g. PP-attachment resolution [51, 52, 46, 38, 20, 16, 30])
- word sense disambiguation ([79, 12]).

Non purely collocational approaches apply at a syntactic level. Syntagmatic information (i.e. syntactic collocates) is extracted from corpora by several robust parsing techniques. Rather than monolithic functions, these parsers are stratified systems [8], finite state devices (([10] or [36]) or shallow recognizers ([18]). *Shallow* here refers not only to the applied strategy (i.e. the grammar) but also

to the supported grammatical inferences. For example, several syntactic disambiguation models are built over shallow syntagmatic information ([52, 20, 30]). The results of these methods are usually large repositories of lexical probabilities applied to PP-attachment resolution.

Semantic information typical of background lexicons relates to sense definition and semantic relations. Several methods based on collocational data (e.g. [79, 13]), or on multiple sources ([60, 78, 63]) have been proposed for automatic WSD and semantic classification. The results of these systems are either sense disambiguation rules in terms of semantic classification scores (e.g. [13, 12]) or word sense preferences, (e.g. [60]) or probabilistic models of semantic classes (e.g. [79]). As suggested in the ARIOSTO architecture a word class domain dictionary should serve the purpose of a naive semantic classification useful to accomplish the required IE actions (e.g. scenario template recognition, as in (1), or interpretation of predicate argument relations).

Background lexicons also list semantic relations used during the interpretation stages. Due to the shallow nature of semantic classes, a shallow representation should be also adopted for the required relational knowledge. Although every system deals with the complexity of the syntax-semantics mapping, there is no consesus on a unified set of relations [45]. Relational and compositional views on sense representation are possible. As a result, different systems suggest very different formalisms. Domain specific conceptual relations have been early suggested to express selectional information [75]. Corpus-driven acquisition of this type of lexical knowledge characterizes systems like ARIOSTO [17, 23]. Among other lexical representation formalisms, the Generative Lexicon (GL, [66, 68]) includes a variety of lexical features to tackle lexical redundancy: exhaustive listing of meanings has several drawbacks as proliferation of senses and poor explanatory power [68]. Complex lexical features are also useful in lexical acquisition. Although these models include deeper lexical information and their full induction is thus harder, approaches have been proposed to integrate GL-based inferences and acquisition ([69], [56] and [34]). Availability of powerful mechanisms for lexical inference (e.g type coercion, [67]) suggest such representation for the more complex tasks typical of the foreground lexicons (see for example, [28]). Other approaches to corpus-driven [49, 70] or MRD-oriented [61] acquisition of selectional information have been also proposed.

The LA contribution to the engineering of background lexical knowledge matches most of outlined requirements for IE systems. Morphosyntactic knowledge bases for lexical preference rules, lexicalized syntactic preference rules, word semantic classification and relational models of the lexicon are the crucial components that can be acquired. Integration within an IE system requires a specific architecture with a clear separation between foreground and background lexical information.

4.2 LA for IE: An integrated architecture

In fig. 4 a possible architecture to integrate LA capability within an IE framework is reported.

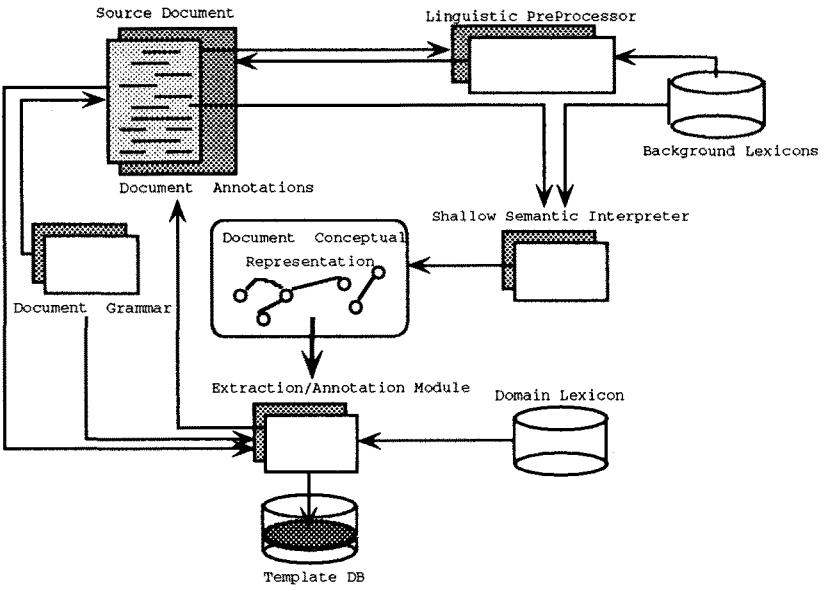


Fig. 4. IE Architecture: an LA approach

The Linguistic Processor provides a set of linguistic annotations as usually produced by "classical" morphosyntactic modules, by exploiting (a set of) basic lexicons (Background Lexicons). Fig. 5 describes the architecture of the linguistic processor used in ARIOSTO. The background lexicons are used by the early preprocessing phases while the foreground lexicon mainly relates to IE actions. The dependence of these lexicons on the application domain, require specific and independent LA processes.

Once the document has been linguistically annotated by POS, syntactic and word sense information, the Shallow Semantic Interpreter carries out a coarse grained semantic analysis. In this phase, interpretation of the main grammatical relations, as they have been (possibly ambiguously) found in the text, is foreseen. A further level of disambiguation based on coarse selectional restrictions is also accomplished. Selectional restrictions at lexical or semantic level (as those shown in fig. 3) are here foreseen. LA methods for this kind of information ([49, 70, 17, 23]) have been discussed in section 4.1.

The results of the Shallow Semantic Interpreter are coarse interpretations characterized by:

1. a reduced syntactic ambiguity, as gross errors in parsing have been removed;
2. an early interpretation of grammatical relations (e.g. Prepositional modifiers) by means of high level semantic relations, either thematic interpreta-

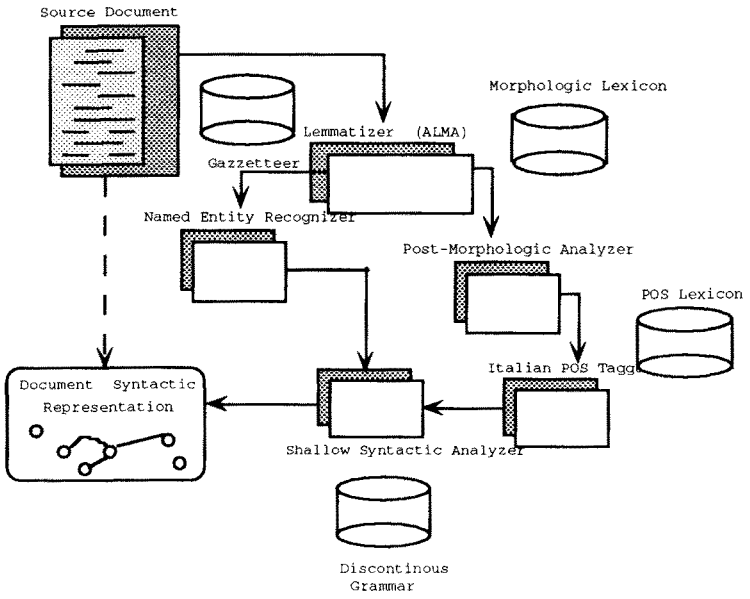


Fig. 5. The functional Architecture of the ARIOSTO Linguistic Processor

tions (e.g. `theme`, `agent`, `location`) or more application-oriented conceptual relations (e.g. `authorship`, `source_Information`, ...);

3. a partial words sense disambiguation: according to the imposed selectional constraints more precise senses can be assigned to ambiguous words (i.e. words belonging to multiple semantic class). Persistent ambiguity is retained, so that multiple interpretations are still allowed. Further domain specific information can be used later to drive the correct interpretation;
4. uncompleteness: the interpretation process may apply to simple fragments of the source sentence. This may improve tolerance to ungrammatical phenomena, still allowing for adequate extraction.

The Extraction/Annotation module completes the interpretation phase according to application specific knowledge, as it is stored in the Domain Lexicon, and accomplishes the suitable scenario recognition and template generation process. The derived information at this stage may result in complex data structures (i.e. templates) or synthetic annotations.

The role of LA within this IE architecture is very relevant:

1. Background lexicons are largely domain specific so that specific acquisition methods are required

2. Robust grammars (i.e. grammatical models used by robust parsers) as opposed to general grammars are not so strictly domain specific so that "*disambiguate as late as possible*" strategies may be supported
3. Partial parsers require more complex syntactic disambiguation. Large scale disambiguation methods have been mentioned: their portability favors an intensive application within IE frameworks.
4. Word senses are highly domain specific. Coarse classifications, rather than more granular sense distinctions (e.g. WordNet), favor a less complex learning, thus ensuring portability throughout application domains. Furthermore, incremental refinement is still allowed even when seeding sense distinctions are rather naive.
5. Selectional information can also be acquired from corpora and used by the Shallow Semantic Interpreter. A crucial aspect is the relationship between shallow semantic information (foreseen in the Background lexicon) and the foreground lexicon. Main problems here are related to the adherence of the former to the domain ontology, usually encoded in the domain (foreground) lexicon:
   - The set of domain concepts (e.g. `governative organization`) should easily map into high level classes: undesired noise due to the generality of the latter, should be minimized. When a tuned domain classification is available (as in ARIOSTO is carried out by the "Semantic Tuning and Classification module") lexical constraints should successfully control the overgeneration.
   - The adherence to the relational description of the ontology should support unambiguously the mapping between the set of coarse grained semantic relations (used in the background lexicon) and the semantic primitives adopted within the foreground lexicon. As the latter are tied to the application they are usually more constrained, so that suitable coding should avoid ambiguity. Whenever a relation in the foreground lexicon (e.g. `leadersihp`) is activated by more than one background relation (e.g `agentiship` or `actorship`), a control of the semantic mismatch is required. Complex mechanisms can be applied (e.g type coercion [67]) in richer lexical formalisms. However, as the size of the foreground lexicon is rather limited, the encoding of such complex lexical control is not a prohibitive enterprise.

4.3 Foreground Lexical Knowledge Acquisition in ARIOSTO

Two main lexical components are mentioned in the architecture of fig. (4): the background lexicon, that is the domain specific lexical information aiming to support document linguistic preprocessing as well as surface semantic interpretation, and a domain (foreground) lexicon mainly devoted to guide the Extraction/Annotation module. In section 3.1 issues related to corpus-driven LA of background lexical information has been discussed.

Hereafter, we will discuss forms of semi-automatic acquisition methods of domain lexical knowledge. As an instance of extraction system, we will refer to

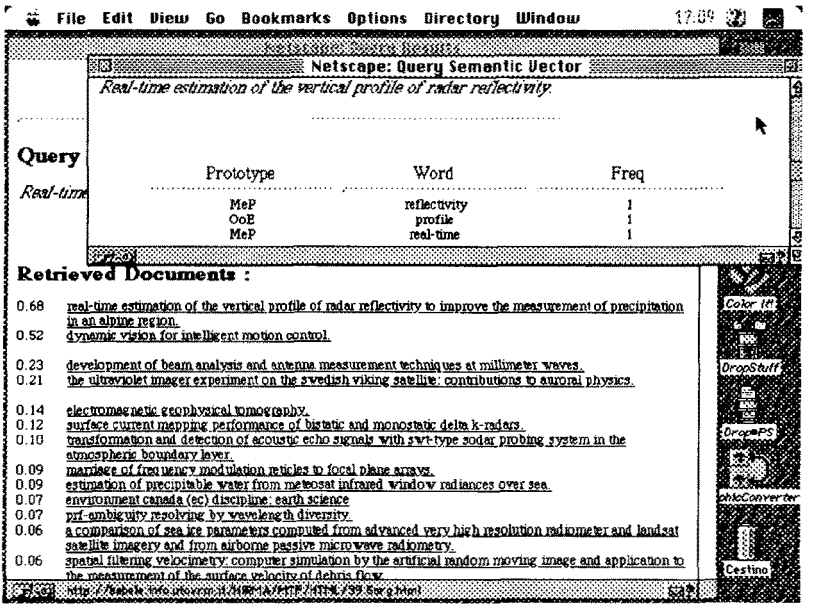


Fig. 6. HIRMA: The result of a Query in the Remote Sensing domain

an NLP system for content driven retrieval and hypertextual authoring of large scale document bases. HIRMA ([14]) is a system for document indexing based on lexical knowledge derived by ARIOSTO. In HIRMA a document index is selected via recognition of word contextual role in the source document. In Fig. 6 the Netscape version of the HIRMA user interface is shown. In the background window the results of a NL query (in English) are suggested as lists of document titles with relevance scores. In the upper right window the interpretation of the user query, as it is produced by HIRMA, is shown. The query (as each document) is interpreted as a set of words augmented with a conceptual description of their role in the query (or in the document). Their frequency is also accounted for, as relevance scores depend on the relative frequency according to an extension of the vector space model [73]. Hence, indexes are words (or terminological word strings) with their specific contextual role and frequency in the document. A *contextual role* is an aspect of a word sense by which a given word is used in a document: the content of the document depends on some relevant words given the set of roles they play in the text.

For example, let us consider two source sentence fragments:

Earth Observation Satellite Systems ... (2)

and

The satellite has been designed by MATRA ... (3)

The following rules (expressed by shallow semantic classes and conceptual relations) can be used to distinguish the two meanings of *satellite* in (2) and (3):

```
Role: Experimental Instrument

[Satellite: Artifact] -(Instrument)->
     [Observation:Act]-(Source_Info)->[Earth:Natural_Object]
```

and

```
Role: Object of a Building Activity

[Satellite: Artifact] -(object)->
     [design:Act]-(Agent)->[MATRA:Human Entity]
```

The above lexical rules express the fact that Satellites are used as Instruments to observe Natural Objects (i.e. the *Experimental Instrument* role in (2)) and that Satellites can be designed by Human Entities (i.e. *Object of a Building Activity* role in (3)).

The two contextual roles are built over the same word sense but describe different aspects, strictly related to the application domain. In the RS domain a well established separation between the active role of instruments (i.e the notion of experimental instrument related to sensors that actually sense geophysical phenomena), and their passive role (i.e. machinery resulting from building activities). Several documents in fact deal with design and development technologies of satellites, sensors, ... Typical user queries focus on these two essentially different kinds of information:

- how a sensor is built, what technologies it uses and where it has been produced/assembled (*Object of Building Activity* role)
- what can be measured by an instrument, how it is used for these purposes (*Experimental Instrument* role)

Hence, while word senses are a typical linguistic information, contextual roles mainly depend on the application, i.e. they are typical foreground lexical information. In HIRMA, contextual roles are assigned on a lexical basis: sentence fragments undergo a partial (surface) semantic interpretation that is used to trigger lexicalized role assignment rules. Note that patterns are usually domain specific, as "*observing*" natural object may be a very odd selectional restriction in domains different from Remote Sensing. Note that rules for role recognition in HIRMA depend on the previous shallow semantic interpretation that is supported by the lexical knowledge acquired by ARIOSTO. Semantic interpretation and role assignment are instances of the "Shallow Semantic Interpreter" and "Extraction/Annotation Module" suggested in the IE architecture of fig. (4).

In HIRMA a feature-vector representation is used to represent documents: thus contextual roles annotations are added to the source text. In HIRMA IR-like queries in natural language are supported and the vector representation

of documents in terms of sets of (word,contextual role) couples better fits the requirements of retrieval process. In the upper right window of Fig. (6), the vector representation of the user query is reported. Two "Measured Parameters" (i.e. **MeP** labels for *reflectivity* and *real-time*) have been found in the query.

Role assignment rules (as for examples (2) and (3)) can be very lexicalized, i.e. triggered by specific sets of words (or word strings), like *satellite, sensor, Meteosat, altimeter* , or expressed in terms of the high level ontologic concepts (i.e. **Instrument**, **Human Entity**, **Abstraction**). Role assignment rules are very close to information employed in typical IE processes, e.g. scenario recognition.

According to the their application oriented nature, the design and development of role assignment rules is very complex: the porting to newer applications is very difficult as most of previous knowledge can be useless. Learning is also an open issue. Corpora, and particularly application corpora, are not repositories of this kind of information. Inductive methods that only rely on the corpus have no hope to discover what is needed. Extending corpora with samples of previously tagged documents or documents indexed by contextual roles may be used to support automatic acquisition. However, the construction of balanced, wide coverage training sets can be a very costly process, and effectiveness of the resulting acquisition is still to be proved.

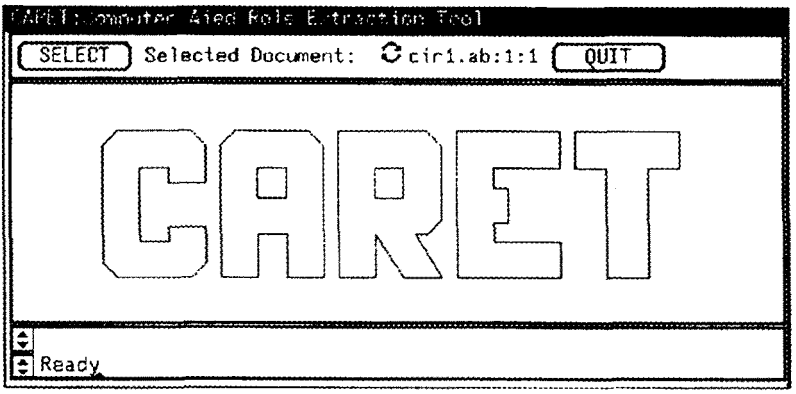


Fig. 7. CARET: the user interface

In order to develop the lexical knowledge required for role assignment, a semi-automatic approach has been defined in HIRMA. A special purpose prototype system supports the knowledge engineer in the definition and implementation of these lexical rules. CARET (Computer Aided Role Extraction Tool) has been fully implemented as a module of HIRMA and tested over three domains: the remote sensing domain in English, a legal domain and an environmental domain in

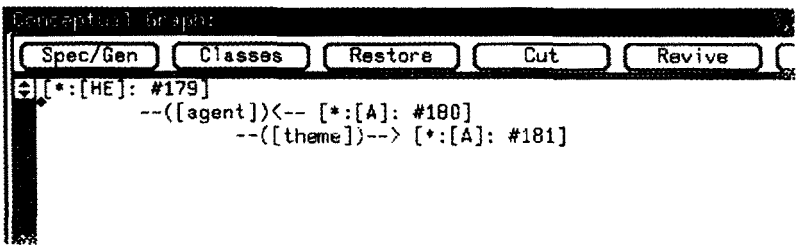


Fig. 8. CARET: GUI for contextual role compilation rules

Italian. The adopted surface semantic description allows to design a unified representation formalism for different domains and natural languages. In CARET, a set of training documents is first selected. Surface interpretation is then run by CARET over the training documents. Ambiguous interpretations are retained. Then a graphical user interface (GUI) allows the knowledge engineer to

1. select a sentence fragment of interest f in a document d
2. access the interpretations of f, due to the possibly ambiguous word senses, to the possible syntactic readings and to the corresponding different surface semantic interpretations of f. Interpretations are conceptual graphs, and they are visualized in their linear notation ([75]).
3. remove the undesired interpretations
4. generalize the interpretation by:
   (a) removing relations that are accidental in the document, i.e. that are not necessary preconditions for the role assignment actions
   (b) generalizing patterns by mapping words in the patterns to their suitable semantic type
5. select a single word that guides the overall interpretation and define it as the role head
6. and, finally, assign the suitable role label to the resulting subgraph.

The final pattern describes thus the set of lexico-semantic constraints sufficient to activate the role assignment in HIRMA. The availability of CARET in HIRMA is very important for issues like portability and large scale document indexing: CARET in fact does not rely on specific lexical knowledge different from the background information built by ARIOSTO (i.e. a domain specific set of classes and a catalogue of semantic relations). The kind of supervised learning supported by CARET allows a user-friendly compilation of the foreground knowledge, otherwise absent in the raw corpus. Semi-automatic large scale acquisition via CARET thus eases the system scale-up, independently from the underlying domain and language.

In Fig. 7 the CARET front-end is shown. In the early interface the user can select a file and a corresponding document in a given domain. When the document is shown in a text window, a given word w (i.e. a potential index) can be selected: the choice of the SELECT button allows the on-line generation of the desired conceptual interpretation (i.e the conceptual graph headed by w is built). In Fig 8 the interactive menu aimed to modify/extend the interpretation is shown.

The button SPEC/GEN activates a contemporary generalization (i.e. words in the pattern are changed into semantic types) or specialization (all semantic types are rewritten into their source words) of all the nodes in the underlying conceptual graph. Button CLASSES lists and allows to select one among the semantic types of a word selected in the graph. Button CUT allows to remove relations acting as undesired constraints for the rule under construction. RESTORE and REVIVE allow to undo some previous CUT actions on the whole graph or on subtrees respectively. Finally the utility STORE is used to compile the target Prolog rules. In Fig 8 a lexical pattern has been already obtained and the corresponding Prolog rule is the following:

```
{---------------------------------------
 Login Name: ariosto
 User Name : enea
 Date      : Tue Dec  6 10:06:23 WET 1996
 ---------------------------------------}
prototype('Human Actor','HE', _179) :-
        rel(agent,_180,[A],_179,[HE]),
        rel(theme,_180,[A],_181,[A]),
        !.
```

Predicates `rel` will guide the pattern matching phase. The extraction process is carried out over the syntactic information derived by the shallow parser SSA. When a component in the syntactic graph satisfies both the required conceptual relations (i.e. predicates `rel(theme, ...)` and `rel(agent, ...)`) the rule is saturated. The word matched by variable `_179` in the rule (i.e. the human entity in the document fragment that is the agent) is selected as potential index for the source document and its role is unified to `Human Actor`.

In Fig. 6 the HIRMA interpretation of a user query is shown: the recognition of the word contextual roles (e.g. `Measured Parameter` (`MeP`) for *reflectivity* or `Object of Experiment` (`OoE`) for *profile*) results in the document (semantic) vector (right upper window).

5 Conclusions

In this paper the contribution of LA methods and tools proposed in the recent CL literature have been discussed. Integration of LA capabilities within an IE system requires a specific architecture. A general architecture, with a clear separation between foreground and background lexical information, has been proposed.

Background lexicons have a large size and are largely domain dependent so that specific acquisition methods are required. The LA contribution to the engineering of background lexical knowledge matches most of outlined requirements for IE. Morphosyntactic knowledge bases for lexical preference rules, lexicalized syntactic preference rules, word semantic classification and relational models of the lexicon are crucial components in an IE system. Robust grammars, typically used in IE, are not so strictly domain specific. However, suitable disambiguation strategies may improve (in a "*disambiguate as late as possible*" fashion) the performances of the linguistic preprocessors in new sublanguages. Several corpus-driven syntactic disambiguation methods have been largely experimented over different languages and may serve this purpose. A further highly domain specific information relates to background word senses. We suggests that coarse classifications, rather than more granular sense distinctions (e.g. WordNet), are better suited. They require simpler learning techniques, supporting incremental refinement as well. Domain specific selectional information can also be acquired from corpora and used by the preliminary phase of shallow semantic interpretation. As an example of a general framework for corpus driven Lexical Acquisition, the ARIOSTO system has been briefly described.

Typical foreground lexicons are highly domain and application dependent. Their portability is thus very limited. Evidence for foreground information is poor in language samples and manual development of training data may be prohibitive.

In this paper, a tool for the acquisition of foreground information used by a NLP-based IR system (HIRMA) has been described. CARET (Computer Aided Role Extraction Tool) supports the semi-automatic development of lexical rules for word contextual roles recognition. In CARET, a set of training documents is first selected. Surface interpretation is then run by CARET over the training documents. Ambiguous interpretations are retained. Then a graphical user interface allows the knowledge engineer to prune and refine the rules, generalize the semantic constraints and compile the corresponding ProLog clauses. A unified formalism for the surface semantic interpretation and the lexical rule representations is adopted throughout different domains and natural languages. CARET relies on domain specific background knowledge (as it is acquired by ARIOSTO), thus supporting large scale acquisition of foreground information, independently from the underlying domain and language. CARET has been fully implemented as a module of HIRMA and tested over application domains in English and Italian.

References

1. In *Proceedings of ACL Conference on Very Large Corpora, 92-97.*
2. In *Proceedings of Conference on Emprical Natural language Processing, 96-97.*
3. In *Proceedings of Conference on New Methodologies for Language Processing, 94-96.*
4. *Computational Linguistics, Special Issue on Very Large Corpora*, 1993.

5. Ecran: Extraction of content: Research at near-market. Technical report, EC ESPRIT - LE n. 2110, 1995.
6. Facile: Fast and accurate categorisation of information by language engineering. Technical report, EC ESPRIT - LE, 1995.
7. Proceedings of the sixth message understanding conference (muc-6). In *Columbia, MD,*. Morgan Kaufmann, 1995.
8. S. Abney. Parsing by chuncks. In *Principle-Based Parsing.* Kluwer Academic Press, 1991.
9. H. Alshawi. Qualitative and quantitative models of speech translation. In *Proc. of the ACL Workshop, The Balancing Act, Las Cruces, New Mexico, USA*, 1994.
10. D. Appelt, J. Hobbs, J. Bear, D. Israel, and T. Mabry. Fastus: a finite-state processor for information extraction from real-world text. In *Proc. of the International Joint Conference on Artificial Intelligence*, 1993.
11. R. Basili, G. De Rossi, and M.T. Pazienza. Inducing terminology for lexical acquisition. In *Preoceedings of the Second Conference on Empirical Methods in Natural Lanaguge Processing, Providence, USA*, 1997.
12. R. Basili, M. Della Rocca, and M.T. Pazienza. Contextual word sense tuning and disambiguation. *Journal of Applied Artificial Intelligence*, 1996.
13. R. Basili, M. Della Rocca, M.T. Pazienza, and P. Velardi. Contexts and categories: tuning a general purpose verbclassification to sublanguages. In *Proc. of International Conference on Recent Advances in Natural Language Processing, Tzigov Chark, Bulgaria*, 1995.
14. R. Basili, F. Grisoli, and M.T. Pazienza. Might a semantic lexicon support hypertextul authoring? In *Proc of the 4th ACL Applied Natural Language Processing, Stuttgart (Germany)*, 1994.
15. R. Basili, A. Marziali, and M.T. Pazienza. Modelling syntactic uncertainty in lexical acquisition from texts. *Journal of Quantitative Linguistics*, 1, 1994.
16. R. Basili, A. Marziali, M.T. Pazienza, and P. Velardi. Unsupervised learning of syntactic knowledge: Methods and measures. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing, Philadelfia, Pennsylvania*, 1996.
17. R. Basili, M.T. Pazienza, and P. Velardi. Computational lexicons: the neat examples and the odd exemplars. In *Proc. of Third Int. Conf. on Applied Natural Language Processing, Povo, Trento, Italy*, 1992.
18. R. Basili, M.T. Pazienza, and P. Velardi. A shallow syntactic analyzer to extract word associations from corpora. *Literary and Linguistic Computing*, 7(2), 1992.
19. R. Basili, M.T. Pazienza, and P. Velardi. Acquisition of selectional patterns. *Journal of Machine Translation*, 8, 1993.
20. R. Basili, M.T. Pazienza, and P. Velardi. Semi-automatic extraction of linguisitc information for syntactic disambiguation. *Applied Artificial Intelligence*, 4, 1993.
21. R. Basili, M.T. Pazienza, and P. Velardi. What can be learned from raw texts? *Journal of Machine Translation*, 8, 1993.
22. R. Basili, M.T. Pazienza, and P. Velardi. *A context driven conceptual clustering method for verb classification.* MIT Press, 1996.
23. R. Basili, M.T. Pazienza, and P. Velardi. An empirical symbolic approach to natural language processing. *Artificial Intelligence*, 85, 1996.
24. R. Basili, M.T. Pazienza, and P. Velardi. Integrating general purpose and corpus-based verb classifications. *Computational Linguistics*, 1996.

25. R. Basili, M. Vindigni, and M.T. Pazienza. Corpus-driven unsupervised learning of verb subcategorization frames. In *Proc. of Conference of the Italian Association for Artificial Intelligence, AI*IA 97, Rome*, 1997.
26. R. Bod. Using an annotated corpus as a stochastic grammar. In *Proceedings of the European Chapter of ACL (EACL-93)*, 1993.
27. B. K. Boguraev and T. Briscoe. *Computational Lexicography for Natural Language Processing.* Longman Group, 1989.
28. A. Bolioli, L. Dini, V. Di Tomaso, A. Goy, and D. Sestero. From ir to ie through gl. In *Proc. of International Workshop on Lexically-Driven Information Extraction*, 1997.
29. E. Brill. A simple rule-based part of speech tagger. In *Proc. of the Third Applied Natural Language Processing, Povo, Trento, Italy*, 1992.
30. E. Brill and P. Resnik. A rule based approach to prepositional phrase attachment disambiguation. In *Proc. of COLING-94, Kyoto, Japan*, 1994.
31. L. Brown. Class-based language models. *Computational Lingustics*, 1994.
32. P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Lafferty J.and Mercer, and P. Rossin. A statistical approach to machine translation. *Computational Linguistics*, 16(2), 1990.
33. P. Brown, S. Della Pietra, V. Della Pietra, J. Lai, and R. Mercer. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1), 1992.
34. P. Buitelar. A lexicon for underspecified semantic tagging. In *Proc. of the Applied Natural Language Processing 97 Workshop on "Tagging text with lexical semantics"*, 1997.
35. C. Cardie. *Embedded Machine Learning Systems for Natural Language Processing: A General Framework.* Lecture Notes in Artificial Intelligence, Springer, 1996. Originally presented at the Workshop on New Approaches to Learning for Natural Language Processing, 14th International Joint Conference on Artificial Intelligence (IJCAI-95), 119-126, 1995. AAAI Press.
36. J.P. Chanod and P. Tapanainen. A robust finite-state parser for french. In J. Carrol, editor, *Proceedings of the ESSLI-96 Workshop on Robust Parsing, Prague*, 1996.
37. K. A. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of Second Conference on Applied Natural Language Processing*, 1988.
38. M. Collins and J. Brooks. Prepositional phrase attachment trough a backed-off model. In *3rd. ACL Workshop on Very Large Corpora*, 1995.
39. J. Cowie, L. Guthrie, and J. Guthrie. Lexical disambiguation using simulated annealing. In *Proc. of COLING-92, Nantes, France*, 1992.
40. J. Cowie and W. Lehnert. Information extraction. *Special NLP Issue of the Communications of the ACM*, 1996.
41. A. Cucchiarelli and P. Velardi. Automatic selection of class labels from a thesaurus for an effective semantic tagging of corpora. In *Proc. of Fifth Applied Natural Language Processing, Washington DC*, 1997.
42. H. Cunningham, K. Humphreys, R. Gaizauskas, and J. Wilks. Software infrastructure for natural language processing. In *Proc. of Fifth Applied Natural Language Processing, Washington DC*, 1997.
43. I. Dagan, M. Marcus, and S. Markovitch. Contextual word similarity and estimation from sparse data. In *Proc. of ACL-93, Ohio, USA*, 1993.
44. R. Del Monte. *Linguistic and Inferential Processes in text analysis by computer.* UniPress, 1992.
45. M. Even. *The relational models of the lexicon.* Cambridge University Press, 1989.

46. A. Franz. A statistical approach to learning prepositional phrase attachment disambiguation. In *Proc. of IJCAI-95 WorkShop on "New Approaches to Learning for NLP"*, 1995.
47. R. Gaizauskas, T. Wakao, K. Humphreys, H. Cunningham, and Y. Wilks. University of sheffield: Description of the LaSIE System as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, Columbia, Maryland, USA, November 1995.
48. R. Grishman. Where is the syntax? In *MUC-6*, 1996.
49. R. Grishman and J. Sterling. Generalizing automatically generated selectional patterns. In *Proc. of COLING-94, Kyoto, Japan*, 1994.
50. L. Guthrie, J. Pustejovsky, Y. Wilks, and B Slator. The role of lexicons in natural language processing. *Communications of ACM*, 39(1), 1996.
51. D. Hindle. Acquiring disambiguation rules from text. In *Proc. of ACL-89*, 1989.
52. D. Hindle and M. Rooths. Structural ambiguity and lexical relation. *Computational Linguistics*, 19(1), 1993.
53. J. Hobbs. The generic information extraction system. In *Fourth Message Understanding Conference, McLean, Virginia*, 1992.
54. P. Jacobs. Text-based intelligent systems. In *Working Notes of the AAAI Symposium, Stanford University*, 1990.
55. K. Jensen, G.H. Heidorn, and S.D. Richardson, editors. *Natural Language processing: the PLNLP approach.* Kluwer Academic, Boston/Dordrecht/London, 1993.
56. M. Johnston, B.K. Boguraev, and J. Pustejovsky. The acquisition and interpretation iof complex nominals. In *Working Notes of AAAI Workshop on "Representation and Acquisition of Lexical Knowledge : Polysemy, Ambiguity and Generativity, Stanford University, CA*, 1995.
57. A. Kilgariff. Foreground and background lexicons and word sense disambiguation for information extraction. In *Proc. of International Workshop on Lexically-Driven Information Extraction*, 1997.
58. A. Luk, B. Vauthey, and O. Ansaldi. Acquisition of domain specific salient semantic features for information extraction. In *Proc. of International Workshop on Lexically-Driven Information Extraction*, 1997.
59. M. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2), 1993.
60. S. McRoy. Using multiple knowledge sources for word sense disambiguation. *Computational Linguistics*, 18(1), 1992.
61. S. Montemagni and L. Vanderwelde. Structural patterns vs. string patterns for extracting semantic information from dictionaries. In *Proc. of COLING-92, Nantes, France*, 1992.
62. J. Neumann, J. Backofen, R.and Baur, M. Becker, and C. Braun. An information extraction core system for real-world german text processing. In *Proc. of Fifth Conference on Applied Natural Language Processing, Washington DC, USA*, 1997.
63. H.T. Ng and H.B. Lee. Integrating multiple knowledge sources to disambiguate word senses : an exemplar-based approach. In *Proc. of ACL-96*, 1996.
64. F. Pereira, N. Tishby, and L. Lee. Distributional clustering of english words. In *Proc. of ACL-93*, 1993.
65. C. Pollard and I. Sag. *Information Based Syntax and Semantics, Vol 1 : Fundamentals.* CSLI n. 13, distrbuted by Chicago University Press, 1987.
66. J. Pustejovsky. The generative lexicon. *Computational Linguistics*, 17, 1991.
67. J. Pustejovsky. *Type Cohercion and Lexical Selection.* 1993.
68. J. Pustejovsky. *The Generative Lexicon.* MIT Press, 1995.

69. J. Pustejovsky, S. Bergler, and P. Anick. Lexical semantic techniques for corpus analsysis. *Computational Linguistics*, 19(2), 1993.
70. P. S. Resnik. *Selection and Information : A Class-Based Approach to Lexical Relationships*. PhD thesis, Pennsylvania University, 1994.
71. E. Riloff and J. Shepherd. A corpus-based approach for building semantic lexicons. In *Proc. of EMNLP-97, Rodhe Island, USA*, 1997.
72. G. Ritchie. *The Lexicon*. 1987.
73. G. Salton, A. Wong, and C.S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11), 1975.
74. F. Smadja. *Macrocoding the Lexicon with co-occurrence knowldege*. Zernik, U.,ed. "Lexical Acquisition" Lawrence Erlbaum, 1991.
75. J. Sowa. *Conceptual Structures : Information Processing in Mind and Machine*. Addison-Wesley, 1984.
76. J. Sowa. *Lexical Structures and Conceptual Structures*, chapter 12. J. Pustejovsky Ed., "The Semantics and the Lexicon", Kluwer, 1993.
77. Y. Wilks, D. Fass, C.M. Guo, J. McDonald, T. Blate, and B. Slator. *Providing Machine Tractable Dictionary Tools*, chapter 16. in Pustejovsky ed., "The Semantics and the Lexicon", Kluwer, 1993.
78. Y. A. Wilks and M. Stevenson. The grammar of sense : Is word-sense tagging much more than part-of-speech tagging ? In *Technical Report CS-96-05, University of Sheffield*, 1996.
79. D. Yarowsky. Word sense disambiguation using statistical models of roget's categories trained on large corpora. In *Proc. of COLING-92, Nantes, France*, 1992.

Technical Terminology for Domain Specification and Content Characterisation

Branimir Boguraev
Apple Research Laboratories
bkb@research.apple.com

Christopher Kennedy
Department of Linguistics, UCSC
kennedy@ling.ucsc.edu

Abstract. The identification and extraction of technical terms is one of the better understood and most robust natural language processing (NLP) technologies within the current state of the art of language engineering. What is particularly interesting here is the clear understanding how to derive, from their linguistic properties, computational procedures for reliable identification and extraction of terms from technical, scientific, prose. In generic information management contexts, terms have been associated both with procedures seeking to identify a term set which uniquely distinguishes a document within a nearly homogenous document collection, and with procedures seeking to extract a representative terms sample which uniquely characterises a document's content. There is a wide range of uses for terminology, commonly identified with e.g. text indexing, computational lexicology, and machine-assisted translation; most of these employ the notion of terminology being representative of a given domain. This paper discusses some specific extensions of the terminology identification technology to make it fully capable of domain specification; it also presents extensions of the technology beyond domain specification, to the purpose of document characterisation. These extensions make terminology identification the foundation of an operational environment for document processing and content characterisation and abstraction; more generally, it becomes an immensely empowering technology in the age of growing information overload.

1 Introduction

1.1 Technical terminology: a background

Within the current state of the art of language engineering, there are few natural language processing (NLP) technologies that are readily applicable to open-ended range of texts and domains. Among those that do have this property, the identification and extraction of technical terms is, arguably, one of the better understood and most robust. There is a rich literature on the linguistic properties of technical terms and how these properties map onto computational procedures for their identification and extraction. Such a mapping is particularly important, as it is understanding how the discourse properties of technical terminology manifest themselves in technical prose that allows the definition of computational procedures for terminology identification and extraction, which are applicable across a wide range of document types, and maintain their quality regardless.

One of the best defined procedures for the identification and extraction of technical terms is the TERMS algorithm developed by Justeson and Katz ([14]), which focuses on multi-word noun phrases occurring in continuous texts. A study of the linguistic properties of these constituents—preferred phrase structures, behaviour towards lexicalisation, contraction patterns, and certain discourse properties—leads to the formulation of a robust and domain-independent algorithm for term identification. The TERMS algorithm accomplishes high levels of coverage (both in comparison to data from terminology dictionaries and as evaluated by the authors of technical prose processed by the algorithm), and it can be implemented within a range of underlying NLP technologies including 'bare' morphologically enhanced lexical look-up [14], part-of-speech tagging [8], or syntactic parsing [20]). It has strong cross-linguistic application (see, for instance, [6]). Most importantly, the algorithm is particularly useful for generating a 'first cut' towards a broad characterisation of the content of the document.

1.2 Uses of technical terminology

Despite the strength of its potential as a tool for NLP applications, terminology identification tends to be viewed as a technology that is primarily suited for a task outside the domain of NLP, namely information retrieval (IR). The typical IR task views a document collection as a homogenous corpus, and seeks to identify those items (words, base forms, stems, or occasionally phrases) that have distributional properties that *distinguish* a document from other documents in the collection. For example, in a set of articles exploring the common theme of programming languages, expressions like *"compiler"*, *"program"*, and *"assembly code"* are likely to be commonly mentioned in all articles, and hence not characteristic of any single one of them. Terminology identification makes it possible to expand the granularity of the indexing task from tokens to larger phrasal units, permitting more refined distinctions among documents in a given domain, as well as more explicit listings of the words and phrases that uniquely identify a particular document (see, e.g. [21], [22]).

In contrast, in the context of NLP, where the goal is not simply document identification, but rather partial understanding, the more general problem is to find those terms which are *characteristic* of a particular document. Quite often, these terms are not the same as those which uniquely distinguish a document from others in a similar domain. For example, in the context of the earlier example of articles on programming languages, a back-of-the-book indexing task would crucially require the extraction of all the terms in a document, including those that might be ignored for the purpose of IR indexing (e.g. *"compiler"*, *"program"*, *"assembly code"*, and so forth) because they might not be sufficiently discriminating.

In general terms, terminology identification in NLP contexts has been applied to tasks that involve 'distilling' a characterisation of specialised domains, such as back-of-the-book indexing. More interesting, and more challenging, are situations where the technology can be leveraged to specify in finer detail such specialised domains. For example, while back-of-the-book indexing requires only

the identification of terms (noun phrases of certain type), a domain specification task, requires in addition the identification of certain generalisations about the relations among, and properties of, the objects (terms) in the domain. Even more interesting are applications where the identification of term-like objects is strongly coupled with special purpose procedures for focused discourse processing, which seek to select, and highlight, those terms which are highly topical, and thus capable of providing broad characterisation of the document's content. The goal of this paper is to show that the identifiable, and reliably computable, discourse properties of terminology-bearing phrases can be used as the primary information base for the dual tasks of domain specification and content characterisation. Before we move to a discussion of the extensions to the traditional notion of a 'term' that are required to augment the basic terminology identification technology for these higher-level tasks, we first address some of the more general issues that must be taken into account when technical terminology is used as the base for higher level document analysis.

1.3 Content characteristics of technical terminology

As was observed in the previous section, conventional uses of technical terminology are most commonly identified with text indexing, both back-of-the-book variety and IR-style ([11], [14]). More recently, applications for technical terminology identification have been found in *computational lexicology*, e.g. for compilation of terminology for technical dictionaries and glossaries, and *machine-assisted translation*. However, it is still uncommon to use technical terms for generating a *representation of the document(s) domain*, as well as for the *topical content of a document*. This appears surprising: [14, appendix] gives examples of term sets derived by the TERMS algorithm for several articles:

> **statistical pattern classification:** *"stochastic neural net", "joint distribution", "feature vector", "covariance matrix", "training algorithm"*, and so forth.
>
> **lexical semantics:** *"word sense", "lexical knowledge", "lexical ambiguity resolution", "word meaning", "semantic interpretation", "syntactic realization"*, and so forth.
>
> **liquid chromatography:** *"mobile phase", "surface area", "packing material", "preparative chromatography", "displacement effect"*, and so forth.

There is a very clear case to be made for the suitability of such term sets for the kinds of tasks which fall within the broad categories for *domain specification* and *document characterisation*. The problem, of course, is finding, within such term sets, the right kind of detail with which to enhance the semantic space defined by a set of term phrases, as well as establishing the right kind of granularity of detail, so that over-generation, on the one hand, and under-representation, on the other, are not detrimental to the task. While intuitively appealing, term sets like these are lacking in several important respects.

First, while admittedly topical within a particular source document, other documents within the same domain are likely to yield similar, overlapping, sets of terms (this is precisely the reason why technical term sets are not necessarily readily usable for document retrieval). This means that if the content of any single document is to be characterised in terms of a representative sets of phrasal units, then these units should include the technical terms identified for the document, but must also include additional material which distinguishes that particular document from others in the domain. In terms of the example term set sketched in Section 1, if all articles mention "compilers", some distinction among them can be brough forward if it becomes clear that in some of them, a compiler is being "optimised", while in others it may be "designed", "modularised", or "built".

Second, for phrasally-based representation of a document, there is a substantial difference between characterising *the content* of a document and characterising *the technical domain* within which a document subject matter belongs. Domain characterisation is, arguably, the more straightforward of the two tasks, as the mapping from a term set to domain label is fairly direct. A term set for a domain can be incrementally developed, by cumulatively growing the set after analysing more than one technical documents relevant to the domain. Alternatively, given an appropriate source document—for instance one that is guaranteed to describe the domain fully, e.g. for instructional purposes (such as a user guide, or a reference manual)—its term set can be assumed to be a near-complete domain characterisation, unlikely to expand further by analysing additional sources.

In contrast, the mapping from a term set to events described in a document requires a strong sense of topicality and relevance. A central aspect of the document characterisation task is that of data reduction: what is the representation of the document content which is smaller than the original document, and yet highly indicative of its central points. Completeness here is defined not in terms of a domain; indeed, there is no domain to specify at all. Instead, the problem is that of compactness and coherence. Term-like entities are clearly a part of such a representation, and focusing on certain aspects of them—in effect by exploiting their lexical semantics which are brought in focus by the relational contexts in which they are found—is essential for supplying the right level of detail. However, what is also essential is to be able to identify just those relational contexts which carry the most, and most central, of the document content. To complete the example we began sketching earlier, the domain specification problem would require the identification of the relational contexts of e.g. "design", "modularisation", and "optimisation", as characteristic of the "compilers" domain. However, in a given document about compilers, it might be issues of "implementing an optimising compiler", further modulated by constraining the source languages to the "functional programming languages" family, that would be representative of this particular document.

1.4 Overview of the paper

This distinction between domain specification and content characterisation raises at least two questions for us. With respect to domain specification, and assuming strictly technical domains, the issue is to what extent can the term set identified by methods similar to the one developed by Justeson and Katz be leveraged for the purposes of *complete* domain description. With respect to content characterisation, we are concerned with a suitable generalisation—and relaxation—of the notion of a term, which still involves identification and extraction of particular phrasal units by essentially the same procedure, but generates a set of objects which are more definitively charactersistic of the core content of a document, both individually, and as a member of a larger document set. This concern leads to a third, more general question: to what extent does the generalization of the notion of technical term which is required to solve the problem of content characterisation provide a means of stretching the application of the technology beyond the purely technical prose? These questions are addressed in the remainder of this paper.

2 Domain specification

The primary problem with using term sets as domain specifications is that they are incomplete: at the very least, a description of a domain based on objects encountered in it ought to provide relational information concerning these objects. For instance, in the domain of lexical semantics mentioned earlier (Section 1.3), we would like to know that the notions of *word sense* and *lexical ambiguity* are related, or that a *semantic interpretation* can be *derived* or *constructed*. As it turns out, acquiring such relational information is not dissimilar to the process of lexical acquisition, where the acquisition is done dynamically, on the basis of linguistic analysis of text sources (see Boguraev and Pustejovsky, [5]). Indeed, it is just a matter of emphasis whether, given a text source, essentially the same operational mechanisms are applied for the purposes of lexicon acquisition alone, or for the purposes of analysing that source. Johnston *et al.* [13] discuss, at some length, the similarities between the acquisition of relational information for domain objects (denoted by technical terms) and the acquisition of functional information for a domain by means of fully mapping out the linguistic analysis of a closed corpus describing this domain.

By way of illustrating this line of argument, we outline below an approach to domain specification, based on analysing a source which is assumed to provide a suitably complete description of the domain in question; in this case, the source is technical documentation found in the form of a reference guide.

2.1 On-line assistance and domain functionality

Apple Guide is an integral component of the Macintosh operating system; it is a general framework for on-line delivery of context-sensitive, task-specific assistance across the entire range of software applications running under the Mac

OS^{TM}. The underlying metaphor is that of answering user questions like *"What is X?"*, *"How do I do Y?"*, *"Why doesn't Z work?"* or *"If I want to know more about W, what else should I learn?"*. For each application, assuming the existence of a database in a certain format, Apple Guide coaches the user through a sequence of definitional panels (describing key domain concepts), action steps (unfolding the correct sequence of actions required to perform a task or achieve a goal), or cross-reference information (revealing additional relevant data concerning the original user query).

An Apple Guide database is typically instantiated by hand, on a per application basis, by trained instructional designers. Viewed abstractly, however, the information in such a database constitutes complete domain specification for the application—in terms of domain objects, their properties, and relations among them. In order to answer questions like *"What is a startup disk?"*, *"How do I prepare a disk for use"*, or *"Why can't I eject the floppy disk?"*, certain aspects of the domain (in this case the general domain of operating system level activities) need to be identified. For example, a 'disk' is a a kind of a domain object; there are several types of disk, including 'floppy disks', 'startup disks', 'internal hard disks', etc.; disks need to be 'prepared for use'; floppy disks can be 'ejected'; and so forth.

It is clear that a terminology identification component, applied to suitably chosen technical documentation for the domain, could be profitably utilised for the purposes of domain specification. What is less clear is that with some degree of additional linguistic analysis which incorporates knowledge of processes like derivational morphology, nominalisation, grammatical function, predicate-argument structure, and compounding, the core set of technical terms can be refined not only to include all (and only) domain objects, but also to enrich the descriptions of these domain objects by deriving relational structures for each of them. In effect, the process is not unlike that of generating lexical entries for a dynamically induced lexicon. There is a strong assumption here, of completeness of the technical documentation source.

This is necessary, for the purpose of concept learning in 'fixed' domains, where it is conventional to assume some version of the "closed world hypothesis". In other words, the domain description (e.g. a reference manual, as in this case) is taken to specify everything that is essential and characteristic. From such an assumption, following a process of lexical context acquisition, feature induction can be limited to literal contexts within which a word, or a term phrase, occurs. This is, of course, the standard assumption in the field of corpus-based lexical acquisition. It turns out, however, that when dealing with *closed* corpora, the special style of writing—aimed at a particular knowledge level of the user, with a very specific depth of description of the information included, and explicit with respect to all relevant details concerning a complete term set—offers an especially rich source for lexical induction and, ultimately, domain specification and acquisition (Johnston *et al.* [13]).

Boguraev ([3]) discusses the details of such an analysis process, comprising a sequence of operations which includes identification of lexical entities, asso-

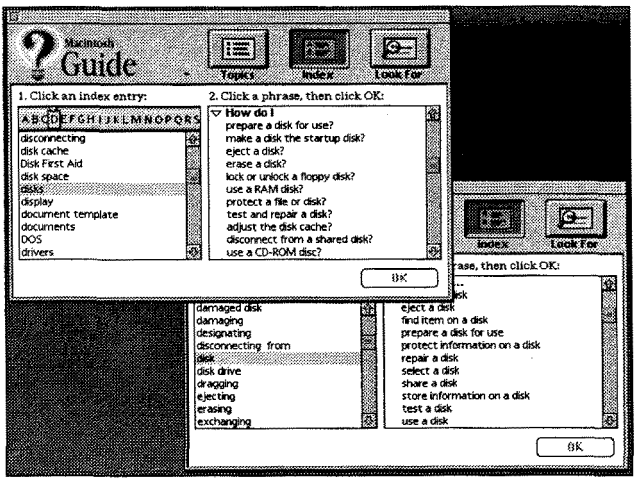
ciation of basic lexical features with them, extraction of selectional restrictions and argument typing information from local context, gathering of data from repeating, similar, contexts across the entire document, and so forth. Due in part to the special nature of the particular genre, and in part to the closed nature of the text corpus, this process can be carried out to an extremely high degree of precision and recall, even though it employs relatively shallow parsing methods. What is of special interest here is the observation that by carrying out progressively refined process of normalisation and data reduction over the source technical text, staged lexical acquisition ultimately derives a conceptual map of the technical domain by mining for core domain objects, relations they participate in, and properties they possess. Mining for objects is conceptually identical to extracting domain-specific vocabulary of technical terms. Mining for relations is identical to instantiating lexical semantic information for such terms. The domain model emerges from incremental refinement of the initial set of core technical terms in the document.

```
Term: startup disk                Term: icon
  Relations:                        Relations:
     conserve space on []              assign color to []
     correct a problem with []         ........
     look for []                    Objects:
     recognize []                      Apple printer
     specify []                        network connection
     use internal hard disk as []      startup disk
  Objects:                             Trash
     System Folder                     ........
```

Seeded from this set, and following further targeted mining for relations between, properties of, and associations among these terms, the domain model takes the form of a *domain catalog*, illustrated above. A complete catalog incorporates several different types of link between terms: `"objects"`, as exemplified here, `"properties"`, `"definition"`, `"used-for"`, and others. For the sake of simplicity, the discussion below focuses on terms and relations only. Mapping from such a domain description to an Apple Guide database is relatively straightforward. A term identified as a salient domain object clearly ought to be in the database—in its simplest form, by providing a definition for it. The same applies to a relation, which naturally maps onto a *"How do I ...?"* panel. For the example fragment above, this would mean definition entries for *"startup disk"*, *"network connection"*, *"System Folder"*; and action sequence panels for *"How do I specify a startup disk?"*, *"How do I use internal hard disk as a startup disk?"*, and so forth. The definitions and task sequences would still have to be supplied externally, but the generation of the database is fully automatic. In fact, the process of technical documentation analysis maintains a complete 'audit trail', relating items in the domain catalog to the relevant fragments in the text source where information concerning them has been found; a prototype implementation augments the database with pointers into the on-line version of the manual.

The operation of such a domain acquisition framework is illustrated below. One of the screen snapshots is from the Macintosh Guide shipping with the

standard Mac OS configuration; the Guide database here has been constructed, manually, by a team of professional instructional designers. The other snapshot displays, through the same delivery mechanism, a database which has been constructed, fully automatically, by the system described above, following an analysis of the primary technical documentation for Mac OS, *Macintosh User's Guide* ([1]). Note, in particular, the *"How do I..."* lists, with entry points to detailed instructions concerning common tasks with specific objects (in this example, disks) in the Mac OS domain. Barring non-essential differences, there is a strong overlap between the two lists: *"prepare a disk for use"*, *"eject a disk"*, *"test (and repair) a disk"*, *"protect a file/information on disk"*, and so forth. (Actions associated with specific types of disk, e.g. *"startup disk"*, *"floppy disk"*, and so forth, appear elsewhere in the automatically generated database; see below.) Moreover, some additional action types have been identified, which are clearly relevant to this domain, but missing from the 'canonical' database: *"share a disk"*, *"find items on a disk"*.



2.2 Domain acquisition and domain structuring

In the previous section, we already brought up one factor which makes terminology identification a powerful enabler for the tasks of domain acquisition and specification: assuming a version of the "closed world hypothesis" allows the induction of structure of fixed domains to be anchored in the process of identifying the core set of objects in the domain, growing from that a set of associated, and relevant, relations and properties, and seeking closure over these sets. What makes the closed world hypothesis convenient for this purpose is that it allows, for every assertion which is not explicitly given in the knowledge base, to assume that its negation is true; this guarantees completeness of the model. For such closed worlds, closure on literals that are not axioms in the domain is a valid method for completing the theory. In general, feature induction without assuming a closed world model is intractable. Fortunately, the particular environment

in which we seek structure to be imposed on the term sets—e.g. software applications in finite domains—falls under that definition: everything of import is described in the manual, and there is nothing essential that is not to be found somewhere in the manual.[1]

The semantic domain acquisition outlined above lends itself naturally to induction of *local ontologies*, which organise related objects in the domain, and *semantic frames*, which define some internal structure to these objects. The standard assumption and procedure in corpus-based lexicon acquisition typically restricts the types of induction techniques that are applicable to the corpus only to the literal contexts within which a word occurs, namely, the positive instances. As we have seen, the documents from which the domain characterising term sets are extracted are examples of *closed* corpora. Because of the intentionally closed nature of these corpora, learning is not restricted to positive data: the absence of a particular pattern with a specific lexical item can be interpreted as a negative instance for that word.

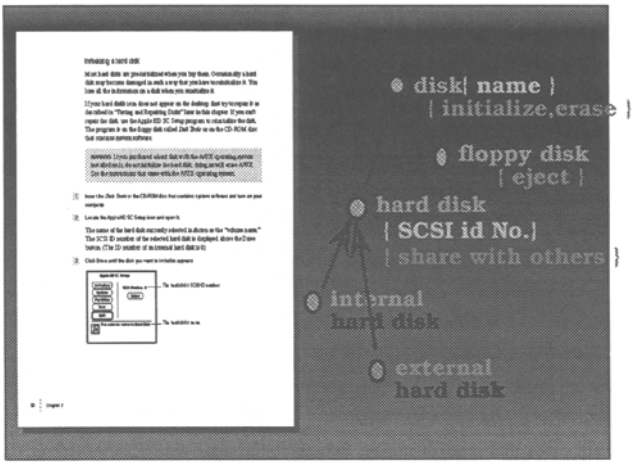
In order to organise the information for the Apple Guide database, the domain catalog is analysed. The analysis seeks to identify structural aspects of, and among, domain objects. In particular, it focuses on determining:

- the internal structure of a domain object,
- the semantic operations through which each object is composed,
- the local ontology for a group of related objects,
- the meaning of a domain object, and
- the cross-classificational structure licensed by relations among objects ([13]).

The remainder of this section outlines the finer grained domain description thus induced.

A number of different semantic operations may be involved when constituents of a domain object compose. For example, in a form like *"hard disk"* the component description *"hard"* acts as a sub-typing modifier, and specifies one of the *formal properties* of the disk. In a form like *"startup disk"*, a component in the same position specifies the *purpose* to which the disk is put. The inferences required for making such distinctions work from closely inspecting the local contexts in which the object occurs; and they crucially depend on the validity of the closed corpus assumption posited above. Local ontologies are hierarchies which capture subdomains of related objects within the content of a particular corpus. They group together domain objects of closely related types. For example, in the *Macintosh Reference* domain there is a local ontology of disks; the most basic type in this ontology is *"disk"*. The subtypes of *"disk"* are *"floppy disk"*, *"RAM disk"*, and *"hard disk"*. The type *"hard disk"* is further subtyped into *"internal hard disk"* and *"external hard disk"* (see the schematic illustration below).

[1] At least, this is how a good manual should be written...



The identification of local ontologies helps determine inheritance relations between lexical entries. Structuring constraints are derived by means of an algebra over relation sets, themselves being loosely interpreted as meanings of the newly acquired domain objects. This is possible precisely because of the validity of the closed world hypothesis: in the domain of personal computers operating systems, a floppy disk is an object which can be initialised, ejected, protected, and so forth. For instance, for the two objects near the top of the lattice in the figure above, *"floppy disk"* and *"hard disk"*, the relation sets derived from the document are shown below.

floppy disk: `name [], erase [], access [], initialize [], use [], test [], save file on [], repair [], eject [], insert [] into disk drive, lock [], unlock [], protect []`

hard disk: `name [], save file on [], repair [], erase [], access [], initialize [], use [], test [], attach [] to printer, reinstall system software on [], copy program disk to []`

Closer analysis of the relation sets leads to a strikingly accurate characterisation of the domain. Hard disks and floppy disks clearly share certain properties: they can be named, initialised, accessed, and tested. On the other hand, floppy disks are things which can be inserted into a disk drive, ejected, locked, unlocked, and protected; no such actions can be applied to hard disks. Furthermore, hard disks can be attached to printers, have system software reinstalled on them, and have program disks copied to them; floppy disks are not. It is this kind of analysis which suggests that shared properties should be established as high in the ontolgy as possible: in this case, we have reliably learned some general information that applies to the whole class of disks, as well as more specific information, applicable to specific sub-classes.

An additional organisational principle over the domain objects set emerges from the notion that semantically related object terms are found in similar contexts, as well as that a family of objects closely related by function in the domain will be classified in a tight category. Thus, while methods for elaborating the local ontology for *"disk"* identify, among others, *"RAM disk"* as a sub-type, clustering techniques similar to those described in [23], but applied to phrasally tagged syntactic contexts, place *"RAM disk"* in a category together with *"battery power"*, *"powerbook computer"*, *"temporary storage device"*, and *"minimal system folder"*. This is precisely the type of information which, while hard to formalise, is very useful for building up the cross-reference links in the on-line assistance framework of Apple Guide (the *see also* links).

2.3 Domain specification: an evaluation

The question of appropriate evaluation metrics and methodology for applied language engineering systems is always a hard one, as often it is not entirely clear just what the right criteria would be. However, in the case of domain specification described here, an evaluation procedure is relatively easy to define, as there exists a convenient—and appropriate—reference point: comparison of automatically derived Apple Guide databases should be carried out against their manually crafted counterparts, designed from the ground up by experienced instructional designers.

In particular, a comparison between the database derived from the *Macintosh Reference* manual (henceforth MACREF), and the manually compiled *Macintosh Guide* database for Mac OS, as it ships with every computer, (henceforth CHIPS, for historical reasons) shows that the domain specification system described here is very accurate. The comparative analysis focuses on precision and recall in coverage; [2] discusses at some length how the sets of terms and relations for the two databases are adjusted to take into account both similarities and differences in the data.

There is a significant overlap in elements common to both databases (*"active window"*, *"virtual memory"*, *"access privileges"*, and many others). Then there are elements which are unique to either database: for instance, the manually crafted one has the notion of *"bus"* (*"Apple Desktop Bus"*, *"NuBus"*) and *"LocalTalk"*, all missing from the automatically derived one; while no entries for *"alarm clock"*, *"time format"* and *"time zone"*, identified automatically, occur in the manual one.

Of greatest interest to a comparative evaluation are the conceptual mismatches between the two databases, which tend to manifest themselves in different ways. For instance, the absence of *"bus"* from the acquired domain is 'legitimate'—in the sense that there is no mention of *"bus"* (nor *"NuBus"*, nor *"ADB"*) in the *Macintosh Reference* manual. There are, however, various divergencies, above and beyond simple lexical mismatch: for instance, the lack of automatic conceptual grouping in the domain specification system fails to assign a category label like *"network services"* to the set *"network service"*, *"network*

access", *"network administrator"*, *"network cable"*, *"network connection"*, *"network control panel"*, *"Network Identity section"*, *"network software"*, *"network user"*, *"networking expansion card"*, *"networking software"*, all of which it identifies as individual domain objects.

The methodology of normalising for such factors is presented in detail in [2]. Ultimately, recall and precision rates are calculated for the automatic database construction system. *Recall* is defined with respect to a 'reference' database, as a normalised count of how many items defined by the instructional designers have also been identified automatically as characteristic of the domain. *Precision* is defined as the ratio of 'good' hits, relative to all items identified by the automatic extraction methods. The table below, which gives a quantitative account of the performance of the domain specification system, demonstrates the practical viability of the domain acquisition techniques described here. By extension, these figures also quantify the recall and precision rates for the specific task of technical terminology identification, as applied to a very real problem.

	terms	relations
recall	94.0%	91.1%
precision	89.5%	88.5%

These are encouraging results, not only for the viability of inducing domain specifications from technical documentation, but also for the utility of terminology identification technology for tasks much stronger than the relatively bland, and unfocused, indexing in general.

3 Content characterisation

The processing environment described above clearly demonstrates that the task of domain acquisition can be effectively tackled by basic terminology identification technology, augmented with the richer toolset of methods and techniques for text-based lexicon acquisition. This section discusses the issues which arise when this technology is applied to the problem of *content characterisation*: generating a suitable representation of the "aboutness" of a document.

This task is of particular importance to applications where the goal is *partial* understanding of arbitrary texts (e.g. press releases, news articles, web pages, mail archives, and so forth); partiality being a necessary constraint in environments where no simplifying assumptions can be made concerning domain dependence and task specificity. Without such assumptions—and in particular, without the closed nature of the technical domains and documentation—it is not clear what use can be made of term sets derived from arbitrary documents.

Certainly, we cannot even talk of "technical terms" in the narrower sense used so far. On the other hand, starting from the position that terms are phrasal units of a certain (albeit very well defined) type, and seeking to extend their ability to characterise the content of technical prose, we may well ask the question whether

the application of similar phrase identification technology generates phrase sets which can be construed as broadly characteristic of the topical content of a document: in the same way in which a term set can be viewed as characterising the domain to which technical prose belongs. In other words, the question concerns the wider applicability of linguistic processing targeted at term identification, relation extraction, and object cross-classification: can a set of phrases derived in this way provide a representational base which enables rapid, compact, and accurate appreciation of the information contained in an *arbitrarily* chosen document?

3.1 Phrasal identification for content characterisation

The problem facing the application of terminology identification technology to arbitrary domains is one of a loss of robustness: outside of the strict confines of technical genres, the reliability of the linguistic analysis which underlies the high accuracy and precision of the TERMS algorithm of Justeson and Katz deteriorates. Several specific problems arise from scaling up the domain specification technology to an open-ended set of document types and genres.

First, since the domains are not closed, there is no strong sense in which notions of sublanguage can be leveraged. The 'clean', typically regular, language of technical documentation dissolves into a variety of writing styles, with wide variation in formality and grammaticality. Second, instead of working with relatively small number of substantially sized documents, the data sets are likely to be made up of a large number of small documents. As a result, the assumptions about how the entities discussed in a document lexicalise do not hold in the way which enables the identification of core technical terms. Finally, a system which extracts phrases on the basis of canonical terminology structure suffers from gross over-generation when applied to a document without regard to domain or genre, with the concomitant result that the number of identified phrases will certainly be larger than a user can absorb without too much cognitive overhead. The specific problems with "vanilla" term sets, when considered as the basis for a content characterisation task, are outlined below.

Undergeneration For a set of phrases to be truly representative of document content, it must provide an *exhaustive* description of the entities discussed in the document. That is, it ought to contain not just those expressions which satisfy the strict phrasal definition of "technical term", but rather *every* expression which mentions a participant in the events described in the text. Phrasal analysis must therefore be extended to include pronouns and reduced descriptions, in addition to the more complex nominals which correspond to true technical terms.

Overgeneration Relaxation of the canonical phrasal definition of technical term leads to information overload. When applied to a document without regard to domain or genre, a system which extracts phrases on the basis of relaxed canonical terminology constraints will typically generate a term set far larger

than a user can absorb without cognitive overhead. At the same time, the term set may contain several distinct phrasal units which refer to the same discourse object. Without some means of resolving anaphoric relations, these crucial connections will be lost.

Differentiation Finally, while a list of terms may be topical for the particular source document in which they occur, other documents within the same domain are likely to yield similar, overlapping sets of terms. Unacceptably, this might result in two documents containing the same or similar terms being classified as "about the same thing", when in fact they might focus on completely different subtopics within the general domain they share.

The consequence, in essence, is that a phrasal grammar alone is a weak technology for the task of unconstrained content characterisation. This is not particularly surprising, and certainly accounts for the fact that terminology identification is not widely considered as an appropriate tool for content analysis. On the other hand, if we allow for the fact that in any set of term-like phrases some are going to be more representative than others, it becomes clear that as long as some mechanism exists for selecting precisely those entities which are most topical, then the phrasal approach becomes not considerably more viable.

It is further clear that such *topic stamps* are essentially phrasal units with certain discourse properties: in particular, they manifest a high degree of salience within contiguous discourse segments. This makes it possible to redefine the task of content characterisation as one of identifying phrasal units with lexico-syntactic properties similar to those of technical terms and with discourse properties which signify their status as "most prominent". It turns out that these discourse properties are computable as a function of the grammatical distribution of the phrase.

3.2 Phrasal selection for content characterisation

The problem of reducing the overhead of a large term set as the characterisation of a document's content can be thought of as the problem of identifying a function that is capable of ordering all terms with respect to some measure of their prominence in the discourse, and selecting from this set only those that are most prominent. This type of filtering would be instrumental in overcoming the lack of coherence in the term sets, which are by definition somewhat diffuse for arbitrary documents.

A problem, however, with arbitrary documents—and certainly with the kinds of documents we want to be able to handle (general news articles, arbitrary press releases, information digests, web pages, and so forth)—is that they are typically smaller than the technical prose for which the canonical terminology identification procedures have been developed. Compounded by effects of wide diversity of genre, this makes frequency-based measures largely unusable. In an effort to solve this problem, we have developed a system which uses a *salience*

feature, rather than a frequency measure, as the basis for determining which members of the large lists of content-bearing phrases identified by the technology constitute the most representative elements of the text.

Although salience does loosely correlate with frequency, our definition of "salience" is considerably more refined than what frequency counts reveal. Roughly speaking, salience is a measure of the relative prominence of an object in a discourse: objects with high salience are the focus of attention; those with low salience are at the periphery. By evaluating the members of a term set according to their salience, a partial ordering can be imposed which, in connection with an appropriate choice of threshold value, provides the basis for a reduction of the entire term set to only those terms which identify the most prominent participants in the discourse. This reduced set of terms, in combination with relational information of the sort discussed in the previous section and folded into an appropriate presentation metaphor, can then be viewed as a characterisation of a document's content. In essence, we substitute a weak selection procedure (frequency-based analysis) which does not 'scale down' well for smaller and arbitrary documents, with a more informed choice of representative phrases, which leverages semantic and discourse factors for reliable selection of salient objects. The fact that the semantic and discourse analysis operates directly from a linguistic basis, rather than indirectly from inferences over frequency counts and probability distributions, makes it applicable to individual documents, of small size, and diverse nature.

This strategy for elaborating the phrasal analysis which underlies standard term identification thus relies heavily on making use of notions like discourse structure and contextual factors—both of which are missing from the traditional technology for 'bare' terminology identification. The questions for this kind of approach to content characterisation can be summarized as follows. First, what formal definition of salience can be applied to a arbitrary sets of phrasal units in order to generate an ordering which accurately represents the relative prominence of the objects referred to in a document? Second, what linguistic information, available through scalable and robust identification technologies, can be levered to inform such a notion of salience? Third, how can general aspects of discourse structure—in particular, discourse segmentation and anaphoric relations—be used most effectively in calculating the topical prominence of expressions across an entire document?

These questions arise as we consider the overriding practical constraints imposed by the completely open-ended nature of the language, domain, style and genre of the texts we want to be able to handle. The next section discusses how we approach the problem in the context of specific examples.

3.3 Content characterisation and discourse analysis

Consider the following news article:[2]

[2] Adapted from an example of S. Nirenburg.

PRIEST IS CHARGED WITH POPE ATTACK

A Spanish Priest was charged here today with attempting to murder the Pope. *Juan Fernandez Krohn*, aged 32, was arrested after *a man armed with a bayonet* approached the Pope while he was saying prayers at Fatima on Wednesday night.

According to the police, *Fernandez* told the investigators today that *he* trained for the past six months for the assault. *He* was alleged to have claimed the Pope 'looked furious' on hearing *the priest's* criticism of his handling of the church's affairs. If found guilty, *the Spaniard* faces a prison sentence of 15–20 years.

Informally, the kind of content characterisation we would seek from this text is best illustrated by its title, *"Priest Is Charged with Pope Attack"*. As a title, it encapsulates the essence of what the story is about; as a condensation of the article content, it highlights the essential components of the discourse: there are two actors, identified by their most prominent characteristics; one of them has been attacked by the other; the perpetrator has been charged; there is an implication of malice to the act. As an instance of content characterisation—brief *and* informative—this works by bringing an adequate set of salient facts together: the priest and the Pope are the main actors of an *attempt to murder* event.

In the general case of document analysis, where no *a priori* domain knowledge is assumed, the task of automatically generating this kind of phrasal summary can be arbitrarily complex; it is certainly beyond the capabilities of present day natural language processing (both in terms of depth of the analysis, and sophistication of the generation). However, an accurate 'approximation' of the content can be delivered by combining the identification of the salient objects with extraction of suitable relational contexts directly from the text. The outcome of such a process is rather akin to a 'telegraphic' summary, as illustrated in the following extract from a novel, *Heart's Journey in Winter* ([7]), where a protagonist is summarising a diplomatic brief:

"... an intellectual and moral renewal," he said. Negotiations in Geneva. Ambassadors Polk and Kvertsovsky. Towards a Zero Solution. Strongest efforts. Reliable alliance partner. Fellow germans on the far side of the Wall and barbed wire.

Our approach to content characterisation attempts to formalise these intuitions, by developing a technology for mining a document for the most salient—and by hypothesis, the most representative–phrasal units, with the goal of establishing precisely the kind of core specification of content that is captured by these examples. At its core, the technology makes substantial use of a terminology identification component. This component, however, requires a number of extensions and enhancements in order to make it appropriate for the tasks discussed here. Three enhancements play a particularly important role: discourse segmentation, referent identification, and determination of salience.

As the example article above demonstrates, two well chosen phrases may be a sufficient indicator for the core content of two paragraphs. Clearly, a two page

document is unlikely to be equally well served by two phrases alone, no matter how salient thay might be. In in order to address the problem of variability in document size, we incorporate a process of *discourse segmentation*. Segmenting a long document into a number of text fragments— where each fragment is characterised by relatively high degree of internal coherence, and segment boundaries are defined as the points in the narrative where there are relatively abrupt shifts in topicality and, hence, flow of disourse—reduces the content characterisation task to that of finding topically coherent units spanning the whole document, and then identifying an appropriate number of *topic stamps* for each unit. We adopt a strategy of discourse segmentation originally proposed by Hearst [9] for the purposes of narrowing the response to user queries in an information retireval system; our implementation is specifically tuned to information derived both from structural analysis of the document and linguistic analysis of its text.

In order to construct the type of salience measure discussed in the previous section, which is intended to indicate the prominence of an object in the discourse, it is crucially important to determine the number and type of references to the object in the text. In the article discussed above, for example, there are altogether eight mentions of the priest; for a system to be able to establish this, however, it needs to go beyond phrasal identification, and engage in subsequent process of *referent identification*. There are numerous aspects to this problem, as exemplified by the identity of the referents for *a spanish priest*, *Juan Fernandez Krohn*, *a man armed with a bayonet*, and two occurrences of the pronoun *"he"* in the second paragraph: they all constitute different references to *the same individual*. We do not yet have the full range of resolution capabilities, but have developed a referent identification subsystem which makes heavy use of a highly effective anaphora resolution algorithm, augmented with identification of reduced nominal forms (Kennedy and Boguraev [16] present an anaphora resolution algorithm driven from a relatively shallow syntactic analysis base).

As discussed in some detail in [16], and following some earlier work by Lappin and Leass [18], resolving anaphors can be particularly effective if the resolution algorithm is informed by a measure of *local salience*. The determination of this parameter is based on taking into account a number of syntactic and contextual factors, thus making it highly indicative of prominence of the discourse object identified as the referent of any given term. As an example of how sych syntactic factors affect the process of content characterisation, consider the title of the article above. *Priest* is in the focus of the title (as opposed to, for instance, in *"Pope attacked by priest"*) not only because of overall frequency (eight references to the priest, and six to the Pope), but also because these references occur in prominent syntactic positions: five are subjects of main clauses, two are subjects of embedded clauses, and one is a possessor. *Pope*, who is of comparable topicality with five references in the article, only gets the position of a secondary object of the title because these references tend to occur in less prominent positions: two are direct objects.

Additional factors, beyond local syntax and context, augment the picture of global distribution of a given discourse referent. In some cases, entities in

the discourse are only brought into focus in a single segment, and then they drift in the background; alternatively, a discourse entity could be brought into focus repeatedly over a larger text span. Again, the intuition here is that a broader picture of prominence is related to overall frequency of reference to the same discourse object. A more precise rendering of this intuition can be made by formalising a notion of a "coreference class", which takes account of which particular text 'mentions' (cf. Mani and Macmillan [19]) of a term refer to the same entity, and mediates that by maintaining a representation of how these references are made, across a discourse segment and across the entire text. One of the computable properties of a coreference class is that of *discourse salience*, which, a function of local salience, also takes into account the distribution of the individual members of the coreference class across the text; Kennedy and Boguraev [17] discuss these issues in more detail.

3.4 Salience-based content characterisation

The interplay of terms and referents identification, determination of membership of coreference classes, calculation of local salience and tracking global salience reflects the distributional properties of a discourse object as the text story unfolds. Ultimately, the non-decreasing salience measure underlies a detailed representation of discourse structure which, when overlayed onto the results of discourse segmentation, gives a coherent representation of the topical prominence of particular referents in specific segments of text. Specifically, it becomes the basis for exactly the type of importance-based ranking of referents discussed in Section 3.2. Using this ranking, we define the topic stamps for a segment `S` to be the `n` highest ranked referents in `S` (where `n` is a scalable value).

A set of topic stamps derived by such methods is clearly representative of the core conceptual space in the document. Even more refined characterisation of the content can be derived by seeking to establish how the discourse referents are related to each other, as well as what additional information can be found which differentiates them in the particular document context(s) (see Karttunen [15] for a general discussion on discourse referents; and Heim's File Card metaphor, [10], for associating contextual information with discourse referents). Generally speaking, this additional elaboration offers a handle not just on the "who" aspect of the content, but also on the "how who does what to whom", further enriching the characterisation. In terms of the earlier example, the priest has been *charged*, while the Pope has been the object of an *attempt to murder* act. This procedure is, in fact, very similar to the process which enriches and refines a domain model by analysing the text of a technical manual for that domain (Section 2.2) and appropriately adorning and structuring the term set derived for that domain. The difference is largely in the relaxation of certain syntactic requirements (which map the semantic notions for term- and relation-hood to grammars for the identification of the extended syntactic contexts for the terms), imposed by the broader nature of the genres and types of documents we are looking at.

To summarise, content characterisation is essentially a five step procedure:

- ❑ discourse segmentation;
- ❑ identification of phrases referring to actors and objects in the story (in effect, "term" identification) and identification of the relational context in which they appear;
- ❑ establishment of referential identity beyond simple string matching; in particular, computation of coreference, including the resolution of pronominal and sortal anaphora; and
- ❑ calculation of salience for each referent in the text;
- ❑ reduction of the referent set to its most prominent members.

These processes *subsume* lower level components which further inform the document analysis; most notably these include syntactic parsing, identification and typing of proper names, determination of appropriate relational contexts. Discussion of these is outside of the central subject of this paper; [16] and [17] present some details concerning how such components have been engineered to circumvent the need for a comprehensive parsing engine. In general—and keeping in the spirit of core terminology identification strategies—our goal in this work has been to develop an inventory of natural language processing techniques, without assuming robust and reliable parsing components which typically seek to deliver in-depth, full, syntactic analysis of text.

3.5 Content characterisation: an example

By definition, the end result of the document analysis process is a compact text-based object, which encapsulates—via its pivotal points, the topic stamps—the core content of the source. In order to distinguish this from what current work in the field refers to as "document summary", we adopt the term *capsule overview*; more discussion concerning the positioning of this research into the space of document summarisation work, and details of the process components described in the previous sections, can be found in Boguraev and Kennedy [4].

We illustrate the procedure of deriving capsule overviews by highlighting certain aspects of the analysis of a recent *Forbes* article ([12]). The document is of medium-to-large size (approximately four pages in print), and focuses on the strategy of Gilbert Amelio (Apple Computer's CEO) concerning a new operating system for the Macintosh. Too long to quote here in full, the following passage from the beginning of the article contains the first, second and third segments, as identified by the discourse segmentation component (see Section 3.3); in the example below, segment boundaries are marked by extra vertical space.

"ONE DAY, everything Bill Gates has sold you up to now, whether it's Windows 95 or Windows 97, will become obsolete," declares Gilbert Amelio, the boss at Apple Computer. "Gates is vulnerable at that point. And we want to make sure we're ready to come forward with a superior answer."

Bill Gates vulnerable? Apple would swoop in and take Microsoft's customers? Ridiculous! Impossible! In the last fiscal year, Apple lost $816 million; Microsoft made $2.2 billion. Microsoft has a market value thirty times that of Apple.

Outlandish and grandiose as Amelio's idea sounds, it makes sense for Apple to think in such big, bold terms. Apple is in a position where standing pat almost certainly means slow death.

It's a bit like a patient with a probably terminal disease deciding to take a chance on an untested but promising new drug. A bold strategy is the least risky strategy. As things stand, customers and outside software developers alike are deserting the company. Apple needs something dramatic to persuade them to stay aboard. A radical redesign of the desktop computer might do the trick. If they think the redesign has merit, they may feel compelled to get on the bandwagon lest it leave them behind.

Lots of "ifs," but you can't accuse Amelio of lacking vision. Today's desktop machines, he says, are ill-equipped to handle the coming power of the Internet. Tomorrow's machines must accommodate rivers of data, multimedia and multitasking (juggling several tasks simultaneously).

We're past the point of upgrading, he says. Time to scrap your operating system and start over. The operating system is the software that controls how your computer's parts (memory, disk drives, screen) interact with applications like games and Web browsers. Once you've done that, buy new applications to go with the reengineered operating system.

Amelio, 53, brings a lot of credibility to this task. His resume includes both a rescue of National Semiconductor from near-bankruptcy and 16 patents, including one for coinventing the charge-coupled device.

But where is Amelio going to get this new operating system? From Be, Inc., in Menlo Park, Calif., a half-hour's drive from Apple's Cupertino headquarters, a hot little company founded by ex-Apple visionary Jean-Louis Gassee. Its BeOS, now undergoing clinical trials, is that radical redesign in operating systems that Amelio is talking about. Married to hardware from Apple and Apple cloners, the BeOS just might be a credible competitor to Microsoft's Windows, which runs on IBM-compatible hardware.

The capsule overview was automatically generated by a fully implemented, and operational, system, which incorporates all of the processing components identified above (see [4]). The relevant sections of the overview (for the three segments of the passage quoted) are as follows:

1 ... APPLE *would swoop in ...*
... *take* MICROSOFT'S *customers?*
... APPLE *lost $816 million;*
... MICROSOFT *made $2.2 billion.*
... MICROSOFT *has a market value ...*
... APPLE *is in a position ...*
... APPLE *needs something dramatic ...*

2 ... *Today's* DESKTOP MACHINES, *he says, are ill-equipped ...*
... *Tomorrow's* MACHINES *must accomodate ...*
... *scrap your* OPERATING SYSTEM ...
... OPERATING SYSTEM *is the software ...*

3 AMELIO *brings credibility ...*
HIS *resumé includes both ...*
... AMELIO *is going to get this* NEW OPERATING SYSTEM*?*
... *radical redesign in* OPERATING SYSTEMS ...
... AMELIO *is talking about.*

The division of this passage into segments, and the segment-based assignment of topic stamps, exemplifies a capsule overview's "tracking" of the underlying coherence of a story. The discourse segmentation component recognizes shifts in topic—in this example, the shift from discussing the relation between Apple and Microsoft to some remarks on the future of desktop computing to a summary of Amelio's background and plans for Apple's operating system. Layered on top of segmentation are the topic stamps themselves, in their relational contexts, at a phrasal level of granularity.

The first segment sets up the discussion by positioning Apple opposite Microsoft in the marketplace and focusing on their major products, the operating systems. The topic stamps identified for this segment, APPLE and MICROSOFT, together with their local contexts, are both indicative of the introductory character of the opening paragraphs and highly representative of the gist of the first segment. Note that the apparent uninformativeness of some relational contexts, for example, '... APPLE *is in a position ...*', does not pose a serious problem. An adjustment of the granularity—at capsule overview presentation time—reveals the larger context in which the topic stamp occurs (e.g., a sentence), which in turn inherits the high topicality ranking of its anchor: '*APPLE is in a position where standing pat almost certainly means slow death.*'

For the second segment of the sample, OPERATING SYSTEM and DESKTOP MACHINES have been identified as representative. The set of four phrases illustrated provides an encapsulated snapshot of the segment, which introduces Amelio's views on coming challenges for desktop machines and the general concept of an operating system. Again, even if some of these are somewhat under-specified, more detail is easily available by a change in granularity, which reveals the definitional nature of the even larger context '*The* OPERATING SYSTEM *is the software that controls how your computer's parts...*'

The third segment of the passage exemplified above is associated with the

stamps GILBERT AMELIO and NEW OPERATING SYSTEM. The reasons, and linguistic rationale, for the selection of these particular noun phrases as topical are essentially identical to the intuition behind *'priest'* and *'Pope'* being the central topics of the example in Section 1. The computational justification for the choices lies in the extremely high values of salience, resulting from taking into account a number of factors: co-referentiality between *'Amelio'* and *'Gilbert Amelio'*, co-referentiality between *'Amelio'* and *'His'*, syntactic prominence of *'Amelio'* (as a subject) promoting topical status higher than for instance *'Apple'* (which appears in adjunct positions), high overall frequency (four, counting the anaphor, as opposed to three for *'Apple'*—even if the two get the same number of text occurrences in the segment)—and boost in global salience measures, due to "priming" effects of both referents for *'Gilbert Amelio'* and *'operating system'* in the prior discourse of the two preceding segments. Even if we are unable to generate a single phrase summary in the form of, say, *'Amelio seeks a new operating system'*, the overview for the closing segment comes close; arguably, it is even better than *any* single phrase summary.

As the discussion of this example illustrates, a capsule overview is derived by a process which facilitates partial understanding of the text by the user. The final set of topic stamps is designed to be representative of the core of the document content. It is *compact*, as it is a significantly cut-down version of the full list of identified terms. It is highly *informative*, as the terms included in it are the most prominent ones in the document. It is *representative* of the whole document, as a separate topic tracking module effectively maintains a record of where and how referents occur in the entire span of the text. As the topics are, by definition, the primary content-bearing entities in a document, they offer *accurate* approximation of what that document is about.

4 Conclusion

The particular strength of terminology identification lies in the guaranteed robustness of the technology, itself due to the clear understanding of how the linguistic properties of technical terms manifest themselves via a variety of lexical, syntactic and discourse factors. Within the confines of a specific genre, namely scientific prose, this makes for a powerful tool for certain types of content management.

What we have shown in this paper is that such a methodology, attempting to map certain syntactic, semantic and discourse properties of a larger set of information bearing units onto their local contexts in text documents, enables the definition of new class of content characterisation algorithms. These can be viewed as a natural extension of term extraction, but result in term-based representations of document content that are nevertheless considerably more coherent than simple term enumeration.

By relaxing the notion of a 'term' to encompass all discourse referents in a wide range of text documents, we develop a comprehensive representation of the core content of documents. By making the transition from a *lexical* notion

of a term to that of a *discourse* notion of a referential entity, we also make it possible to bring strong semantic and discourse factors (such as anaphoric reference, coreference, salience, and so forth) in the contextual analysis of the long range behaviour of the discourse referents. This, in its own turn, allows for the use of discourse referents as anchors around which to organise their additional distinguishing factors.

While relaxation of the notion of term necessarily brings about large sets of referents—certainly too large to use constructively for characterisation purposes—the very fact that now we are dealing with discourse (rather than with text) entities becomes instrumental in being able to associate with each of these entities highly informative discourse information. This encapsulates the analysis of their distributional properties, and ultimately makes it possible to distill, from the unstructured original set of all discourse referents in a document, a highly representative subset, itself enriched by salient information pertinent to these core discourse entities. By applying different methods for tracking entities through the document, by enforcing different criteria for calculating salience, and by employing different degrees of granularity of contextual information associated with the topical set of discourse referents, we are now in a position to develop a family of term-based representations of individual documents, as well as document collections with varying degrees of intra-document coherence.

References

1. Apple Computer, Inc., 20525 Mariani Avenue, Cupertino, CA 95014–6299. *Macintosh User's Guide*, 1994.
2. B. Boguraev. WORDWEB and APPLE GUIDE: a comparative evaluation. Technical report, Internal Report, Advanced Technologies Group, Apple Computer, 1995.
3. B. Boguraev. Content analysis via lexical semantics. *The Apple Research Labs Review*, pages 2–13, September 1996.
4. B. Boguraev and C. Kennedy. Salience-based content characterisation of text documents. In *Proceedings of ACL'97 Workshop on Intelligent, Scalable Text Summarisation*, Madrid, Spain, 1997.
5. B. Boguraev and J. Pustejovsky, editors. *Corpus processing for lexical acquisition.* MIT Press, Cambridge, Mass, 1996.
6. D. Bourigault. Surface grammatical analysis for the extraction of terminological noun phrases. In *14th International Conference on Computational Linguistics*, Nantes, France, 1992.
7. J. Buchan. *Heart's journey in winter.* Harvill Collins, London, 1996.
8. I. Dagan and K. Church. Termight: identifying and translating technical terminology. In *4th Conference on Applied Natural Language Processing*, Stuttgart, Germany, 1995.
9. M. Hearst. Multi-paragraph segmentation of expository text. In *32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, 1994.
10. I. Heim. *The semantics of definite and indefinite noun phrases.* PhD thesis, University of Massachusetts, Department of Linguistics, Amherst, MA, 1981. unpublished.

11. J. Hodges, S. Yie, R. Reighart, and L. Bogges. An automated system that assists in the generation of document indexes. *Natural Language Engineering*, 2:137–160, 1996.
12. N. Hutheesing. Gilbert Amelio's grand scheme to rescue Apple. Forbes Magazine, December 16, 1996.
13. M. Johnston, B. Boguraev, and J. Pustejovsky. The structure and interpretation of compound nominals. In *AAAI Spring Symposium on Generativity and the Lexicon*, Stanford, 1994.
14. J. S. Justeson and S. M. Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering*, 1(1):9–27, 1995.
15. L. Karttunen. Discourse referents. In J. McCawley, editor, *Syntax and Semantics*. Academic Press, New York, NY, 1968.
16. C. Kennedy and B. Boguraev. Anaphora for everyone: Pronominal anaphora resolution without a parser, In *Proceedings of COLING-96 (16th International Conference on Computational Linguistics)*, Copenhagen, DK, 1996.
17. C. Kennedy and B. Boguraev. Anaphora in a wider context: Tracking discourse referents. In W. Wahlster, editor, *Proceedings of ECAI-96 (12th European Conference on Artificial Intelligence)*, Budapest, Hungary, 1996. John Wiley and Sons, Ltd, London/New York.
18. S. Lappin and H. Leass. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561, 1994.
19. I. Mani and T. R. MacMillan. Identifying unknown proper names in newswire text. In B. Boguraev and J. Pustejovsky, editors, *Corpus Processing for Lexical Acquisition*, pages 41–60. MIT Press, 1996.
20. M. M. McCord. Slot grammar: a system for simpler construction of practical natural language grammars. In R. Studer, editor, *Natural language and logic: international scientific symposium*, Lecture Notes in Computer Science, pages 118–145. Springer Verlag, Berlin, 1990.
21. G. Salton. Syntactic approaches to automatic book indexing. In *26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York, 1988.
22. G. Salton, Z. Zhao, and C. Buckley. A simple syntactic approach for the generation of indexing phrases. Technical Report 90-1137, Department of Computer Science, Cornell University, 1990.
23. S. Waterman. Distinguished usage. In B. Boguraev and J. Pustejovsky, editors, *Corpus processing for domain acquisition*, pages 143–172. MIT Press, Cambridge, MA, 1996.

Short Query Linguistic Expansion Techniques: Palliating One-Word Queries by Providing Intermediate Structure to Text

Gregory Grefenstette

Rank Xerox Research Centre,38240 Meylan, France
Gregory.Grefenstette@grenoble.rxrc.xerox.com

Abstract. The usual approach to finding information on the WWW via existing Web browsers is to use a one or two word query. Browsers return a number of documents containing these words, and the user examines those documents, or their abstracts, sees how the word or words in their query are being used and alters their initial query accordingly. This contrasts markedly with the Information Retrieval models explored by researchers over the past thirty-five years. These models were designed for longer queries and do not provide an adequate response to the user needs. On the other hand, recent advances in natural language processing permit the extraction of typed information that is axed on one or two words. We review a selection of this typed information and describe how it could be used to present an intermediate structure for the user fitting between their short queries and the documents found in a heterogeneous text collection such as the WWW.

1 The Gap Between Practice and Research

There is a well-known gap between the characteristics of the typical user studied by Information Retrieval (IR) researchers and the typical information seeker on the Web. Most real searchers use only a few words[33], but the bulk of Information Retrieval research since the 1960's has been based on longer queries.

There are two primary reasons for this gap between practice and research. The first is that most of the research in information retrieval was funded by government agencies, and it is standard practice in government intelligence agencies to express information needs as long, detailed descriptions of the subjects that interest them. Government agents will willingly research and list all the ramifications touching a topic. These descriptions generate long queries that can be used for filtering documents.

Secondly, in order to prove that one information retrieval technique works better than another, also a necessary task to secure research funding, one must have objective bases for judging when a system has gotten the right answer. Cleverdon *at al.* [12] helped devised the first information retrieval testbeds, called the Cranfield collections[1], which are composed of a set of documents, a set of

[1] One of these Cranfield collections is still publically available, at ftp://ftp.cs.cornell.edu/pub/smart/cran. This ftp server also contains the Information retrieval testbeds: *adi, cacm, cisi, med, and npl;* as well as a research version of the information retrieval system SMART.

natural language queries on these documents, and a relevance mapping, listing which documents are relevant to which queries.

In order that the relevance judgments represent some objective standard against which to measure, the queries must be relatively explicit about what is being searched for, which means they must be verbose. A short one or two word query could not be used in such experiments without the relevance judgments either becoming subjective (the judger reinterpreting what the searcher was looking for), or being reduced to a simple judgment of when a pattern was found in a text (something any file system search tool provides). These two reasons, then, that government funding agencies used long queries, and that long queries are necessary in order to objectively measure retrieval accuracy, led to the creation of a number of information retrieval testbeds, many of which can be found on the ftp server at Cornell University, and to the structure of queries tested in the current series of Text Retrieval Conferences (TREC) sponsored by the National Institute of Standards since 1992[26].

1.1 The Vector Space Model and Query Length

As a result of the above, the vector space model of information retrieval has been favored as an underlying IR implementation in research. This model[37] considers each word in the language as an axis is a highly dimensional space. Each dimension, each axis, in this space corresponds to a unique word found in the language. Each document and query are plotted in this space, with the distance along each axis being dependent upon the number of times the word appears in the document or the query. This distance is normalized using the document length[6]. To find the closest documents to a query, one calculates the cosine of the angle between the query point and all the document points, taking the points found closest as the relevant documents. This geometrically based theory is very attractive, theoretically. But this model works best when many axes (many query words) are being used to circumscribe the space of relevant documents, and when the model can exploit the repetition of certain important words within the query in order to give greater weight, and thus focus the query more accurately.

For short queries, the vector space model is inadequate. This inadequacy is recognized by traditional Information Retrieval researchers. Kwok[32] remarks that short queries lack two elements that make vector space retrieval over long queries successful: the variety of terms used to match the query to documents, and the indication of the importance of the words in the query. This latter point, which most affects precision according to Kwok, is automatically estimated in traditional information retrieval systems from the frequency with which the word appears in the query. Of course, in short queries, each word will usually only appear once and frequency can not give us any clues of the terms importance.

In order to show the interest of giving more weight to the important words in a short query, Kwok first performed a series of experiments in which a human user manually picked the important words in a query, presumably something that one might ask of Web Browser user, and then these words were repeated twice in the query submitted to the IR system. This simple technique improved the average precision of the short queries by 10%, a significant gain in precision in information retrieval circles.

Kwok then devised an automatic method of weighting query words by replacing their initial weight of 1 by a weight representing the average number of times a word appears in a document, when it appears, divided by a function of the absolute frequency of the word in the whole document database. This weighting gives rare words which appear fairly often in the documents they appear in more weight than common words or rare words that appear sporadically. He showed[32] that this automatic weighting performed as well as having a user manually choose the most important words in a query.

1.2 Actual Web Use

Most queries produced by users on WWW browsers are only a few words long. The user is essentially casting out a fishing line into an enormous ocean of pages to see what bites. The words that are used in the query are weighted based on their relative frequency[2] in the pages indexed and returned by the browser. No one can say how good a browser is. The user is happy to get any reaction from the system, ignorant of what they might be missing, a common situation with real use of information retrieval systems[2]. It is difficult to judge goodness because, not only are there no objective testbeds such as those created in the information retrieval community for long queries, but given the typical short query, one can not objectively say whether a given Web page containing that word or those words is relevant to a given user, since it is not clear what the user is really looking for. The users may not know themselves. This is why the term "browser" fits these information retrieval tools so well.

These browsers return more precise results when either more words are given or some positional restrictions are specified on the query words. Some search engines allow the user to specify a positional arrangement so that the query words appear "near" each other, or in a certain order. The burden of predicting how the words are used is left nonetheless on the user, the browser considering the whole space of the Web as a flat index structure. As one response to this flat structure, a few search engines, such as Yahoo![3], provide a hierarchical structure into which pages are inserted, but this insertion is done manually which limits the number of pages that can be indexed.

---

[2] This method of ranking pages has led to the "word spamming problem" in which a malicious HTML programmer fills in a WWW page with a large number of occurrences of a hidden key word (such as a competitor's company name) so that the programmer's page is shown first to the unsuspecting user.

[3] http://www.yahoo.com

1.3 Feedback from Short Query Results

A method for improving precision of a query by including frequently occurring terms from an initial retrieved document set can be implemented in the DIALOG information retrieval system using the RANK command[4],

first introduced in 1994. Once a user performs a search, RANK will sort all the indexed words appearing in the result set and display them to the user, who can then use them to make their query more precise. This feedback step can happen, after performing a search, but before the user examines the first set of documents returned.

Recently, François Bourdoncle of the Ecole des Mines de Paris has developed a somewhat similar version of this intermediate approach between query and documents, called LiveTopics, for the Altavista[5] browser. LiveTopics shows, in either a graph or a list, the words which appear in the neighborhood of the original query words in the retrieved documents[6]. One can click on these words which are then added to the query to improve the precision of the original query.

The above methods are a first response to the fact that a short query can often pick up many different aspects of meaning in the document collection containing the words in the query. A more sophisticated response is given in Hearst *et al.*, a paper that describes a system in which the documents returned from an initial query are clustered along these strands of meaning. This clustering is automatically performed on the documents that contain the short query terms by a technique developed in a collection browsing system called Scatter-Gather[14]. This linear-time technique works by splitting the returned documents into a fixed set of clusters, each cluster is then represented by its most representative terms, which are displayed to the user along with the first few titles in each cluster. The user can then easily see which of the groupings correspond most closely to the information that they were looking for and use the whole cluster as a new request via a traditional relevance feedback technique[25].

For example, Hearst *et al.* [27] show that, given the one-word query *star* over the text of a general encyclopedia, the 400 documents returned are automatically clustered by the Scatter-Gather technique into groups characterized by words such as **Cluster 2:** *flag, rug, weave, carpet, pattern, stripe, force*; **Cluster 3:** *game, player, team, league, ball, football, professional*; **Cluster 4:** *energy, hydrogen, radiation, planet, temperature, gas*, etc. These clusters show different aspects of the meaning of the polysemous word *star*, as a pattern, as a person, as a physical phenomena, and so on. This technique, which retrieves and then clusters documents, gives a general view of the words used in the same documents as short query words, and provides an alternative dynamic approach to using the more fine-grained pre-stored structures proposed below.

---

[4] http://www.dialog.com/dialog/publications/rank-qrc.html

[5] http://altavista.digital.com

[6] The same idea was described by Doyle [17] in the early 1960's as a realization of her Semantic Road Maps.

2 Information Extraction and Information Structuring

It is a very simple matter to say whether or not a certain string appears in text stored on a computer. In essence, this is what Web browsers do. They crawl[10] through the Web, retrieve pages, tokenize the text, and store each token with a pointer to its Web page. Given a one-word query, the browser returns the pages pointed to by that string. The pages are usually ranked, so that if the string appears in the `<title>` field of the HTML page, the page is ranked higher. The page is also ranked higher in function of the frequency of the word in the page. Most browsers will match on a string without regard to upper or lower case, maybe giving a slightly higher weight to the pages in which the case of the search word and the token in the page match.

But the browsers could be finding out much more about a word than its simple presence in the page. The word appears in a context that determines the word meaning. Current natural language processing allows much of this structure to be recognized, and stored, as a page is being retrieved and indexed by the Web crawler. If this structure is stored at index time, then an alternative route from a one or two-word query could pass through this static previously-recognized structure, before Web pages are presented to the user. Rather than delving straight into the documents as is the current case, the user would be able to see what abstract linguistic structures a word appears in. These more precise structures can then be clicked to reach the final Web-based documents.

Given the structures in which a word is found, its meaning can often be deduced. For example, "dog" can mean many things. But one easily sees its nuances of meaning when given such phrases as "dog biscuit" or "rabid dog" or "walk a dog" or even "hot dog."

These and other types of linguistically-motivated structures can be automatically extracted from text using present-day natural language technology. Not only for English, but for many other languages, there exists robust natural language technology[19] for performing basic processing of raw text. In the remaining sections we will overview these natural language processing technologies, covering how they work, and what they can extract. Then in section 3 we will show how this information might be succinctly presented to a user in response to their short, one word query.

2.1 Preliminary Natural Language Processing

Language Identification In the context of documents brought back from a Web crawler, one of the first tasks after removing HTML markup and before performing any language-specific processing is, of course, determining what is the original language of the text to be treated, if this is not mentioned in the HTML metalanguage which it rarely is. As an example of this problem, the one-word query "parole" on Altavista in the Spring of 1997 gave back six French, three Italian, eleven English and one Japanese page among the first twenty document returned. An automatic Web crawler that brings back these pages for

Danish	Dutch	English	French	German	Italian	Norweg.	Portug.	Spanish	Swedish
i	de	the	de	der	di	og	de	de	och
af	van	and	la	die	e	det	a	la	i
og	het	to	le	und	il	han	que	que	att
at	een	of	à	den	che	i	o	el	som
til	en	a	et	in	la	er	e	en	en
for	in	in	des	von	a	på	do	y	är
	til	da	a	på					
	at	não	los	det					
del	som	um	del	av					
	var	em	se	för					
	jeg	para	por	med					
	med	com	las	den					
	ikke	se	con	till					
	jeg	é	un	har					
	med	os	para	de					
	ikke	uma	una	ett					
	for	no	El	inte					
	har	-	su	om					
	så	na	al	-					

Fig. 1. Most frequent short tokens per language derived from the ECI Multilingual Corpus.

indexing must be able to identify which language processing tools to apply to which pages.

Fortunately, language identification[23] is a rather simple task. One could attempt to rely on Internet domain names, but many domains, such as **.ca** for Canada of **.be** for Belgium, cover pages written in many languages. It is better to rely on characteristic short words of the common languages (Figure 1) or characteristic character sequences (Figure 2), which can be calculated from any large body of text in a given language. Automatic language identifiers have been available on the Web since 1996[7].

Tokenization Once the language is identified, language specific tokenizers can be applied. A tokenizer is a device that segments an input stream into an ordered sequence of *tokens*, each token corresponding to an inflected word form, a number, a punctuation mark, or other kind of unit to be passed on to subsequent natural language processing.

Most sequences of uninterrupted alphabetic characters compose a token in most languages, but the use of separators inside words varies from language to

[7] http://www.rxrc.xerox.com/research/mltt/Tools/identifier.html

Danish	Dutch	English	French	German	Italian	Norweg.	Portug.	Spanish	Swedish
er_	en_	_th	_de	en_	_di	et_	_de	_de	en_
en_	de_	he_	es_	er_	to_	en_	de_	de_	er_
for	_de	the	de_	_de	_de	er_	os_	os_	et_
et_	et_	nd_	ent	der	di_	_de	do_	_la	tt_
ing	an_	ed_	nt_	ie_	_co	_ha	que	el_	_de
_fo	n_d	_an	_le	ich	la_	an_	_qu	la_	ar_
	de_	_co	que	för					
	-,-	as_	as_	-,-					
	-,-	ent	ue_	för					

Fig. 2. Most frequent trigrams per language derived from the ECI Multilingual Corpus.

language. For example, the sequence `l'amour` might split into two tokens in French, the article `l'` and the noun `amour` ; while `won't` might be considered as a single token in English, as it is in the Brown Corpus, a 1 million hand-tagged sample corpus heavily used in natural language processing. On the other hand, in certain language-dependent instances, a sequence of words (e.g., `hot dog, ein bisschen, a priori, e. g., parce que, a fuera de, in order to`) may be considered as a single token for further linguistic treatment.

Though proper tokenization is not necessary when words are taken as isolated indexing units, for natural language processing of text is is important to do it right. In order to recognize structure in text, it is necessary to recognize sentence boundaries[20], since parser tools work on entire sentences as input. It is also necessary to correctly recognize each word, since information used by the parser, such as whether the word is a noun, a verb, an adjective, a number, etc., has to be correctly associated with each token during morphological analysis.

One approach[38] to tokenization is to provide a cascade of language-dependent finite-state transducing tokenizers. These tokenizers segment text by introducing a token boundary (usually a new-line) into the output stream. The cascade is composed of a *basic tokenizer* which segments any sequence of input characters into simple tokens (i.e. no multiword units) and one or several *multiword staplers* which identify multiwords and group them together as single units. The development and implementation of a finite-state longest match operator [31] has made this development both practical and possible.

Morphological Analysis Once the text of the WWW page is tokenized, each token has be identified for subsequent language processing. In order to recognize the syntactic structures that word participates in, it is necessary to know what grammatical roles a word can play. For example, the token "watches" can be a plural noun or the present singular tense of a verb. On the other hand, the token "witches" can only be a plural noun. Information about what parts-of-speech a

word can play in a language is contained in a lexicon[8] for that language[7, 8]. A program which uses this lexicon to classify tokens is called a morphological analyzer[9][35].

Morphological analysis usually returns, for each analyzed token, the parts of speech that token can play, as well as the normalized dictionary entry form, the *lemma* of the word. The proper lemma for each word is chosen after the part-of-speech the word plays is disambiguated, see the next section. Morphological analysis can be contrasted with the traditional information retrieval of *stemming*.

A stemmer, such as the Porter[10] stemmer[34], contains a set of suffixes to be stripped from words. The suffix is removed under certain conditions, such as being preceded by a consonant. When the suffix is removed, another suffix may be added, for example, removing "ies" and adding "y." The purpose of a stemmer is, using the most productive derivational mechanisms of a given language, to reduce all the words that derive from a same source to a unique stem, so that if a user in an information retrieval setting uses one of the words, they will get answers concerning any of them. Most stemmers contain a set of exceptions so that words like "better" are not stemmed to "bet." And as the list of exception and the number of transformation rules grow, stemmers approximate linguistically motivated morphological analyzers better and better[28] at recognizing variants of words, but stemmers do not provide the part-of-speech information needed for subsequent natural language processing.

Input Phrase: *pending antitrust investigations involving an alleged violation*
Stemmed Version: *pend antitrust investig involut an alleg violat*
Lemmatized version: *pending antitrust investigation involve an allege violation*

Fig. 3. Difference between normalization of words from an inflectional lemmatization and from Porter stemming of an English phrase.

Part of Speech Tagging In order to recognize linguistic structure, a parser must know or decide what role each word is playing in a given sentence. It is

[8] See samples of lexicons for English, French, Spanish and German at http://www.lpl.univ-aix.fr/projects/multext/MUL5.html. The MULTEX project is working to make a wide variety of lexicons and natural language processing tools available for the research community. EAGLES (http://www.ilc.pi.cnr.it/EAGLES/home.html), a partner project, is concerned with normalizing lexicon formats.

[9] Interactive morphological analyzers for English, Dutch, French, German, Italian, Portuguese, and Spanish, as well as part-of-speech taggers for these languages, are available at http://www.rxrc.xerox.com/research/mltt/toolhome.html.

[10] http://www.cs.jhu.edu/~weiss/stem.c lists source code for the English version of the Porter stemmer.
http://wwwots.let.ruu.nl/UPLIFT/stinfon.ps describes a Dutch version of the Porter stemmer.

common to use a part-of-speech tagger to eliminate most or all of these ambiguities before parsing. For example, "watches" is ambiguous as we said above, but "the watches" is not, since an article is much more likely to be followed by a noun than a verb. A standard practice is natural language processing is to use these probabilities to eliminate unlikely readings for words.

The most common method of developing a probabilistic part-of-speech tagger is to first create a manually tagged corpus, i.e. for a certain amount of text, a human decides what part-of-speech tag should be assigned to each word. From this manually tagged text, probabilities are calculated about the frequency with which tag sequences are found[15]. If enough text is tagged, then the frequency with which a certain tag appears with a certain word can also be stored[5]. Then, from this statistical information, a Hidden Markov Model is built and used to disambiguate the parts-of-speech by calculating the most probable path through all the possible tags[9][Chap 3.3].

Alternative techniques are to train a tagger over untagged text so that the entropy of tag transitions is minimized[13][11], or to use a manually tagged corpus to generate automated correction rules[4][12], or to create a set of rules describing which transitions (such as article followed by verb) are impossible[41][13]

Once the part-of-speech that each word plays in a sentence is determined, the words can be lemmatized, i.e. reduced to their normalized forms. This reduction allows different variants of similar structures to be recognized as being identical, which will allow the subsequent counting of the event to be more accurate.

Noun Phrase Recognition One of the most important multiword structures that one can recognize easily after part-of-speech tagging is the simple noun phrase. Much of the terminology found in a document collection is composed of noun phrases, and when a word is included in one of these phrases, one has a good indication of the word's meaning. For example, alone, the noun "star" is ambiguous, but not when it is embedded in the noun phrase "one star hotel."

In order to recognize noun phrases, one can define a regular pattern[39] of part of speech tags describing the contour that such phrases can take. For example, Figure 4 shows the output of a part-of-speech tagger and a definition of a noun phrase filter. When the filter is applied to the tagged text, the following noun phrases are matched: *industrial products*, *red warning lights*, *computerized design of integrated radio circuits*.

Let's reconsider the example *star*. Running a noun phrase filter over a large British corpus yields noun phrases containing *star* such as: *favourite pop star*, *orange K-type star*, *former England star*, *central star cluster*, *bright new star*, *white dwarf star*, *western film star*, *unorthodox ballroom star*, *red giant star*, *poor star vehicle*, *major league baseball star*, *international film star*, *very heavy star*,

[11] The Common Lisp source code for such a tagger, version 1.2 of the Xerox part-of-speech tagger, is available at ftp://parcftp.xerox.com/tagger-1-2.tar.Z.

[12] This tagger created by Eric Brill for English can be found via ftp://ftp.cs.jhu.edu/pub/brill/.

[13] The ENGCG tagger can be tested by sending an e-mail to engcg@ling.helsinki.fi.

Input Phrase: *industrial products, such as red warning lights, are now produced by computerized design of integrated radio circuits*

original	*lemma*	*tag*
industrial	industrial	+JJ
products	product	+NNS
,	,	+CM
such as	such as	+IN
red	red	+JJ
warning	warning	+NN
lights	light	+NNS
,	,	+CM
are	be	+BER
now	now	+RB
produced	produce	+VBN
by	by	+RB
computerized	computerize	+VBN
design	design	+NN
of	of	+IN
integrated	integrate	+VBN
radio	radio	+NN
circuits	circuit	+NNS

Tags	*meaning*
+BER	form of verb 'to be'
+CM	comma
+IN	preposition
+JJ	adjective
+NN	noun
+NNS	plural noun
+RB	adverb
+VBN	past participle

Noun Phrase Filter:
define **PRE** *as one of the following tags:* +JJ, +VBN, +NN, +NNS
define **NOUN** *as one of:* +NN *or* +NNS
define Filter as: **(PRE)* NOUN ((of (PRE)* NOUN))***
Phrases matched by Filter:
industrial products,
red warning lights,
computerized design of integrated radio circuits

Fig. 4. Defining a noun phrase filter as a regular expression over tags and words. The star(*) stands for any number of repetitions, including zero. The filter matches any number of members of the PRE class followed by a noun, followed by any number of the sequence "of" followed by possibly modified nouns.

very bright star, *very average star*, *near naked-eye star*, *huge red star*, ... In nearly every case, except maybe for *very heavy star*,the use of *star* in these phrases is no longer ambiguous. In the current situation, though, even though these phrases are automatically extractable, the Web user has no access to them except by guessing their existence and posing a positional query to a Web browser.

Light Parsing Extending the techniques for noun phrase recognition, one can define regular patterns which recognize verb groups, and which identify the heads of the noun groups and verb groups; low-level syntactic relations can then be drawn between these group heads[21, 18]. Finite-state compilers[30, 40] are very powerful tools for describing the patterns and the constraints that patterns must respect in order be recognized as a syntactic group. Just as one technique for tokenizing text is to feed the input to a cascade of transducers that insert token boundary markers such as newlines in the input, higher level syntactic markings can be inserted into tagged text[1].

Figure 5 shows how additional markings can be added into tagged text using more elaborate regular expressions[29] compiled into transducers that insert or remove symbols. These additional markings allow one to easily define other regular expressions for recognizing specific syntactic configurations, for example, a configuration corresponding to subjects of passive verbs. These filtering expressions can include transducing elements that delete all the unimportant elements in the configuration, and which insert a label specifying the configuration. A passive direct object filter will extract a relation between *correlation* and *obtain*: PDOBJ(correlation,obtain). This relation can be made to correspond to relations extracted by the direct object filter which would like DOBJ(obtain,correlation). In this way, surface positional variation can be abstracted away from relations between words, just as lemmatization remove surface variation from different inflectional forms of a single word.

Elaborate patterns for around 100 nominalizations of communication verbs in English and French were extracted in the framework of the European project DECIDE [24]. This project, involving RXRC, the University of Liege (Belgium), and the University of Stuttgart (Germany), used automatic means of exploring corpora in order to extract collocations, in particular, support verbs for nominalizations. An online multilingual lexicon[14] was produced of French, German, and English equivalents of these corpus-derived collocations.

3 Short Query Linguistic Expansion Techniques

It was argued earlier that in the current state of affairs, the burden is one the Web user to know or to discover how words are used in the pages indexed by the Web Browser. The previous section sought to demonstrate that it is possible, with current linguistic technology, for Web Browsers to extract much more meaning-identifying structure from the text that they index. The Web user should have access to this structure to browse, just as they have access to the original Web pages. A person who uses an indexed book can see how a word is used before reading the pages on which the word is found.

In this section, we will examine a more extended example of the information can be derived from a large heterogeneous corpus around a given word. For this

[14] Consultable at http://engdep1.philo.ulg.ac.be/decide/lexicon.

Original Phrase: *Significant correlations were obtained between the maternal and fetal glucose levels and the maternal and fetal ffa levels.*

Group	*Heads*	*original*	*lemma*	*tag*
<NG		Significant	significant	+JJ
	*HeadN	correlations	correlation	+NNS
NG>				
<VG		were	be	+BED
	*PasV	obtained	obtain	+VBN
VG>				
<NG		between	between	+IN
		the	the	+AT
		maternal	maternal	+JJ
		and	and	+CC
		fetal	fetal	+JJ
		glucose	glucose	+NN
	*PrepN	levels	level	+NNS
		and	and	+CC
		the	the	+AT
		maternal	maternal	+JJ
		and	and	+CC
		fetal	fetal	+JJ
		ffa	ffa	+JJ
	*PrepN	levels	level	+NNS
NG>				

Tags	*meaning*
+BED	form 'to be'
+CC	conjunction
+AT	article
+JJ	adjective
+NN	noun
+NNS	plural noun
+VBN	past participle
*HeadN	head of noun phrase
*PrepN	tied to preposition
*PasV	passive verb head
<NG	begin Noun group
NG>	end Noun group
<VG	begin Verb group
VG>	end Verb group

Fig. 5. Additional marking inserted by low-level parser. Just as one can describe the contour of noun phrases using regular expressions, it is possible to define the contour of Noun Groups, and Verb Groups. Once these marks are inserted, other regular expression insert head markings.

example, we extracted from the British National Corpus[15] all the sentences that contain the string "watch." There are 15881 such sentences. This 2 M-byte corpus corresponds in a sense to gathering a large number of WWW pages containing any form of that string.

Running the BNC *watch* corpus through a noun phrase extractor produces a list of noun phrases, such as "neighbourhood watch," "watch tower," "digital watch," "watch dog," "Rolex Show," etc. 1066 different simple noun phrases can be found, 63 of which appear more than twice. Each of these phrases can be linked back to the pages from which they were extracted, in the WWW setting.

But how should this linguistically structured information be presented to the user? In some cases, *watch* is the head of the noun phrase, and in other cases it modifies some other noun. We cannot expect the naive user of the WWW to know or care about these linguistic distinctions. But we can rephrase these relations in a way that may be more understandable to the lay user. We can label those

[15] The BNC is a 100 million word corpus of British written and spoken English. See http://info.ox.ac.uk/bnc/index.html for information. We parsed files *a* to *g*.

phrases in which "watch" appears at the end as "types of watches." And those in which it appears in the beginning as "things involved in watches." Though the actual relations that a word can play in a two word phrase are multifarious [36, 42], these labels are vague enough to cover many of them. Given these labels, we can present the noun phrases structures in order of decreasing frequency to the user in a way such as in Figure 6. Due to tagging errors, some of words "watch" are actually being used a verb in the things listed as *watch things*. Still, the association of "watch" with another word often makes its meaning clearer.

types of **watch**:	gold watch, close watch, good watch, night watch, digital watch, careful watch, just watch, fob watch, wrist watch, stop watch, pocket watch, own watch, neighbourhood watch, ...
watch *things:*	watch television, watch TV, watch tower, watch people, watch video, watch glass, watch film, watch face, watch child, watch strap, watch spring, watch shop, watch salesman, watch press, watch mark, watch duty, watch cricket, watch chain, watch case, watch alarm

Fig. 6. Explaining the uses of "watch" in noun phrases.

For languages such as English which use capitalization as a proper name indicator, it easy to use the patterns of upper and lower case letters characteristic of names to recognize a large number of proper names[21, p. 150]. Alternative methods uses lists of proper name markers[3]. Recognized names can be semantically typed [16] by decoding their subcomponents. Once noun phrases have been recognized, one can isolate those noun phrases containing non-sentence-initial upper case from lower-case noun phrases, shown in Figure 7, providing a further automatic structuring to the WWW user.

Named **watch**:	Rolex watch, Swiss watch, Cartier watch, Seiko watch, Rab watch, Dali watch, British watch, American watch, Abberley watch, Z114 watch, Tudor watch, Thomas Pride watch, Taiwanese watch, Tag Heuer watch...
Names involving **watch**:	Neighbourhood Watch, Black Watch, Watch Committee, Night Watch,, Americas Watch, International Dolphin Watch, Green Leaf Watch, Bank Watch, Whitehall Watch, Watchorn Alfreton, Watches of Switzerland,..., Rolex Watch Co.

Fig. 7. Noun phrases in which "watch" appears with a proper name.

In addition to noun phrases, verbal phrases involving "watch" used either as a verb or as a noun can be extracted using Light Parsing technology mentioned in the previous section. When our "watch" corpus is parsed in this way, we find the typical direct objects of the verb "watch" and the typical verbs associated with the noun watch. For example, the most frequently found direct object of "watch" (other than a personal pronoun or "it") was "television" (34 times in 5134 sentences). The naive user of the WWW will not want to know about direct objects and subjects. In fact, most naive users of the WWW will hardly remember what a noun or a verb is. The information may be conveyed in a more simple way by using circumlocutions such as *Things one can watch* instead of the more academic *direct objects of watch.* In a similar the information concerning the use of *watch* as a verb can be presented as in Figure 8.

Things one can **watch**:	watch somebody, watch it, watch television, watch man, watch film, watch face, watch TV, watch video, watch news, watch game, watch people, watch play, watch match, watch programme, watch woman, watch one, watch show, watch movie, watch world, watch child, watch this, watch girl, watch walk, watch programmes, watch bird, watch football, watch space, watch scene, watch move, watch light, watch performance, ...
Who can **watch**:	somebody watch, people watch, man watch, it watch, eye watch, child watch, those watch, one watch, worth watch, stand watch, woman watch, anyone watch, everyone watch, sit watch, crowd watch, mother watch, time watch, girl watch, boy watch, hour watch, all watch, day watch, bird watch, million watch, face watch, audience watch, there watch, other watch, window watch, friend watch, ...
How can one **watch**:	watch out, watch just, watch back, watch down, watch on television, watch still, watch also, watch even, watch closely, watch somebody, watch carefully, watch now, watch too, watch home, watch over somebody, watch away, watch on TV, watch by, watch always, watch able, watch with interest, watch often, watch again, watch from window, watch never, watch in silence, watch together, watch with somebody, watch for somebody

Fig. 8. Verb phrases involving the verb "watch."

Above we saw how noun phrases involving the word *watch* might be presented

to the linguistically naive Web user, the uses of the noun *watch* with verbs might also be explained by listing things that a *watch* can do, or things that are done to a *watch*, as in Figure 9, which cover the uses of the word as a direct object and as a subject.

Things that one does to a **watch**:	keep watch, have watch, check watch, take watch, stand watch, consult watch, wear watch, sit watch, get watch, go watch, could watch, make watch, say watch, come watch, see watch, set watch, can watch, do watch, leave watch, receive watch, find watch, stop watch, wait watch, put watch, steal watch, buy watch, sell watch, need watch, watch watch, close watch, lose watch, ...
What can a **watch** *can do*:	watch stop, watch say, watch tell, watch happen, watch get, watch go, watch lie, watch move, watch leave, watch know, watch see, watch pull, watch make, watch let, watch hold, watch take, watch ask, watch like, watch agree, watch cost, watch cause, watch demand, watch learn, watch work, watch find, watch think, watch miss, ...

Fig. 9. verb phrases in which "watch" appears as a noun.

The linguistically derived structures mentioned in the previous sections show the words that are syntactically tied to "watch." If the WWW users pose a one-word query "watch" then presenting them with these lists would provide a useful overview for distinguishing the meaning the searcher was initially interested in. In every document in which "watch" appears, there are additional words, not directly entering into syntactic relations with "watch" but sharing a first order affinity [22], that is, often found in the same vicinity, though not syntactically attached to "watch." These words give another view of how the word is used, and like with the RANK function of Dialog and LiveTopics of Altavista, can be added to a boolean query to make the results more precise. In our "watch" corpus, the words most often used in the same sentences with "watch" outside of the words syntactically tied to it, and outside of stopwords such as article, prepositions, etc., are "time", "look," "stand," "way," "face," "night," "small," "years," etc. If a reference corpus exists, then these words can be further classified by applying a statistic such as "mutual recall" as has become popular in lexicography [11].

4 Conclusion

We have presented a number of natural language processing tools that could be used to provide an intermediate structure between short queries and the WWW. The tools provide for each word, a list of ways in which that word is used with

other words over the indexed documents. The presence of the other words allows the user to see the possible uses of the word, determine the meaning and to decide if they want to access the documents with those uses. The user has access to the way a word is used without having to access Web pages, The burden of the work is placed upon the computer during initial indexing, but existing natural language processing tools are already capable of providing the robust text analysis necessary, and the question of additional storage costs that such indexes would incur becomes less meaningful daily.

References

1. Salah Ait-Mokhtar and Jean-Pierre Chanod. Incremental finite-state parsing. In *ANLP'97*, pages 72–79, Washington, 1997.
2. D.C. Blair and M.E. Maron. An evaluation of retrieval effectiveness. *Communications of the ACM*, 28:289–299, 1985.
3. C. Borkowski. An experimental system for the automatic identification of personal names and personal titles in newspaper texts. *American Documentation*, 18:131, July 1967.
4. Eric Brill. A simple Rule-Based part of speech tagger. In *Proceedings of the Third conference on Applied Natural Language Processing*, Trento, Italy, 1992. ACL.
5. Ted Briscoe, Greg Grefenstette, Lluís Padró, and Iskander Serai. Hybrid techniques for training hmm part-of-speech tagger. Technical Report MLTT-007, Rank Xerox Research Centre, 1994.
6. Chris Buckley, Amit Singhal, and Mindhar Mitra. New retrieval approaches using smart: Trec4. In D.K. Harman, editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. U.S. Department of Commerce, 1996. NIST Special Publication 500–236.
7. John Carroll and Ted Briscoe. The derivation of a large computational lexicon for english from ldoce. In B. Boguraev and T. Briscoe, editors, *Computational Lexicography for Natural Language Processing*, London, 1989. Longman.
8. J.P. Chanod and P. Tapanainen. Creating a tagset, lexicon and guesser for a french tagger. In *Proceedings of the ACL SIGDAT Workshop*, Dublin, Ireland, 1995.
9. Eugene Charniak. *Statistical Language Learning.* MIT Press, Cambridge, Mass, 1993.
10. Fah-Chun Cheong. *Internet Agents: Spiders, Wanderers, Brokers and 'Bots.* New Riders Publishing, Indianapolis, 1996.
11. Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, March 1990.
12. Cyril W. Cleverdon. The significance of the cranfield tests on index languages. In A. Bookstein, Y. Chiaramella, G. Salton, and V. V. Raghavan, editors, *Proceedings of the 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 3–131, New York, Oct 13-16 1991. SIGIR'91, Association for Computing Machinery. Special issue of the SIGIR Forum.
13. Doug Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. A practical part-of-speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, April 1992.

14. Douglas Cutting, Jan O. Pedersen, David Karger, and John W. Tukey. Scatter/Gather: A cluster-based approach to browsing large document collections. In *Proceedings of SIGIR'92*, pages 318–329, Copenhagen, Denmark, June 21-24 1992. ACM.
15. Steven J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, Winter 1988.
16. David D. Donaldson. Internal and external evidence in the identification and semantic categorization of proper names. In B. Boguraev and J. Pustejovsky, editors, *Proceedings of the SIGLEX Workshop on Acquisition of Lexical Knowledge from Text*, pages 32–43, Columbus, OH, 1993.
17. Lauren B. Doyle. Semantic road maps for literature searchers. *Journal of the ACM*, 8(4):553–578, October 1961.
18. G. Grefenstette. Light parsing as finite state filtering. In *Workshop on Extended finite state models of language*, Budapest, Hungary, Aug 11–12 1996. ECAI'96.
19. G. Grefenstette and F. Segond. Multilingual natural language processing. *International Corpus of Corpus Linguistics*, 2(1), 1997.
20. G. Grefenstette and P. Tapanainen. What is a word, what is a sentence? Problems of tokenization. In *3rd Conference on Computational Lexicography and Text Research*, Budapest, Hungary, 7–10 July 1994. COMPLEX'94. http://www.rxrc.xerox.com/publis/mltt/mltt-004.ps.
21. Gregory Grefenstette. *Explorations in Automatic Thesaurus Discovery.* Kluwer Academic Press, Boston, 1994.
22. Gregory Grefenstette. Corpus-derived first, second and third-order word affinities. In *Sixth Euralex International Congress*, Amsterdam, Aug 3—Sept 3, 1994.
23. Gregory Grefenstette. Comparing two language identification schemes. In *Proceedings of the 3rd International Conference on the Statistical Analysis of Textual Data, JADT'95*, Rome, Dec 11-13, 1995.
24. Gregory Grefenstette, Ulrich Heid, and Thierry Fontenelle. The DECIDE project: Multilingual collocation extraction. In *Seventh Euralex International Congress*, University of Gothenburg, Sweden, Aug 13-18, 1996.
25. Donna Harman. Relevance feedback revisited. In *Proceedings of SIGIR'92*, Copenhagen, Denmark, June 21-24 1992. ACM.
26. Donna Harman, editor. *The First Text REtrieval Conference (TREC-1).* U.S. Government Printing Office, Washington, 1993. NIST Special Publication 500–207.
27. Marti A. Hearst, David Karger, and Jan O. Pedersen. Scatter/gather as a tool for the navigation of retrieval results. In Robin Burke, editor, *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA, November 1995. AAAI.
28. David A. Hull. Stemming algorithms: A case study for detailed evaluation. *JASIS*, 47(1), January 1996. Special Issue on the Evaluation of Information Retrieval Systems.
29. L. Karttunen, J.P Chanod, G. Grefenstette, and A. Schiller. Regular expression for language engineering. *Journal of Natural Language Engineering*, 1997.
30. Lauri Karttunen. Finite-state lexicon compiler. Technical Report ISTL-NLTT–1993-04-02, Xerox, Palo Alto Research Center, April 1993.
31. Lauri Karttunen. Directed replacement. In *Proceedings of the 34rd Annual Meeting of the ACL*, Santa Cruz, CA, 1996.
32. K. L. Kwok. A new method for weighting query terms for ad-hoc retrieval. In *Proc. of the 19th ACM/SIGIR Conference*, pages 187–196, 1996.

33. X. A. Lu and R. B. Keefer. Query expanison/reduction and its impact on information retrieval effectiveness. In Donna Harman, editor, *The Thirs Text REtrieval Conference (TREC-3)*, pages 231–239, Washington, 1995. U.S. Government Printing Office. NIST Special Publication 500–225.
34. M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.
35. G. Russell, S. Pulman, G. Ritchie, and A. Black. A dictionary and morphological analyser for english. In *11th International Conference on Computational Linguistics*, pages 277–279, Bonn, Germany, 1987.
36. Gerard Salton. A note on information retrieval models. In *RIAO'85*, pages 2–27, Grenoble, France, March 18-20 1985. CID, Paris, and IMAG.
37. Gerard Salton and M. McGill. *An Introduction to Modern Information Retrieval.* McGraw-Hill, New York, 1983.
38. Anne Schiller. Multilingual finite-state noun phrase extraction. In *Workshop on Extended finite state models of language*, Budapest, Hungary, Aug 11–12 1996. ECAI'96.
39. Anne Schiller. Multilingual part-of-speech tagging and noun phrase mark-up. In *15th European Conference on Grammar and Lexicon of Romance Languages*, University of Munich, Sept 19–21 1996.
40. Pasi Tapanainen. RXRC finite-state compiler. Technical Report MLTT-020, Rank Xerox Research Centre, Grenoble, April 1995.
41. Atro Voutilainen, Julia Heikkila, and Arto Anttila. A lexicon and constraint grammar of english. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France, July 1992. COLING'92.
42. Beatrice Warren, editor. *Semantic Patterns of Noun-Noun Compounds.* Acta Universitatis Gothoburgensis, Goteborg, Sweden, 1978. Gothenburg Studies in English, 41.

Information Retrieval: Still Butting Heads with Natural Language Processing ?

Alan F. Smeaton

Dublin City University, Glasnevin, Dublin 9, IRELAND
asmeaton@CompApp.dcu.ie

Abstract. Information retrieval (IR) is about finding documents which may be of relevance to a user's query, from within a corpus or collection of texts. While apparently a simple task at first glance, IR is in fact a hard problem because of the subtleties introduced by the use of natural language in both documents and in queries. The automatic processing of natural language clearly represents significant potential for improving information retrieval tasks because of the dominance of the natural language medium on the whole IR task. Information extraction is also fundamentally about dealing with natural language albeit for a different function. It is thus of interest to the IE community to see how a related task, perhaps the most-related task, IR, has managed to use the same NLP base technology in its development so far. This is an especially valid comparison to make since IR has been the subject of research and development and has been delivering working solutions for many decades whereas IE is a more recent and emerging technology.

1 Information Retrieval and Information Extraction

1.1 Information Retrieval

Information Retrieval (IR) is the task of finding relevant documents from a text corpus or collection in response to a user's information need. As a discipline and as an application, IR has grown enormously in importance in the last few years. Part of this has been due to the increased availability of information in digital form and part also is due to the increased inter-connection between people via computers. As more and more information has been made available in electronic format, especially text information, the increased volume of information available to us has meant that our requirement for effective search has also grown. Since the mid-1980s we have also seen the emergence of personal computing allowing people direct access to data and information. In previous IR applications people searched for information via an intermediary or expert searcher who acted as a conduit between users and their information needs, and the information itself. The combination of an increased amount of information and the direct access by endusers to that information has meant that IR is now large-scale, common and cheap whereas it used to be specialist and expensive.

Despite the fact that IR is now a commonplace operation with many tens of millions of searches on the WWW for example run daily, the techniques available and used in mainstream IR applications are the same or similar to the

technologies used by IR specialists years ago. There has been little transfer of technologies from specialist retrieval to the untrained users of today with their 1-, 2- and 3-word queries to the world wide web (WWW). An exception to this is the LiveTopics feature introduced in early 1997 by Alta Vista[1]. This is a feature whereby once a user's initial search has been run and has generated a ranking of some hundreds of web pages (documents), index terms from those ranked documents are then processed to determine the most commonly occurring and offered to the user as potential extra search terms for that search. Furthermore, index terms from throughout the document corpus which have co-occurred a significant number of times with those index terms suggested for query expansion are also listed and the whole collection of candidate extra search terms is presented graphically.

While AltaVista's LiveTopics may sound useful, and indeed it is innovative and an improvement in WWW search engines' functionality, to the naive enduser the fact that he is swamped with extra candidate terms to choose from with no guidance and no help and no expertise available, is as much a hindrance as a useful feature. The real measure in choosing terms for query expansion involves examining term frequencies and co-occurrence frequencies but the point here is that the LiveTopics feature is useful to a user only if he knows how to use it, and most endusers of IR systems, including WWW search engines, do not know what they are doing. Thus we find that contemporary information retrieval as used by the vast majority of users, is inadequate.

Within the evolution of IR there have been several significant developments over the most recent years worth mentioning. The first is that IR is becoming a basic technology underlying other applications. Whereas we will always have a requirement to search through information directly in its own right we are beginning to see IR embedded in other applications for example in agents that autonomously perform tasks that require a search component or as part of mail applications that require search through email archives. A second development is the emergence of other IR applications besides basic searching. The central procedure in an information retrieval operation is the matching operation between text and query but this basic function can be used in tasks related to information retrieval like the following:

- *categorisation* is a task wherein documents in a collection are assigned one or more static pre-defined categories from a closed set of such categories.
- *filtering* involves streaming documents against one or more user profiles, each reflecting a fixed information need from a user.
- *routing* also requires the distribution of incoming documents to groups or individuals based on content.
- *clustering* is where similar documents are grouped together for subsequent browsing or retrieval.

What all these tasks have in common is that they have at their core the matching between documents and some representation of a user's information need, where that need is either static or dynamic [1].

[1] http://www.altavista.digital.com

The final recent development in IR worth mentioning is the application of IR tasks to other media besides text. As more and more information has become available in media other than text, e.g. image, audio, etc., we are seeing techniques for content-based retrieval being used for these media, though they have some progress to make before reaching the same level of effectiveness as obtainable with text [14].

These major developments are important to information retrieval but they are really only contemporary information retrieval approaches wrapped around new applications or scenarios. In this chapter we are concerned with more fundamental issues relevant to information retrieval.

At a very abstract level, information retrieval appears to be a simple technology. Users indicate what information they seek and the system retrieves documents about that topic. In practice it is much more complicated than that. There are so many degrees of freedom and this is not just to do with the retrieval operation. For the most part, users of an IR system do not know exactly what they are looking for. If they did know then a database or other exact-match system would suffice but as users search they are exploring the document space and their own information needs also. There is also another degree of freedom at the other end of the IR process in that even if a user did manage to come up with the ideal query which fully encapsulates their information requirements it is unlikely that a document author would have used the exact same terminology to express the same content [7] thus matching query terms against document terms must be fuzzy. Furthermore, because documents are about many things a user may be interested in only a subset of a document's content so the query-document matching must account for this.

By far the biggest issue to complicate the information retrieval operation is the fact that we are dealing with natural language, often both in documents and in user queries and it is the properties of natural language that make IR so non-trivial. Text[2] consists of word tokens from a surprisingly small lexicon or dictionary, each of which can independently and in isolation convey some meaning. The morphology of words can be changed as they are concatenated into units called sentences for which there is a grammar of allowable syntactic combinations to which sentences must conform. The grammar specifying the legitimate combinations of word tokens affects the order of tokens and their morphology but this is not always strictly enforced, in user queries, for example. Sentences are in turn concatenated to make prose which makes up documents. When documents reach a large enough size they may be structurally organised, typically into a hierarchy, to ease navigation through the document, so we can have chapters, sections, sub-sections, paragraphs etc. A document corpus can be one single, large structured document like the technical documentation for a computer system, or a collection of many independent documents as in a newspaper archive.

[2] We cover written text here but many of these features are also applicable to spoken dialogue.

Word tokens do not have unique spelling and many common word tokens are polysemous in their base form like *bar*, *table* and *pen*. Many words can also be polysemous in their declined form like "lights" which can be a plural noun or third person singular present tense of the verb "to light", or "drove" which can have the same set of word classes.

Within society there are many sub-languages, each with their own sub-grammars. For example technical documentation, electronic mail messages, weather forecasts, legal documents, newspaper articles, fault reports, all are written in different styles with slightly different grammars. Technical documentation is terse, tight prose consisting of complex phrases and complex individual sentences are needed because we are usually conveying complex information, whereas electronic mail messages are often ungrammatical, short, full of abbreviations with local dialects and slang and tend to use simple grammatical constructs. Newspaper texts would be grammatically uncomplicated and short.

Text has an intricacy and a complexity and it is filled with synonymy and ambiguity, variations in capitalisation and spelling, syntax, grammar and the use of different word forms and we do not realise this until we try to process language computationally. Given that this natural language is the medium that matters in IR and IR applications it is no surprise to find IR a difficult task.

1.2 Information Extraction

Information extraction (IE) is a task which is related to IR and is well-described elsewhere in this volume. Whereas IR retrieves texts for users in response to their queries, IE processes texts into fixed format unambiguous data, perhaps for indexing documents as a component part of an overall IR process. IE can also exist on the same level as an IR system being used to automatically extract data on some topic of interest to a user from a corpus of texts and use this structured data as input to a spreadsheet or database.

In [4] information retrieval and information extraction are compared with the following highlightss:

- IE systems are more difficult and knowledge intensive to build;
- IE systems are more tied to given domains and scenarios;
- IE systems are more computationally intensive than IR;
- IE systems generally have higher precision than IR and so are potentially more efficient to use because of the possibility of reduced user requirements to read fewer non-relevant texts;
- IE lends itself to cross-lingual operations;

The last point is less of an issue now because cross-lingual information retrieval (CLIR) is currently generating huge interest in IR research with workshops, papers and a special track in TREC-6 all devoted to CLIR.

The relationship between IE and IR is quite multi-faceted as the two operations can potentially be combined in a number of ways such as the following:

1. IR could be embedded within IE to pre-process a large document collection to a manageable subset on which the more computationally expensive IE techniques can be applied;
2. IE can bs used as a component within IR where IE document analysis is used to identify index terms for document representation. For example, named entity recognition [4] is an important part of IE and would certainly help IR;
3. IR combined with IE wherein as a user browses an information space with a combination of IR search and subsequent browsing[3], IE can be used to summarise search output into some coherent whole;

The contribution of the present chapter to this volume is to present an overview of information retrieval from a technical perspective, concentrating particularly on the role of NLP in IR. The automatic processing of natural language clearly represents significant potential for improving information retrieval tasks. As we have seen, IR is fundamentally about dealing with language, language in documents and language in user queries and we will see that many of the difficulties with IR which make it a non-trivial task are due to phenomena of natural language. Information extraction is also fundamentally about dealing with natural language albeit for a different function. It is thus of interest to the IE community to see how a related task, perhaps the most-related task, IR, has managed to use the same NLP base technology so far. This is an especially valid comparison to make since IR has been the subject of research and development and has been delivering working solutions for many decades whereas IE is a more recent and emerging technology.

In this section we have shown how IR is indeed a non-trivial task due to its many degrees of freedom. In the next section we will generalise the features of current operational IR systems, not research prototypes or ideas but operational systems which have been shown to handle of the order of hundreds or megabytes or gigabytes of text. Following that we then discuss the various ways in which we can represent documents and queries and we look at some of the techniques which can be incorporated into the retrieval or matching operation.

The main contribution of this chapter is the discussion of the role of NLP in IR and so in section 5 we give a brief overview of the levels of NLP that concern IR, namely lexical, syntactic and semantic processing and we follow that, also in section 5, with a review of NLP as actually used in operational IR. Having covered the present contribution of NLP to operational IR we then discuss the prospects for the development of the relationship between NLP and IR in section 6, which is the final section.

2 Operational IR Systems

As we have seen, information retrieval is concerned with finding information of relevance to user's information needs. Traditionally IR has found application in

[3] This is the current paradigm for the vast majority of users browsing the WWW.

bibliographic searching of libraries, patent offices and legal applications but as more and more information has become available electronically this has led to users performing their own searches directly. The trickle-down phenomenon is now complete with literally millions of untrained users madly seeking information in the largest collection of globally distributed electronic information available, the world wide web, and generally not getting what they want but accepting that.

To implement information retrieval, the general approach taken is to represent both documents and queries as bags of index terms. For simplicity these index terms can be normalised forms of words such as word stems or word base forms, or perhaps the terms may consist of linguistically or statistically motivated phrases. The "matching" criterion is to count the number of terms in common between query and document and normalise this count by some function of the document and/or query lengths, factoring in such things as passage-level evidence, relevance feedback, query expansion, and so on. Clearly this is inadequate and demands that authors and searchers use a common vocabulary, a proposal which is never palatable.

Given that IR is addressing a problem where user's information needs vary over time it is desirable to add some dynamicity into the IR process. This can be manifest as incorporating relevance feedback from the user back into the retrieval process to help retrieval by feeding back to the system which of the retrieved documents are relevant. Alternatively, or in addition, a system should allow a user to explicitly alter their initial query in mid-search by adding or deleting query terms from the working query. Another desirable feature of IR systems is that they provide ranked output of documents rather than sets. With ranked output, ranked according to estimated similarity to the query, a user decides when to stop looking for documents, i.e. the user takes on the responsibility for this; in set-based retrieval as we get with boolean searches, the system presents an unordered set of documents and it behoves the user to find relevant documents from within this unordered set.

Apart from technical issues, the main philosophical problem with operational information retrieval systems is due to the representation of documents and queries as bags of terms which yields information systems which match at the symbolic,or literal level. Implicit in this are the following assumptions

- users know what they want and can articulate it accurately;
- users' information needs stay fixed during a search;
- authors of documents know what they want to say and can also articulate it accurately;
- users know what terms are used in documents;

Clearly in view of what we have seen in the earlier part of this chapter, all of these assumptions are false and so operational information retrieval must fall short as a solution to the specification of what we really want from IR which is to satisfy user's real information needs.

The IR systems used today are based on matching bags of index terms which as we saw is inadequate, yet operational information retrieval does provide func-

tionality and has done so for decades. The principles of document ranking etc. developed 20 years ago or more have filtered into operational IR and both theoretical and empirical research into improving this has been an active area. Sections three and four of this chapter describe just some of the approaches which have tried to represent (or index) documents and queries and to perfect the matching operation.

The evaluation and comparison of approaches has been an important feature of IR research with the widespread benchmarking of new and innovative approaches to IR components being the norm. Such evaluations have normally been carried out under scientific conditions with the use of test collections. These collections of documents, topics or queries, and relevance assessments for each of those topics, have been expensive to construct and up until recently have been small in scale. Since the early 1990s the United States Department of Defence have been part-funding a number of competitive evaluations in human language technologies. These include evaluations in speech processing, in information extraction from messages (MUC) and in information retrieval (TREC).

The annual series of TREC conferences [12, 11] have had a significant impact on experimental information retrieval in the United States and elsewhere [27]. While TREC can be credited with helping to spawn innovations in a number of IR areas such as collection merging and data fusion, its chief contribution is to have forced issues of scale on the IR community. IR experimentation is now regularly reported on collections of documents which are gigabytes in size and much of this is due to TREC. We will now look at how documents and queries can be represented and matched internally within an IR system.

3 Representing and Matching Documents and Queries

3.1 Representing Text

Information retrieval is supposed to be about matching a user's query against a set of documents but in practice it is about matching a *representation* for a user's query against the *representations* for documents. It is clearly evident that the form of representation and the way that that representation is achieved, is fundamental to any IR system.

Broadly and very simplistically speaking, language is about real world concepts and relationships between those concepts, albeit modified by some modalities or otherwise as we shall discuss later. The task we are faced with is to turn this into some representation whose combined semantic meaning is equivalent to the content of the text. The most popular representation for text in an IR system is as a set of index terms and while one could spend a long time pointing out the inadequacies of such an approach the simple fact remains that this is almost a *de facto* representation approach used in operational IR.

The identification of a set of index terms to represent texts can be done on several levels as follows:

- we can identify word-level equivalencies between words in the text to be represented and index terms in the representation. Thus we can determine that words like "vibration" and "undulation" in documents in an aeronautics domain all refer to a concept which can be represented by the term "oscillation". More simply however we can determine that each word in a document, with the exception of some non-content bearing stopwords, represents a single concept and we can represent the document by the set of non-stopwords, normalising different word forms as appropriate (see later).
- we can attempt to identify equivalencies at the conceptual level where occurrences of phrases such as "sonographic detection of fetal ureteral obstruction", "obstetric ultrasound" and "ultrasound in child gestation" all refer to the same concept of *prenatal ultrasonic diagnosis* and wherever such concepts occur in documents as represented by such phrases, we always index by the concept of *prenatal ultrasonic diagnosis.*

Ideally, concept level indexing represents documents by phrases rather than individual words and it is semantically rich but it is costly, laborious, specialised and domain-dependent, and it is manual. In between these two extremes of simplicity and complexity there is somewhat of a middle ground, though it does lean towards the more simplistic of the approaches which is unfortunate. The idea of transforming

$$\text{text word} \rightarrow \text{index term}$$

rather than

$$\text{text word} \rightarrow \text{concept} \rightarrow \text{index term}$$

is more commonplace in information retrieval and we shall now look at the most common technique used in practice, stemming, and at a number of linguistically-motivated approaches later.

The English language, like most others, modifies word forms depending on the role played by that word in the text or dialogue in question. This we normally add -S to the end of a noun to turn it from a singular form into a plural noun, we add endings like -S, -ED and -ING to verbs to conjugate them and we add endings like -LY to a noun to turn it into its adverbial form. These simple rules are compounded by a large number of exceptions and there is great overlap between the intended meanings of a large number of word form occurrences. Because of the various ways of modifying a word form in language, matching one form of a word in a user's query against all possible other forms of that same word can create a problem. In IR this is handled by normalising all word forms in both documents and queries into a single form and the most common approach to doing this is word stemming.

Word stemming is a crude pseudo-linguistic process which removes word suffices to reduce words to their word stem. In an ideal language this would always reduce a word form occurrence to a real word base form but because of the large number of exceptions to word modification rules, words are more

often than not reduced to an artificial word. For example, one of the commonly used stemming algorithms is Porter's stemmer [18] for English which turns the ending -IES into the ending -I if the remaining word is long enough and also turns an ending -Y into an -I under the same conditions. This means that word occurrences such as PONY and PONIES are stemmed to PONI which is not a real word in the sense of small horses but in information retrieval it does not matter what the actual word stems mean so long as they are matched. Thus a query for PONIES will match against documents containing PONY, a document on COMPUTING will be matched against a query on COMPUTERS because both will reduce to the artificial

word stem COMPUT, and so on.

Word stemming is a simple process, simple conceptually and simple to implement but despite, or perhaps because of, its simplicity it has found widespread application in IR systems. The true effectiveness of using word stemming has been shown by Harman [10] but more recent work [13] has shown that the properties of even this apparently simple aspect of an IR system have yet to be fully understood.

3.2 Retrieving Text

The function of the retrieval operation in an IR system is to compute some degree of overlap between the representation of a document and the representation of a query, for each document in the corpus, and to rank documents by this Retrieval Status Value (RSV) or score. There are some simple metrics for doing this such as Dice's coefficient, or the Cosine coefficient [29] but these heuristic methods generally do not perform well because they do not account for the relative importance of index terms in the collection.

Much work in IR has been devoted to the development of mathematical models for the IR task, in particular for retrieval. The most important of these have been based on probability theory and on vector space modelling respectively. Such models have been extended to incorporate the most desirable features of retrieval, namely query expansion and relevance feedback. There is a wealth of published literature in this area, going back 20 years, but the essential background points related to conventional retrieval are covered in [19]. This article presents many of the aspects incorporated into contemporary term weighting functions wherein an IR system assigns a weight or degree of importance to index terms in a collection depending on their collection frequency (number of documents in a collection containing a term) and within-document frequency (number of times a term occurs within a given document). The popular $tf \times IDF$ weighting function which combines the two term frequencies above, has

proved to be particularly robust and is in widespread use.

Besides the common model of matching query terms against sets of document terms there are other possible aspects to the retrieval operation listed below.

1. *Cluster-based retrieval* depends on clustering documents into groups of similar documents, usually done *a priori*, and using these clusters as part of

retrieval. Alternatively a user may be presented with documents clusters as a result of a search and may be encouraged to browse through the clusters.

2. *Retrieval as a combination of several retrieval strategies* often involving data fusion approaches; this has found particular favour in the TREC experiments and involves a combination of rankings from more than one document ranking approach into one, consolidated ranking. Data fusion may also involve generating more than one version of the same query to run against a document collection and in this case the individual document rankings are combined [2]. Data fusion has consistently been shown to improve overall retrieval effectiveness when the rankings come from retrieval strategies which are conceptually independent of each other.
3. *Latent semantic indexing* is based on the statistical technique of singular value decomposition where an $m \times n$ matrix is reduced to an $m \times \delta n$ matrix where the transformation of $n \rightarrow \delta n$ is a reduction in the number of index terms for the document collection and δn is of the order of 100 to 300. This dimensionality-reduction technique allows term-term dependencies and relationships to be statistically aggregated and incorporated into the reduced term space and although very computationally expensive, especially for very large collections like TREC [6], has been shown to lead to effective information retrieval [8]
4. IR normally delivers entire documents in response to user queries and on those documents users can make relevance judgements but we have also seen the emergence of *passage retrieval*, where the retrieved items are sections within documents [20]. This is known as passage retrieval and is difficult to evaluate, which is something IR has traditionally liked to do, but appears to be worthwhile for handling long documents.
5. An aspect related to passage retrieval is the problem of applying standard IR techniques to *heterogeneous length documents*. One can simply normalise a document's RSV by its length but this pre-supposes that documents are about topics which are treated equally throughout the length of a (long) document. This is not so as long documents are compositional and the topics covered in a text are treated unevenly and in different document segments. Significant improvements in retrieval effectiveness may be obtained by incorporating document length into the document scoring procedure as shown in [23] and [19].

The above list represents just a snapshot of some of the areas of retrieval research commanding lots of attention in the IR research community and illustrates that this is still a very active area.

4 Natural Language Processing and Information Retrieval

Computational linguistics is the study of computer systems for performing automatic natural language processing and like IR, NLP and computational linguistics has had a long history of evolution. Computational linguistics aims to

develop systems for processing natural language and aims to handle *most* cases of natural language. NLP systems do not mind occasional failures and are more concerned with getting systems working.

The automatic processing of natural language is often divided into a number of levels or strata which represent the levels at which language exists. NLP as a whole is well-described elsewhere in this volume and a re-description here would be inappropriate. Rather than that we choose to present a summary overview of the levels of language processing which are relevant for information retrieval, namely lexical, syntactic and semantic and we follow that with a review of how those levels of language analysis are used in IR systems. Before we proceed it is worth pointing out that this is a very short and simplified view of a large field which has been active for many decades and such a short review must surely be unable to do justice in terms of coverage. What we do present here is the view of NLP from the information retrieval perspective and all its innate biases.

4.1 Brief Overview of NLP

Lexical Level Language Processing. At a lexical level of language processing we seek to identify words and their grammatical classes. We process each word individually and out of context, we handle word morphology meaning different word endings and so on, and we generally make use of a lexicon or dictionary.

In an ideal language[4] where lexical ambiguity did not exist, we would take each word token in text and look it up in a lexicon to determine its base form and grammatical class. In English this is not possible as we have much lexical ambiguity where nouns can act as verbs, where noun plurals are created by appending -S to the word which is the same way in which the third person singular present tense of a regular verb is constructed. Thus when we encounter the words "leaves" or "covers" we do not know whether these are plural nouns or verb forms. When we also take into account the ambiguities introduced by irregular verb and other word forms we have even more ambiguity. For example, "DROVE" is both a noun and verb form.

It is impossible to resolve the many instances of lexical ambiguity by processing at only the lexical level and it requires higher levels of language analysis to do this.

Syntactic Level Language Processing. Traditionally syntax means the structure of a sentence. It means the parts-of-speech and their set of rules acting on them to determine grammaticality, or put simply, the set of rules which determines legitimate sequences of words in a language. Researchers at the syntactic level of language analysis have primarily been concerned with the construction of wide-coverage grammars and the development of efficient parsing strategies. Grammatical formalisms have also been studied in order to try to capture the vagaries of natural language and these have led to the development of phrase

[4] An ideal language for automatic analysis that is !

structure grammars, context-free grammars, context-sensitive grammars, transformational grammars, definite clause grammars, constraint grammars and many more.

Natural language has proved notoriously difficult to capture in its entirety as a set of rules as there are always exceptional sentences or clauses which make the complexity of grammars huge, hence there is no definitve "grammar for English", or any other natural language.

The aim of syntactic processing is to determine the structure of a sentence but that structure itself can be ambiguous and there is that word "ambiguous" again which causes so many of the problems in NLP. The input to the syntactic analysis process (probably) has lexical ambiguities and structural ambiguity can arise within the resultant syntactic structure itself due sometimes but not always to an underlying lexical analysis. For example, the sentence "I saw her duck" can have two structural interpretations parenthesised as either:

(((I saw) her) duck)
((I saw) (her duck))

In the first case we refer to somebody viewing somebody else's pet water fowl whereas in the second case we refer to somebody viewing somebody else who was crouching perhaps to avoid a low-flying object ! This structural ambiguity is caused by the lexical ambiguity of the word "duck" which can act as either a noun or verb form. The sentence "sheep attacks rocket" is similarly ambiguous with "attacks" being either a noun or verb. On the other hand the classic "I recognised the boy with the telescope" is genuine pure structural ambiguity without any underlying lexical ambiguity.

There are many sources of syntactic ambiguity in English but the three most common sources are caused by prepositional phrase attachment, co-ordination and conjunction and noun compounding.

- **PP Attachment:** Prepositional phrases (PP) are a linguistic feature consisting of a preposition followed by a noun phrase, which can be attached to almost any syntactic category, including itself, in order to act as a modifier. Thus we can have PPs modifying noun phrases, verb phrases, adjectival phrases, and so on. The following sentence, although somewhat contrived, has a total of 13 PPs:

  Example 1. "I broke the seal from the fuel pump with the red top to the right of the engine in the car with the dent in the back from a crash on the road to Dublin during the icy spell of weather in 1988".

  In this case, as in all PPs, each prepositional phrase is used to modify some other single construct. So, *fuel pump* is modified by the PP *with the red top*; *car* is modified by *with the dent*, and so on. The problem with using PPs is in finding out to what they should be attached as modifiers. For example, consider the following:

  "Remove the bolt with the square head"
  "Remove the bolt with the square wrench"

Both sentences are identical except for the last words, *head* and *wrench*, both of which can be either a noun or a verb, yet there is a syntactic ambiguity in that we do not know, syntactically, what the PP *with the square XXX* is used to modify. From our semantic understanding of the words in the sentences we know that bolts may have square heads but we do not normally have such things as square wrenches and that the removal operation may be completed with a square wrench but not with a square head and it is this higher level semantic processing that disambiguates the syntactic alternatives for us.

- **Co-ordination and conjunction:** Conjunction or co-ordination is one of the most frequently used constructions in natural language but the scope of the conjunctions, i.e. what is being conjoined, can almost always be ambiguous. For example, we can have conjunction among the heads of a noun phrases as in:

  "Inspect the bearing cups and cones"
  "Inspect the hub and bearing components"

  In the first case it is not clear whether we are to inspect bearing cones as well as bearing cups and in the second case it is not clear whether we are to inspect hub components. Conjunctions can appear in natural language almost anywhere, among modifiers, among PPs, among heads, among clauses, and they are used to make language more concise, but at the cost of increased ambiguity.
- **Noun compounding:** Noun or nominal compounds occur when a noun or nouns are used as a modifier of another noun, making a compound structure as in:

  "computer performance evaluation"

  In this case we have *performance*, a noun, modifies *evaluation*, another noun. *Computer*, another noun, modifies what .. *performance evaluation* or just *performance* ? We genuinely don't know from simple syntactic analysis, hence the ambiguity.

  Another feature of nominal compounding is the ambiguity caused by the kind of relationship might exist between the nouns being compounded [9]. A *fighter plane* is a plane made for fighting, a *garden party* is a party held in a garden and a *timber house* is a house made from timber.

The final problem with syntactic ambiguities is that they are potentially multiplicative rather than additive, so long and complex sentences, as are found normally in technical documentation will be likely to have much ambiguity at this level. On the other hand the advantages of syntactic level processing are that can be reasonably efficient and the rules of syntax are general and concepts like word classes are abstract, meaning that the syntactic analysis process is fairly domain dependent once the word tokens are in the lexicon.

Semantic Level Language Processing. Semantic level language analysis is concerned with context-independent meaning, taking one sentence at a time, independently of its more global context in the text or discourse. It focuses on broad questions such as what type of knowledge representation formalism to use and how to interpret things like

"John only introduced Mary to Sue"

which could actually mean any of the following:

"John did nothing else with respect to Mary"
"John introduced Mary to Sue but to no one else"
"John introduced Mary and no one else to Sue"

Generally, semantic level NLP involves defining a formal language into which natural language can be processed. The earliest attempts at understanding meaning used various forms of logic but more recently artificial intelligence represents knowledge by specifying primitive or simple concepts and then combining or structuring them in some way to define complex, real-life concepts. These can be used to capture permanent universal objects like physical objects and relationships between them but natural language discusses more than concepts and relationships between them. NL involves notions of modality (possibility, necessity), of belief and of time among others and it is necessary or at the very worst case desirable for any semantic representation of language to capture these elements of natural language. This is non-trivial and there is no universally-agreed knowledge representation formalism which does this.

Semantic level language analysis should be able to analyse grammatically parsed text into a knowledge representation format and should also be able to "parse" the semantics of the input, to note and to respond to nonsense or violations of real-world constraints or axioms. The reason for wanting to do this is that a sentence may have a number of semantic interpretations, possibly arising from a number of syntactic interpretations, and we want to eliminate as many of these as possible, especially those that would not make common sense. The sentence

"I noticed a man on the road wearing a hat"

has (at least) two syntactic interpretations with the participal phrase "wearing a hat" modifying either the man or the road. Semantic level interpretation should tell us that hats are word by animate objects such as men and donkeys and not by roads, and thus the latter interpretation should be discarded. In order to do this, however, a huge amount of domain knowledge is needed, even for restricted domain applications, and this is generally not available.

4.2 The Role of NLP in IR

So far in this chapter we have looked at information retrieval and at how it is generally implemented by representing documents as bags of terms and we have

looked at NLP techniques noting the problems of ambiguity at different levels of processing. The conventional approaches to IR based on bags of terms and their statistical distributions will always have inherent limitations and possibilities for text retrieval because of the following:

1. We can have different words used to convey the same meaning as in *throttle* equals *accelerator* when referring to automobiles; a *stomach pain after eating* is a *belly-ache*, and we can have differing perspectives and thus different language used to refer to the same concept as in *the accident* vs. *the unfortunate incident* depending on whether you are for the prosecution or defence in a court case.
2. We can also have the same words used to convey different meanings. A *blind Venetian* is not a *venetian blind* and a *juvenile victim of crime* is not the same as a *victim of juvenile crime*.
3. We can also have a single word or phrase have different meaning depending on the domain; *sharp* may refer to a measure of pain intensity in medicine or the quality of a cutting tool in a gardening handbook.

Restrictions like these which intuitively set an upperbound on the processing capabilities of the contemporary approaches to IR provide a simple motivation and a justification for attempting to use NLP within IR. Large-scale applications of NLP tend to be domain-dependent and this arena of large-scale is the one in which information retrieval systems must operate. Natural language processing, excluding the simple NLP techniques used in spelling error detection and correction [3] also require much coding of knowledge bases, simple but large lexicons in the case of syntactic analysis but much more complex structures in the case of anything more sophisticated. Because this semantic-level processing requires such specialist resources it is clear that we are not going to get fully interactive, domain-independent language processing of large text bases for retrieval but the question must be asked whether we need such processing or not for IR ?

It is believed by many that the problems NLP wrestles with, especially at the semantic level, are not important for information retrieval which as we have seen already has so much vagueness and imprecision inherent. Thus IR has a great tolerance for "noise" in its processing. If a user wants to retrieve some documents about apples or about elephants then an IR system does not need to know what an apple or an elephant is, or what the difference between them might be, it just needs to find areas of its corpus which **might** be about apples or about elephants because in an IR system the decision on relevance is something that is loaded upon the user as a responsibility. In IR we do not need to comprehend or to wrestle with meaning at all, we only need to distinguish texts from each other in the context of a specific query. Perhaps these might be sub-texts and perhaps we have to generate a ranking of texts, but nonetheless information retrieval is unlike information extraction in that it does not do any

more processing on documents other than deliver them to the user. Systems which do more than this are not information retrieval systems.

The lessening of ambition with respect to what IR sets out to do may be seen as a "cop out" but if this is so it is because information retrieval must deliver

operational systems which scale up to large volumes and have rapid response times and it is this over-riding constraint of operation that determines this field. Given that this is so, what can NLP offer to information retrieval. The following are possibilities:

- NLP in document (and query) indexing, perhaps as a way to identify co-ordinated terms or good phrases or something else to be used as content representatives. This leads to an alternative to the "bag of words", namely the "bag of phrases".
- NLP in query formulation where NLP analysis of a user's query dialogue to support information seeking may help a user to more accurately formulate and refine a query.
- NLP in the matching operation where matching a query with a document may incorporate dynamic, on-the-fly NLP analysis of documents, perhaps involving inference on the document content from such an analysis.
- others ?

In practice it is document indexing, and by implication retrieval, which has received the most attention in applying NLP to IR and it raises the fundamental question of what should we replace the bag of words with ? Although for the remainder of this chapter we look at indexing using NLP, the retrieval operation which would have to follow normally defaults to being statistically-based retrieval as the greatest impact of NLP on IR processes has been to try to improve the quality and range of the internal representation of document and query, and the retrieval operation simply follows this.

4.3 Simple NLP as Used for Information Retrieval

Indexing by Base Forms If we process the text in documents (and queries) simply at the word level then we may try to use morphological analysis of word tokens as they occur in text to equate variations of word forms due to pluralisation and different verb forms. This would mean taking a word occurrence and determining the base form of the word from a lexical lookup and some processing of the word form. In theory this sounds attractive but there are a few significant hurdles to this becoming commonplace. In the first case, all potential words in documents and queries must be in the lexicon and building and maintaining this is likely to be expensive and incomplete. Many words in text are proper names, the names of people, places, companies, etc. and identifying these in text is difficult, in fact it is a significant part of text pre-processing for the information extraction application [4]. A second reason while indexing into base forms is not practical is that processing text at the word-only level leaves us with lex

ical ambiguity which as we saw earlier is only resolved with higher levels of language analysis.

The final and most significant reason why representing text by word base forms is unattractive is that for all the effort in lexical lookup and word processing, it can only be slightly better than simple, domain-independent word

stemming. Word stemming is simple and it is crude and it makes mistakes, but it makes the same mistakes for stemming query words as it does for stemming document texts and the resultant incorrect word stems, even though they my not be linguistically accurate, do match against each other thus supporting retrieval. In experiments we have consistently seen that indexing by stemming and by true word base forms are approximately equal in terms of overall retrieval effectiveness [25] so overall it is not worth the computational effort compared to simple word stemming.

Indexing by Word Senses A very active area in information retrieval research in the 1990s has been the application of the recently available machine readable dictionaries (MRDs) to a variety of tasks, including text indexing for information retrieval. For each word an MRD will contain a short textual description of each valid sense of that word and so for polysemous words with more than one semantic meaning there will be more than one entry. For example, the word "ball" will have entries for the senses referring to a solid or hollow sphere, an accoutrement used in games, a missile fired from a cannon, part of the base of the foot, a type of bone joint, a social gathering or assembly, a verb meaning to squeeze material into a ball shape, and so on. It would be clearly desirable to index texts and queries accurately not by words but by the word **senses** intended.

If word sense indexing could be done accurately and with reasonable efficiency then one of the big problems with the "bag of words" approach to information retrieval, word mis-matching, would be solved ! It is thus no surprise to find that using MRDs to perform word sense indexing has attracted so much interest since the increased availability of MRDs over the last few years.

Word sense disambiguation (WSD) is a linguistic problem that has many applications in NLP besides information retrieval and as a problem in its own right attracts researchers. The most common approach to WSD is based on matching each textual sense description from the MRD against the local context or window in which a polysemous word occurs in text. Thus if some of the sense descriptions of the word "ball" include

1. a missile fired from a cannon
2. a joint between bones
3. a social gathering of people
4. a hollow sphere used in playing various sports and games
5. ...others

and we have the following extracts of text in documents

"The General ordered the cannons to be fired
and the cannon balls breached the barracade"
"The centre forward put the ball in the back of
the net and the game was over"
"The ladies dressed for the ball that evening"

In the first example we take the local context for the word "ball", i.e. the words surrounding it and search for occurrences of those words in the sense descriptions from the dictionary. We find the word "cannon" occurring adjacent to the word in question and that same word "cannon" appearing in a sense description. We also find the word "fired" occurring 3 words before the word being disambiguated and also in that same sense description and so deduce that the sense intended is the first one listed.

For the second sentence we find the word "game" occurring 9 words after the word being disambiguated (ball) and also occurring in sense description 4. Finally for the last of the sentences we have no words[5] immediately surrounding the word being disambiguated so we have to resort to a second-level process and look up the sense definitions of all words in the input text surrounding the word "ball" until we eventually get connections between words surrounding the word being disambiguated, and the various sense definitions on offer from the MRD.

These simple canned illustrations illustrate the mechanism behind a simple form of word sense disambiguation as a problem and illustrate the potential it has for information retrieval. If we could correctly index documents (and queries) by word senses then we would never retrieve documents about cannon projectiles when seeking information about a part of the foot nor would we receive documents about social gatherings of people when seeking information about an implement used in playing games. Unfortunately, in practice, automatic word sense disambiguation is a notoriously difficult problem and approaches based on word experts, neural networks, spreading activation and dictionary lookup have all been tried. While the accuracy of a WSD problem is dependent on the number of sense definitions in the MRD, figures of about 70% accurate recognition of the unique word sense among all words in input texts seems to be about the best.

In a series of information retrieval simulations in which he used the concept of a pseudo-word to incorporate a synthetic value for the accuracy of a word sense disambiguator, Sanderson [21] showed that a WSD accuracy of at least 90% is required before the payback of WSD begins to have a positive effect on the effectiveness of subsequent information retrieval. With performance figures less than that the amount of noise introduced by incorrect disambiguation choices brings down the relative performance of retrieval when compared to indexing and retrieval based on simple word stems. A series of subsequent experiments [22] confirmed earlier results as well as exploring a number of other WSD approaches for an information retrieval application.

The current thinking among those who work on using WSD for IR seems to be to use multiple sources of evidence in indexing by word senses by using parts-of-speech, word morphology, etc., and instead of trying to uniquely identify **the** correct sense of a word occurrence to rule out the incorrect senses and to weight the likely senses highly. This is progress in representation when compared to encoding a word occurrence as all senses equally likely which is what indexing by words or word stems, but the idea has to be explored much further.

[5] Excluding stopwords of course !

Another school of thought is not to try WSD into dictionary senses of words which may be archaic and are certainly static, but to dynamically define word senses from within a whole corpus and to index into only those word senses which exist in a corpus. At present this idea is exploratory and requires much work in corpus linguistics but it is intuitively attractive and removes the dependency on machine readable dictionaries.

4.4 Using NLP to Index by Phrases

We now turn our attention to indexing text into units which are larger and more complex than single words, stems or word senses, i.e. phrases. In information retrieval a phrase is a concatenation of two or more word forms and the set of possible phrases for a corpus is larger and richer than the set of word stems or senses. Phrases are generally more content-bearing than their individual components (words) used in isolation and so if we can successfully index text into phrases then we have a richer representation of text and hence the possibility of getting more effective retrieval.

It has been assumed by researchers in IR that in text it is the noun phrases that are the content-bearing elements and certainly noun phrases are more content-bearing than single words but phrases are not a full representation of meaning, just better content indicators than single words and in IR that is all we want.

There are two approaches to phrase identification in text termed "statistical" and "linguistic". Statistical phrase indexing approaches are simple and crude and very common among the IR systems benchmarked in TREC [12]. Typically they pre-process a sample of text in order to determine a phrasal lexicon, effectively a list of phrases that occur with reasonable frequency in the corpus sample. Subsequent indexing of document texts is based on identifying potential phrases on the fly at indexing time and looking up such phrases in this phrasal lexicon to see if the phrase occurs with any reasonable frequency in the domain and if so then the document in question is indexed by that phrase. For example, in [15] we have used a sample text size of 20 Mbytes and if a phrase occurs more than 25 times in this sample then it is entered in the phrasal lexicon. Using these figures our phrasal lexicon is normally around 200,000 to 250,000 entries.

The technique used to identify phrases in text, both at pre-processing time to determine the phrasal lexicon and at document indexing time is normally very straightforward. Typically it involves stemming words and determining phrases to be sequences (pairs or triples) of non-stopwords that do not span sentence or paragraph boundaries, or sequences of capitalised words to identify person, place or company names (named entity recognition [4]).

Statistical phrase indexing has really crept into IR systems in recent years and the true contribution of statistical phrases has been difficult to isolate since systems reported in TREC and elsewhere normally incorporate a large number of techniques besides phrase indexing to improve retrieval effectiveness. In a recent series of experiments the effect of phrase indexing on retrieval performance has been isolated and evaluated and it has been shown that even a simple approach

to statistical phrase identification is a profitable feature to incorporate into an IR system, *without great computational* or resource overheads [15].

NLP techniques have also been incorporated into phrase indexing yielding "linguistic phrases". Normally these are based on a syntactic analysis of document texts and such analysis can be used to determine the boundaries of noun phrases in texts and in queries. The problem with automatically indexing by noun phrases is the variety of ways of representing a concept which is so complex that it needs a noun phrase to represent it. This can lead to the kind of linguistic problems discussed earlier in the section on the role of NLP in IR and in IR we need to represent documents and queries by some normalised derivative of the occurrence form in order to address these variances. For single words the normalisation is normally done by word stemming; for phrases three approaches have been identified previously [24]

1. **Ignore:** This approach allows text to be indexed directly by phrases as they occur in documents and depends on the matching or retrieval to do something about the problems of ambiguity, different ways of expressing the same concept, etc. Generally this can only work if the phrases are limited to 2 words in length and the user is forced to enter variants of their search phrases to ensure coverage.
2. **Normalise the indexing phrases:** Here we index texts by some processed version of sets of words as they have occurred in document texts. The advantage with this approach is that it does yield a smaller vocabulary and makes subsequent retrieval less complex as syntactic variants in texts and in queries will always be normalised to the same form and the retrieval process can default to the techniques used to match single word terms.
   An example of such an approach is the CLARIT system from CLARITECH Corp. [5]. This system operates in a similar way to using statistical phrases in that a phrasal lexicon, called a first order thesaurus, is created by linguistically analysing a sample of the document corpus. This analysis is based on identifying phrases from a linguistic motivation and not just from word adjacency and frequency of co-occurrence. For example, a headnoun and its noun pre-modifier is normally a good candidate for a phrase, as is a headnoun and its modifying adjective. Once the sample text has been pre-processed to generate the phrase list, document indexing can commence by linguistically analysing document texts to identify good candidate phrases. As these are identified they are searched for in the phrasal lexicon and if they exist literally with an exact match then the phrase is used to index the document. If document phrases do not exist in the lexicon but a phrase or phrases in the lexicon are found as const
   ituents of candidate document phrases then the document is indexed by the constituent phrases. Finally, if there are any leftovers then these require special processing. For example if we have the terms *"autonomous robot"* and *"robot navigation"* in the lexicon and we encounter a document with the text:

   "...and the autonomous robot navigation system is designed to ..."

then the text will be indexed by the phrases *"autonomous robot"* and *"robot navigation"* even though a linguistic analysis of the document suggests the phrase *"autonomous robot navigation system"* as a good indexing phrase. The key thing about CLARIT is that it indexes documents only by phrases already existing in the phrasal lexicon and in this way it achieves phrase normalisation.

3. **Index by structures to capture linguistic ambiguities:** The third approach to handling phrasal variation in matching query and document terms is to represent either the document or query phrases as a structured representation from which all semantic interpretations can be derived and to allow the retrieval or matching operation to handle the ambiguity automatically. As reported in [24] this is attractive in theory but in practice has not yielded any fruitful developments in retrieval performance.

An alternative to identifying phrases in text for use as indexing units is taken by Strzalkowsky [28] where a linguistic analysis (parse) of document texts is performed in order to identify pairs of words which are used as indexing units. Conceptually this lies somewhere between indexing by single word forms and indexing by phrases as they occur in texts. The word pairs identified by Strzalkowsky correspond to linguistic dependencies such as a headnoun and its modifier. In indexing by word pairs this approach circumvents the problems of matching phrases as they occur in queries and in documents by indexing by word pairs which are another form of normalised simple phrases. The effectiveness of this approach in experimental evaluation in TREC has been impressive and represents another of the few examples where NLP techniques have been shown to improve document retrieval.

4.5 Using Linguistic Resources in IR

The development of natural language processing has yielded and continues to produce resources which may be of use to information retrieval. We have already seen how machine readable dictionaries can be used in indexing by word senses. The other great resource from NLP, knowledge bases, may also be of use. In terms of large-scale, domain-independent knowledge bases there are really only two possibilities at present, CYC [16] and WordNet [17].

The problem with using CYC at present is that it is still under development and is unlikely to be made available for general use. WordNet on the other hand, is freely available and has been used in IR applications to varying degrees of success, and failure. The principal task that using something like WordNet does is to address the automatic handling of related terms which is an inadequacy of keyword based IR and in the present author's own work we have used WordNet as a basis for computing word-word semantic distances as a basis for IR [26] which is ongoing research. It remains to be seen whether linguistic resources such as thesauri and knowledge bases can be exploited really effectively in information retrieval tasks, something we would expect to happen but which has not been delivered upon yet.

5 Prospects for NLP in IR

In discussing the prospects for NLP in information retrieval tasks we must define what we mean by NLP. If NLP includes such low level processes such as spelling error correction [3] then there is a lot of NLP already in IR but this is all quite straightforward stuff. As we saw, NLP exists at many different levels and if we exclude stemming and word normalisation we find that there is not too much work to report on. The reasons for this used to be because of processing costs and the poor availability of NLP techniques and resources but these reasons no longer appear to be valid. What does seem to be important is that IR and NLP are inherently different processes. IR is inexact and vague NLP has been developed for applications like machine translation and user interfaces where vagueness and imprecision are simply not tolerated. These fundamental differences in approach seem to point to an uncomfortable alliance, notwithstanding the systems mentioned earlier which are exceptions.

If we accept the evidence that current approaches to NLP and to IR are not going to lead to a groundswell in the application of NLP to information retrieval then what are the prospects for the relationship between IR and NLP. Current approaches to what we call natural language processing cannot help to progress what we currently call information retrieval and clearly the "butting of heads" which we see at present with IR attempting to cherry-pick any appropriate techniques from NLP, is not going to have any long-term impact. On the other hand, as we saw earlier, NLP and IR both deal with language and there should be some overlap in what the respective disciplines attempt to do.

It is this author's belief that what we currently call information retrieval systems, which rank documents in response to a set of user's query terms, will be enhanced by the kinds of functionality hinted at earlier in this chapter. Long documents, or even short ones, should be abstracted dynamically in the context of a given user query and these summaries presented to users; a single summary of all top-scored documents should be generated for users to browse; user's should be presented with clusters of related documents, not a single list; query formulation should be a process of dialogue and browsing the information or term space, not just a simple query input dialogue box. All of these techniques will require some elements of NLP and will deliver systems for retrieving information which go much further to satisfying a user's information seeking task than what we currently have. Such systems may use what we currently call information retrieval as a base technology underlying their operation and historic

al inertia may dictate this to be so but these systems will integrate techniques used in IR, information extraction and NLP. How far they go to satisfying our information seeking requirements remains to be seen but for certain they should be an improvement over what we currently call information retrieval.

References

1. Belkin, N.J. and Croft, W.B. : Information Filtering and Information retrieval: Two Sides of the Same Coin ? **Communications of the ACM**, 35(12), 29-38, 1992.

2. Belkin, N.J., Kantor, P., Fox, E.A., and Shaw, J.A. : Combining the Evidence of Multiple Query Representations for Information Retrieval. **Information Processing & Management**, 31(3), 431-448, 1995.
3. Church, K.W. and Rau, L : Commercial Applications of Natural Language Processing **Communications of the ACM**, 38(11), 71-79, 1995.
4. Cunningham, H.: Information Extraction —A User Guide. Department of Computer Science, University of Sheffield Research Memo **CS-97-02**, January 1997.
5. Evans, D.A., Milić-Frayling, Lefferts, R.G. : CLARIT TREC-4 Experiments. in [11].
6. Foltz, P.W. and Dumais, S.T. : Personalised information Delivery: An Analysis of Information Filtering Methods. **Communications of the ACM**, 35(12), 51-60, 1992.
7. Furnas, G.W., Landauer, T.K., Gomez, L.M. and Dumais, S.T. : Analysis of the Potential Performance of Keyword Information systems. **Bell Systems Technical Journal**, 62(6), 1753-1806, 1983.
8. Furnas, G.W., Deerwester, S., Dumais, S.T., Landauer, T.K., Harshman, R.A., Streeter, R.A., and Lochbaum, K.E. : Information Retrieval Using a Singular Value decomposition Model of Latent Semantic Indexing. in Proceedings of the 11th International Conference on Research and Development in Information Retrieval, Grenoble, France, ACM Press, 465-480, 1988.
9. Gay, L.S. and Croft, W.B. : Interpreting Nominal Compounds for Information Retrieval. **Information Processing & Management**, 26(1), 21-38, 1990.
10. Harman, D. : How Effective is Suffixing ? **Journal of the American Society for Information Science**, 42(1), 7-15, 1991.
11. Harman, D.H. (Ed.) : The Fourth Text Retrieval Conference. NIST Special Publication 500-236, 1996.
12. Harman, D.H. (Ed.) : The Fifth Text Retrieval Conference. NIST Special Publication (in press), 1997.
13. Hull, D. : Stemming Algorithms – A Case Study for Detailed Evaluation. **Journal of the American Society for Information Science**, 47(1), 1996.
14. Finding the Right Image: Content-Based Image Retrieval Systems. Special issue of **IEEE Computer**, V.N. Gudivada and V.V. Raghavan (Eds.), 28(9), 1995.
15. Kelledy, F. and Smeaton, A.F. : Phrase Indexing for Information Retrieval. In. **Information retrieval Research, Aberdeen, 1997: Proceedings of the 19th Annual BCS-IRSG Colloquium on IR Research**, London: Springer-Verlag, in press, 1997.
16. Lenat, D.B. : CYC: A Large-Scale Investment in Knowledge Infrastructure. **Communications of the ACM**, 38(11), 33-38, 1995
17. Miller, G.A. : WordNet: A Lexical database for English. **Communications of the ACM**, 38(11), 39-41, 1995
18. Porter, M.F. : An Algorithm for Suffix Stripping. **PROGRAM**, 14, 130-137, 1980.
19. Robertson, S.E. and Sparck Jones, K. : Simple, Proven Approaches to Text Retrieval. Technical Report 356, **University of Cambridge Computer Laboratory**, 1996.
20. Salton, G. : Approaches to Passage retrieval in Full Text information Systems. in: **Proceeedings of the 16th ACM-SIGIR Conference**, Pittsburgh, 1993, 49-58, ACM Press.
21. Sanderson, M. : Word Sense Disambiguation and Information Retrieval. in **Proc. 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval**, Dublin, Ireland, 142-151, Springer-Verlag, 1994.

22. Sanderson, M. : Word Sense Disambiguation and Information Retrieval. PhD thesis, **Department of Computing Science, University of Glasgow**, Scotland, 1997.
23. Singhal, A., Buckley, C. and Mitra, M. : Pivoted Document Length Normalization. in **Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**, Zürich, Switzerland, 21-29, ACM Press, 1996.
24. Smeaton, A.F. : Progress in the Application of NLP to Information Retrieval Tasks. The Computer Journal, 26(3), 268-278, 1992.
25. Smeaton, A.F. TREC-4 Experiments at Dublin City University: Thresholding Posting Lists, Query Expansion with WordNet and POS Tagging of Spanish. in [11].
26. Smeaton, A.F. : Using NLP or NLP Resources for Information Retrieval Tasks. in **Natural Language Information Retrieval** T. Strzalkowski (Ed.), Kluwer Academic Publishers, (in press), 1997.
27. Smeaton, A.F. and Harman, D.H. : TREC and its Impact on Europe ? **Journal of Information Science**, (in press) 1997.
28. Strzalkowski, T. and Carbello, J.P. : Natural Language Information Retrieval: TREC-4 Report. in [11].
29. van Rijsbergen, C.J. : Information Retrieval (2nd Edition). Butterworths, 1979.

Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration

Nicola Guarino
LADSEB-CNR, National Research Council
Corso Stati Uniti 4, I-35127 Padova
guarino@ladseb.pd.cnr.it
http://www.ladseb.pd.cnr.it/infor/Ontology/ontology.html

Abstract. The task of information extraction can be seen as a problem of semantic matching between a user-defined template and a piece of information written in natural language. To this purpose, the ontological assumptions of the template need to be suitably specified, and compared with the ontological implications of the text. So-called "ontologies", consisting of theories of various kinds expressing the meaning of shared vocabularies, begin to be used for this task. This paper addresses the theoretical issues related to the design and use of such ontologies for purposes of information retrieval and extraction. After a discussion on the nature of semantic matching within a model-theoretical framework, we introduce the subject of Formal Ontology, showing how the notions of parthood, integrity, identity, and dependence can be of help in understanding, organizing and formalizing fundamental ontological distinctions. We present then some basic principles for ontology design, and we illustrate a preliminary proposal for a top-level ontology develped according to such principles. As a concrete example of ontology-based information retrieval, we finally report an ongoing experience of use of a large linguistic ontology for the retrieval of object-oriented software components.

1. Introduction

With the dramatic increase of the possibilities of information access via the World-Wide-Web – paralleled by the constant increase of the costs of knowledge acquisition from scratch – the necessity of suitable tools for information organization, extraction, and integration has become more and more evident. In the "global information" perspective, the *added value* of a piece of coded information is no more only bounded to the particular application which motivated its acquisition, but tends to increase in dependence of its *reusability*, i.e. its suitability to be dynamically integrated within various, different bodies of information.

In this situation, the crucial characteristic of a piece of coded information is *what it is about,* i.e. the entities it refers to. It is this referential meaning that needs to be

made explicit and organized, in order the relevant information to be retrieved and used when necessary. It is easy to see how ontological aspects play a fundamental role here, and how the value of an information resource turns to be crucially bound to its *semantic transparency*. This is especially evident in the case of information extraction from text [Cowie and Lehnert 1996], where a user-defined template is to be "filled" by the information conveyed by a natural language statement. Here the problem is the *semantic matching* between the terms used to define the template and those appearing in the text: only if their meaning is clear, then it is possible to decide whether they match or not. In other words, the problem is to make explicit the *intended models* of the vocabulary used to convey and request information, clarifying the ontological assumptions implicit in the terms adopted for concepts, relations, attributes.

The purpose of this paper is to introduce the ontological notions underlying the problem of semantic matching, and discuss the emerging role of *ontologies*, i.e. theories of various kinds expressing the meaning of shared vocabularies, in the specific field of information retrieval and extraction as well as in the more general area of knowledge and language engineering. In the next section I analyze in some detail the nature of *semantic matching* within a model-theoretical framework, and the related notion of *conceptualization*. After a clarification of the meaning of the term "ontology" in the current practice of knowledge engineering (Section 3), I introduce in Section 4 the subject of formal ontology, showing how the basic notions of parthood, integrity, identity, and dependence can be of help in understanding, organizing and formalizing fundamental ontological distinctions. In Section 5 I propose some basic principles for ontology design, while in Sections 7 and 8 I present a preliminary proposal for the top-level ontology of, respectly, *particulars* (entities of the world) and *universals* (relations involving entities of the world). Finally, as a concrete example of ontology-based information retrieval, I report in Section 9 an ongoing experience of use of a large linguistic ontology like SENSUS [Knight and Luk 1994, Swartout *et al.* 1996] for a project of ontology-based retrieval of object-oriented software components.

Most of the material presented here is an adaptation and a systematization, with didactic purposes in mind, of work already appeared in previous papers. References to the relevant original works appear throughout the paper. Sections 6 and 7, presenting a concrete proposal of a top-level ontology, are rather sketchy and compact, partly for the sake of conciseness, but more importantly because they refer to very recent work, introduced in a preliminary form in [Guarino 1997a].

2. Semantic Matching

Just to keep things simple, suppose you want to produce a list of natural language descriptions satisfying a certain property, say Red(X), by automatically extracting them from a body of text. You use a lexical item "red" to describe your template, but, without making clear what its *intended meaning* is, the system can only perform a syntac-

tic matching, without distinguishing between its different meanings (Fig. 1).

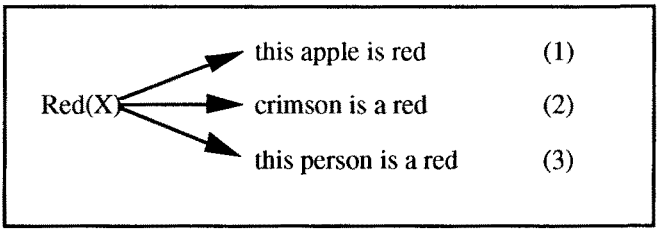


Fig. 1. If the *intended meaning* of the template is not specified, problems of semantic match arise. (From [Guarino *et al.* 1994]).

Let us try to state the problem in more general terms. Consider a logical language **L** using a certain set **V** of predicate symbols, called the *vocabulary* (or the *signature)* of that language. When an agent **A** uses **L** for some purpose, the *intended models* of **L** according to **A** will constitute a small subset of the set **M(L)** of all models of **L** (Fig. 2). We call such set of intended models the *conceptualization* of **V** according to **A**.

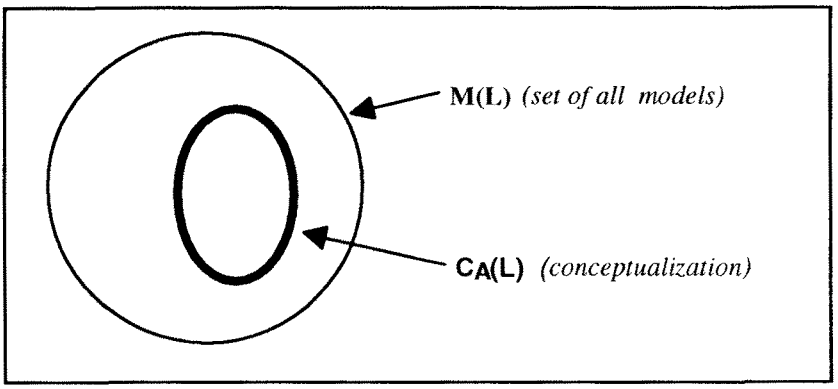


Fig. 2. $C_A(L)$ is the *conceptualization* of the language **L** according to the agent **A**. It is a subset of the set **M(L)** of all models of **L**, representing those models (the *intended models*) which are compatible with the intended meaning of the vocabulary used.

Consider now two different agents, **A** and **B**, using the same language **L**: in order to give the same meaning to the vocabulary used they must either already share the same conceptualization, or otherwise agree on adopting a common conceptualization which is the intersection of the two distinct original conceptualizations (Fig. 3). In the context of information extraction, the process of semantic matching between the template and the data implies this kind of agreement. As we shall see, the key role of ontologies in information extraction is to help establishing this agreement.

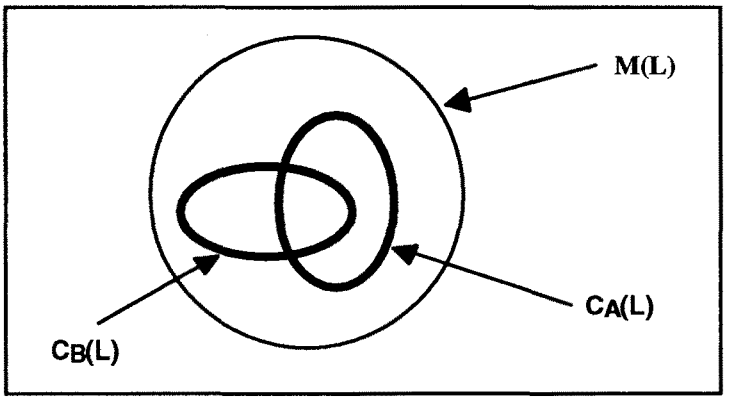


Fig. 3. Two agents **A** and **B** using the same language **L** can communicate only if their conceptualizations $\mathbf{C}_A(\mathbf{L})$ and $\mathbf{C}_B(\mathbf{L})$ overlap.

3. What is an Ontology

But what is exactly *an* ontology? How does it differ from Ontology, with the capital O, intended as a branch of philosophy? Together with Pierdaniele Giaretta, I have addressed this question in [Guarino and Giaretta 1995]; I report here some of the discussion appeared there (also further developed in [Guarino 1997b]).

In the AI literature, an ontology (intended as an engineering artifact, constituted by a particular piece of code) has been defined as a "specification of a conceptualization" [Gruber 1995]. In the model-theoretical framework introduced above, this definition can be interpreted according to Fig. 4 below.

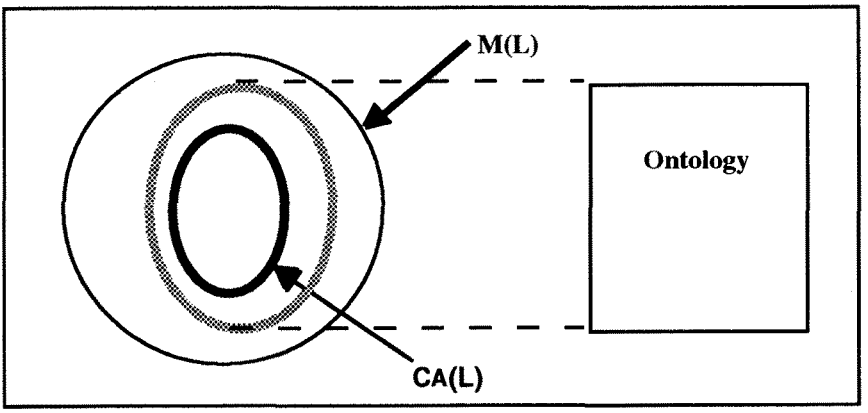


Fig. 4. An ontology characterizes the conceptualization of a language by constraining the intended models of such language

An ontology, according to such an interpretation, is a logical theory whose models *constrain* a particular conceptualization, whithout exactly *specifying* it. In other words, an ontology is an axiomatic characterization of the *meaning* of a logical vocabulary. In many cases, the axioms of an ontology only express subsumption (ISA) relationships between unary predicates, but of course a more detailed axiomatization is often necessary in order to exclude unwanted interpretations.

3.1 What is a conceptualization

The notion of conceptualization I have introduced here requires some further comments, since some confusions about this term exist in the literature. In particular, Gruber's definition of ontology reported above explicitly refers to the notion of "conceptualization" introduced by Genesereth and Nilsson in their well-known textbook on AI [Genesereth and Nilsson 1987]; such a notion is however more akin to that of "state of affairs", while Gruber seems to appeal to the intuitive meaning of the term "conceptualization". Let us consider the example given by Genesereth and Nilsson. They take into account a situation where two piles of blocks are resting on a table (Fig. 5a). According to them, a possible conceptualization of this scene is given by the following structure:

$$<\{a, b, c, d, e\}, \{on, above, clear, table\}>$$

where $\{a, b, c, d, e\}$ is the *universe of discourse*, consisting of the five blocks we are interested in, and $\{on, above, clear, table\}$ is the set of the relevant relations among this blocks, of which the first two, *on* and *above*, are binary and the other two, *clear* and *table*, are unary. The authors make clear that objects and relations are extensional entities. For instance, the *table* relation, which is understood as holding of a block if and only if that block is resting on the table, is just equal to the set $\{c, e\}$. It is exactly such an extensional interpretation that originates our troubles.

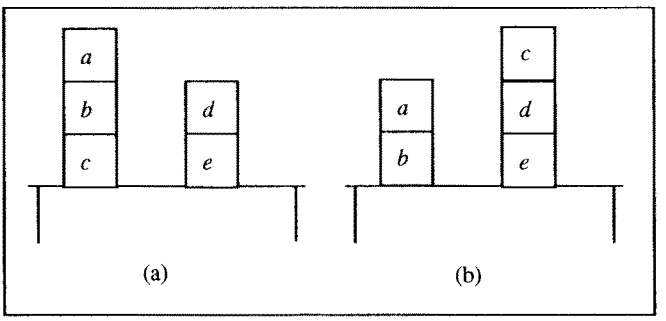


Fig. 5. Blocks on a table. (a) A possible arrangement of blocks. (b) A different arrangement. Also a different conceptualization? (from [Guarino and Giaretta 1995])

Referring to the example given, consider a different arrangement of blocks, where *c* is on the top of *d* and *a* and *b* form a separate stack standing on the table (Fig. 5b). The corresponding structure would be different from the previous one, generating therefore – according to Genesereth and Nilsson – a different "conceptualization". Of course there is nothing wrong in such a view, if one is only interested in isolated snapshots of the world. But the meanings of the terms used to denote the relevant relations are still the same, since they are invariant with respect to the possible configurations of blocks. In fact, in the metalanguage adopted in their book, Genesereth and Nilsson would adopt the same symbols (*on, above, clear, table*) to denote the new conceptualization. We prefer to say in this case that the *states of affairs* are different, but the conceptualization is the same, as it is related to *the way* we assign a particular relation to each symbol in the new situation.

Formally, the meaning of a symbol like *on*, for instance, can be seen as a function f: $W \rightarrow 2^{D2}$, where W is the set of all possible states of affairs and $D=\{a, b, c, d, e\}$ is the universe of discourse. In other words, a conceptualization expresses the intended meaning of each symbol independently of the particular situation at hand: the rules which tell us whether a certain block is *on* another one remain the same, independently of the particular arrangement of the blocks. These rules can be expressed by suitable axioms, like for instance $\neg on(x,x)$, which constrain the intended models in the sense described in Fig. 4.

3.2 Ontologies and knowledge bases

Having made clear that an ontology corresponds to a logical theory, the obvious question arising is the difference between an ontology and a knowledge base. The answer is related to the *purpose* of an ontology, which is a *particular* knowledge base, describing facts assumed to be *always true* by a community of users, in virtue of the agreed-upon meaning of the vocabulary used. A *general* knowledge base, instead, may also describe facts and assertions related to a particular state of affairs or a particular epistemic state.

We can classify ontologies according to two dimensions: their *level of detail* and their *level of dependence* on a particular task or point of view. On the first dimension, a very detailed ontology gets closer to specifying the intended meaning of a vocabulary (and therefore may be used to *establish consensus* about sharing that vocabulary, or a knowledge base which uses that vocabulary), but it usually pays the price of a richer representation language. A very simple ontology, on the other hand, may be developed with particular inference services in mind, in order to be shared among users which *already agree* on the underlying conceptualization. We can distinguish therefore between *reference ontologies* and *shareable ontologies,* or maybe *off-line* and *on-line ontologies.* Very simple ontologies like thesauri can be kept on-line, while sophisticated theories accounting for the meaning of the terms used in a thesaurus can be kept off-line.

On the second dimension, we may distinguish between *top-level ontologies, domain ontologies, task ontologies* and *application ontologies*:

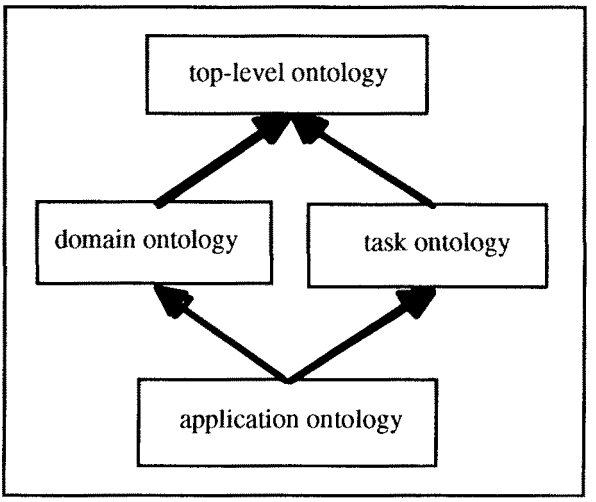


Fig. 6. Kinds of ontologies, according to their level of dependence on a particular task or point of view. Thick arrows represent specialization relationships.

- *Top-level ontologies* describe[1] very general concepts like space, time, matter, object, event, action, etc., which are independent of a particular problem or domain: it seems therefore reasonable, at least in theory, to have a single, unified top-level ontology.
- *Domain ontologies* and *task ontologies* describe, respectively, the vocabulary related to a generic domain (like medicine, or automobiles) or a generic task or activity (like diagnosing or selling), by specializing the terms introduced in the top-level ontology.
- Application ontologies describe concepts depending both on a particular domain and task, which are often specializations of both the related ontologies. These concepts often correspond to *roles* played by domain entities while performing a certain activity, like *replaceable unit* or *spare component*.

The interested reader may refer to [Uschold and Gruninger 1996, Van Heijst *et al.* 1997] for a general introduction on the use of ontologies in the practice of knowledge engineering, and [Guarino 1997b, Van Heijst *et al.* 1997] for an account of the current debate about the role and the nature of task ontologies and application ontologies.

[1] I prefer to use the verb "describe" rather than "define", since it is very rarely possible to completely *define* the meaning of a term.

3.3 Ontologies and KR languages

A further, separate kind of ontology is constituted by what have been called *representation ontologies* [Van Heijst *et al.* 1997]. They are in fact meta-level ontologies, describing a classification of the primitives used by a knowledge representation language (like concepts, attributes, relations...)[2]. An example of a representation ontology is the *Frame Ontology* [Gruber 1993], used to support translations within different knowledge representation languages. A further example is the ontology of meta-level primitives presented in [Guarino *et al.* 1994], which differs from the Frame Ontology in assuming a non-neutral ontological commitment for the representation primitives. Such a position has been further discussed in [Guarino 1994, Guarino 1995], where the notion of *ontological level* for knowledge representation languages has been introduced.

4. The Tools of Formal Ontology

In the current practice, the process of ontology building lacks well-established principles and methodologies. In the past, ontological issues have been rather neglected in computer science and especially in artificial intelligence, where reasoning and problem-solving have been largely emphasized with respect to knowledge representation and conceptual analysis. Some methodological proposals have been made in recent times [Bateman *et al.* 1990, Bouaud *et al.* 1995, Gruber 1995, Mahesh 1996, Uschold and Gruninger 1996], but most of the ontologies currently in use seem to be the result of a mixture of ad-hoc creativity and naive introspection. Recently, however, a different line of research began to emerge, characterized by a highly interdisciplinary perspective: while staying on the solid grounds of logic and computer science, it shows an open-minded aptitude towards the subtle distinctions of philosophy and the slippery issues of natural language and commonsense. The philosophical field inspiring this trend is that of *formal ontology*, which has been defined as "the systematic, formal, axiomatic development of the logic of all forms and modes of being" [Cocchiarella 1991]. As such, formal ontology is a recent expression of traditional ontology, intended as the branch of philosophy whih deals with the *a priori* nature of reality[3]. In its current shape, formal ontology can be seen as the confluence between a school of thought which has addressed metaphysical problems within the mainstream of analytic philosophy, and another school more closely related to phenomenology, in the tradition of Brentano and Husserl. The former school includes a multitude of philosophers,

---

[2] Representation ontologies are therefore special cases of ontologies of universals (Section 7).

[3] The genuine interpretation of the term "formal ontology" is still a matter of debate (see for instance [Poli 1995]): in the Husserlian sense,"formal" has to be intended as opposite to "material", and means therefore "independent of a particular content"; here the adjective is intended as synonymous of both "rigorous" and "a priori", including both "analytic a priori" and "synthetic a priori". For instance, the properties of matter and space would be considered as "material" in the Husserlian sense, while they are included in the scope of the notion of formal ontology adopted here.

which roughly agree on the idea of "descriptive metaphysics" proposed by Strawson [Strawson 1959, Aune 1991]. The latter sees the philosophers of the so-called "school of Manchester" [Smith 1982, Smith and Mulligan 1983, Simons 1987, Mulligan 1992] as its principal defenders. The reader may refer to [Guarino and Poli 1995] for an overview of the role of formal ontology in the information technology. For a general reference on metaphysics and ontology, see [Burkhardt and Smith 1991].

In practice, formal ontology can be seen as the *theory of distinctions*:

- between entities of the real world, or *particulars* (physical objects, events, regions of space, amounts of matter...)
- between the categories we use to model the real world, or *universals* (concepts, properties, qualities, states, relations, roles, parts...)

The study of formal ontological distinctions can be organized around a number of core theories (or, better, theoretical tools), which have always been at the basis of philosophical research. I describe here briefly these theories, which in my opinion may contribute to establish a solid and fruitful ground for ontological analysis. See [Guarino 1997] for further details.

Theory of parts. A theory of parthood is at the basis of any form of ontological analysis. Relevant questions that must be addressed are:

- What counts as a part of a given entity?
- What are the properties of the parthood relation?
- Are there different kinds of parts?

An important example of a theory of parthood is given by *extensional mereology*. Much work must be addressed however in order to come up to a satisfactory theory of *intensional mereology*, where integrity and identity are taken into account. See [Simons 1987] for a thorough reference to the problems of mereology.

Theory of wholes. A given entity (like a collection of disconnected pieces) can have parts without being considered as a single whole. The theory of wholes (or theory of *integrity*) studies the ways of connecting together different parts to form a whole. Relevant questions that must be addressed are:

- What counts as a whole? What does make it a whole?
- In which sense are the parts of whole *connected*? What are the properties of such connection relation?
- How is the whole isolated from the background? What are its boundaries?
- What is the role played by the parts with respect to the whole?

Notice that in order to understand the various forms of part-*whole* relation [Winston *et al.* 1987, Artale *et al.* 1996] the general theory of parthood must be sup-

plemented with a theory of integrity. Together, the two theories form what may be called *mereotopology* [Varzi 1996].

Theory of identity. The theory of identity builds up on the theory of parthood and the theory of wholes, studying the conditions under which an entity can be considered as identical to another one. Relevant questions that must be addressed are:

- How can an entity change while keeping its identity?
- What are its essential properties?
- Under what conditions does an entity loose its identity?
- Does a change of parts affect identity?
- Does a change of "point of view" change the identity conditions?

The last question is especially relevant in our case to distinguish between ontological strata (see below). For instance, consider the classical example of a vase of clay. Should we consider the vase and the clay it is made of as two separate individuals, or just as two different points of view about the same individual? The answer may be difficult, but a careful ontological analysis tells us that the two views imply different identity criteria: when the vase looses its identity by crashing to the floor, the clay is still there. This is because the clay has an *extensional* criterion of identity, since it always coincides with the sum of its parts, while the vase requires a particular arrangement of its parts in order to be a vase. Therefore, we are in presence of two different individuals. A rigorous analysis of this argument requires some care, of course, but the example gives the idea. See [Hirsch 1982] for an account of the identity problems of ordinary objects, and [Noonan 1993] for a review of the philosophical research in this area.

Theory of dependence. The theory of dependence studies the various forms of existential dependence involving specific individuals and the class they belong to. We refer here to the notion of existence as "ontological existence", not as "logical existence". In this sense, existence can be represented by a specific predicate (like in [Hirst 1991]) rather than by a logical quantifier. Relevant questions that must be addressed are:

- Does the existence of an individual necessarily imply the existence of another specific individual? *(Rigid dependence)*
- Does the existence of an individual necessarily imply the existence of some individual belonging to a particular class? *(Generic dependence)*
- Does the belonging of an individual to a particular class imply the belonging of another individual to another class? *(Class dependence)*

An example of rigid dependence may be the relationship between a person and his/her brain, while the relationship between a person and his/her heart is an example of generic dependence (because the heart can be substituted with another heart, and the identity of the person does not change). Finally, an example of class dependence is

the relationship existing between the class "Father" and the class "Child". In our opinion, the notion of dependence is at the core of Peirce's distinction between Firstness, Secondness and Thirdness, proposed in [Sowa 1995]. However, the examples reported by Sowa are far from being clear, mainly because they don't take into account the distinction between particulars and universals (see below).

Theory of universals. According to Aristotle, *particulars* are entities that "cannot be said of anything" (except themselves); they correspond to individuals existing either in the actual or in a possible world. *Universals*, on the other hand, are entities which "can be said of something", usually corresponding to classes and relations. If these classes and relations are *reified*, and taken therefore as elements of a domain, the theory of universals can be seen as a sort of meta-level ontology, which builds on the theory of particulars to introduce useful distinctions among universals.

Notice however that what we usually call class and relations cannot have – in an ontological setting – the standard semantics of sets of tuples, since their *meaning* is supposed to be independent on particular states of affairs (worlds). For these reasons, we give them a Montague-style intensional semantics, as discussed in [Guarino and Giaretta 1995]. A relation of arity n is not just a set of tuples, but rather a function from the set of all possible worlds to the set 2^{D^n}, where D is the domain where the relation is defined.

The following formal properties of classes turn to be of particular relevance for ontological purposes[4]:

- *Countability*: A class is countable if, for all of its instances, none of their proper parts is an instance of the same class [Griffin 1977] ("Person" is countable, "Water" is not)
- *Rigidity*: A class C is rigid if, when C*x* is true, then it must be *always* true ("Person" is rigid, "Student" is not).

5. Some Basic Principles for Ontology Design

After this review of the core theories of Formal Ontology, let us show now how these theories can be exploited in order to establish a methodological foundation for ontology design, and to enlighten the fundamental distinctions that must be recognized in every task involving the analysis and the organization of a body of information. In the field of information retrieval and extraction, the study of these distinctions may help developing less user-dependent ways of organizing information and making explicit the intended meaning of linguistic terms.

---

[4] The definition reported are simplified. See [Guarino *et al.* 1994] for a more detailed account.

A common problem of many current top-level ontologies, like CYC [Lenat and Guha 1990], PENMAN/PANGLOSS [Bateman *et al.* 1990, Knight and Luk 1994] or Mikrokosmos [Mahesh 1996], is that they present a tangled taxonomic structure, so that it is difficult to isolate a basic backbone, a sort of natural skeleton or "conceptual coat rack" [Woods 1986] to be used for cognitive and organizational purposes. In the following, I present a set of principles I have used to isolate a preliminary taxonomy of top-level ontological concepts aimed to conciliate clarity with semantic rigour, generality and commonsense, which *results* to be a tree, although being a tree is not its most important (nor especially desired) property. These principles can be summed up as follows:

1. Two distinct ontologies are assumed for *particulars* and *universals.*
2. Not all the unary relations appearing in the ontology of universals correspond to *explicit* concepts in the top-level ontology of particulars. Such concepts are called *taxons*, and must be chosen according to suitable organization criteria.
3. Taxons correspond to what are called in [Strawson 1959] *sortal* categories, i.e. unary relations which imply a specific identity criterion for their instances (like *physical body* and *event*). Examples of *non-sortal* categories are *decomposable entity* or *red thing* , which do not carry a specific identity criterion.
4. Taxons are all primitive. Formal ontological properties (many of whom are defined) contribute to characterize their meaning, in terms of necessary conditions.
5. Taxons are mainly organized according to their specific identity criteria. Different identity criteria correspond to disjoint taxons. Identity criteria can be grouped in classes, corresponding to a stack of *ontological strata* linked together by a dependence relationship [Borgo *et al.* 1996].

In summary, I argue that a top-level ontology should not explicitly represent as concepts all the relevant properties, but only those corresponding to *sortal* properties; the *non-sortal* properties, which contribute however to characterize the meaning of concepts, can be represented by means of slots and axioms. The result of this choice, together with the application of the principle of stratification according to different kinds of identity criteria, is such that the structure of the top-level becomes considerably simplified, guaranteeing at the same time a rigorous semantics.

6. The Ontology of Particulars

In the following, I shall first introduce what I consider as a basic "backbone" of distinctions among particulars, organized around sortal categories; then I present three possible dimensions of non-sortal categories which can be used to characterize and further specialize the basic backbone. The first of these dimensions introduces a stratification on the basic backbone on the basis of identity criteria, while the other two are based on considerations of integrity and dependence. A preliminar, schematic top-level for the ontology of particulars is reported in Appendix 1.

6.1 Locations, Substrates, Objects, and Qualities

The basic "backbone" of the distinctions I propose is reported in Fig. 7. All these categories are assumed as primitive sortal categories. I give in the following an informal account of their properties, and the relationships among them. A preliminary formal account, limited to the case of space, matter, and physical continuants, appears in [Borgo and Guarino 1997]. Just to visualize the perspective I have in mind, suppose to have a workshop for the production of plasticine artifacts: you have a lump of plasticine on one side, a clean table where to put your artifacts, a side desk with colors and a clock in front of you. We are interested in modelling a "world" which is the table area, where various objects may appear or disappear at different times: for instance, we may decide to destroy an object in order to reuse its plasticine later, or to substitute some of its pieces.

Particular	
Substrate	
Location	
Space	*(a region of the table)*
Time	*(a time clock)*
Matter	*(an amount of plasticine)*
Object	
Concrete object	
Continuant	*(a plasticine artifact)*
Occurrent	*(the movement of an artifact on the table)*
Abstract object	*(Pitagora's theorem)*
Quality	*(the color of a particular piece of plasticine)*

Fig. 7. The basic "backbone" in the ontology of particulars.

Notice that we distinguish the *actual existence* of an object on the table from its *logical existence* as a possible artifact. Its actual existence is a matter of contingency, while its logical existence is constrained by the "laws of nature". This assumption is needed in order to classify objects according to their behavior across possible worlds, avoding at the same time to adopt a modal logical framework. The aim is not only that of keeping things simple from the technical point of view, but also that of offering a graspable notion of the "possibility" dimension, which appears to be unavoidable in any ontological endeavour.

Substrates are the ultimate entities on which objects depend. Their main characteristic is that they have an *extensional identity criterion*. Matter is the classical example of a substrate in the Aristotelian sense. I include here also *locations* like space and time

within this category, since they present similar characteristics for what concerns their identity criteria and their depedence relationships[5].

A *location* is either a region of (absolute) space or an interval of (absolute) time. I do not enter here in the debate between regions and points: we can either assume regions are sets of points or adopt a "pointless" topology thinking of "points" as regular regions of minimal granularity. See [Borgo *et al.* 1996] for a formal account of the latter approach in the domain of space.

Regarding *matter*, notice that I do not assume that an amount of matter has always a spatial location: it acquires *actual existence* when it gets a spatial location in a specific interval of time (i.e. when, in our example, a certain amount of plasticine is taken out from the lump and put on the table). In this way we can quantify over different worlds having a varying total amount of matter.

Concrete objects (or simply "objects", if the context is clear) are intended here in a very general sense, to refer to entities which can be located in space and/or in time and somehow depend on a certain amount of matter, like me and my typing on the computer right now. They are characterized by the way they depend on other objects, and the way they *can* extend in space and time. As in the case of matter, they acquire actual existence (distinguished from logical existence) when they get a location in spacetime.

Within concrete objects, I assume here for granted the classical distinction between *continuants* and *occurrents*. These two terms are less ambiguous than other terms commonly used such "object", "event", "process", or "eventuality".

In order to have actual existence, continuants must get a location in space, but this location can vary with time. They can have spatial parts, but they do not have a temporal location, nor temporal parts: my hand is a part of my body, but my youth is not part of me: it's part of my life. I am not part of my life, but rather I *take part* to my life... In other words, I refuse to admit that continuants have a *location* in time coinciding with their temporal existence, since I want to say that they *fully exist* also in each time interval within their life. All continuants *depend* on matter (which is considered therefore as their *substrate*), since their identity criteria are bound to the ways matter exhibits specific properties (in particular, mereo-topo-morphological properties).

Occurrents are "generated" by continuants, according to the ways they behave in time. Examples of occurrents are the change of location of a piece of matter (an *event* which extends on a given time interval), but also the permanence of a piece of matter in a given location for a given time (a state occurrence, i.e. a *static event*). Occurrents

---

[5] I don't exclude other possible substrates; for instance , it is not clear to me whether abstract objects have a substrate, too (knowledge?).

depend on continuants, but in a way different from the way continuants depend on their substrate, since we can say that matter *constitutes* physical bodies, while the latter *take part* to occurrents whithout constituting them nor being parts of them. Occurrents have temporal parts (they *perdure* in time), while continuants do not (they *endure* in time). Occurrents always have a unique temporal location, while their exact spatial location is less obvious, and it is however bound to the location of the participating continuants. In the following, when mentioning the location of an occurrent, I shall refer to its temporal location. For a thorough review of the philosophical work on occurrents, see [Casati and Varzi 1996].

Abstract objects are included here just for completeness, but are in fact kept outside the current analysis. Notice however that I refer here to abstract *particulars*, such as Pitagora's theorem or (maybe?) the number "1". Most entities usually called abstract objects are however *universals* (see below).

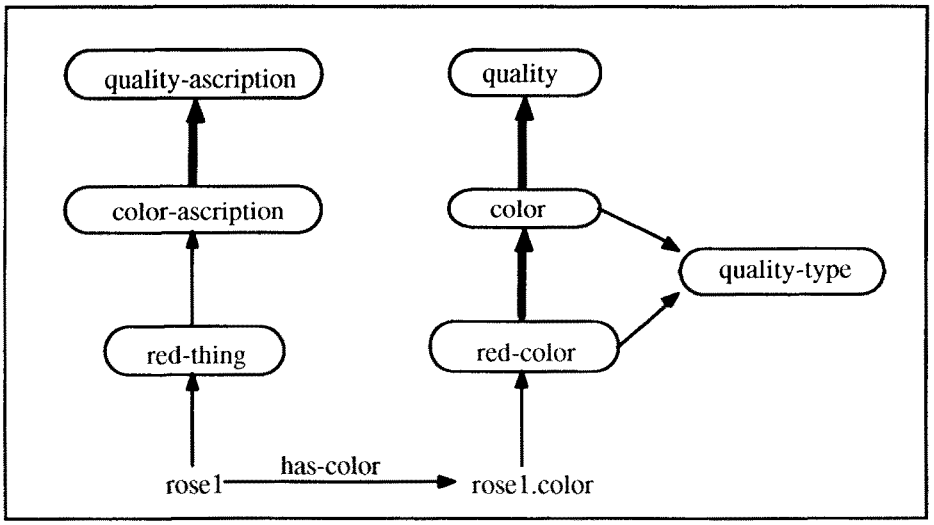


Fig. 8. *rose1.color* is a quality of *rose1*. Both are considered as particulars. *red-color* is a class of qualities having certain common characteristics, while *red-thing* is the class of particulars exhibiting such characteristics. Such a class is an instance of the metaclass *quality-ascription*, while *color* and *red-color* are both instances of the metaclass *quality-type*. Thin unlabelled arcs denote instance-of relationships, while thick arcs subsumption relationships.

Qualities are included here among particulars. I endorse here the position of those philosophers who – following Husserl – admit the ontological existence of "individual qualities" [Mulligan *et al.* 1984]. Therefore the color of a particular rose, the height of a particular mountain, the length of my nose are all unique individuals, not just properties like "being red" or "being 2.5 cm. long". If x is a rose, it is *its* color, not the color of another rose, which is responsible of making the proposition Red(x) true.

Such a color is an individual dependent on the rose itself, but different from it. Briefly, the opportunity of admitting individual qualities comes from:

- the presence of distinct perception stimuli;
- the difficulties of giving a formal account to sentences like "the color of this rose turned from red to purple in two hours";
- the naturality of the individual interpretation for expressions like "the area of New York", "the volume of a body", "the velocity of a body" (intended as an applied vector);
- the difficulties of representing the relationship between specific properties like having a particular speed and "property spaces" like "speed" if this is intended as the set of all such properties, especially in presence of multiple scales of measurement.

Consider for example the case of a particular rose, *rose1* (Fig. 8). Its color (*rose1.color*) can be modeled as an instance of *red-color*, which is a subclass of *color*, which itself is subclass of *quality*. *Rose1* will be then an instance of *red-thing*, which is an instance of the metaclass *color-ascription*, which itself is a subclass of *quality-ascription*. We can explain in this way the relationship between *color*, *red-color* and *red-thing*. Recalling the example of Fig. 1, the meaning of "red" in statement 1 is that of "red-thing", while the meaning of the same term in statement 2 is that of "red-color". Instances of quality types correspond to what have been called *determinable* properties, while instances of quality ascriptions correspond to *determinate* properties [Cleland 1991]. Consider another example of determinable, like "speed". It turns to be just a class of individual qualities, while each set of determinates relative to a particular scale, like {"1 m/s", "2 m/s", ...} forms a *gradient* of mutually exclusive subclasses of "speed". These gradients have been called "quality spaces" in [Lombard 1979]. Under this approach, a *unit of measure* is nothing more that a specific individual quality taken as a reference. A relevant relation between qualities is the *congruence* relation, which allows us to *measure* an individual quality by comparing it with a reference quality.

6.2 Ontological Strata

Let us now introduce a set of *non-sortal* categories useful to establish further distinctions and characterizations in the main backbone described above, on the basis of considerations bound to the theory of identity and the theory of dependence. The idea of *strata* of reality has been inspired by [Poli 1996, Albertazzi 1989]. The notion adopted here, introduced in [Borgo *et al.* 1996], is however more explicitly related to the theory of identity.

I have already underlined that *substrates* have an extensional criterion of identity, in the sense that any two of them are identical if they have the same parts; objects, on the other hand, have an *intensional* criterion of identity, in the sense that they are more that mere sums of parts. Within objects, further distinctions can be made ac-

cording to the identity criteria ascribed to them. For instance, an animal can be conceptualized as an intentional agent, as a biological organism or just as a piece of matter. I argue that different identity criteria correspond to *disjoint* concepts: in the case of an animal, three distinct individuals, corresponding to the three conceptualizations above, share a common spatial location. Since the animal depends on the underlying biological organism, as well as the biological organism depends on the underlying amount of matter, I call these different sub-concepts *ontological strata.*

The proposed classification of ontological strata is reported below (Fig. 9). Strata correspond to different *kinds* of identity criteria (IC) for particulars, distinguished according to the properties used to express them. A *dependence relation* links higher strata to lower strata: an animal depends on its body, which depends on body parts having a certain functionality, which depend on pieces of matter having specific properties, and so on. Notice that this dependence is of a *generic* kind: a vase depends on *some* amount of clay, but not necessarily always the same specific clay (it can loose parts, it can be repaired with a different piece of clay).

Ontological stratum	
Mereological	*(an amount of matter)*
Physical	
Static	*(a state or a state occurrence)*
Topological	*(a piece of matter)*
Morphological	*(a cubic block)*
Functional	*(an artifact)*
Biological	*(a human body)*
Intentional	*(a person or a robot)*
Social	*(a company)*

Fig. 9. Ontological strata describe *disjoint sets* of particulars, according to the different identity criteria adopted to conceptualize them.

At the *mereological stratum,* the IC is extensional: two entities are the same if they have the same parts. As I said, substrates belong to this stratum.

The *physical stratum* corresponds to ICs bound to spatial configuration of matter (i.e., to topo-morphological properties). Notice that these ICs are assumed to be *intensional,* in the sense that mereological identity is not always assumed as a necessary condition for physical identity. The physical stratum can be split into three separate layers:

At the *static layer,* all the non-temporal[6] properties of a particular contribute to its

[6] This means that we exclude properties such as "having occupied a certain location in the past", or "having been part of a certain object in the past".

identity: if one of these changes, identity is lost. In this stratum only very peculiar objects are defined, namely *states* in the case of continuants and *state occurrences* in the case of occurrents. Such objects will play a crucial role in our (not yet fully developed) formal machinery, since they avoid the introduction of modal framework.

A *state* is a a sort of "frozen" continuant having a static IC, corresponding to a specific pattern of (non-temporal) properties holding for a specific amount of matter. For instance, a particular configuration of plasticine (including shape, position, color and any other non-temporal property for each single piece of plasticine) in a certain area of our table is a state.

A*world state* is a state involving a maximal amount of matter. A *world snapshot* (or simply *snapshot*) is the occurrence of a world state in a certain interval of time. A *world evolution* (or simply *world*) is a mereological sum of temporally connected snapshots. Snapshots are the ontological equivalent of temporally indexed possible worlds, and worlds are just temporal sequences of these indexed possible worlds. An occurrent *occurs* in a world if it is part of it. A continuant *is defined* in a world state if its substrate exhibits the required properties for the existence of that continuant in that state. A continuant *exists* in a snapshot if it is defined in a state that occurs in that snapshot. Finally, a continuant *exists* in a world if it exists in a snapshot being part of that world.

After the static layer we have the *topological layer*, characterizing those particulars whose IC is bound to topological properties: at this layer, topological self-connection is a necessary property to maintain identity: a *piece* of matter belongs to the physical stratum (at the topological layer), while a (possibly disconnected) amount of matter belongs to the mereological stratum. The two things are *distinct* entities, since a piece of matter can cease to exist (generating new pieces) while the same amount of matter is still there.

At the *morphological layer*, the IC is bound to morphological properties (or, in general, *gestaltic* properties), like spatial shapes or temporal patterns. A change of this properties can influence identity. A cube-shapedblock is an example of an instance of this level: if its shape changes (above a certain limit) it is not *the same cube* any more, while still being the same piece of matter.

The strata above the physical stratum are related to ICs bound to properties related to the way objects interact with the external world. At the *functional stratum*, the IC is bound to functional and pragmatic properties: identity is destroyed when functionality is destroyed. At the *biological stratum*, the IC is bound to properties related to life: identity is destroyed when biological activity ceases. At the *intentional stratum*, the IC is bound to capability of intentional behavior: identity is destroyed when such capa-

bility ceases. At the *social stratum*, the IC is bound to social rules and conventions involving the interaction of intentional objects. Identity is destroyed when some of these rules change.

6.3 Singular and Plural Objects

Let us now examine some distinctions which only hold among concrete objects. An important one is that hodling between singular and plural objects. I shall assume that *singular objects* are those whose location is topologically self-connected (although not necessarily *maximally* self-connected). For occurrents, this means that they must extend on a continuous stretch of time, while for continuants I assume *strong* self-connection, as discussed in [Borgo *et al.* 1996]. In this way, the mereological sum of two pieces of matter touching in a point is considered as a plural object. Notice that this definition is not related with identity considerations: for instance, a person can be considered as a singular object although its identity criterion is not based (only) on topological properties.

Identity considerations play however an important role in distinguishing among different plural objects: although the simplest example of a plural object is the mereological sum of two disconnected pieces of matter, plural objects are not *just* mereological sums of disconnected singular objects, since such sums may have their own identity criteria, different from those of the objects participating to the sum (see below).

6.4 Bodies and Features

A further distinction among concrete objects is presented in Fig. 10. *Bodies* are objects *constituted by* a self-connected particular. *Constitution* is a primitive relation holding between a stratum and its immediate inferior, which implies dependence. For singular bodies the connection relation is the topological strong connection mentioned above, while for plural bodies *it depends on the stratum or layer where the body belongs*. For example, at the topological layer I shall adopt standard topological connection, in such a way that a lump of coal (whose single pieces are point-connected) will count as a plural body, while at the morphological layer I shall adopt some "gestaltic" notion of connection, in such a way that a constellation still counts as a plural *body*.

Object	
Body	
Whole	*(a person; a lump of coal)*
Piece	*(a hand)*
Feature	*(a hole)*

Fig. 10. Bodies and Features

Features are places marked by spatial changes of the properties of a body. I use this generic term (taken from mechanical design) to include *discontinuities* like boundaries, and *disturbances* like holes, scratches, grooves, ridges, edges, corners, spots of color, or pauses in music performances [Karmo 1977, Simons 1987, Casati and Varzi 1994]. Less obvious examples of features are certain spaces marked by intrinsically oriented objects, like the *underneath* of a table of the *front area* of a house. Differently from pieces, features are dependent on their bodies, which are called "hosts". Features belong to the more general class of *parasitic objects* like shadows [Casati and Varzi 1994], which I do not consider here.

Within bodies, we can further distinguish between *wholes* and *pieces.* A whole is a body such that: i) its constituent is *maximally* self-connected in each state where the body is defined; ii) it is independent on the existence of other bodies belonging to the same stratum. What we call "whole" here captures the important notion of Aristotelian substance, discussed for instance in [Smith 1989]. Differently from Smith, I give here a crucial importance to the quantification on all possible states: looking only at single world states, we should admit that a hand *becomes* a whole as soon as it is detached from the body, while we have good cognitive reasons to assume that the property of being a whole is rigid across all possible worlds. A hand is therefore a *piece* for us, i.e. just a body which is not a whole. A piece becomes a *fragment* when it detached from a whole.

7. Ontology of Universals

Let us now introduce the top-level ontology of universals, limiting ourselves to unary universals for the sake of conciseness (Fig. 11). The basic distinction is between *taxons* (which could be called also *concepts*), corresponding to sortal universals, and *properties*, corresponding to non-sortal universals. In [Guarino *et al.* 1994] we proposed a similar classification, mainly based on countability and rigidity as discriminating properties. The classification discussed below refines such a taxonomy, taking criteria of identity and individual qualities into account.

As discussed in Section 5, the difference between taxons and properties is that the former have an identity criterion associated with them: they are therefore *sortals,* in Strawson's sense. Within taxons, a preliminary distinction is made between *types* and *roles*: the former are rigid, in the sense introduced in Section 4 above and discussed in more detail in [Guarino *et al.* 1994]; the latter are not. Types are rigid classes of particulars endowed with a specific identity criterion; they can be further distinguished on the basis of the ontological properties of their instances, reflected by the top-level structure of particulars described in the previous section. We distinguish therefore between *substrate types,* i.e. classes of substrates, having an *extensional* identity criterion; *object types*, i.e. classes of objects, with an *intensional* identity criterion; and *quality types*, i.e. classes of individual qualities. According to the discussion made in

[Guarino *et al.* 1994], object types are further distinguished into *substantial* and *non-substantial* types, depending on their countability. Non-substantial types have been called "categorial predicates" [Guarino *et al.* 1994], and "super-sortals" in [Pelletier and Schubert 1989].

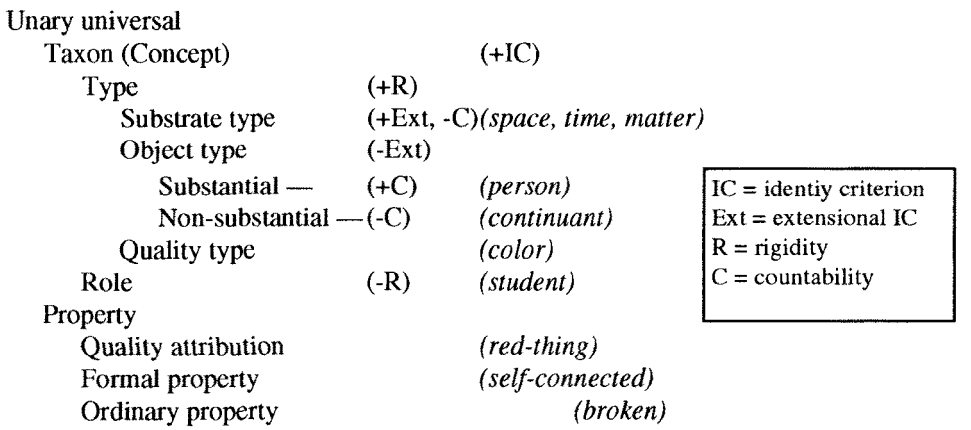


Fig. 11. Top-level ontology of unary universals. The names in italics denote reified classes of particulars, appearing as instances of the metaclasses constituting the nodes of this ontology. Symbols in parentheses refer to the formal properties holding for the instances of the corresponding class.

Those unary universals which are not taxons are called (mere) *properties.* Within these, we distinguish those properties which are the result of a process of *quality attribution*, i.e. those properties which hold for an object in virtue of the fact that some of its individual quality is an instance of a certain quality type. A further class is that of *formal properties* like *atomic, independent* or *self-connected*, which characterize an individual in terms of the formal ontological properties introduced in Section 4. All other unary properties are called *ordinary properties.*

8. The ONTOSEEK project: using a large linguistic ontology for information retrieval

As a concrete example of the use of ontologies for information retrieval, I would like to conclude this paper reporting about OntoSeek, a general purpose tool for ontology-based information retrieval being developed by CORINTO[7] with the cooperation of LADSEB-CNR, as part of a project on retrieval and reuse of object-oriented software components [Borgo *et al.* 1997]. One of the key choices of this project has been avoiding the construction of an ontology from scratch, relying instead on a large ontology built for purposes of natural language translation. The ontology chosen, SENSUS [Knight and Luk 1994] is the result of many years of effort, and amounts to a simple taxonomic structure (no meaning axioms) of about 50,000 nodes, mostly resulting from the merging of the Wordnet thesaurus [Miller 1995] into the PENMAN top-level ontology [Bateman *et al.* 1990]. PENMAN's top-level distinctions are based on the analysis of structural invariants across multiple languages, rather than on theoretical ontological analysis like the one developed in this paper, or on "brute force" empirical effort as in the case of CYC [Guha and Lenat 1990]. The use of SENSUS ontology is an opportunity to perform an experiment of large-scale ontology reuse: its broad covering gives us the possibility to address real life applications with a moderate initial effort, making possible at the same time to verify the ontological validity of language-based distinctions on the basis of *both* concrete examples and theoretical analysis. We aim in this way to contribute to the realization of a more general, robust and well-founded top-level structure, to be used as a common *reference ontology* for multiple purposes of information interchange[8].

8.1 The ONTOSEEK Architecture

OntoSeek adopts a language of limited expressiveness, privileging the simplicity of use as the most important requirement. The system has been designed to:

- encode an information item described by a single NL phrase into a simple graph of concepts and relations tagged by arbitrary English words;
- query a database of previously encoded information tokens by performing a process of ontology-driven *semantic match*;

We adopt a very simple graph structure for representing both queries and information data, but – differently from most of current systems – we do not assume the user

---

[7] COnsorzio di RIcerca Nazionale Tecnologia Oggetti, as a partnership of IBM Semea, Apple Italia and Selfin SpA.

[8] Within the ANSI X3T2 committee on Information Interchange and Interpretation, and *ad-hoc group on Ontology Standard* has been recently formed, with the purpose exploring the feasibility of such a common reference ontology.

to have familiarity with the vocabulary used for information encoding, relying on the ontology – supplemented by lexical information – to perform the match between queries and data.

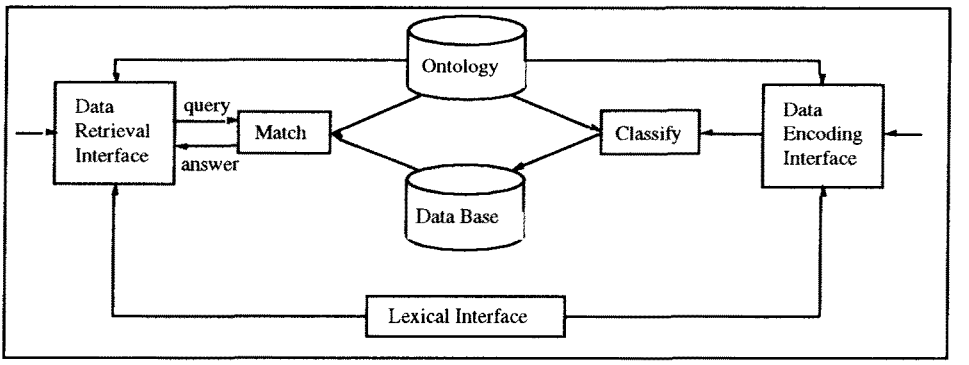


Fig. 12. Basic architecture of *OntoSeek* (from [Borgo *et al.* 1997]).

The basic architecture of the system is sketched in Fig. 12. In the encoding phase (which we assume to be an *interactive* process supported by NL understanding techniques), the input phrase is converted by the *Data Encoding Interface* into a simple graph like the one reported in Fig. 13, where nodes and arcs are labelled with English words. A special semantics is adopted for this graph, which is called *Lexical Conceptual Graphs* (see below). English words appearing in the graph are recognized by a *Lexical Interface* based on WordNet [Miller 1995], which asks the analyst to choose among possibly different senses associated to each word. The graph of *words* is therefore translated into a graph of *senses*, each one corresponding to a node in the *Ontology*. The sense graph is the input of a *Classification* phase, where the graph describing the component is added to the *Database*, suitably linked to the *Ontology* in order to exploit efficient search algorithms. The data retrieval process works roughly in a dual way: the query is represented again as a word graph, which is then refined into a sense graph within the *Data Retrieval Interface*. The database is then searched in order to find the information items described by those sense graphs which are subsumed by the query, according to the taxonomic constraints present in the ontology.

8.2 Lexical Conceptual Graphs

Let us now briefly discuss the semantics of the graph reported in Fig. 13. The basic idea of the whole project is to exploit available lexical resources like WordNet to describe and retrieve information items. Now, most of the labels currently used in modelling formalims to denote binary relations (like "part-of", "function-of", "to-the-right-of"...) do not correspond to lexical entries. In fact, people are usually forced to invent ad-hoc relation labels when using such formalisms. Many of these labels (like "function-of", "has-part"...) are however formed from standard nouns; such nouns are indeed called "relational nouns" since they have a direct relational import, in the sense

that their meaning can be fully understood only in the context of a binary relation: the noun "part" denotes the property of being a part (of something not specified), but also the range of the relation "has-part". This situation has been discussed in [Guarino 1992], where a relation like "has-part" is defined as the *relational interpretation* of the noun "part". According to that paper, only relations of this kind (i.e., relational interpretations of concepts) should be modeled as attributes (also called "roles", or "slots") *of* objects; the remaining relations should be rather modelled as constraints *between* objects.

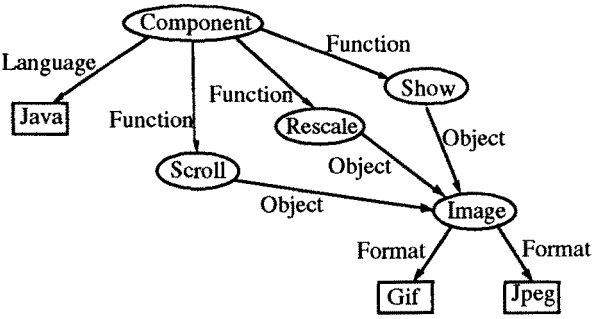


Fig. 13. A Lexical Conceptual Graph representing a software component (from [Borgo *et al.* 1997]).

In the light of this discussion, we can assume that arcs labelled with nouns denote the relational interpretations of such nouns, while arcs labelled with (transitive) verbs denote the relations corresponding to such verbs. In this way we are able to always guarantee a *lexical handle* to interpret the arcs appearing in our graph[9]. More formally, we call *Lexical Conceptual Graph* [10] a simplified form of conceptual graph [Sowa 1984] like the one reported in Fig. 12, where:

1. ellipses denote concepts and rectangles denote individuals;
2. ellipses and arcs are labelled with nouns or verbs[11]; only transitive verbs are admitted for arcs;
3. if a verb appears as a label of an ellipsis, then it denotes the concept corresponding to the nominalization of that verb.

---

[9] It is interesting to note that the opportunity of a linguistic restriction on role names has been advocated in a historical paper by Bill Woods [Woods 1975], who proposed a "linguistic test" which can be paraphrased as follows: A is a good role-name for X if for each filler F we can say that *F is an A of X*. See [Guarino 1992] for a full discussion.

[10] Lexical Conceptual Graphs have been called *Lexical Semantic Graphs* in [Borgo *et al.* 1997]. The present definition is more appropriate, since these graphs are simplified versions of Sowa's conceptual graphs, with a special constraint on their semantics.

[11] Infinitive forms are supposed to be abbreviations for the corresponding nominalizations: "show" denotes the concept of "showing" (whose instances are events).

4. if a transitive verb appears as a label of an arc, it denotes the relation existing between the subject and the object of that verb.
5. if a noun C appears as a label of an arc, it denotes a relation whose range is restricted to the concept C (i.e., the relational interpretation of C). In particular, an arc with label C from the concept (or individual) A to the concept (or individual) B denotes a non-empty relation with domain A (or {A}) and range C ∩ B (or C ∩ {B}) (Fig. 14).

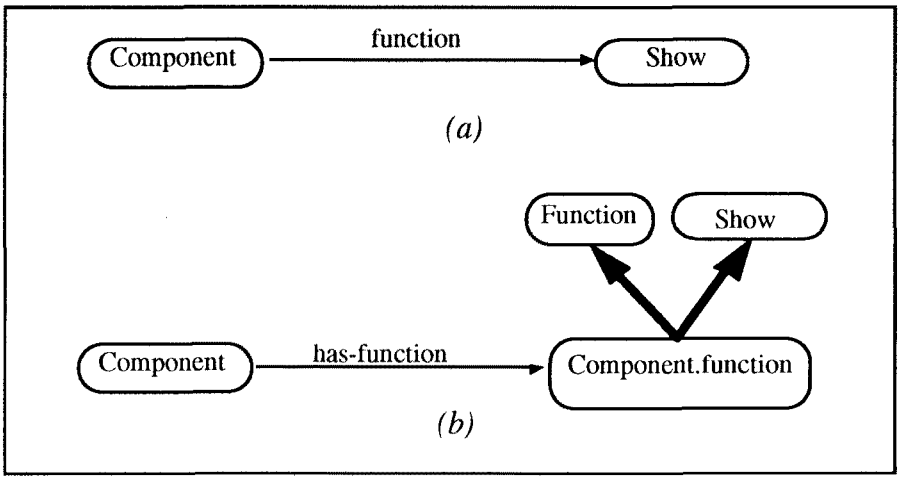


Fig. 14. Semantics of Lexical Conceptual Graphs (LCG). The LCG in (a) is a shorthand for the ordinary semantic graph reported in (b). The advantage is that the relation "has-function" is represented by means of the lexical item "function" (adapted from [Borgo *et al.* 1997]).

8.3 The advantages of lexical-driven information retrieval

Let us now discuss the approach presented above in the light of the current literature, with special regard to systems based on faceted classification schemes. With respect to these approaches, our main points are:

- a simple (but semantically rigorous) representation language of intermediate expressivity (roughly situated between facets and description logics);
- a very large choice of description terms both in the encoding and retrieval phase, due to the use of full English lexicon combined with a large ontology;
- the possibility of semantic checks guided by the ontology;

Regarding facets-based schemes, a well known limitation is related to the fixed number of descriptors (keywords). When we consider a large repository of non-homogeneous data, the choice of such descriptors can be very difficult. Classifying

these data in a satisfactory way implies defining a big set of descriptors, whose exact meaning is difficult to control, document and maintain. It may even happen that the intended meaning of a descriptor varies depending on data types. When dealing with homogeneous data, the importance of this problem decreases, but it does not disappear. Suppose we have a repository of products of similar kind: a few facets may be enough to describe it, but the keywords to be allowed under each facets may be quite many.

In the lexicon-driven approach we have described these problems are overcome. The classification is flexible enough to describe large non-homogeneous repositories, where it is necessary to represent a variety of coarse distinctions, as well as homogeneous repositories, where fine-grained distinctions are important. Moreover, the ontology (supported by the lexical interface) makes possible to clarify the meaning of each term used.

In sum, the real importance of facets-based classification systems lies in the particular structure of keywords they propose, designed to be effective for the information retrieval task. Within our approach we can still use this structure as a *template*, offering however to the user three extra-services: the possibility to *paraphrase* the description, the possibility to *add meaning* to the description, and the possibility to *better understand* the meaning of the words adopted. In this perspective, a lexicon-driven system can be seen as a smooth but powerful improvement with respect to a facets-based system.

Of course, the system we have discussed has many limitations, which need to be taken into due account. As an example taken from our application in the domain of software components, consider an item called "RAM doubler" which does not magically double the physical RAM (unfortunately), but does enrich the performance of a computer. If the software analyst uses the very same words, "Ram" and "double", to classify this software, then a query like "enhancing performance", would never match the above object. Note that even in the case where the item would be described by the expression "enhancing memory", the query could not be satisfied, unless knowing that the performance of a computer can be enhanced by increasing the perfomance of a *part* of it, namely its memory. This kind of problem is faced in each approach based on just natural language descriptions, where no background domain knowledge is available.

Acknowledgements

I hope to have offered a contribution showing the possible advantages of philosophical distinctions in the practice of knowledge engineering, and information retrieval and extraction in particular. Many of these distinctions still present serious philosophical problems, but I believe they can be however very useful for practical applications.

Hopefully, their introduction in engineering systems can stimulate further fundamental research.

This work has been partially funded by a contract between LADSEB-CNR and the Apple/IBM Consortium "CORINTO". I am grateful to the co-authors of the cited papers (Stefano Borgo, Massimiliano Carrara, Pierdaniele Giaretta, Claudio Masolo, Guido Vetere) for their support.

Bibliography

Albertazzi, L. 1989. *Strati.* Riverdito, Trento.

Artale, A., Franconi, E., Guarino, N., and Pazzi, L. 1996. Part-Whole Relations in Object-Centered Systems: an Overview. *Data and Knowledge Engineering*, **20**(3): 347-383.

Aune, B. 1991. Metaphysics of Analytic Philosophy. In H. Burkhardt and B. Smith (eds.), *Handbook of Metaphysics and Ontology*. Philosophia, Munich: 539-543.

Bateman, J. A., Kasper, R. T., Moore, J. D., and Whitney, R. A. 1990. A General Organization of Knowledge for Natural Language Processing: the PENMAN upper model. USC/Information Sciences Institute, Marina del Rey, CA.

Borgo, S. and Guarino, N. 1997. An Ontological Theory of Physical Objects. In *Proceedings of 1997 Workshop on Qualitative Reasoning (QR 97)*. Cortona, Italy.

Borgo, S., Guarino, N., and Masolo, C. 1996. A Pointless Theory of Space based on Strong Connection and Congruence. In *Proceedings of Principles of Knowledge Representation and Reasoning (KR96)*. Boston, MA, Morgan Kaufmann: 220-229.

Borgo, S., Guarino, N., and Masolo, C. 1996. Stratified Ontologies: the Case of Physical Objects. In *Proceedings of ECAI-96 Workshop on Ontological Engineering*. Budapest.

Borgo, S., Guarino, N., Masolo, C., and Vetere, G. 1997. Using a Large Linguistic Ontology for Internet-Based Retrieval of Object-Oriented Components. In *Proceedings of 1997 Conference on Software Engineering and Knowledge Engineering*. Madrid, Knowledge Systems Institute, Snokie, IL, USA.

Bouaud, J., Bachimont, B., Charlet, J., and Zweigenbaum, P. 1995. Methodological Principles for Structuring an "Ontology". In *Proceedings of IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing*. Montreal, Quebec, Canada.

Burkhardt, H. and Smith, B. (eds.) 1991. *Handbook of Metaphysics and Ontology*. Philosophia Verlag, Münich.

Casati, R. and Varzi, A. (eds.) 1996. *Events*. Dartmouth, Aldershots, USA.

Casati, R. and Varzi, A. C. 1994. *Holes and Other Superficialities*. MIT Press/Bradford Books; revised paperback edition 1995., Cambridge (MA) and London (UK). Ed. Italiana: Buchi ed altre superficialita' (Trad. it. di Libero Sosio), Milano, Garzanti 1996.

Cleland, C. 1991. On the Individuation of Events. *Synthese*, **86**: 229-254. Reprinted in R. Casati and A. Varzi (eds.), *Events*, Dartmouth 1996.

Cocchiarella, N. B. 1991. Formal Ontology. In H. Burkhardt and B. Smith (eds.), *Handbook of Metaphysics and Ontology*. Philosophia Verlag, Munich: 640-647.

Cowie, J. and Lehnert, W. 1996. Information Extraction. *Communications of the ACM*, **39**(1): 80-91.

Genesereth, M. R. and Nilsson, N. J. 1987. *Logical Foundation of Artificial Intelligence*. Morgan Kaufmann, Los Altos, California.

Griffin, N. 1977. *Relative Identity*. Oxford University Press, Oxford.

Gruber, T. 1995. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal of Human and Computer Studies*, **43**(5/6): 907-928.

Gruber, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition*, **5**: 199-220.

Guarino, N. 1992. Concepts, Attributes and Arbitrary Relations: Some Linguistic and Ontological Criteria for Structuring Knowledge Bases. *Data & Knowledge Engineering*, **8**: 249-261.

Guarino, N. 1994. The Ontological Level. In R. Casati, B. Smith and G. White (eds.), *Philosophy and the Cognitive Science*. Hölder-Pichler-Tempsky, Vienna: 443-456.

Guarino, N. 1995. Formal Ontology, Conceptual Analysis and Knowledge Representation. *International Journal of Human and Computer Studies*, **43**(5/6): 625-640.

Guarino, N. 1996. Understanding, Building and Using Ontologies. LADSEB Internal Report 6/96.

Guarino, N. 1997a. Some Organizing Principles for a Unified Top-Level Ontology. In *Proceedings of AAAI Spring Symposium on Ontological Engineering*. Stanford, CA, AAAI Press.

Guarino, N. 1997b. Understanding, Building, and Using Ontologies: A Commentary to "Using Explicit Ontologies in KBS Development", by van Heijst, Schreiber, and Wielinga. *International Journal of Human and Computer Studies (in press)*(46): (in press).

Guarino, N., Carrara, M., and Giaretta, P. 1994. Formalizing Ontological Commitment. In *Proceedings of National Conference on Artificial Intelligence (AAAI-94)*. Seattle, Morgan Kaufmann: 560-567.

Guarino, N., Carrara, M., and Giaretta, P. 1994. An Ontology of Meta-Level Categories. In D. J., E. Sandewall and P. Torasso (eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*. Morgan Kaufmann, San Mateo, CA: 270-280.

Guarino, N. and Giaretta, P. 1995. Ontologies and Knowledge Bases: Towards a Terminological Clarification. In N. Mars (ed.) *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing 1995*. IOS Press, Amsterdam: 25-32.

Guarino, N. and Poli, R. (eds.) 1995. *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Special issue of the International Journal of Human and Computer Studies, vol. 43 n. 5/6, Academic Press.

Guha, R. V. and Lenat, D. B. 1990. CYC: a Mid-Term Report. *AI Magazine*, **11**(3): 32-59.

Hirsch, E. 1982. *The Concept of Identity*. Oxford University Press, New York, Oxford.

Hirst, G. 1991. Existence Assumptions in Knowledge Representation. *Artificial Intelligence*, **49**: 199-242.

Karmo, T. 1977. Disturbances. *Analysis*, **37**: 147-148.

Knight, K. and Luk, S. 1994. Building a Large Knowledge Base for Machine Translation. In *Proceedings of American Association of Artificial Intelligence Conference (AAAI-94)*. Seattle, WA.

Lombard, L. B. 1979. Events. *Canadian Journal of Philosophy*, **9**(3). Reprinted in Roberto Casati and Achille Varzi (eds.), Events, Dartmouth 1996.

Mahesh, K. 1996. Ontology Development for Machine Translation: Ideology and Methodology. New Mexico State University, Computing Research Laboratory MCCS-96-292.

Miller, G. A. 1995. WORDNET: A Lexical Database for English. *Communications of ACM*(11): 39-41.

Mulligan, K. (ed.) 1992. *Language, Truth and Ontology*. Kluwer, Dordrecht.

Mulligan, K., Simons, P. M., and Smith, B. 1984. Truth Makers. *Philosophy and Phenomenological Research*, **44**: 287-321.

Noonan, H. (ed.) 1993. *Identity*. Dartmouth, Aldershot, USA.

Pelletier, F. J. and Schubert, L. K. 1989. Mass Expressions. In D. Gabbay and F. Günthner (eds.), *Handbook of Philosophical Logic*. Reidel: 327-405.

Poli, R. 1995. Bimodality of Formal Ontology and Mereology. *International Journal of Human and Machines Studies*, **43**(5/6): 687-696.

Poli, R. 1996. Ontology and Knowledge Organization. In *Proceedings of 4th Conference of the International Society of Knowledge Organization (ISKO 96)*. Washington.

Simons, P. 1987. *Parts: a Study in Ontology*. Clarendon Press, Oxford.

Smith, B. (ed.) 1982. *Parts and Moments: Studies in Logic and Formal Ontology*. Philosophia Verlag, München.

Smith, B. 1989. The Primacy of Place: an Investigation in Brentanian Ontology. *Topoi*, **8**: 43-51.

Smith, B. and Mulligan, K. 1983. Framework for Formal Ontology. *Topoi*, **2**: 73-85.

Sowa, J. 1995. Top-level ontological categories. *International Journal of Human and Computer Studies*, **43**: 669-685.

Strawson, P. F. 1959. *Individuals. An Essay in Descriptive Metaphysics*. Routledge, London and New York.

Swartout, B., Patil, R., Knight, K., and Russ, T. 1996. Toward distributed use of large-scale ontologies. In *Proceedings of 10th Knowledge Acquisition for Knowledge-Based Systems Worskhop*. Banff, Canada.

Uschold, M. and Gruninger, M. 1996. Ontologies: Principles, Methods and Applications. *The Knowledge Engineering Review*, **11**(2): 93-136.

Van Heijst, G., Schreiber, A. T., and Wielinga, B. J. 1997. Roles aren't Classes: a Reply to Nicola Guarino. *International Journal of Human and Computer Studies*(in press).

Van Heijst, G., Schreiber, A. T., and Wielinga, B. J. 1997. Using Explicit Ontologies in KBS Development. *International Journal of Human and Computer Studies*(in press).

Varzi, A. 1996. Parts, Wholes, and Part-Whole Relations: The Prospects of Mereotopology. *Data and Knowledge Engineering*, **20**(3): 259-286.

Winston, M., Chaffin, R., and Herrmann, D. 1987. A Taxonomy of Part-Whole Relations. *Cognitive Science*, **11**: 417-444.

Woods, W. 1986. Important issues in knowledge representation. *Proceedings of the IEEE*, **74**(10): 1322-1334.

Woods, W. A. 1975. What's in a Link: Foundations for Semantic Networks. In D. G. Bobrow and A. M. Collins (eds.), *Representation and Understanding: Studies in Cognitive Science*. Academic Press: 35-82.

Appendix 1. A *preliminary* top-level for the ontology of particulars

NOTE - This ontology is in no sense stable and complete. Formal axioms are intended to characterize the meaning of the concepts used, but they are not reported here due to the preliminary nature of this work.

Notation

1. Each line denotes a node in the taxonomy. Parenthesized expressions in regular style denote alternative names for nodes.
2. Indentation represents subsumption.
3. The symbol — is used to avoid the repetition of the term used at tbe above level.
4. Lines in italics always denote instances of the concept denoted by the above level.
5. The direct children of a node denote mutually exclusive classes. Such classes are also intended as exhaustive if the node "..." does not appear as a child.

Particulars

Substrate
- Location
  - Space
  - Time
- Matter
  - Iron
  - Wood
  - ...

Object
- Concrete object
  - Continuant
    - Singular —
      - Singular body
        - Physical body
          - Topological body
            - Topological whole
              - *(a planet)*
            - Topological piece
              - *(a piece of wood)*
          - Morphological body
            - Morphological whole
              - *(a cube of wood)*
            - Morphological piece
              - *(a mountain)*
          - Functional body
            - Functional whole
              - Artifact
              - ...
            - Functional piece
              - Artifact component
              - Organic structure
              - ...
      - Singular feature
        - Physical feature
          - Topological feature
            - *(the boundary of a blob)*
          - Morphological feature
            - *(the top of a mountain)*
            - *(a hole in a piece of cheese)*
          - Functional feature
            - *(the head of a bolt)*
            - *(the profile of a wing)*

- Plural —
  - Plurality (Plural body)
    - Physical plurality
      - Mereological plurality
      - Topological plurality
        - *(a lump of coal)*
      - Morphological plurality
        - *(a line of trees)*
        - *(a pattern of stars)*
      - Functional plurality
        - *(a disassembled artifact)*
    - Plural feature
      - *(a hole in a lump of coal)*
      - *(the sword of Orion)*
- Occurrent
  - Static occurrent
    - World state
    - ...
  - Topological occurrent
  - Morphological occurrent
  - Functional occurrent
  - Biological occurrent
  - Intentional occurrent
    - Action
    - Mental event
    - ...
  - Social occurrent
    - Communicative event
    - ...
- Abstract object
- **Quality**
  - *(the color of a rose)*
  - *(the style of a talk)*

Machine Learning for Information Extraction

Filippo Neri and Lorenza Saitta

Dipartimento di Informatica, Università di Torino,
Corso Svizzera 185, 10149 Torino (Italy)
e-mail: neri@di.unito.it, saitta@di.unito.it

Abstract. This paper presents a brief overview of the history and trends of Machine Learning, organized according to its goal and principal methodologies. More details are given for the concept learning task, one of the most mature in the field. Applications to Information Extraction tasks are discussed.

1 Introduction

The recent advances in information transmission and the widespread availability of accesses to information networks have exponentially increased the amount of information and data among which a user has to find what really interests him/her. It becomes then necessary to both improve the quality of the information received and to reduce the time devoted to browsing. This is especially true when the information is contained in documents in textual form.

As long as the number of available documents was limited, methods from classical Information Retrieval, based on text's syntactic features, or, most often, co-occurrence of a few words, were sufficient. But, currently, more effective methods, for extracting information from large collections of potentially relevant documents, are needed. In order to be useful, these methods should take into account at least two new aspects: the first one is related to the documents themselves, and concerns the ability to exploit, to some extent, the document's "semantic" content; the second one is related to the user, and concerns the possibility of building up a "user's profile", to be used as an effective "filter" in the selection of truly relevant pieces of information.

Regarding the documents, the construction of intermediate representations (for texts) between words and full text semantics could be of great value to reach the above goal. One way to synthesize information to help focussing selection, is to "categorize" a document according to a set of predefined labels. Given the size of current document collections, it seems very impractical to ask humans to classify them. One possibility to alleviate this burden is to provide only a sample of labelled documents, and to exploit Machine Learning techniques to automatically assign the labels to the remaining ones, as well as to incrementally added new documents.

Categorization can be phrased, in Machine Learning terms, as a learning from examples (or concept learning) problem , which is one of the most well studied task up to now. Learning can be done in both supervised and unsupervised

mode. In the supervised mode, the set of categories is assigned a-priori, and the human provides the learning system with a set of classified training examples. The human expert has also to decide what features must be used to describe each document. The system shall infer a relation between each category and a combination of the given features. This relation can be represented in a variety of ways, ranging from decision trees to neural networks, to sets of production rules.

In unsupervised mode (also called concept formation or conceptual clustering), the categories are not assigned a-priori, and no training set of classified examples is required. The human expert has still to define the document descriptive features, and also to give a criterion of "similarity" or a "distance" measure among documents. The learning system shall automatically group documents, by exploiting their similarity. The difficulty of choosing a good metric, meaningful with respect to the target problem, makes unsupervised learning a more difficult task.

On the user side, Machine Learning can be exploited to build relations between document descriptions and labelling such as "interesting" or "not interesting", based on a previous set of labelled documents provided by the user. It could also be possible to use some domain theory (specific information on the user), to build up such a profile. Moreover, the learning process could be made incremental, requiring the user to continuously add its rating on each document he/she has examined, in order to improve the filtering ability of the profile. Such a learning capability would be of great value in applications such as e-mail screening. On the other hand, it must be said that very high perfomances cannot be expected, given the difficulty of defining suitable features on every kind of incoming mail. However, even a moderate reduction of the number of mails to be read could provide a significant reduction in the user's waste of time.

In order to give to the reader, foreign to the field, an idea of the complexity and variety of the tasks currently within reach of Machine Learning, an overview of the current approaches and methodologies is provided in the Sections 4 and 5. In Section 8, some already existing applications of Machine Learning methods to problems of information extraction (or retrieval), will be briefly illustrated.

2 Machine Learning: Roots and Origin

Machine Learning (ML) denotes the discipline concerned with *learning phenomena in artificial systems*, which is rooted in at least three different fields: Pattern Recognition (and, before, Statistics and Information theory), Recursive Function theory and Philosophy of Science.

In Statistical Pattern Recognition, learning meant essentially to determine parameters in given probability distributions or, more in general, in discriminant functions dividing classes of different objects [1]. In Syntactic Pattern Recognition, acquisition of more complex pattern descriptions have been widely considered, such as grammars, automata or graphs of different complexity [29, 41].

In addition to this common background of shared tasks, Machine Learning has also more fundamental connections with Pattern Recognition. For instance, the Computational Learning Theory [89, 35], took over works by Vapnik [91], and has its methodological foundations, even though not explicitely recognized [78], in a theorem by Devroye [24].

The history of Neural Networks deserves a special mention. Neural nets were introduced, in Pattern Recognition, in the 50's [75], were later abandoned, because of the critics by Papert and Minsky [58], and were then proposed again, with a more complex structure, in the 80's [76].

An aspect that differentiates these *ante litteram* ML approaches from the current view is their stress on numerical representations and the almost exclusive interest in performance as evaluation measure. The aspects of comprehensibility and justification of the learned knowledge were foreign to these studies.

Another way, orthogonal to the previous one, of approaching learning has been that of trying to formally model the learning process with methods of computation theory. The proposed formalizations followed alternative paths. In the recursive-theoretical approach [13, 40] an unknown recursive function f(x) has to be discovered from a set of examples, i.e. pairs <x, f(x)>. Different modalities of presenting the examples define different classes of learnable functions.

A somewhat different, but related, approach has been proposed by Gold [33]: with the "learning in the limit" paradigm, he established a precise learnability criterion and investigated, with respect to it, learnability of functions and grammars of various complexity. Learning is seen as an endless activity, in which more and more data help refining a tentative hypothesis that explains the data provided so far.

Finally, Solomonoff [84] proposed a formalization of learning in terms of Turing computability and Information Theory. According to this view, learning consists in supplying a Turing machine with a stream of data and the machine shall output a string, i.e., a hypothesis, which synthesizes the data. A way of choosing among alternative plausible hypotheses is to evaluate their entropy: shorter (simpler) strings are more probable.

Solomonoff's assumption about the probability distribution over strings is an instance of the "Minimum Description Length" principle [73].

The not truth-preserving nature of inductive hypotheses was already stressed by Hume, and, since then, several attemps to introduce a principled hypothesis ranking, based for instance on probability theory, have been done [17, 26]. One such criterion, related to the Minimum Description Length principle, is to use the notion of simplicity of the hypotheses: Occam's Razor principle states that, *ceteris paribus*, a simpler hypothesis has to be preferred over a more complex one.

3 Historical Synopsis

All the mentioned approaches share the underlying assumption that learning is basically an algorithmic process. A substantial change occurred when people

started to realize that, on the contrary, learning could be fruitfully considered as a search process in a hypothesis space. This change of perspective was the beginning of "modern" Machine Learning. In the first works by Michalski [55], this new view of Machine Learning was implicit. A little later, Mitchell reconsidered the nature of the induction task and explicitely defined it as a search problem [61]. This new view of Machine Learning shows, as well as general AI, a bias toward symbolic representations rather then numerical ones, and toward heuristic approaches rather than "algorithmic" ones. These dimensions separate AI and Cognitive Science from mainstream Computer Science and Pattern Recognition, and Machine Learning is more closely associated with the former two areas then the latter ones.

When working systems started to be built up, the complexity of machine learning tasks clearly emerged. This aspect is stressed in the editorial to the first issue of the *Machine Learning* journal [46]:

> "One can study learning phenomena in such diverse domains as problem solving, natural language understanding, and perception. In addition, one cannot study learning in isolation from work on models of representation and performance, so that integration with traditional areas of AI and cognitive science seems an inherent feature of the field."

The realization of the difficulties and complexities of learning convinced researchers in the field that the only viable way to achieve some concrete results was to give up the idea of formal models in favor of more experimental and computational approaches [47].

With the lack of support from formal models, new criteria for evaluation emerged:

- task specific performance measures;
- understandability for a human user of the system's result;
- comparison of results obtained from different learning algorithms on the same tasks;
- experimental analysis of the effects determined by specific system components, notably the system's parameters;
- ability to handle the presence of noise in the data;
- sensitivity to the order in which learning instances are examined, for incremental learning methods.

In the meantime, the broad scope of learning tasks (supervised concept learning, unsupervised concept formation, acquisition of heuristics for planning and problem solving, scientific discovery), the variety of the application domains, and also a renewed interest in theoretical foundations, triggered the widening of the field. We can mention some of the pursued research issues:

Explanation Based Learning (EBL) [63, 23]. EBL introduces the use of a prior deductive reasoning on a "domain theory" coupled with a subsequent generalization step as a learning method;

Connectionism [76], which takes inspiration from known facts about (human) brain, and is concerned with learning systems whose basic architecture is a network of artificial neurons;

Reinforcement Learning [86, 6], which is concerned with agents that learn to accomplish a task by doing, i.e., by interacting with an external environment from where they receive feedback about their actions;

Classifiers [37] and *Genetic Algorithms* [21]. They consider learning as a search activity, simulating a Darwinian-like evolution, in a space of potential solutions.

Computational Learning Theory (COLT) [89, 36]. COLT is concerned with the theoretical analysis of learning methods and algorithms indipendently of any example distribution. The field of computational learning theory was started by Valiant's seminal paper in which he introduced the Probably Approximately Correct model of learning (PAC-learning). Essentially, Valiant's focus was on how many examples an Inductive Concept Learner needs to be supplied with, in order to achieve a performance rate higher than a given bound, with an assigned probability.

These new research directions generated a diversification of the field along several dimensions:

- *Accuracy vs. efficiency.* The existence of a trade-off between the computational resources required by the task execution and the resulting accuracy clearly emerged, both in human and in artificial learning. Actually, one or the other measure have been used separately, depending on whether the research was concerned more with modeling the learning process (accuracy is more relevant) than the pure acquisition of knowledge to deal with a specific task (efficiency is more relevant).
- *Incremental vs. non incremental learning.* The former approach is concerned with learning something more any time new examples are available, by modifying the previously acquired knowledge. The second approach is concerned with "batch" learning from a set of examples.
  A same method may have different variants for incremental and non incremental learning.
- *Theoretical vs. experimental studies.* Commonly used learning algorithms are very complex from an analytical point of view. COLT studies arose hopes in the possibility to obtain an analytical comprehension of concrete learning algorithms. Unfortunately, current COLT results are still not useful in practice because of the too general assumptions made to prove them. In fact, concrete learning algorithms are more task/domain-oriented, i.e., they make quite specific assumptions about the learning task at hand. ¿From an experimental point of view it turns out that, in average and across domains with "similar features", concrete algorithms behave better than how can be predicted by the theory.
- *Justified vs. Unjustified learning.* The distinction is concerned with the human comprehensibility of the results of a learning system. This distinction may affect the knowledge representation, as in the dichotomy *symbolic vs.*

subsymbolic learning, or the learning strategy, as in the dichotomy *inductive vs. deductive* learning.
The distinction symbolic vs. subsymbolic learning concerns algorithms adopting symbolic representations, such as logical expressions, rather than subsymbolic representations, such as the patterns of weights in neural networks. The distinction inductive vs. deductive learning concerns the use of a purely inductive leap from instances to general rules, rather than the intensive exploitation of the available domain knowledge. The inductive approach is more involved in increasing the learner competence, i.e., in generalizing the learner ability, whereas the deductive approach is more involved in improving the learner's efficiency.

- *Knowledge-intensive vs. knowledge-free learning.* A related issue concerns the "quality" and "amount" of knowledge provided to the learner. Investigation on several related aspects has already started but no general results are available. Typical related problems concern: the effect of increasing the "amount" of knowledge available to the system, the effect produced by the use of relevant, irrelevant or incorrect knowledge, the effect of incremental knowledge acquisition in incremental learner.

4 Machine Learning Today

Current researches in Machine Learning can be grouped into three mainstreams:

- THEORETICAL ANALYSES. The theoretical analysis of learning methods and algorithms indipendently from any specific application domain.
- COGNITIVE SIMULATION OF HUMAN LEARNING. The investigation and simulation of human learning activity.
- EMPIRICAL MACHINE LEARNING. It includes the development and analysis of methodologies in a predetermined set of tasks and the development of computer systems with learning capabilities to solve *real-world* applications.

Each mainstream has motivations and directions determined both by the evolution and ripening of the methodologies themselves, which allow more ambitious goals to be aimed at, and by the needs set forth by new classes of potential users.

4.1 Empirical Machine Learning

We assist to the establishment of the learning paradigms emerged in the past years (Symbolic learning, Genetic Algorithms, Neural Networks, Reinforcement Learning, Case Based Learning [92]). A main issue in representation problems is the interest in Shift of bias and Constructive Induction [88, 3, 4], which aim at dynamically changing the representation language during learning.

Also related to representation issues is the increasing attention to learning in First Order Logic environments [12, 14, 71, 70, 31], ILP [64].

With the increasing number and variety of the proposed systems and approaches, well founded methods for validating the quality of the learned knowledge become necessary. As a consequence, statistical tools, proposed in Pattern Recognition, have been adapted to the novel needs [42].

In the last years, ML applications are rapidly covering new domains. The complexity and the specificity of these domains often require that "specialized" learning approaches are developed to cope with the requirements of the target class of problems. This is the case of Knowledge Discovery in Databases (KDD) [7], learning in robotics [8], learning from text and for Information Retrieval [85], molecular biology [5], and development of softbots working in network environments [62].

4.2 Theoretical Analyses

Even though the COLT approach is still actively investigated, the expected support from it in the analysis of the behaviour of heuristically developed learning systems was not successful. On the other hand, new directions emerged inside the field, which have the characteristics of being at the same time practical and theoretically well founded. In particular, studies about how to make efficient use of multiple independent runs of weak learning algorithms appeared [82, 15, 28]. These works are concerned with the approximation of an unknown target concept arbitrarily well by combining a collection of hypotheses.

Theoretical results have been obtained in several fields, for instance in Reinforcement Learning [11], Neural Networks [38], Genetic Algorithms [68], and Inductive Logic Programming (ILP) [64].

4.3 Models of Human Learning

Recently, there has been a growing interest in joint research with cognitive scientists working on the investigation and simulation of human learning. In particular, this research focuses on the attempt of building computational models that could account for human learning behaviours.

¿From the psychologist's perspective, this would force to precisely specify a theory of human learning. Moreover, the comparison of the performances of the computational model with those of the human learner can help in identifying shortcomings of existing theory and suggesting areas for future research.

¿From the point of view of the ML researcher, the experimental findings on human subjects can call attention to inadequacies of current computational models of learning. For instance, Thau [87] provides experimental evidence that suggests that human learners, unlike some computational models of unsupervised learning, selectively allocate attention to different dimensions during learning. Moreover, human learners are the closest thing to general learning machines that are available for study. Then, insights into how people perform tasks can provide useful starting points for machine learning.

Examples of the current research topics are: comparisons between the symbolic and connessionist paradigms in modeling learning of verbs [54], modeling

human understanding about the balance scale [83], modeling the children conceptual change in physics [67, 93], learning of causal models [69].

5 Machine Learning: a Computational View

There are a number of ways in which learning tasks and approaches can be organized, privileging one or another dimension, according to the goal of the classification. Starting from the viewpoint of a learning program as a component inside a software development cycle, in which cooperation between program and human user plays a fundamental role, we have choosen to classify learning approaches according to the degree of involvement a human expert must have in providing information to the learning system. In this way, learning approaches can be partitioned into *supervised, unsupervised* or *partially supervised* ones:

- Supervised Learning
  - Learning declarative knowledge
    - * Concept Learning
  - Learning procedural knowledge
    - * Learning problem solving operator/heuristics
- Unsupervised Learning
  - * Concept Formation, Scientific Discovery and Theory Formation
- Partially supervised learning
  - * Reinforcement Learning

In the following a more detailed account of the different learning tasks will be given.

5.1 Supervised Learning

Supervised Learning [56] is characterized by the active presence of the knowledge engineer and the domain expert both in the configuration step of the learning process and during the evaluation, when the results produced by the system are analysed.

A widely accepted definition of Machine Learning is "A system is said *to learn* if it can acquire (synthesize) declarative knowledge from data and/or it displays performance/competence improvement through practice". The expression "performance improvement" indicates an increase in the task execution speed, whereas the expression "competence improvement" states an increase in the number of situations that can be successfully handled (i.e., previously unsolvable tasks became solvable).

Most works in Supervised Learning can be grouped into two classes of tasks: *Concept Learning* and *Learning to Plan*. The Concept Learning task consists in learning declarative knowledge useful to recognize positive instances of a given concept from negative ones. The Learning to Plan task consists in learning procedural (control) knowledge to be used in problem solving tasks.

Learning Declarative Knowledge: Concept Learning

A widely accepted characterization of the Concept Learning task as a *search process in a space of generalizations* has been given by Mitchell [61]:

Generalization Problem:
Given:
(1) A language in which to describe instances.
(2) A language in which to describe generalizations.
(3) A matching predicate that matches generalizations to instances.
(4) A set of positive and negative training instances of the target concept.
Determine:
Generalizations within the provided language that are consistent with the presented training instances.

More in general, concept learning can be considered as a search process in a *hypothesis space*. The term *generalization* stresses the internal structuration of the space according to a partial order induced by the *more general than* relation, which Mitchell [61] defines extensionally for a non structured hypothesis language.

Even though the above definition has been proposed with symbolic approaches in mind, it can be immediately extended to any other approach, in particular, to the connessionist one, provided that the representation languages are suitably chosen.

The view of learning as a search process allows alternative search methods to be used, without modification of the other components. This is the case of introducing genetic algorithms for the exploration of the hypothesis space [22, 39, 32, 34, 30, 66, 65].

Later, Mitchell himself [63] (in parallel with deJong [23]) introduced a new perspective in learning: learning is considered as a process that is able to exploit a body of background knowledge of the domain, in order to improve the quality of the learned knowledge. This new view of learning, called Explanation Based Learning (EBL) proved actually to be rather impractical, because of the complexity in encoding and handling a domain theory. Nevertheless, it was very important from the methodological point of view, because it introduced the idea that learning is related to the notion of *explanation*. This link provided the motivation for knowledge intensive approaches, which went beyond the purely inductive approach.

Learning Procedural Knowledge: Learning to Plan

The Learning to Plan task consists in learning plans to deal with problem solving; it requires that a planning system cooperates with a learner, helping it to learn procedural (control) knowledge. More specifically, the goal of learning is to increase the planner's competence and performace through the acquisition of operators, of conditions on when to apply operators, and/or of transformations of operator sequences into general plan to handle special situations.

Learning to Plan is a more difficult task than concept learning, because of the complexity involved in the evaluation of learned macro-operators (plus the associated precondition set). In fact, until a plan reaches a successul/failing state, no precise evaluation of the macro-operator can be given. Also, the macro-operator contribution to reach a successful/failing plan should be determined. The latter problem is also known as the "Credit Assignment" problem (i.e., a reward/blame evaluation has to be backpropagated along the operator chain up to the macro-operator concerned).

Learning procedural knowledge lets an interesting aspect of learning emerge: the *utility problem* [59], which shows that learning in an uncontrolled way may be dangerous. In fact, the number of learned macro-operators can became very large, causing the planning engine to become slower and slower in solving problems, because it becomes busy in searching which macro-plan could be employed in solving efficiently the current problem.

Well known plan learners are Soar [44] and Prodigy [60], which are general problem solving frameworks that exploit learning during the planning activity. More ambitiously, Soar proposes itself as a general cognitive architecture based on the *chunking mechanism*. Prodigy is a complex, multi-strategy learner, in which induction, EBL, and analogy cooperate to achieve the learning goal.

5.2 Unsupervised Learning

Unsupervised Learning [49] is characterized by a very limited human intervention in the learning configuration step. Human activity is in fact reduced to the collection and selection of data.

Unsupervised learning is related to the investigation of cognitive/psychological aspects of learning, specifically the *categorization* problem [74] and the formation of theories.

Conceptual Clustering: Concept Formation

The Conceptual Clustering [57] task is based on the idea that a stand-alone learning system should be able to form informative data clusters and to describe them. To reach this goal, the system tries to minimize (maximize) some intra-cluster (inter-cluster) distance measures [27], a priori defined by a domain expert.

It turned out that these systems could find some meaningful (for a human) clustering of the data only if the measures were considering quite specific structural data information. In other words, no domain-independent and goal-independent measures have been found, then no general clustering system exists. The problem is open even for the cognitive sciences.

Scientific Discovery: Theory Formation

The Scientific Discovery task aims to have a stand-alone learning system able to discover new scientific knowledge. Scientific knowledge may take the format of regularities in the data that, for instance, could help to minimize their description

[73]. The seminal works in the field are Lenat's AM and Eurisko systems [50, 51] and Langley's BACON [48].

The first two systems were concerned with discovering number theory concepts. They viewed the discovering activity as a problem solving process in a theoretical setting. AM applied a predefined set of heuristics to learn, whereas EURISKO's activity was also concerned to enlarge the set of available heuristics. Unfortunately, when the systems were applied to domain different from that of numbers, they were unable to produce useful results.

The BACON system considered the discovering activity as the discovery of mathematical relations holding among raw data. It was shown, for instance, how BACON was able to discover the *Ideal Gas Law* by performing (simulated) experiments on gases, analysing the variations of gas' pressure, temperature, etc., hypothesizing mathematical relations among them, and proposing new experiments to verify the current hypothesis. More recent approaches to Scientific Discovery may be found in [2].

5.3 Reinforcement Learning

Reinforcement Learning [86] comes from the integration of Dynamic Programming and Animal Conditioning researches. The original objectives of Reinforcement Learning were concerned with the learning of control functions and the learning of functions to evaluate actions, including the learning of a world model.

The typical Reinforcement Learning framework contains an agent that learns to accomplish a task by interacting with the world, which returns now and then some scalar evaluation, called reinforce, of the world state the agent is in. The reinforce may be positive (reward) or negative (punishment). The agent's goal is to try to maximize its rewards over a period of time.

Reinforcement Learning is characterized by the same limited human activity of Unsupervised Learning and by the presence of feedback signals during the learning process. Reinforcement Learning is a partially supervised form of learning, because of the presence of the punishment/reward information relative to the currently acquired knowledge.

6 Concept Learning

Concept Learning, which is the ML task best suited for Information Retrieval, includes a variety of different approaches to infer classification rule from a set of data. They include symbolic, subsymbolic/connessionist, statistical, genetic, and case-based paradigms. Even though the uniform superiority of one paradigm over the others cannot be proved [81], according to the type of data, knowledge available, kind of needed results (human readable or not), and user preferences, one paradigm may be more easily exploited than others.

Concept Learning, as many subfields of Machine Learning, seems to have a twofold nature: on one side, it may be considered as a successfull attempt to enhance statistical techniques in terms of readability and ability to handle

symbolic data, but, on the other, it may be viewed as an ambitious investigation about the modeling of human cognitive abilities.

The Concept Learning task hystorically derives from the Classification/Regression task of the Pattern Recognition (PR) area. Goal of this task is to determine a method to discriminate objects belonging to different classes by identifying differentiating pattern in a set of a priori labeled data. If the available classes are a discrete set, the problem is a Classification one. If the values of a continuous function (real numbers), are to be predicted, the problem is a Regression one.

Concept Learning brings substantial novelties in the way it deals with the classification task:

- the ability to handle symbolic (categorical) data and not only numeric ones,
- the idea to express the classification rule by using a symbolic formalism, and
- the extensive use of non-parametric methods.

Finally, Concept Learning enriched of semantics the Pattern Recognition terms: the classes are viewed as concepts, and, then, the potential class descriptions are view as "hypothetic" concept descriptions.

Concept Learning, in its starting phase, is essentially pure induction of general knowledge from specific facts.

As the inductive inference is not truth preserving, the knowledge acquired by concept induction cannot be completely validated, and, in particular, several alternative candidate concept descriptions may comply with the facts available. Consequently, the induction of a concept description needs to select a description in a set of candidate descriptions, according to a criterion that is not supported by the observations.

In practice, the first inductive algorithms, when they found more than one complete and consistent hypothesis, did select one of the alternative descriptions according to some criterion (*bias*) embedded in the algorithm itself. Instance of these biases may are: the first complete (describing all the concept instances) and consistent (not describing any counter-example) description found, the (syntactically) simplest description found, or something else.

These considerations suggest to develop a general framework for the concept learning task, where the *biases* adopted by the learning algorithm would be separated by the inductive algorithm and defined in a declarative way. This would allow a better comprehension of the effect produced by the learning biases.

In this framework, the Concept Learning process is viewed as a search activity for a concept description in a space of symbolic descriptions, the *hypothesis space*. The hypothesis space is defined a-priori, and it constitutes the *language bias*, i.e. it defines the set of the candidate concept descriptions to be used to explain the data. A general search algorithm could then work by progressively reducing the hypothesis space to the set of the complete and consistent hypotheses with respect to the training data. ¿From this set of descriptions, a target concept description is extracted by employing some additional *preference bias*. The Version Space and the *candidate-elimination algorithm* [61] are examples of this philosophy.

Unfortunately, the candidate elimination algorithm is exponential in the number of examples to be considered and in the number of attributes used to describe the instances.

To deal with the computational complexity involved in the hypothesis space exploration, the inductive algorithms should adopt, in the inference, a domain-independent *bias* that results in pruning the explorative paths while searching the hypothesis space. However, after the learning bias problem was elucidated, a greater care in defining and analysing alternative learning biases was taken. The study on the minimum description length or on the syntactic simplicity of the descriptions are an example of this trend.

Pure induction has essentially two kinds of limitations: first, given a set of complete and consistent hypotheses, no additional support could be obtained from the given data in selecting the target concept description. Second, it is difficult, if not impossible, to the exploit the further eventually available information on the domain. In order to overcome the limitations of the pure concept induction, the use of background knowlegde, as an additional bias for the acquisition of knowledge, has been proposed. The basic idea is that if any domain knowledge is available, it may be used to drive the search process by justifying (confirming) an induced description [63]. The use of background knowledge makes possible a more effective exploration of the hypothesis space by determining a more constrained set of alternative, complete and consistent, concept descriptions: the descriptions not supported by the background knowledge are not valid alternatives. Thus, the use of background knowledge is another type of *learning bias* [61, 88] that can be effectively exploited to deal with the complexity of the learning process.

7 Alternative Learning Frameworks

Inductive inference is not the only learning mechanism that has been explored. Deductive learning [63], case-based learning [10], analogical learning [16], and abductive learning [79, 19] are other research topics faced at the same time. They explicitely use background knowlegde in the learning process, with the exception of the case-based learning, where the domain knowledge should be exploited a priori to define a meaningful (with respect to the application at hand) distance.

According to this view, the idea of using the background knowledge expressed in many formats (set of production rules, mechanichal models, and so on) to successfully constrain the hypothesis space search, has been exploited in the multistrategy learning framework. In this framework, several inferential techniques contribute to the exploitation of available domain knowledge [79].

8 Machine Learning and Information Extraction

In this section, some examples of Machine Learning applications to problems of Information Extraction (or Retrieval) are briefly summarized. The approched

problems include text categorization and information filtering, as well as related problems.

In order to improve tha quality of text retrieval, more complex information than simple statistics on two or three words co-occurrence has to be extracted from the text. This is made possible by the advances in Natural Language Processing methodologies, which allow more or less "deep" knowledge to be extracted from a text without requiring a complete and, hence, time consuming syntactic and semantic analysis.

Statistical word-based techniques have obviuos limitations, but, on the other hand, they are fully automatizable, easy to implement, and they run fast on large collections of documents. On the other hand, knowledge-based approaches require large knowledge engineering efforts to be effective.

Riloff and Lehnert [72] present three algorithms, of increasing cmplexity, which exploit information extraction from texts to improve the precision of text classification. The first one, the relevancy signatures algorithm, uses linguistic phrases; the second one, the augmented relevancy signatures algorithm, uses phrases and local contexts, whereas the third one, the case-based text classification algorithm, exploits larger contexts. Both phrases and contexts are acquired automatically from a training corpus. The authors report significant improvements on test sets from the MUC-4 corpus.

Semantic features attached to words are also used in [53] to categorize texts for multiple users, by evaluating the similarity of each analysed document with respect to a user profile. A more direct application of Machine Learning to text categorization is provided by Cohen [18], who applied the system FOIL and showed that, using this representation language, classification rules with a lower error rate and higher degrees of precision and recall than with a propositional language are obtained. He also shows that further improvements can be achieved by a process of relation selection, an analog to feature selection.

Text categorization is a problem characterized by the presence of many features (for instance all the words occurring in the texts), but only few of these are set to true in any given example. Moreover, the negative examples for each class (category) largely outnumber the positive ones. Finally, as categories may not be exclusive, Cohen argues in favor of considering a set of binary classification problems, by considering each category in turn, rather than a single multiclass classification. As usual, evaluation parameters are precision and recall.

As experimental test bed, he uses the ten problems described in [52]. In each of these problems, the goal is to assign AP newswire headlines as relevant or irrelevant to a give topic. Each headline has an average of 9 words, for a total of 67,331 words, deriving from a corpus of 371,454 documents, partitioned into a set of 319,463 training samples and a set of 51,991 test samples. Due to its large size, the corpus has been split into ten smaller subsamples, each containing 999 examples, according to the technique of uncertainty sampling [52].

For learning, each document has been described in terms of a set of predicates specifying the exact position of each word in the document, and of others asserting neiborghhood relations between pairs of words. By running FOIL [71] and

a propositional equivalent of it, comparable results are obtained. The average (over the problems) recall rate was 0.465 for propositional and 0.470 for relational encoding, respectively; the precision was 0.302 and 0.354 for propositional and relational encoding, respectively. The above figures have been obtained by performing document retrieval on the basis of the assigned categories.

A new set of experiments was then performed, by trying to reduce the set of features used, in particular, by discarding words whose number of occurrences in the training corpus was less than k. The results were that, for low values of k (k=3), both recall and precision improved, whereas for larger values of k, precision improved at the expense of recall.

Another approach to automatic text categorization has been presented in [77], based on the Multiple Cause Mixture Model (MCMM). The method is unsupervised, i.e., it does not require to have atraining set of pre-labeled examples, and can assign multiple categories to a document.

Each document is represented as a J-dimensional binary vector, each entry of which asserts the occurrence/non occurrence of a corresponding word in the document. The basic idea is that the "topic" of a document is a cause that forces groups of words to co-occur in it. Then, clusters of words become indicative of a topic.

The method has been tested on the Reuters document collection, by extracting from it two subsets, namely DS1 containing 983 documents with 9 labels and a vector of 372 words, and DS2 containing 240 articles with 3 labels and a vector of 360 words. The method has been used both as a supervised and as a unsupervised classifier, obtaining acceptable results, even though it required large computational resources.

In a related work Feldman and Dagan [25] propose using a text categorization paradigm to annotate text articles with hierarchical concepts, allowing a structure to be introduced in texts, so as to facilitate discovery of interesting patterns, data summarization and trend analysis.

One of the challenge that text categorization sets on Machine Learning resides in the possibly very large number of categories. In fact, many learning systems have difficulty in facing multiple concept learning, especially when these concepts are many and also possibly overlapping. In fact, most often, more than one category has to be assigned to the same document.

Lang [45] proposes a method for automatically building up a profile of a net user, to be used later to filter his/her interesting news. The user him/herself must rate on a scale (1-5) each read article, and this rating is used by a learning system, NewsWeeder, to infer the user's preferences, exploiting the Minimum Description Length (MDL). First of all, the text is parsed into tokens, and, then a vector of (unstemmed) token counts is associated to a document. Learning with NewsWeeder has been compared to the Term- Frequency/Inverse-Document-Frequency (tf-idf) weighting method [80], embedded in the Stanford Netnews Filtering Service [94], obtaining large improvements over this last.

In [9] the problem of learning co-occurrence of two word categories, for instance verbs and nouns, is handled. The presented framework is based on an

MDL estimation of all the probability distributions involved, and is applied to the problem of acquiring case-frame patterns, important for many natural language processing tasks. The method has been trained on texts from the tagged Wall Street Journal corpus, containing 126,084 sentences, and the considered associations were triples of the form (verb, case-slot, noun) or (noun, case-slot, noun). The test set was taken from the Penn Tree Bank corpus, and the task was that of disambiguation of sentences.

A multiple layered database approach is suggested in [95] to handle the resource and knowledge discovery in global information bases. In particular, information retrieval, data mining and data analysis are used to extract and transform information from a lower level (storing primitive information) to a higher one (storing summaries of the information) in a database. As an experimental test bed the Unified Computer Science Technical Reports Index (UCSTRI) [90] has been used.

Kohonen's self-organizing maps [43], are proposed, in the system WEBSOM, for automatic organization of full- text document collections. Based on short word contexts, documents cluster themselves, in an unsupervised manner, in such a way that related documents lie in nearby regions of the document space. They can then be browsed and visualized, by means of a suitable interface.

9 Conclusions

According to [20] two are currently the problems most well suited to be approached with Machine Learning techniques, namely routing (and information filtering) and categorization, which both have available large training sets. Routing involves building profiles, based on selected and weighted features, using large sets of relevant and non-relevant documents. Categorization, as mentioned before, involves assigning to text documents categories chosen from a predefined set. Large databases, previously labeled, are typically available for training. Machine Learning methods that have been used include decision tree construction, Bayesian classifiers and rule learning.

References

1. In J. Kittler, K. S. Fu, and L. F. Pau, editors, *Pattern Recognition Theory and Applications*. Reidel Publ. Co., Boston, MA, 1982.
2. In J. M. Zytkow, editor, *Machine Learning (Special Issue on Machine Discovery)*, volume 12. 1993.
3. In K. Morik, F. Bergadano, and W. Buntine, editors, *Machine Learning (Special issue on Evaluating and Changing Representation)*, volume 14. 1994.
4. In M. desJardins and D. F. Gordon, editors, *Machine Learning (Special issue on Bias Evaluation and Selection)*, volume 20. 1995.
5. In J. Shavlik, L. Hunter, and D. Searls, editors, *Machine Learning (Special Issue on applications in Molecular Biology)*, volume 21. 1995.
6. In L. Kaelbling, editor, *Machine Learning (Special Issue on Reinforcement Learning)*, volume 22. 1996.

7. In E. Simoudis, J. Han J., and U. Fayyad, editors, *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining.* AAAI Press, Menlo PArk, CA, 1996.
8. In J. A. Franklin, T. M. Mitchell, and S. Thrun, editors, *Machine Learning (Special Issue on Robot Learning)*, volume 23. 1996.
9. N. Abe and H. Li. Learning word association norms using tree cut pair models. In *Proceedings of the 13th Conference on Machine Learning*, pages 3–11, Bari, Italy, 1996. Morgan Kaufman.
10. D. W. Aha and D. Kibler. Noise-tolerant instance-based learning algorithms. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 794–799, Detroit, MI, 1989.
11. L. Baird. Residual algorithms: Reiforcement learning with function approximation. In *12th International Conference on Machine Learning*, pages 30–37, Lake Tahoe, CA, 1995.
12. F. Bergadano, A. Giordana, and L. Saitta. Learning concepts in noisy environment. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-10:555–578, 1988.
13. M. Blum and L. Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
14. M. Botta and A. Giordana. SMART+: A multi-strategy learning tool. In *IJCAI-93, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 937–943, Chambéry, France, 1993.
15. L. Breiman. Stacked regression. *Machine Learning*, 24(1):49–64, 1996.
16. J.G. Carbonell. Learning by analogy: formulating and generalizing plans from past experience. In J.G. Carbonell, R.S. Michalski, and T. Mitchell, editors, *Machine Learning, an Artificial Intelligence Approach*, pages 137–161. Morgan Kaufmann, 1983.
17. L. J. Cohen. Inductive logic 1945-1977. In E. Agazzi, editor, *Modern Logic.* D. Reidel Publ. Co., 1980.
18. W. Cohen. Text categorization and relational learning. In *12th International Conference on Machine Learning*, pages 124–132, Lake Tahoe, CA, 1995.
19. W. W. Cohen. Incremental abductive explanation based learning. *Machine Learning*, 15:5–24, 1993.
20. B. Croft. Machine learning and information retrieval. In *12th International Conference on Machine Learning*, pages 587–587, Lake Tahoe, CA, 1995.
21. K. A. De Jong. *Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* PhD thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI, 1975.
22. K. A. De Jong, W. M. Spears, and F. D. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 13:161–188, 1993.
23. G. F. DeJong and R. J. Mooney. Explanation based generalization: an alternative view. *Machine Learning*, 1:145–176, 1986.
24. L. Devroye. Any discrimination rule can have an arbitrarily bad probability of error for finite sample size. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-2:154–157, 1982.
25. R. Feldman and I. Dagan. Knowledge discovery in textual databases (kdt). In *Proceedings of the First International Conference on Knowlede Discovery and Data Mining*, pages 112–117, Montreal, Quebec, 1995. AAAI Press.
26. T. Fine. *Theories of Probability: an examination of foundations.* Academic Press, New York, NY, 1974.

27. D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.
28. Y. Freund and R. E. Schapire. A decision-theorethic generalization of on-line learning and an application to boosting. In *Second European Conference on Computational Learning Theory*, pages 23–37. Springer-Verlag, 1995.
29. K. S. Fu. *Syntactic Pattern Recognition.* Academic Press, New York, NY, 1974.
30. A. Giordana and F. Neri. Search-intensive concept induction. *Evolutionary Computation*, 3 (4):375–416, 1995.
31. A. Giordana, F. Neri, L. Saitta, and M. Botta. Integrating multiple learning strategies in first order logics. *Machine Learning*, To appear, 1997.
32. A. Giordana and C. Sale. Genetic algorithms for learning relations. In *9th International Conference on Machine Learning*, pages 169–178, Aberdeen, UK, 1992.
33. E. M. Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
34. D. P. Greene and S. F. Smith. Competition-based induction of decision models from examples. *Machine Learning*, 13:229–258, 1993.
35. D. Haussler. Quantifying inductive bias - ai learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36:177–221, 1988.
36. D. Haussler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4:7–40, 1989.
37. J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, Mi, 1975.
38. K. Hornik, M. Stinchcombe, and H. White. Multilayer feed-forward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
39. C.Z. Janikow. A knowledge intensive genetic algorithm for supervised learning. *Machine Learning*, 13:198–228, 1993.
40. K. P. Jantke. Case-based learning and inductive inference. In *5th Annual ACM Workshop on Computational Learning Theory*, pages 218–223, Pittsburgh, PA, 1992.
41. F. Jelinek. Continuous speech recognition by statistical methods. In *Proceedings of IEEE*, volume 64, pages 532–556, 1976.
42. R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial intelligence*, pages 1137–1143, Montreal, Quebec, 1995. AAAI Press.
43. T. Kohonen. *Self-Organizing Maps.* Springer-Verlag, Berlin, 1995.
44. J. E. Laird, A. Newell, and P. S. Rosenbloom. Soar: an architecture for general intelligence. *Artificial Intelligence*, 33, 1987.
45. K. Lang. Newsweeder: Learning to filter netnews. In *12th International Conference on Machine Learning*, pages 331–339, Lake Tahoe, CA, 1995.
46. P. Langley. Editorial: On machine learning. *Machine Learning*, 1:5–10, 1986.
47. P. Langley. Editorial: Machine learning as an experimental science. *Machine Learning*, 3:5–8, 1988.
48. P. Langley, G. L. Bradshaw, and H. A. Simon. Bacon.5: The discovery of conservation laws. In *International Joint Conference on Artificial Intelligence*, pages 121–126, Vancouver, Canada, 1981.
49. P. Langley, G. L. Bradshaw, H. A. Simon, and J. M. Zytkow. *Scientific Discovery: computational explorations of the creative processs.* MIT Press, Cambridge, MA, 1987.
50. D. B. Lenat. *AM: an artificial intelligence approach to discovery in mathematics as heuristic search.* McGraw-Hill, New York, NY, 1982.

51. D. B. Lenat. EURISKO: A program that learns new heuristics and domain concepts. the nature of heuristics iii: Program design and results. *Artificial Intelligence*, 21, 1983.
52. D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *11th International Machine Learning Conference*, New Brunswick, NJ, July 1994.
53. E. D. Liddy, W. Paik, and E. S. Yu. Text categorization for multiple users based on semantic feature from a machine readable dictionary. *ACM Transaction on Information Systems*, 12:278–295, 1994.
54. C. X. Ling and M. Marinov. Answering the connessionistic challenge: a symbolic model of learning the past tenses of english verbs. *Cognition*, 49:235–290, 1993.
55. R.S. Michalski. Pattern recognition as a rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2:349–361, 1980.
56. R.S. Michalski. A theory and methodology of inductive learning. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning, an Artificial Intelligence Approach*, volume I, pages 83–134. Morgan Kaufmann, Los Altos, CA, 1983.
57. R.S. Michalski and R. Stepp. Learning from observation: conceptual clustering. In R. Michalski, J. Carbonell, and T. Mitchell, editors, *Machine Learning, an Artificial Intelligence Approach*, volume I, pages 83–134. Morgan Kaufmann, Los Altos, CA, 1983.
58. M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambride, MA, 1969.
59. S. Minton. *Learning Search Control Knowledge: an Explanation-based Approach*. Kluwer, Boston, MA, 1988.
60. S. Minton, J. G. Carbonell, C. A. Knoblock, D. R. Kuokka, O. Etzioni, and Y. Gil. Explanation-based learning: a problem solving perspective. *Artificial Intelligence*, 40:63–118, 1989.
61. T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
62. T.M. Mitchell. Webwatcher: a learning apprentice for the world wide web. In *AAAI Spring Symposium*, Stanford, CA, 1995.
63. T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli. Explanation based generalization: an unifying view. *Machine Learning*, 1:47–80, 1986.
64. S. Muggleton, editor. *Inductive Logic Programming*. Academic Press, London, UK, 1992.
65. F. Neri. *First Order Logic Concept Learning by means of a Distributed Genetic Algorithm*. PhD thesis, University of Torino, Italy, 1997. Available at http://www.di.unito.it/~neri/phd/thesis.ps.gz.
66. F. Neri and L. Saitta. Exploring the power of genetic search in learning symbolic classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, PAMI-18:1135–1142, 1996.
67. F. Neri, L. Saitta, and A. Tiberghien. Modelling physical knowledge acquisition in children with machine learning. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, page In press, Stanford, 1997.
68. A. Nix and M. Vose. Modeling genetic algorithms with Markov Chains. *Annals of Mathematics and Artificial Intelligence*, 5:79–88, 1992.
69. M. Pazzani, M. Dyer, and m. Flowers. Using prior learning to facilitate the learning of new causal theories. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 277–279, Milan, Italy, 1987.
70. M.J. Pazzani and D. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 14:57–94, 1992.

71. J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.
72. E. Riloff and W. Lehnert. Information extraction as a basis for high precision text classification. *ACM Transaction on Information Systems*, 12:296–333, 1994.
73. J. Rissanen. Universal coding, information, prediction, and estimation. *IEEE Transaction on Information Theory*, IT-30:629–636, 1984.
74. E. W. Rosch. Principles of categorization. In E. W. Rosch and B. Lloyd, editors, *Cognition and Categorization*. Earlbaum, Hillsdale, NJ, 1978.
75. F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.
76. D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Parts I & II.* MIT Press, Cambridge, Massachusetts, 1986.
77. M. Sahami, M. Hearst, and E. Saund. Applying the multiple cause mixture model to text categorization. In *Proceedings of the 13th Conference on Machine Learning*, pages 435–443, Bari, Italy, 1996. Morgan Kaufman.
78. L. Saitta and F. Bergadano. Pattern recognition and valiant's learning framework. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-15:145–155, 1993.
79. L. Saitta, M. Botta, and F. Neri. Multistrategy learning and theory revision. *Machine Learning*, 11:153–172, 1993.
80. G. Salton. Development in automatic text retrieval. *Science*, 253:974–980, 1991.
81. C. Schaffer. A conservation law for generalization performance. In *11th International Conference on Machine Learning*, pages 259–265, New Brunswick, NJ, 1994.
82. R. E. Schapire. The strenght of weak learnability. *Machine Learning*, 5:197–227, 1990.
83. T. R. Shultz, D. Mareschal, and W. C. Schmidt. Modeling cognitive development on balance scale phenomena. *Machine Learning*, 16:57–86, 1994.
84. R. J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:1–22, 224–254, 1964.
85. P. Suppes, M. Bottner, and L. Liang. Comprehension grammars generated from ml on nl sentences. *Machine Learning*, 19:133–152, 1990.
86. R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.
87. D. Thau. Primacy effects and selective attention in incremental clustering. In *Fourteenth Annual Conference of the Cognitive Science Society*, pages 219–223, Hillsdale, NJ, 1992. Lawrence Erlbaum Associates.
88. P.E. Utgoff. *Machine learning of Inductive Bias.* Kluwer Academic Press, 1986.
89. L. G. Valiant. Learning fallible deterministic finite automata. *Communications of the ACM*, 27:1134–1142, 1984.
90. M. VanHeyningen. The unified computer science technical reports index: Lessons in indexing diverse resources. In *Proceedings of the 2nd Int. Conf. on the World Wide Web*, 1994.
91. V. N. Vapnik and Y. A. Chervonenkis. Necessary and sufficient conditions for the uniform convergence of means to their expectations. *Theory Probability Applications*, 26:532–553, 1981.
92. M. Veloso and J. Carbonell. Automatic case generation, storage and retrieval in prodigy. In *Proceedings of the First Workshop on Multistrategy Learning*, pages 363–377, Harpers Ferry, WV, 1991.

93. S. Vosniadou and W. F. Brewer. Mental models of the earth: A study of conceptual change in childhood. *Cognitive Psychology*, 24:535–585, 1992.
94. T. W. Yan and H. Garcia-Molina. Index structures for selective dissemination of information. Technical Report TRSTAN-CS-92-1454, Stanford University, Stanford, CA, 1992.
95. O. R. Zaane and J. Han. Resource and knowledge discovery in global information systems: A preliminary design and experiment. In *Proceedings of the First International Conference on Knowlede Discovery and Data Mining*, pages 331–336, Menlo Park, CA, 1995. AAAI Press.

Modeling and Querying Semi-structured Data

Sophie Cluet

INRIA, Domaine de Voluceau, 78153 Le Chesnay, France

1 Introduction

Since its creation in 1989, the Web has known a tremendous success due to its capacity to deliver very inexpensively information world wide.

Introduced as a mean to link the academic research community, the Web now evolves rapidly in three directions:

- Internet applications in which we find electronic commerce, marketing or technical support; this kind of applications enables commercial companies to deliver their products, information about them or services twenty-four hours a day, seven days a week.
- Intranet applications such as CIS (Corporate Information Systems) uses the same technology as Internet applications but isolated behind a corporate network, data is kept away from inquisitive eyes.
- Extranet applications use the internet infrastructure to build cross-domain secure applications. These applications are good candidates to use emerging technologies such as CORBA, DCOM, Internet tunneling in conjunction with HTML or JAVA applets.

The Web is thus no longer used as a support for hyperlinked individual pages but as an infra-structure for running world wide or corporate applications.

Common requirements can be pointed out for these applications.

- Web applications involve a very large amount of inter-related data that require some level of consistency.
- They are highly interactive. This implies strong security and concurrency mechanisms.
- Another obvious requirement is to allow users to find the information they are looking for. This enforces the need for structuring the information and providing declarative query languages.

Building applications with standard Web tools is a laborious task. The information is stored in poorly structured HTML pages, consistency and concurrency are managed by *ad hoc* programs, navigation is the only mean to access information and it can be a very tedious process. Furthermore, programs have to access data through complex protocols such as CGI or proprietary Web servers API.

Given the requirements stated above, database technology should be a good candidate to support Web applications. As a matter of fact, the database vendors and research community have been very active in this field in the last few years. The typical scenario for Web applications using the database technology

is the following: data are collected and stored in a database system, applications written on top of this system and output data formatted in HTML to be displayed on the Web. This scenario raises mainly three issues that concern (i) gathering information, (ii) importing and managing new kinds of data, often poorly structured (commonly called semi-structured [1]) and (iii) exporting data on the Web.

Gathering Information

Few database vendors have studied the problem of collecting Web data. Most probably, the reason for this lack of interest is that the applications targeted by database vendors are those of their current customers who already possess the information they plan to broadcast on the Internet. However, there exists a market of data providers whose job is to provide information relevant to some topic, information that has to be collected over various sources including the Web.

The database research community has been very active in that domain (see for example [10, 19, 26, 27, 29, 37]). The postulate at the basis of most of these research projects is that search engines as found on the Web (e.g., Altavista [18], Yahoo [40]) are only partial answers to the problem of gathering information. Although they are fast and cover most of the available data, (i) they cannot be adapted easily to the requirements of a specific user, (ii) their query language is poor, and (iii) they often return too many answers, badly ordered and mainly irrelevant. The advocated solutions usually rely on database-like Web query languages and customized structured views.

Importing and Managing Semi-Structured Information

The industrial answer to the problem of Web data import (either virtual or real) mainly relies on Corba [31] or ODBC (e.g., O_2[30], Versant[38]). The programmers are thus in charge of specifying the interfaces allowing the import of the various (potentially heterogeneous) data. Once the data is stored in the database system, the programmers are supposed to used standard database tools, sometimes enhanced (notably with full text index capacities [30]), to write their applications.

Whereas the database vendors have, somehow, skirted the problem of data heterogeneity (it is up to the programmer to solve it), the research community has mainly tackled this specific problem. A few experiments have shown that database models are not always suitable when one has to integrate heterogeneous and poorly structured data. The reason for this is their lack of flexibility. New models have thus been proposed, integration models in which the notion of type/schema disappears or is loosen (see for example [9, 13, 33, 34, 35, 36, 37]). Each of these models supports a query language. Some migration tools (e.g., [2, 8, 12, 13]) and optimization techniques are also proposed (e.g., [8, 14]).

Exporting Data on the Web

Most database vendors now provide tools allowing automatic display on the Web of database stored information (see for example O_2 [30] and Oracle [32]). These tools generate HTML pages in a rather simple way. For instance, an

object = an HTML page, a complex value = an HTML list of items, an atomic value = a string, a reference to some other object = a URL, a URL = a query on the database. However, this default mapping is sometimes inadequate and appropriate customization is then either difficult or impossible. The outward simplicity of the Web exportation problem may explain the disinterest of the database research community. To our knowledge, there exist few studies on this topic ([2, 37]).

The readers should now be convinced that Web applications raise many interesting problems for database researchers. This paper does not develop all of them but concentrates on the problems of modeling and querying (efficiently) semi-structured data. It is by no means exhaustive. It compares various works with some experiments led by the VERSO project at INRIA in the context of three European projects: AQUARELLE, OPAL and WIRE.

2 Modeling Data

A Web application usually deals with data found in various systems using different formats, e.g., HTML[21], SGML[23], or LaTex [28] for text, ASN.1 [22] or STEP [5] for technical data, relational data [16], JPEG[39] or MPEG[24] for images, RealAudio[7] for sound.

Most of these formats can easily be integrated in a commercial database system either because the system supports the format or because one can define a mapping to the database format. In the best case (the database model is more general than the imported data format), this mapping has no loss of information (e.g., JPEG data can easily be mapped to the "Bits" type found in most commercial systems, STEP data have a structure similar to those found in ODMG systems). The data is then well structured using the database constructors (relations, lists, tuples, etc.). In other cases (some characteristics of the imported model cannot be captured by the target model), information is lost in the process or the database model is used in a degenerated way.

The ODMG-like schema below illustrates this using the SGML format. SGML, among other things, features a choice connector, that can be seen as a union type and that complicates its import in a database system. For instance, one may specify that a paragraph is a list of elements that are either text or figures or references to other documents. This either-or construction is common to many formats, notably the simple and widely used BNF format. Instead of capturing the structure relevant to a class of documents (as is usually the case in database applications), the following schema captures the (pseudo-)syntactical tree of any SGML document. In other words, it models the imported data representation model.

```
Class Element type tuple (name: string,
                          attributes: list(Attribute),
                          content: list(Element))
Class Attribute type tuple (name: string,
                          value: string)
```

Figure 1 shows an instance of this schema.

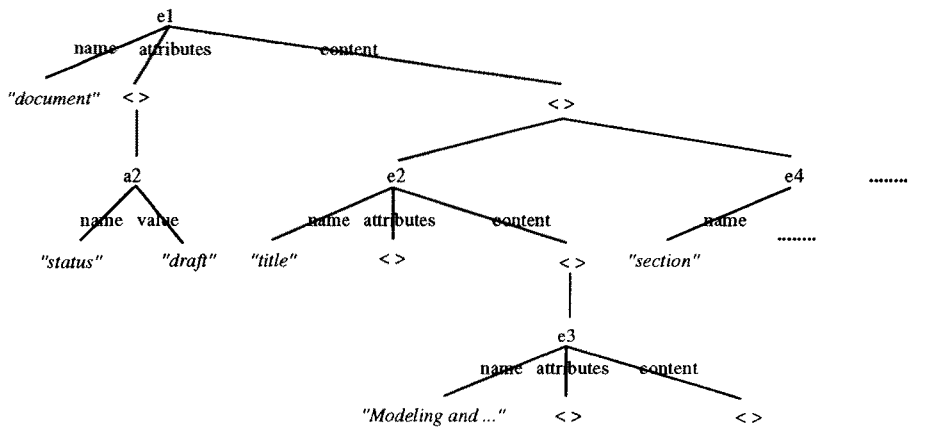


Fig. 1. Database View of a Syntactical Tree

As can be noticed, the structure of the document (i.e., the fact that a document is composed of a title, a status, some sections with titles, etc.) is part of the data itself. A consequence of this degenerated use of a database schema is that database tools are no longer adequate. As an illustration, one may try to formulate the following query using a database query language.

What are the titles contained in my document?"

To palliate this, two solutions have been considered. Both rely on the assumption that the difficulty in representing semi-structured data in a database system is due to the lack of flexibility of database models. The first solution advocates the effective support of union types in database systems. The second solution is more radical. The idea is to build new (lightweight) systems (from scratch or on top of existing ones) without types or schema. We now illustrate these two approaches.

2.1 The Union Type Approach

To capture data heterogeneity, one can use union types. This approach adds union types to an existing object-oriented database system in order to give it the flexibility required by semi-structured data. It has been followed by the Verso group at INRIA [12, 13]. In a first phase, it was used in the context of the AQUARELLE project whose goal is to manage, using Web technologies, the cultural inheritance of the European countries. Ongoing work on that model is done in the context of two Esprit projects: WIRE (Web Information Repository

for the Enterprise) and OPAL (Integrated Information and Process Management In Manufacturing Engineering).

Despite of their rare presence in database systems, union types exist in various programming languages (among others, C and ML) as well as in theoretical database models (e.g., [3]). The idea is to introduce flexibility in the type system while keeping some measure of type safety and efficiency in terms of space allocation. Type safety is usually obtained through marked unions which require explicit access to the instantiated component of a union typed data and thus allow static typing of the programs. Memory is saved by allocating the space required by the largest member of the union and not the sum of the various members size.

When asked about the absence of union types in their systems, most database vendors answer that they can be either (i) captured by inheritance with appropriate methods or (ii) implemented using the tuple constructor. These *ad hoc* solutions are hardly satisfactory. They present the following drawbacks.

1. One way to implement union types using inheritance is the following: a union type between two classes C_1 and C_2 is represented by a class C super-class of both C_1 and C_2. Then, to access the C_1 component of a union typed object, one cast the object to C_1. This solution is inadequate.
   Inheritance is defined on classes (as opposed to types) and requires some structural compatibility. In that context, defining data as being either real or string requires some rather inelegant manipulation.
   In most systems, classes cannot be generated dynamically[1]. Complex types can. For example, an OQL query usually creates one or more new types and we would like new union types to be created as well.
2. OODB systems do not allow a redefinition of the operations associated to tuple types. Thus, using tuples to implement union types forces the programmer to check (by hand) the correctness of his/her programs given the semantics he/she has in mind and that may not be shared by his/her fellow programmers.
   Furthermore, the solution is greedy in terms of allocated space and there is no way to distinguish system generated tuples (e.g., as an answer to a query) from system generated union types.

Given this and the need for more flexibility, [13] proposes a model with marked union types that is an extension of IQL[3]. We assume here that the readers are familiar with object models without union types (e.g., the ODMG[11] model, but any model supported by an OO system would do) and simply (i) define marked union types, (ii) their domain and (iii) the main sub-typing rules involving marked union.

1. if $\tau_1, \ldots, \tau_n$ are distinct types and $a_1, \ldots, a_n$ are distinct attribute names, $(a_1 : \tau_1 + \ldots + a_n : \tau_n)$ is a (union) type.

[1] Classes have to be defined using a data description language preliminary to object creation

2. $dom(a_1 : \tau_1 + \ldots + a_k : \tau_k) = \{[a_i : v_i] \mid 1 \leq i \leq k \wedge v_i \in dom(\tau_i)\}$.
3. – if $\forall i \in [1..n], \exists j \in [1..m] : a_i = b_j \wedge \tau_i \leq \tau_j^{'}$
   $(a_1 : \tau_1 + \ldots + a_n : \tau_n] \leq (b_1 : \tau_1^{'} + \ldots + a_m : \tau_m^{'})$.
   – if $\exists j \in [1..m] : a_i = b_j \wedge \tau_i \leq \tau_j^{'}$
   $[a_i : \tau_i] \leq (b_1 : \tau_1^{'} + \ldots + b_m : \tau_m^{'})$.

The following schema with a union type captures the SGML document of Example 1 in a more natural way.

```
Name Documents: set(Document)
Class Document
  tuple (title: string, status: string, sections: list(Section))
Class Section
  tuple (title: string, paragraphs: list(Paragraph))
Class Paragraph
  list (text: string + figure: Figure + reference: Document)
```

Figure 2 represents a partial instance of that schema.

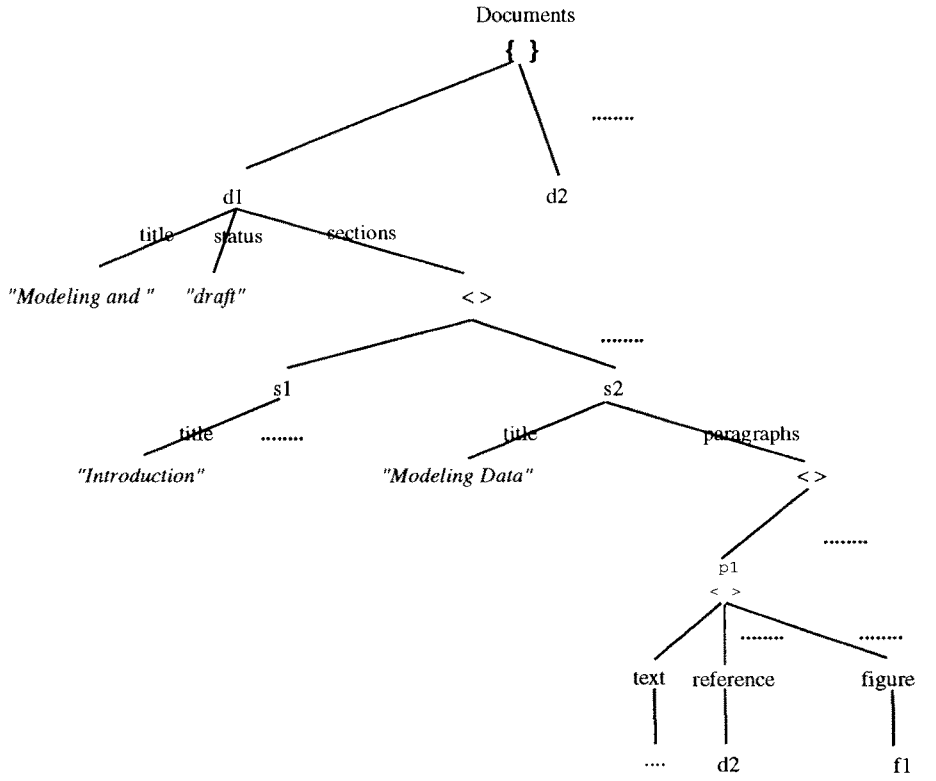


Fig. 2. Database View of a Set of Documents

We are back to a more traditional use of a database schema which captures the structure of the stored data. We will see in the next sections that this representation allows easy interrogation and that standard administration tools (e.g., clusters, indexes) can also be used (even though they might be insufficient in some cases). Thus, the main advantage of this approach is that its implementation requires few modifications of a database system.

However, it also has some drawbacks.

1. It is not clear whether union types are the panacea to all problems raised by the integration of heterogeneous data. It is reasonable to think that we will find data on the Web that will require some (too many?) manipulations in order to be modeled. Only an experimental study can validate/invalidate this approach.
2. Before accessing data of a union type, one has to check which component is instantiated. Thus, if too frequently used, union types may seriously downgrade the performance of a database application.

2.2 The No-Schema Approach

The No-Schema approach is the most popular [4, 8, 9, 37] for modeling semi-structured data. We illustrate it using two representative projects: the Lore system[4] of Stanford University and the system built by the database team at U. Pennsylvania[8, 9].

The approach consists in building new systems that do not support the notion of schema/type. Data is represented as a labeled graph/tree and new data can be added in a very flexible way by simply adding a branch to the existing graph/tree (i.e., no type restriction). Let us now study the two models.

The OEM (Object Exchange Model) Model of Lore manipulates objects. Each object has a unique identifier. Some objects are atomic, others are complex: their value is a set of object references, denoted by a set of (*label*, *edge*) pairs. The labels are string. Figure 3 illustrates this. $d1$ is the identifier of an object whose complex value is a set containing, among others, the pair ("title", $a1$). $a1$ is the identifier of an atomic object whose value is "Modeling and ...".

More formally, an OEM instance of R, R being a set of names, consists of (i) a finite labeled graph $(V_a \cup V_c, E)$ where V_a and V_c are disjoint sets of oid's corresponding respectively to atomic and complex objects, and the edges in E are labeled by strings; (ii) a *name* function from R to $V_a \cup V_c$; and (iii) a value function *val* that maps the objects in V_a to atomic values. The instance must also satisfy the following two conditions:

1. Atomic vertices have no outgoing edges.
2. Each vertex is reachable from some object $name(N)$, N in R.

The tree model of [9] is defined recursively in the following way.

- {} is the empty tree

- $\{l \Longrightarrow t\}$ is the tree whose root has an edge labeled l towards the sub-tree t.
- $t_1 \cup t_2$ is the tree constructed by merging the roots of both t_1 and t_2 trees.

It is further extended to support cycles (i.e., graphs) but we will study this simple form here. Figure 4 illustrates the definition.
$\{title \Longrightarrow \{Modeling\ and.... \Longrightarrow \{\}\}\}$ is a linear tree.
The tree representing the whole document is the union of this linear tree with other trees.
We introduced "key" components to represent the references between document.

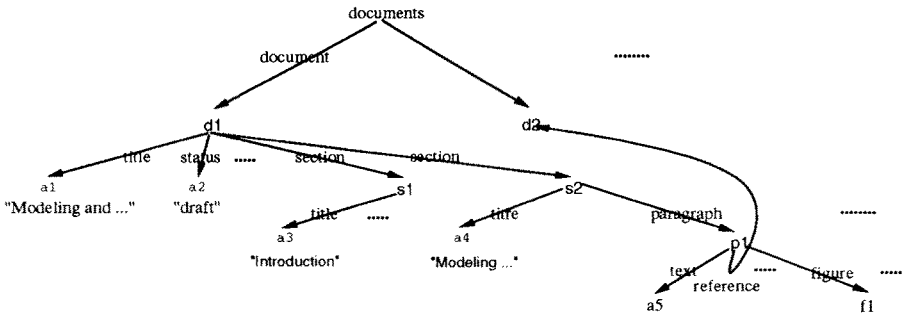


Fig. 3. An OEM Tree ([4])

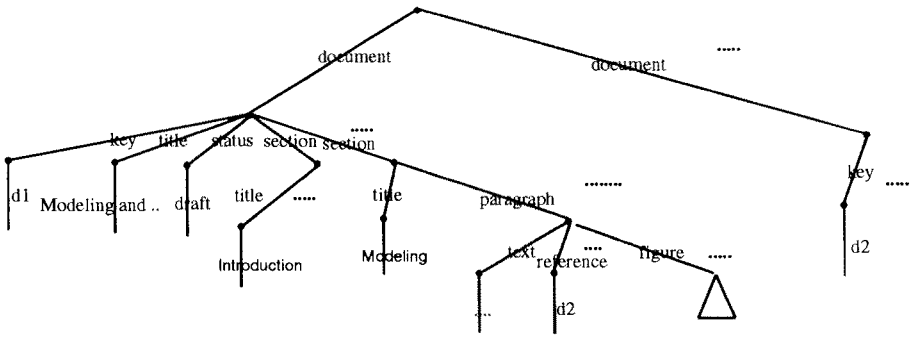


Fig. 4. A Data Tree According to [9]

As can be noted, the OEM model, as opposed to that of [9], features node identifiers. They can be used as an alternative to model cyclic references. Another

difference is that only the edges contain information in [9] whereas [4] favors both edges (for the structural information) and nodes (for atomic information).

The main advantages of the No-Schema approach are its simplicity and its flexibility. The integration of a new format is straightforward. We will see in the sequel, that these new models also support declarative languages.

However, and this will be illustrated in Section 4, the No-Schema approach has an important drawback. It entails a dynamic evaluation of queries. In a traditional database system, query evaluation is performed in two phases. First, the query is statically analyzed, using schema information and language semantics, in order to generate an execution plan. Second, the plan is evaluated without unnecessary access to schema information. Let us consider for instance a query that returns the main title of the documents contained in a database. A static analysis of the corresponding database schema will allow to retrieve the address of all documents as well as the offset to the attribute *title* in these documents. In a model without schema this is no longer feasible and the information has to be found dynamically and, in the absence of appropriate structure, may require a scan over the whole database.

This leads to an open and interesting issue: what new index structures will allow query optimization in the No-Schema context?

3 Querying Data

Once the data is stored in some system, one would like to access/query it. Let us first consider the union type approach. Accessing a member of a marked union type is similar to accessing an attribute of a tuple type. Thus, there is no *a priori* need for query features other than those found in a standard database system.

However, a Web application potentially contains highly heterogeneous information whose structure will probably be incompletely known by the programmers. On the other hand, standard database query languages require a complete knowledge of the schema in order to formulate a query. Consequently, there is a need to extend database languages in order to offer a more flexible access to the stored data.

It is then interesting to note that the POQL language [12, 13] that was designed to palliate this lack of flexibility, offers many similarities with the Lorel language [4] that was designed for the OEM model. Both languages rely on generalized path expressions (GPE) to query data with no or partial knowledge of the schema[2].

Other query languages for tree/graph models are those of [9, 8]. The first relies on structural recursion and differs a lot syntactically and semantically from the two previous languages. The second is more similar in syntax to that of [13, 4] and will just be briefly commented here.

In this section, we do not present the Lorel language. The reason is that it is very similar in syntax to POQL and that its semantics is defined in Section 4.

[2] GPE's are not new. They were introduced in different contexts [6, 25]

We first present POQL (with some references to Lorel), then the language from [9] will be presented .

3.1 Languages with GPE's

POQL[13] is an extension of OQL, the ODMG standard query language[11]. This design choice seems natural since the language is intended to support an object-oriented model with union types. OQL is a functional language. It is defined by a set of basic functions/queries and a way of building new ones through composition and iterators. The language iterators have an SQL-like syntax. The functional nature of OQL makes it easily expendable. One simply has to add new basic functions.

POQL extends OQL by adding (i) new textual boolean operators and (ii) generalized path expressions. We do not elaborate on the first extension: the textual operators are those found in most information retrieval systems (e.g., *contains* - does a text contain a word or a group of words?, *near* - is a word near another in a document?). The second extension is more significant since GPE's are primitives to query data with no or little knowledge of the schema, an essential feature in the semi-structured context.

A standard path expression is a combination of elementary access to object/value components (e.g., $.sections[0].title$). POQL GPE's extends these by adding to the model, on top of standard data variables, two new kinds of variables: attribute and path variables. For example, $.\#A@P(V)$ is a GPE containing an attribute variable ($\#A$, the symbol # prefixes attribute variables), a path variable ($@P$, the symbol @ prefixes path variables) and a data variable (V). If we apply this GPE to the object $d1$ of Example 2, one possible instantiation of the variables could be $\#A = sections$, $@P = [0].title$ and $V = "Modeling..."$. Adding these new variables implies adding the corresponding data types in the model along with their corresponding domains. The domain of type *Attribute* is the set of all attribute names. The domain of type *Path* is given below along with the syntactical definition of a GPE.

Domain of type Path

1. ϵ (the empty path) is in $dom(Path)$;
2. if a is in $dom(Attribute)$, then $.a$ (access to an attribute) is in $dom(Path)$;
3. if i is in $dom(integer)$, then $[i]$ (access to the element of a list) is in $dom(Path)$;
4. if p is in $dom(Path)$ and d is an object/value, then $p(d)$ (access to a value/object) is in $dom(Path)$.
5. if p is in $dom(Path)$ and d is an object/value, then $p\{d\}$ (access to the element of a collection) is in $dom(Path)$.
6. if p is in $dom(Path)$, then pq (path combination) is in $dom(Path)$.

Some example of paths are given below:

```
[0].title
.sections[0].title(''Modeling'')
.sections{s2}.paragraphs{p1}
```

Syntax of a GPE

Let $\mathcal{A}$ be the set of attribute variables taking their values in $dom(Attribute)$, and $\mathcal{P}$ the set of path variables taking their values in $dom(Path)$. The set E_G of GPE basic elements is constructed as follows:

$E_G = \{.a \mid a \in dom(Attribute)\} \cup \{.A \mid A \in \mathcal{A}\} \cup \{P \mid P \in \mathcal{P}\}$
$\cup \{(V) \mid V \in \mathcal{V}\} \cup \{\{V\} \mid V \in \mathcal{V}\} \cup \{\{\}\}$

Then, $\mathcal{G}$, the set of generalized path expressions is defined as follows:

- $\epsilon \in \mathcal{G}$ (the empty GPE);
- $\forall e \in E_G, e \in \mathcal{G}$ (the elementary GPE);
- $\forall g, g' \in \mathcal{G}, gg' \in \mathcal{G}$ (composition).

Applied on some value/object, a GPE returns the set of all "correct" valuations of the variables it contains. Each valuation is represented by a tuple with one attribute per variable. Thus, used in the *from* clause of an OQL query, a GPE allows to iterate over the different values of these variables. The notion of "correctness" for a valuation is defined below.

Correct Valuations

A valuation ν of a GPE g is said to be correct according to some data (value or object) d, iff:

- ν maps data variable to values/objects, path variables to paths and attribute variables to attributes.
- $d\ \nu(g)$ is a concrete path (see below) and $d\ \nu(g) \neq \perp$.

A concrete path corresponds to the application of a path p to some data value d. The value associated to a concrete path is defined as follows.

1. if $p = \epsilon$ then $d\ p = d$.
2. if $p = .a$, then if $d \leq [a : d']$ (note that this takes care of union types) or d is an object whose attribute/property a is equal to d',
   - then $d\ p = d'$
   - else $d\ p = \perp$.
3. if $p = [i], i \in [0..n]$, then if $d =< d_0, \ldots, d_n >$
   - then $d\ p = d_i$
   - else $d\ p = \perp$.
4. if $p = (d')$, then if $d = d'$
   - then $d\ p = d'$
   - else $d\ p = \perp$.
5. if $p = \{d_i\}, i \in [1..n]$, then if $d = \{d_1, \ldots, d_n\}$ of $d =< d_1, \ldots, d_n >$
   - then $d\ p = d_i$
   - else $d\ p = \perp$.
6. if $p = p_1 p_2$ and $d\ p_1 = d'$ then $d\ p = d' p2$

Note that this definition does not forbid cyclic paths. Thus, potentially, the set of valuation is infinite. The implementation of the valuation function avoids that by not considering path that traverses twice the same class. This is, of course, very restrictive. A better solution avoids cycle at the data level rather than at the schema level.

Let us now illustrate the language using some examples.

Q1 : what are the titles of the documents?

Supposing that one has a perfect knowledge of the database schema, this query can be formulated in OQL in the following way:

select d.title
from d in Documents
union
select s.title
from d in Documents, s in d.sections

The two subqueries of *Q1* correspond to the two attributes named *title* in the database schema. Clearly, if we are interested in an attribute occurring more often in the schema, OQL is not a good solution. Furthermore, we have to check the database schema beforehand to be sure that we do not forget any occurrences. Now, let us consider the corresponding POQL query.

select t
from Documents @P.title (t)

The *from* clause contains a generalized path expression. The path variable $@P$ is defined as being a path starting from the set of documents and ending in some object/value with a *title* attribute/property whose value will be assigned to the data variable t.

Let us now suppose that paragraphs have an attribute named *title* whose type is not a string but an object of class *Text*. Then, variable t may be instantiated with data from two different types (string and *Text*). This implies the use of union types for GPE's data variables. This has two important consequences. (i) Since system generated union markers cannot be known by the user, the language must give implicit access to the component of a union type. For example, t *like*"$Web*$" must be interpreted as "t is instantiated with a string and, assuming a is the union marker corresponding to the string type, $t.a$ *like*"$Web*$"". (ii) There is a need to perform an analysis of the query to reduce the generated union type. Considering the same example, we can reduce the type of t to a string. These problems are more complex than it may appear and are treated in [12]. Note that they do not exist in Lorel since there is no typing process. Below is the Lorel query corresponding to *Q1*.

select Documents.document ♯.title

¿From the database root, one accesses all edges labeled *document*, then all possible paths are followed (denoted by the ♯ symbol) until one reaches an edge labeled *title*. The query returns a tree consisting of a new root with as many outgoing edges as *title*'s edges found following any path from a document.

Let us now consider query *Q2*.

Q2: what are the documents containing the word "Web"?

Again, supposing that one has a complete knowledge of the schema and some time ahead of him, this query can be expressed in OQL.

select distinct d
from d in Documents,
s in d.sections,
p in Paragraphs
where d.title like "*Web*" or
s.title like "*Web*" or
.... or p.text->contains("Web")

The corresponding POQL and Lorel expressions are simpler:

POQL
select d
from Documents{d} @P(e)
where e contains "*Web*"

Lorel
select Documents.document
where Documents.document # contains "*Web*"

It is interesting to note that this query, although simple, may require a full traversal of the database. Standard database index cannot be used. Apart from the fact that they do not support high-level textual predicates, they are specified on fixed paths and their cost forbids a total coverage of the database.

We end this presentation with a query that cannot be formulated in OQL[3] and that is formulated in the same way in Lorel and POQL.

Q3: What structural difference is there between the current and the previous versions of my document?

We suppose that the names *MyDocument* and *MyOldDocument* represent, respectively, the current and previous versions of my document.

select @P
from MyDocument @P
minus
select @P
from MyOldDocument @P

The result of this query could be:

```
d.sections[0].paragraphs[4]
d.sections[3]
d.sections[3].title
```

3.2 Structural Recursion as a Query Language

Traditionally, construction and traversal of a tree are done recursively. Most of the time, the recursion follows the same pattern (either breadth or depth first). Some variations of that pattern allow to stop the recursion at "interesting" places or at a certain depth.

[3] The fact that it cannot be expressed in OQL is due to the absence of type *Path* in the ODMG model.

Based on that, [9] proposes a language for querying trees using a simple recursive pattern, that of the recursive definition of their model given in the previous section. The language is based on three simple functions that allow the recursive traversal of a data tree. The traversal is guided by the tree structure, thus the name of "structural recursion". The first function (*ext*) stops the recursion at the first level of the tree. The second function (*text*) traverses the whole tree. The last function (*text'*) allows a conditional traversal (i.e., stops in the recursion are possible).

1. Given a function $f : label \times tree \rightarrow tree$, *ext* applies f in the following way:
   - $ext(\{\}) = \{\}$
   - $ext(\{l \Longrightarrow t\}) = f(l,t)$ /* end of the recursion */
   - $ext(t_1 \cup t_2) = ext(t_1) \cup ext(t_2)$
2. Given a function $f : label \times tree \rightarrow tree$, *text* applies f in the following way:
   - $text(\{\}) = \{\}$
   - $text(\{l \Longrightarrow t\}) = f(l, text(t))$ /* recursive call */
   - $text(t_1 \cup t_2) = text(t_1) \cup text(t_2)$
3. Given a function $f : label \times tree \times tree \rightarrow tree$ *text'* applies f in the following way:
   - $text'(\{\}) = \{\}$
   - $text'(\{l \Longrightarrow t\}) = f(l,t,text'(t))$ /* treatment of t before recursive call */
   - $text'(t_1 \cup t_2) = text'(t_1) \cup text'(t_2)$

We now reformulate Queries *Q1* and *Q2* to illustrate this language. Given the language description[9], we did not find a way to formulate Query *Q3*.

Q1 : What are the titles of the documents ?

$text'(\lambda(l,t,r).$ **if** $(l$='title) **then** t **else** $r)$(Documents)

Q2 : What are the documents containing the word "Web" ?

$text'(\lambda(l,t,r).$ **if** $(l$ = 'document)
then (**if** $[text(\lambda(l2,r2).$ **if** ($l2$ contains 'Web)
then $l2 \Longrightarrow \{\}$
else $r2)$ (t) = $\{\}]$
then {}
else t)
else r) (Documents)

Obviously, this language lacks "user-friendliness". It was originally designed to support optimization of the tree traversals. In [8], the authors propose a new SQL-like language. Its semantics is based on a calculus called UnCal which can itself be captured using structural recursion.

4 Optimizing Queries

We now consider the problem of query evaluation. As mentioned in Section 2, the Union-Type approach is more advantageous in terms of optimization opportunities since it is possible to work on schema and language before considering

data. As a matter of fact, it allows all the optimization techniques that have been developed in the OO context. However, standard OO optimization does not consider GPE's and we will see in this survey that GPE's do require new techniques.

Although difficult, we will then show that optimization in the No-Schema context is possible.

4.1 Optimizing GPE's with Schema

To our knowledge, GPE's were first introduced in [6] along with a (naive) evaluation technique that was also adopted in [13]. The technique implements somehow an intuitive understanding of GPE's:

1. look for all possible instantiations of attribute and path variables,
2. replace the attribute and path variables by their instantiations,
3. eliminate the not well-typed alternatives,
4. union the remaining instantiated queries,
5. perform standard optimization and evaluate the resulting query.

In other words, applied on the POQL formulation of Query *Q1*, Step 4 of this technique results in the corresponding OQL query. This solution has the following drawbacks:

- The number of subqueries explodes exponentially in the number of attribute or path variables, but also in the number of attribute of each tuple type involved.
- If a distinct is required within the original query, expensive duplicate elimination becomes necessary.
- No rewriting previous to, or intermixed with, the GPE's instantiation is possible.

To palliate these inconvenients, [14] proposes an algebraic treatment that we review now.

In the naive approach, GPE's are processed from a schema perspective before being evaluated on the object base. The approach of [14] relies on the fact that these two instantiations are somehow unavoidable but offers some optimization possibilities that should be exploited. The main idea is thus to integrate schema lookup and object base lookup in an algebra, and thereby be able to apply optimization techniques in a homogeneous fashion to both lookups. For this, [14] extends the algebra of [15] with two new operators, *S_inst* and *D_inst*, that instantiate GPE's from a schema and a data perspective. We do not define the algebra here, simply gives an intuition, using an example, of the optimization process and of its potential. Readers are referred to [14, 12] for more details.

The example is the following:

Q4 : What are the couples of documents referencing each other and such that the first document contains the word "Web"?

select d1, d2
from Documents{d1} @P (x),
Documents{d2} @Q (y),
where d2=x **and** d1=y **and** d.title contains "Web"
The algebraic translation of Query *Q4* is illustrated on Figure 5.

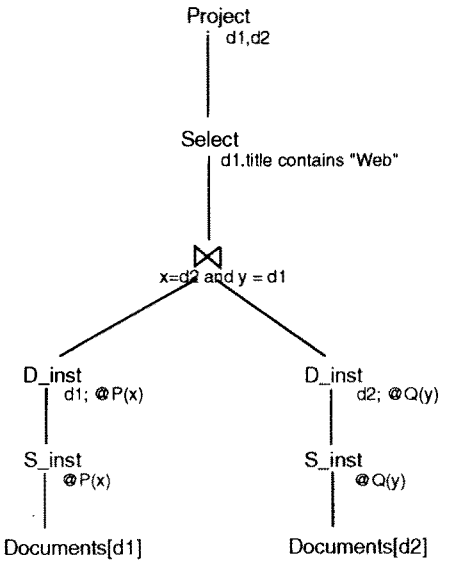


Fig. 5. Algebraic Translation of Query Q4

The left sub-tree (resp. right) represents the schema and data instantiations of the first (resp. second) GPE. The two branches are followed by a join, a selection and, finally, a projection operation.

Let us now study how this expression could be optimized.

1. Left and right sub-trees are similar; a factorization should be possible.
2. The selection operation could be pushed down the query tree. This would present several advantages. First, we may reasonably assume that the (expensive) *D_inst* operation will be applied to a smaller set. Second, this will allow the use of an index to evaluate the selection.
3. By analyzing the join predicate, we could reduce the schema lookup phase.

Figure 6 illustrates this. It shows an algebraic rewriting of the expression of Figure 5.

Note that the second *S_inst* operation has disappeared and be replaced by a renaming (*Map*) of the result of the first. The join operation has been integrated to the second *D_inst* operation, which follows @Q from the $d2$ data elements that can be found at the end of @P. This somehow takes care of the first join

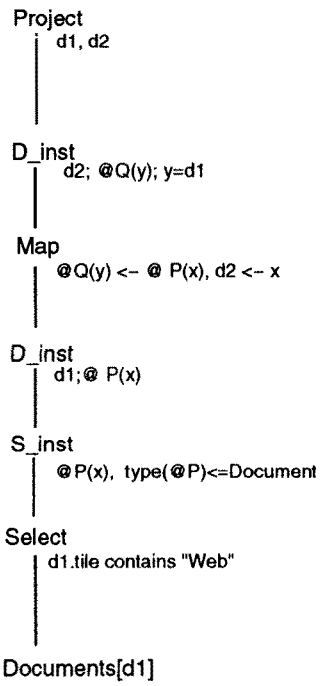


Fig. 6. Algebraic Optimization of Query Q4

condition $(x = d2)$. The second join condition $(y = d1)$ is given to the *D_inst* operation that will consider only elements satisfying it.

This algebraic treatment offers already much more than the naive technique. However, it is still insufficient. This can be shown using query *Q2* of the previous section. The algebraic expression corresponding to this query is presented in Figure 7.

Since the selection operation carries over elements found at the end of path $@P$, it cannot be pushed. This is rather bad. It means that, in order to evaluate the query, most of the database must be scanned. This clearly implies that some alternative solutions must be found. Unsurprisingly, we propose to use full text indexes. This is still ungoing work and will not be presented here.

4.2 Optimization in the No-Schema Context

As far as we know, there have been only two proposals to process queries in the No-Schema context and little or no optimization *per se*.

The first proposal concerns the UnQL language[8]. Queries are translated into a calculus and are then processed using structural recursion. The proposed optimization consists mainly in pushing selections and factorizing common sub-expressions.

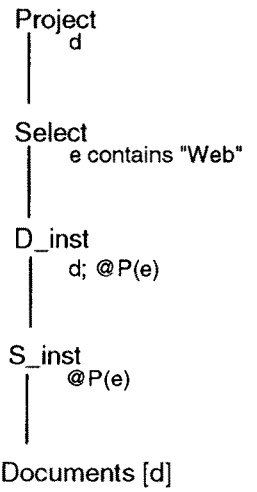


Fig. 7. Algebraic Optimization of Query Q2

We do not comment this work more. The reason is that we are not familiar with the formalism and don't really see what could be done to improve the optimization process. On the other hand, we are more familiar with the formalism used in the second proposal which is that of the Stanford group[4]. We study it here.

The first step taken in order to process a query is to rewrite it into an OQL-like query. For example, consider the following query:

Q5 : What are the titles of the sections of the documents whose title contains the word "Web"?

select Documents.document.section.title
where Documents.document.title contains "Web"

It is translated in OQL in the following way:

select t
from d in Documents.document, s in d.section, t1 in s.title
where exists t2 in d.title: t2 contains "Web"

Note that all common prefixes of paths are factorized and that for each of them a new variable is introduced in the added *from* clause. Additionally, since all attribute values are thought of as set-valued (leaving a node, there may be several edges with the same label), additional variables are introduced for paths occurring in the *select* clause (t in the example) so as to obtain an unnested result. Last — again since attribute are set-valued — comparisons between simple values and sets are removed by introducing explicit existential quantifiers. In the above Lorel query, the simple value *Web* is compared with the set of titles of the database documents. Hence, an explicit existential quantifier is introduced in the corresponding OQL query.

After this step, the logical query plan given in Fig. 8 is constructed. It consists of several *scan* operators, three *d-joins* [15], some *select* and *project* operators

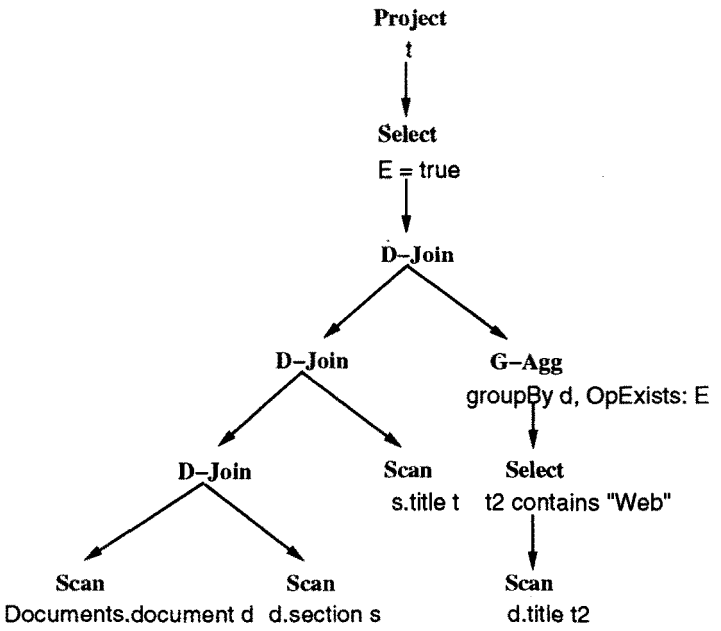


Fig. 8. Lorel Plan for Query Q5

as well as a *generalized aggregate* operator [17]. The *d-join* is a dependent join operator, sometimes called functional or implicit join. Its right input contains a free variable that is bound in the left input. Hence, the evaluation of the right input of a *d-join* depends on the left input. The *generalized aggregate* operator performs a group-by operation and then evaluates an aggregate function on each group. In the example plan, the aggregate is an existential quantifier.

After the query plan construction, query optimization, translation into a physical query plan, the plan execution follows. However, the authors of [4] note that the plan shown in Fig. 8 is — apart from pushing selections — *essentially evaluated directly*. The following disadvantages become obvious:

1. The *scan* operator of [4] returns , given an object identifier and a generalized path expression, the set of object identifiers at the end of the path. Since the scan operator is not allowed to branch, it is not able to extract information that is possibly stored together. For every single piece of information (e.g., corresponding to an attribute value), an extra scan is necessary. Clearly, this is a waste.
2. Since the information flows from one scan to another scan (object identifiers which are the output of one scan become the input of another scan by a transfer via the d-join), the access to the information is no longer sequential in nature. Instead, direct access via the object identifiers has to be performed. This is known to be far slower than a sequential scan.

3. Since only one path can be followed by the *scan* operator, the information in the tree is flattened. Assume that a document has many sections. Then, due to the recursive iterator [20] organization of their *scan* operator [4] there is a new object assignment for each section. There is no possibility to derive all the sections in a single step and work with the set of all sections afterwards. This way, all the leaf nodes that could be thought of as a set-valued attribute get unfolded, resulting in (1) a waste number of recursive iterator calls and (2) large intermediate results if they have to be materialized. This clearly slows down the evaluation process.
4. The extensive use of *d-joins* has several negative implications. First, the right argument cannot be evaluated independently. This implies a nested-loop where for each element of the left input the right input is evaluated. A nested-loop evaluation is the slowest evaluation procedure known. Further, due to the dependency, there is no way to reorder *d-joins*, which implies that the most powerful optimization technique — join reordering — cannot be applied.
5. Finally, the implementation of queries featuring GPE's is still unclear.

We believe that these drawbacks can be fixed and are currently working in that direction. The idea is to use a more powerful scan operator. More specifically, this operator should be able to do whatever is possible in a single (!) traversal of the data tree. This is in analogy to the *scan* over relations in the relational context or an extent scan in the object-oriented context. This will minimize access to the data storage while leaving the control of the more sophisticated processing to the optimizer.

5 Acknowledgments

This paper summarizes some of the research themes of the VERSO group. Many thanks to Serge Abiteboul, Bernd Amann, Vassilis Christophides, Claude Delobel, Tova Milo, Guido Moerkotte, Michel Scholl and Jérôme Siméon with whom I had the pleasure to work.

References

1. S. Abiteboul. Querying semi-structured data. In *Proc. of International Conference on Database Theory*, Athènes, January 1997.
2. S. Abiteboul, S. Cluet, and T. Milo. Correspondence and translation for heterogeneous data. In *Proc. ICDT, Athènes, Grèce*, 1997.
3. S. Abiteboul and P. Kanellakis. Identity as a query language primitive. In *Proc. SIGMOD, Portland, Oregon*, 1989.
4. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The lorel query language for semistructured data, 1996. http://www-db.stanford.edu.
5. S. Arbouy, A. Bezos, A.-F. Cutting-Decelle, P. Diakonoff, P. Germain-Lacour, J.-P. Letouzey, and C. Viel. *STEP : Concepts fondamentaux*. AFNOR, 1994.

6. E. Bertino, F. Rabitti, and S. Gibbs. Query Processing in a Multimedia Document System. *ACM Transactions on Office Information Systems*, 6(1):1–41, January 1988.
7. brinkley.realaudio.com. Realaudio, 1996. http://brinkley.realaudio.com/index.html.
8. P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996.
9. P. Buneman, S. Davidson, and D. Suciu. Programming constructs for unstructured data. In *Proc. Workshop on Database Programming Languages (DBPL)*, 1995.
10. T. Catarci, S. Chang, D. Nardi, and G. Santucci. Wag: Web-at-a-glance. Technical report, Universita di Roma La Sapienza, 1997.
11. R.G.G Cattell. *The Object Database Standard: ODMG-93.* Morgan Kaufmann Publishers, San Francisco, CA, 1994. with contributions from Tom Atwood et.al.
12. V. Christophides. *Documents Structurés et Bases de Données Objet.* PhD thesis, CNAM-Paris, Octobre 1996.
13. V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In *SIGMOD'94*, pages 313–324. ACM, 1994.
14. V. Christophides, S. Cluet, and G. Moerkotte. Evaluating queries with generalized path expressions. In *Proceedings ACM SIGMOD, International Conference on Management of Data*, pages 413–422, Montreal, Quebec, Canada, June 1996.
15. S. Cluet and G. Moerkotte. Nested queries in object bases. In *Proc. Workshop on Database Programming Languages (DBPL)*, 1993.
16. E. Codd. A relational model of data for large shared data banks. *Communications ACM*, 13(6), 1970. pp 377-387.
17. U. Dayal. Of nests and trees: A unified approach to processing queries that contain nested subqueries, aggregates, and quantifiers. In *VLDB*, pages 197–208, 1987.
18. Digital. Altavista, 1997. http://altavista.digital.com/.
19. P. Atzeni et. al. The araneus project, 1996. http://poincare.inf.uniroma3.it:8080/Araneus/araneus.html.
20. G. Graefe. Query evaluation techniques for large databases. *ACM Computing Surveys*, 25(2), 1993.
21. I. Graham. Html- documentation and style guide, 1994. http://www.utirc.utoronto.ca/HTMLdocs/NewHTML/htmlindex.html.
22. ISO. Specification of astraction syntax notation one (asn.1), 1987. Standard 8824, Information Processing System.
23. ISO 8879. Information processing—text and office systems—Standard Generalized Markup Language (SGML), 1986.
24. C. Fogg J. L. Mitchell, D. Le Gall. *MPEG Video Compression Standard.* Chapman Hall, 1996.
25. M. Kifer, W. Kim, and Y. Sagiv. Querying object-oriented databases. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, pages 393–402, 1992.
26. D. Konopnicki and O. Shmueli. W3QS: A query system for the World Wide Web. In *Proceedings of the Twenty First International Conference on Very Large Data Bases*, pages 54–65, 1995.
27. Laks V. S. Lakshmanan, Fereidoon Sadri, and Iyer N. Subramanian. A declarative language for querying and restructuring the Web. In *Proc. of 6th. International Workshop on Research Issues in Data Engineering, RIDE '96*, New Orleans, February 1996. In press.

28. L. Lamport. *Latex: A Document Preparation System.* AW Computer and Engineering Publishing Group, 1994. second edition.
29. A. Mendelzohn, G. A. Mihaila, and T. Milo. Querying the world wide web, 1996. draft, available by ftp: milo@math.tau.ac.il.
30. O2Technology. The O_2 server. http://www.o2tech.fr/ ou http://www.o2tech.com/.
31. OMG ORBTF. *Common Object Request Broker Architecture.* Object Management Group, Framingham, MA, 1992.
32. Oracle. The oracle server. http://www.oracle.com:81/.
33. Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *Proc. International Conf. on Very Large Data Bases*, Bombay, 1996. to appear.
34. Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object exchange across heterogeneous information sources. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 251–260, Taipei, Taiwan, March 1995.
35. D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. Querying semistructured heterogeneous information. Technical report, Stanford University, December 1995. Full version of [QRS+95a], available by anonymous ftp from db.stanford.edu.
36. A. Rafii, R. Ahmed, M. Ketabchi, P. DeSmedt, and W. Du. Integration strategies in Pegasus object oriented multidatabase system. In *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, Volume II*, pages 323–334, January 1992.
37. D. Suciu. Proceedings of the workhop on management of semi-structured data. http://www.research.att.com/ suciu/workshop-papers.html.
38. Versant. The versant server. http://www.versant.com/.
39. K. Gregory Wallace. The jpeg still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.
40. Yahoo. The yahoo search engine, 1997. http://www.yahoo.com/.

Lecture Notes in Artificial Intelligence (LNAI)

Vol. 1114: N. Foo, R. Goebel (Eds.), PRICAI'96: Topics in Artificial Intelligence. Proceedings, 1996. XXI, 658 pages. 1996.

Vol. 1115: P.W. Eklund, G. Ellis, G. Mann (Eds.), Conceptual Structures: Knowledge Representation as Interlingua. Proceedings, 1996. XIII, 321 pages. 1996.

Vol. 1126: J.J. Alferes, L. Moniz Pereira, E. Orlowska (Eds.), Logics in Artificial Intelligence. Proceedings, 1996. IX, 417 pages. 1996.

Vol. 1137: G. Görz, S. Hölldobler (Eds.), KI-96: Advances in Artificial Intelligence. Proceedings, 1996. XI, 387 pages. 1996.

Vol. 1147: L. Miclet, C. de la Higuera (Eds.), Grammatical Inference: Learning Syntax from Sentences. Proceedings, 1996. VIII, 327 pages. 1996.

Vol. 1152: T. Furuhashi, Y. Uchikawa (Eds.), Fuzzy Logic, Neural Networks, and Evolutionary Computation. Proceedings, 1995. VIII, 243 pages. 1996.

Vol. 1159: D.L. Borges, C.A.A. Kaestner (Eds.), Advances in Artificial Intelligence. Proceedings, 1996. XI, 243 pages. 1996.

Vol. 1160: S. Arikawa, A.K. Sharma (Eds.), Algorithmic Learning Theory. Proceedings, 1996. XVII, 337 pages. 1996.

Vol. 1168: I. Smith, B. Faltings (Eds.), Advances in Case-Based Reasoning. Proceedings, 1996. IX, 531 pages. 1996.

Vol. 1171: A. Franz, Automatic Ambiguity Resolution in Natural Language Processing. XIX, 155 pages. 1996.

Vol. 1177: J.P. Müller, The Design of Intelligent Agents. XV, 227 pages. 1996.

Vol. 1187: K. Schlechta, Nonmonotonic Logics. IX, 243 pages. 1997.

Vol. 1188: T.P. Martin, A.L. Ralescu (Eds.), Fuzzy Logic in Artificial Intelligence. Proceedings, 1995. VIII, 272 pages. 1997.

Vol. 1193: J.P. Müller, M.J. Wooldridge, N.R. Jennings (Eds.), Intelligent Agents III. XV, 401 pages. 1997.

Vol. 1195: R. Trappl, P. Petta (Eds.), Creating Personalities for Synthetic Actors. VII, 251 pages. 1997.

Vol. 1198: H. S. Nwana, N. Azarmi (Eds.), Software Agents and Soft Computing: Towards Enhancing Machine Intelligents. XIV, 298 pages. 1997.

Vol. 1202: P. Kandzia, M. Klusch (Eds.), Cooperative Information Agents. Proceedings, 1997. IX, 287 pages. 1997.

Vol. 1208: S. Ben-David (Ed.), Computational Learning Theory. Proceedings, 1997. VIII, 331 pages. 1997.

Vol. 1209: L. Cavedon, A. Rao, W. Wobcke (Eds.), Intelligent Agent Systems. Proceedings, 1996. IX, 188 pages. 1997.

Vol. 1211: E. Keravnou, C. Garbay, R. Baud, J. Wyatt (Eds.), Artificial Intelligence in Medicine. Proceedings, 1997. XIII, 526 pages. 1997.

Vol. 1216: J. Dix, L. Moniz Pereira, T.C. Przymusinski (Eds.), Non-Monotonic Extensions of Logic Programming. Proceedings, 1996. XI, 224 pages. 1997.

Vol. 1221: G. Weiß (Ed.), Distributed Artificial Intelligence Meets Machine Learning. Proceedings, 1996. X, 294 pages. 1997.

Vol. 1224: M. van Someren, G. Widmer (Eds.), Machine Learning: ECML-97. Proceedings, 1997. XI, 361 pages. 1997.

Vol. 1227: D. Galmiche (Ed.), Automated Reasoning with Analytic Tableaux and Related Methods. Proceedings, 1997. XI, 373 pages. 1997.

Vol. 1228: S.-H. Nienhuys-Cheng, R. de Wolf, Foundations of Inductive Logic Programming. XVII, 404 pages. 1997.

Vol. 1236: E. Maier, M. Mast, S. LuperFoy (Eds.), Dialogue Processing in Spoken Language Systems. Proceedings, 1996. VIII, 220 pages. 1997.

Vol. 1237: M. Boman, W. Van de Velde (Eds.), Multi-Agent Rationality. Proceedings, 1997. XII, 254 pages. 1997.

Vol. 1244: D. M. Gabbay, R. Kruse, A. Nonnengart, H.J. Ohlbach (Eds.), Qualitative and Quantitative Practical Reasoning. Proceedings, 1997. X, 621 pages. 1997.

Vol. 1249: W. McCune (Ed.), Automated Deduction – CADE-14. Proceedings, 1997. XIV, 462 pages. 1997.

Vol. 1257: D. Lukose, H. Delugach, M. Keeler, L. Searle, J. Sowa (Eds.), Conceptual Structures: Fulfilling Peirce's Dream. Proceedings, 1997. XII, 621 pages. 1997.

Vol. 1263: J. Komorowski, J. Zytkow (Eds.), Principles of Data Mining and Knowledge Discovery. Proceedings, 1997. IX, 397 pages. 1997.

Vol. 1266: D.B. Leake, E. Plaza (Eds.), Case-Based Reasoning Research and Development. Proceedings, 1997. XIII, 648 pages. 1997.

Vol. 1265: J. Dix, U. Furbach, A. Nerode (Eds.), Logic Programming and Nonmonotonic Reasoning. Proceedings, 1997. X, 453 pages. 1997.

Vol. 1285: X. Jao, J.-H. Kim, T. Furuhashi (Eds.), Simulated Evolution and Learning. Proceedings, 1996. VIII, 231 pages. 1997.

Vol. 1286: C. Zhang, D. Lukose (Eds.), Multi-Agent Systems. Proceedings, 1996. VII, 195 pages. 1997.

Vol. 1299: M.T. Pazienza (Ed.), Information Extraction. Proceedings, 1997. IX, 213 pages. 1997.

Lecture Notes in Computer Science

Vol. 1257: D. Lukose, H. Delugach, M. Keeler, L. Searle, J. Sowa (Eds.), Conceptual Structures: Fulfilling Peirce's Dream. Proceedings, 1997. XII, 621 pages. 1997. (Subseries LNAI).

Vol. 1258: D. van Dalen, M. Bezem (Eds.), Computer Science Logic. Proceedings, 1996. VIII, 473 pages. 1997.

Vol. 1259: T. Higuchi, M. Iwata, W. Liu (Eds.), Evolvable Systems: From Biology to Hardware. Proceedings, 1996. XI, 484 pages. 1997.

Vol. 1260: D. Raymond, D. Wood, S. Yu (Eds.), Automata Implementation. Proceedings, 1996. VIII, 189 pages. 1997.

Vol. 1261: J. Mycielski, G. Rozenberg, A. Salomaa (Eds.), Structures in Logic and Computer Science. X, 371 pages. 1997.

Vol. 1262: M. Scholl, A. Voisard (Eds.), Advances in Spatial Databases. Proceedings, 1997. XI, 379 pages. 1997.

Vol. 1263: J. Komorowski, J. Zytkow (Eds.), Principles of Data Mining and Knowledge Discovery. Proceedings, 1997. IX, 397 pages. 1997. (Subseries LNAI).

Vol. 1264: A. Apostolico, J. Hein (Eds.), Combinatorial Pattern Matching. Proceedings, 1997. VIII, 277 pages. 1997.

Vol. 1265: J. Dix, U. Furbach, A. Nerode (Eds.), Logic Programming and Nonmonotonic Reasoning. Proceedings, 1997. X, 453 pages. 1997. (Subseries LNAI).

Vol. 1266: D.B. Leake, E. Plaza (Eds.), Case-Based Reasoning Research and Development. Proceedings, 1997. XIII, 648 pages. 1997 (Subseries LNAI).

Vol. 1267: E. Biham (Ed.), Fast Software Encryption. Proceedings, 1997. VIII, 289 pages. 1997.

Vol. 1268: W. Kluge (Ed.), Implementation of Functional Languages. Proceedings, 1996. XI, 284 pages. 1997.

Vol. 1269: J. Rolim (Ed.), Randomization and Approximation Techniques in Computer Science. Proceedings, 1997. VIII, 227 pages. 1997.

Vol. 1270: V. Varadharajan, J. Pieprzyk, Y. Mu (Eds.), Information Security and Privacy. Proceedings, 1997. XI, 337 pages. 1997.

Vol. 1271: C. Small, P. Douglas, R. Johnson, P. King, N. Martin (Eds.), Advances in Databases. Proceedings, 1997. XI, 233 pages. 1997.

Vol. 1272: F. Dehne, A. Rau-Chaplin, J.-R. Sack, R. Tamassia (Eds.), Algorithms and Data Structures. Proceedings, 1997. X, 476 pages. 1997.

Vol. 1273: P. Antsaklis, W. Kohn, A. Nerode, S. Sastry (Eds.), Hybrid Systems IV. X, 405 pages. 1997.

Vol. 1274: T. Masuda, Y. Masunaga, M. Tsukamoto (Eds.), Worldwide Computing and Its Applications. Proceedings, 1997. XVI, 443 pages. 1997.

Vol. 1275: E.L. Gunter, A. Felty (Eds.), Theorem Proving in Higher Order Logics. Proceedings, 1997. VIII, 339 pages. 1997.

Vol. 1276: T. Jiang, D.T. Lee (Eds.), Computing and Combinatorics. Proceedings, 1997. XI, 522 pages. 1997.

Vol. 1277: V. Malyshkin (Ed.), Parallel Computing Technologies. Proceedings, 1997. XII, 455 pages. 1997.

Vol. 1278: R. Hofestädt, T. Lengauer, M. Löffler, D. Schomburg (Eds.), Bioinformatics. Proceedings, 1996. XI, 222 pages. 1997.

Vol. 1279: B. Chlebus, L. Czaja (Eds.), Fundamentals of Computation Theory. Proceedings, 1997. XI, 475 pages. 1997.

Vol. 1280: X. Liu, P. Cohen, M. Berthold (Eds.), Advances in Intelligent Data Analysis. Proceedings, 1997. XII, 621 pages. 1997.

Vol. 1281: M. Abadi, T. Ito (Eds.), Theoretical Aspects of Computer Software. Proceedings, 1997. XI, 639 pages. 1997.

Vol. 1282: D. Garlan, D. Le Métayer (Eds.), Coordination Languages and Models. Proceedings, 1997. X, 435 pages. 1997.

Vol. 1283: M. Müller-Olm, Modular Compiler Verification. XV, 250 pages. 1997.

Vol. 1284: R. Burkard, G. Woeginger (Eds.), Algorithms — ESA '97. Proceedings, 1997. XI, 515 pages. 1997.

Vol. 1285: X. Jao, J.-H. Kim, T. Furuhashi (Eds.), Simulated Evolution and Learning. Proceedings, 1996. VIII, 231 pages. 1997. (Subseries LNAI).

Vol. 1286: C. Zhang, D. Lukose (Eds.), Multi-Agent Systems. Proceedings, 1996. VII, 195 pages. 1997. (Subseries LNAI).

Vol. 1289: G. Gottlob, A. Leitsch, D. Mundici (Eds.), Computational Logic and Proof Theory. Proceedings, 1997. VIII, 348 pages. 1997.

Vol. 1292: H. Glaser, P. Hartel, H. Kuchen (Eds.), Programming Languages: Implementations, Logigs, and Programs. Proceedings, 1997. XI, 425 pages. 1997.

Vol. 1294: B.S. Kaliski Jr. (Ed.), Advances in Cryptology — CRYPTO '97. Proceedings, 1997. XII, 539 pages. 1997.

Vol. 1299: M.T. Pazienza (Ed.), Information Extraction. Proceedings, 1997. IX, 213 pages. 1997. (Subseries LNAI).

Vol. 1300: C. Lengauer, M. Griebl, S. Gorlatch (Eds.), Euro-Par'97 Parallel Processing. Proceedings, 1997. XXX, 1379 pages. 1997.