Natural Language Processing for Question Answering (NLP4QA)

edited by

Maarten de Rijke   Bonnie Webber

EACL 2003 Workshop

Monday, April 14, 2003

Order copies of this and other ACL proceedings from:

Priscilla Rasmussen
Association for Computational Linguistics (ACL)
3 Landmark Center
East Stroudsburg, PA 18301
USA
Phone: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

SPONSOR:

Language and Inference Technology Group (University of Amsterdam)

INVITED SPEAKER:

Pierre Zweigenbaum, Université Paris 6

PROGRAM COMMITTEE:

Maarten de Rijke, University of Amsterdam, Co-chair
Bonnie Webber, University of Edinburgh, Co-chair

Steve Abney, University of Michigan
Johan Bos, University of Edinburgh
Eric Brill, Microsoft Research
Sabine Buchholz, Tilburg University
Charles Clarke, University of Waterloo
Oren Etzioni, University of Washington
Claire Gardent, CNRS Nancy
Brigitte Grau, LIMSI
Donna Harman, NIST
Bernardo Magnini, ITC-IRST, Trento
Mark Maybury, MITRE
Dan Moldovan, University of Texas at Dallas
Karen Sparck Jones, Cambridge University

WORKSHOP WEBSITE:

`http://www.science.uva.nl/~mdr/NLP4QA/`

INTRODUCTION

This volume contains the six papers accepted for presentation at Natural Language Processing for Question Answering, an EACL 2003 workshop held on April 14, 2003, just preceding the 10th Conference of the European Chapter of the Association for Computational Linguistics.

The aim of the workshop is to enable participants to hear about, discuss and assess where Natural Language Processing (NLP) can make a contribution to current and future Question Answering (QA) systems. Specific questions raised in the call for papers include

- Is the role of NLP restricted to domains where the amount of available data is insufficient for redundancy-based approaches to work—where one needs to answer a question based on 1 or 100 or 1000 documents rather than millions?
- Are there kinds of questions that NLP is needed in order to answer, such as subjective questions, temporal questions, why questions, questions involving assessment, information fusion, etc.?
- Can NLP be made to exploit the semi-structured nature of more and more web-based documents in QA? Can QA systems use NLP to exploit the emergence of the semantic web?
- Can QA take anything from previous work on Natural Language interfaces to databases? Is there potential synergy between the two?
- In evaluating system performance, can NLP provide new methods of answer assessment, to help move QA evaluation beyond time-consuming manual assessment?

The six accepted papers touch on many aspects of these questions, while Pierre Zweigenbaum's invited lecture on question answering in the medical domain will illustrate the impact of domain restrictions on question answering and the role of NLP.

We gratefully acknowledge the support of the Language and Inference Technology Group at the University of Amsterdam for our invited speaker, Pierre Zweigenbaum. Finally, we thank our reviewers for doing an excellent job within the very short time available to them.

Maarten de Rijke
Bonnie Webber
February 2003

WORKSHOP PROGRAM

Monday, April 14

14:00-14:05	Welcome
14:05-15:05	*Question Answering in Biomedicine* Pierre Zweigenbaum
15:05-15:30	*NLP for Answer Extraction in Technical Domains* Diego Mollá, Rolf Schwitter, Fabio Rinaldi, James Dowdall and Michael Hess
15:30-16:00	BREAK
16:00-16:25	*Generating Annotated Corpora for Reading Comprehension and Question Answering Evaluation* Tiphaine Dalmas, Jochen L. Leidner, Bonnie Webber, Claire Grover and Johan Bos
16:25-16:50	*Getaruns: A Hybrid System for Summarization and Question Answering* Rodolfo Delmonte
16:50-17:15	*Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering* Mark A. Greenwood and Robert Gaizauskas
17:15-17:40	*Learning Paraphrases to Improve a Question-Answering System* Florence Duclaye, François Yvon and Olivier Collin
17:40-18:05	*Selectively Using Relations to Improve Precision in Question Answering* Boris Katz and Jimmy Lin
18:05-18:10	Wrap-Up

Table of Contents

Question answering in biomedicine

Pierre Zweigenbaum
Mission de recherche en Sciences et Technologies de l’Information Médicale (STIM)
Assistance Publique - Hôpitaux de Paris
pz@biomath.jussieu.fr http://www.biomath.jussieu.fr/~pz/

Abstract

The recent developments in Question Answering have kept with open-domain questions and collections, sometimes argued as being more difficult than narrow domain-focused questions and corpora. The biomedical field is indeed a specialized domain; however, its scope is fairly broad, so that considering a biomedical QA task is not necessarily such a simplification over open-domain QA as represented in the recent TREC evaluations. We shall try to characterize salient aspects of biomedical QA as well as to give a short review of useful resources to address this task.

1 The task

1.1 What for whom

Question answering can be seen as an extension of Information Retrieval, and accordingly, potential users of biomedical QA are the present users of biomedical IR: the general public, medical students, health care professionals and researchers.

The general public increasingly consult knowledge resources, especially the Web, before or after seeing a doctor, for themselves or for relatives, to obtain information about the nature of a disease, the indications and contraindications of a treatment, etc. Medical students, as other students, use the Web for preparing assignments; in parallel, medical schools put an increasing proportion of their teaching material online. For health care professionals, online knowledge resources participate in continuous medical education. The traditional bibliographic databases (Medline) are now complemented with direct Web search, and discussions are seen on physicians newsgroups (and articles are published) about the best search strategies with different resources. Alper et al. (2001), looking for answers to family physicians’ clinical questions, note that “*the average time to obtain an adequate answer ranged from 2.4 to 6.5 minutes.*” They also mention that “*One study (Ely et al., 1999) found that physicians spent less than 2 minutes on average seeking an answer to a question. Thus, most clinical questions remain unanswered.*” The potential for QA technology is clear in that respect. Biomedical researchers, as other researchers, use both Web search and specialized knowledge bases (*e.g.*, FlyBase, flybase.bio.indiana.edu).

1.2 Language specialization

Medicine is notorious for its specialized ‘jargon’. Different levels of language specialization are found depending on the sources and their intended audience. Similarly, queries use more or less technical terminology depending on their authors. The potential gap in technicity between user questions and target documents may therefore be larger than in other domains. The issue of providing the general public access to specialized knowledge through ‘consumer vocabulary’ has been given particular attention a couple of years ago with the development of general public access to the Web (see, *e.g.*, www.nlm.nih.gov/medlineplus/). Medical QA may then be confronted with a more acute need to cater for terminological variation.[1] At

[1] For instance, Dalmas and Rivoallan (2002) have compiled patterns for finding expressions of answers to generic questions such as *“Is disease X inherited”*. Many variant words are clues of an interesting passage, including (we give English equivalents) *“genes”*, *“genetic”*, *“auto-*

the same time, one may question the relevance of (too?) technical documents as a source of knowledge to non-specialists. The situation might be compared to that of cross-language information retrieval, where users may be confronted with documents in a language which they do not master: shall we need automated translation of the target documents into less specialized terms? As in other domains, the difficulty of finding an answer depends on the distance between the question and the available source material. Indeed, things are easier when the source material contains the answer and when its formulation is closer to the question. The amount and nature of reasoning processes involved when the formulation gap is larger still remain to be assessed.

1.3 Trustworthy sources

An extremely delicate question is that of the reliability of medical knowledge sources, especially when consulted by the general public. Whereas search engines may propose different types of Web sites as the result of a search, the kind of service provided by a QA system involves a higher degree of answer categorization and selection, and therefore may be construed as implying more endorsement of the answers. The confidence that can be assigned to a knowledge source is thus a key feature which must be taken into account. The distinct role of *evidence-based medicine* is also to be considered.

1.4 Availability

The volume of data and knowledge available on the Web shows great variation from one domain to another. Whereas questions related to 'trivia' (*e.g.*, geography/tourism: *"What is the size of the Eiffel Tower"*) correspond to a well-developed area of the Web, the nature of biomedical knowledge available online must be assessed before question answering is addressed. For instance, in contrast with computer science, where technical details for many procedures are explained and discussed at length, every diagnostic or surgical technique might not be detailed on the Web. The preceding issue (reliability conditions) bears on this one, since it restricts the number of sources which are eligible for looking for answers. Another related dimension is that of language: if another language than English is considered, the quantity of online source material decreases accordingly. In our experience (Jacquemart and Zweigenbaum, 2003) with a test set of 100 student questions in a specialized domain (oral surgery) and a language other than English (French), a thorough manual search with Google was unable to obtain relevant answering documents within the top five hits for 40% of the questions. This contrasts with the 75–95% of answers obtained by (Alper et al., 2001) on a set of 20 clinical questions when searching 'electronic medical databases'.

2 The resources

Medical information processing has a long tradition of compiling large-scale resources for dealing with medical knowledge and information. We review a few of these resources which are relevant to biomedical question answering.

2.1 Health information sources

Web directories have been developed to help users find classified information. In a similar way, health information gateways now propose to perform this task for the biomedical domain. For instance, the *CISMeF* quality-controlled health gateway (www.chu-rouen.fr/cismef/, Darmoni et al. (2000)) selects resources according to quality criteria (*e.g.*, www.medcertain.org), which addresses the above-mentioned trust issue; it indexes medical Web sites with a structured thesaurus (the MeSH thesaurus, NLM (2001)), which helps to identify potentially relevant documents more precisely.

These directories are undoubtedly a place to visit if one is looking for precise, endorsed online biomedical information. Among the types of resources which can be found, teaching material occupies a good rank. An example is the project which has been lead for a couple of years by a consortium of French medical universities to create an online *French-language Virtual Medical University* (UMVF, Le Beux et al. (2002)). Another type of resource, practice guidelines, compile the most

somic", "hereditary", "inherited", "transmit", "transmission", "predisposition", "familial form", etc.

up-to-date and scientifically verified ('evidence-based') clinical knowledge.

A wealth of biomedical information also exists on commercial sites and CDROMs: drug knowledge bases (*e.g.*, in France, the Vidal drug monographs), encyclopedias and other resources (*e.g.*, www.dynamicmedical.com) provide authoritative knowledge which is precious as a source of answers to questions. Their online versions however often have restricted access.

2.2 Types of questions

Ely et al. (2000) have studied 1396 questions collected from more than 150 physicians, mainly family doctors, and propose a taxonomy of generic clinical questions. The main question types are listed in table 1. Such a taxonomy, according to the British Medical Journal's comments, "*has four potential uses: to organise large numbers of real questions, to route questions to appropriate knowledge resources by using automated interfaces, to characterise and help remedy areas where current resources fail to address specific question types, and to set priorities for research by identifying question types for which answers do not exist.*" Jacquemart and Zweigenbaum (2003) studied and classified 100 questions by students in oral surgery. Since both authors and domain are different, the main question types are also different.

What is the drug of choice for condition X?
What is the cause of symptom X?
What test is indicated in situation X?

Table 1: Most frequent generic questions derived from questions by primary care doctors (from (Ely et al., 2000)).

2.3 Linguistic and terminological resources

Open-domain QA draws liberally on linguistic resources: lexicons indeed, but also the ubiquitous WordNet thesaurus. The biomedical domain, at least in the English language, proposes its own specialized lexicon as well as plenty of structured thesauri.

The resource here is the *Unified Medical Language System* (UMLS, Lindberg et al. (1993)). Its main component is the *Metathesaurus*, which compiles and cross-references one hundred biomedical terminologies (in version 2003AA: more than 800,000 concepts and 2,000,000 strings), with their hierarchical and transversal relations. Its *Semantic Network* adds a common structure above these imported terminologies. Additionally, its *Specialist Lexicon* provides a large English lexicon with an emphasis on biomedical words, including derivational knowledge. Tools have been built around the UMLS to address terminological variation (*e.g.*, MetaMap, Aronson (2001)). The UMLS and its companion tools can be obtained free of charge from the US National Library of Medicine.[2] While the Specialist Lexicon and most of the terminologies included in the Metathesaurus are in English, many of these terminologies are international and exist in other languages. Medical lexicons are also being created in other languages (Weske-Heck et al., 2002; Zweigenbaum et al., 2003).

The UMLS can, to some extent, be compared with WordNet: it provides terms, synonyms, hierarchical relations, but most of its organization and structure is that of a thesaurus rather than that of a formal ontology. *Galen* is an actual medical ontology, expressed in a description logic formalism (Rector et al., 1997). It precisely defines several thousand medical concepts based on more primitive concepts and relations. Produced by the Galen European project, it can be downloaded from the OpenGalen web site (www.opengalen.org). Galen does propose sound, formal concept descriptions; its use is also more complex than that of a simple term hierarchy.

The *Gene Ongology* is a more recent controlled vocabulary dedicated to genomics, more specifically the description of gene products and their associated molecular functions, biological processes and cellular components. The Gene Ontology can be downloaded from www.geneontology.org. It provides terms, synonyms, multiple hierarchies with explicit *"is-a"* and *"part-of"* relations.

[2] www.nlm.nih.gov/research/umls/licence.html.

3 Conclusion

The biomedical domain raises new challenges for question-answering systems, but at the same time already proposes some resources to address these challenges: quality-controlled health information gateways offer a thorough indexing of trustworthy biomedical sources (section 2.1); taxonomies of question types rank and categorize interesting questions, taking into account their frequency of occurrence (section 2.2); biomedical lexicons, terminologies and ontologies are there to help manage domain-specific terms and concepts (section 2.3).

Although a few questions in the past TREC QA tracks have touched on the medical domain, no specific evaluation of medical QA has yet been performed. This is to happen in the French QA evaluation initiative *EQueR* (Grau, 2002), which is to take place in the coming year, where a medical QA track is planned. The preparation and execution of this track will tell us more about how the above considerations materialize in actual systems.

References

[Alper et al.2001] Brian S. Alper, James J. Stevermer, David S. White, and Bernard G. Ewigman. 2001. Answering family physicians' clinical questions using electronic medical databases. *J Fam Pract*, 50(11):960–965. Available at http://www.jfponline.com/content/2001/11/jfp_1101_09600.asp.

[Aronson2001] Alan R. Aronson. 2001. Effective mapping of biomedical text to the UMLS Metathesaurus: The MetaMap program. *Journal of the American Medical Informatics Association*, 8(suppl).

[Dalmas and Rivoallan2002] Tiphaine Dalmas and Renaud Rivoallan. 2002. Système de questions réponses dans le domaine médical. Projet de DESS, Intelligence Artificielle, Université Paris 6, April.

[Darmoni et al.2000] Stéfan J. Darmoni, J.-P. Leroy, Benoît Thirion, F. Baudic, Magali Douyere, and J. Piot. 2000. CISMeF: a structured health resource guide. *Methods of Information in Medicine*, 39(1):30–35.

[Ely et al.1999] John W. Ely, Jerome A. Osheroff, Mark H. Ebell, et al. 1999. Analysis of questions asked by family doctors regarding patient care. *BMJ*, 319:358–361.

[Ely et al.2000] John W. Ely, Jerome A. Osheroff, Paul N. Gorman, Mark H. Ebell, M. Lee Chambliss, Eric A. Pifer, and P. Zoe Stavri. 2000. A taxonomy of generic clinical questions: classification study. *BMJ*, 321:429–432. Available at http://bmj.com/cgi/content/full/321/7258/429.

[Grau2002] Brigitte Grau. 2002. EQueR: Évaluation de systèmes de Questions Réponses. Project definition, LIMSI-CNRS, Orsay, France. Part of the EVALDA evaluation initiative, ELDA, Paris, France.

[Jacquemart and Zweigenbaum2003] Pierre Jacquemart and Pierre Zweigenbaum. 2003. Towards a medical question-answering system: a feasibility study. In Pierre Le Beux and Robert Baud, editors, *Proceedings Medical Informatics Europe*, Amsterdam. IOS Press. *To appear*.

[Le Beux et al.2002] Pierre Le Beux, Franck Le Duff, Jacques Weber, Stéfan Darmoni, and Albert-Claude Benhamou. 2002. Intégration des nouvelles technologies éducatives dans l'Université Médicale Virtuelle Francophone. In Pascal Staccini, Mario Fieschi, Daniel Benchimol, and Régis Beuscart, editors, *Formation médicale et technologies de l'information et de la communication*, volume 14, pages 3–12. Springer, Paris.

[Lindberg et al.1993] Don A B Lindberg, Betsy L Humphreys, and Alexa T. McCray. 1993. The Unified Medical Language System. *Methods of Information in Medicine*, 32(2):81–91.

[NLM2001] National Library of Medicine, Bethesda, Maryland, 2001. *Medical Subject Headings*. http://www.nlm.nih.gov/mesh/meshhome.html.

[Rector et al.1997] A. L. Rector, S. Bechhover, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. 1997. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9(2):139–171.

[Weske-Heck et al.2002] G. Weske-Heck, Albrecht Zaiß, M. Zabel, Stefan Schulz, Wolfgang Giere, M. Schopen, and Rudiger Klar. 2002. The German Specialist Lexicon. *Journal of the American Medical Informatics Association*, 8(suppl).

[Zweigenbaum et al.2003] Pierre Zweigenbaum, Robert Baud, Anita Burgun, Fiammetta Namer, Éric Jarrousse, Natalia Grabar, Patrick Ruch, Franck Le Duff, Benoît Thirion, and Stéfan Darmoni. 2003. Towards a Unified Medical Lexicon for French. In Pierre Le Beux and Robert Baud, editors, *Proceedings Medical Informatics Europe*, Amsterdam. IOS Press. *To appear*.

NLP for Answer Extraction in Technical Domains

Diego Mollá
Rolf Schwitter
Centre for Language Technology
Macquarie University
{diego,rolfs}
@ics.mq.edu.au

Fabio Rinaldi **James Dowdall**
Michael Hess
Institute of Computational Linguistics
University of Zurich
{rinaldi,dowdall,hess}
@cl.unizh.ch

Abstract

In this paper we argue that question-answering (QA) over technical domains is distinctly different from TREC-based QA or Web-based QA and it cannot benefit from data-intensive approaches. Technical questions arise in situations where concrete problems require specific answers and explanations. Finding a justification of the answer in the context of the document is essential if we have to solve a real-world problem. We show that NLP techniques can be used successfully in technical domains for high-precision access to information stored in documents. We present ExtrAns, an answer extraction system over technical domains, its architecture, its use of logical forms for answer extractions and how terminology extraction becomes an important part of the system.

1 Introduction

Early question-answering (QA) systems were complex AI-based systems that converted a natural language query into a knowledge base query, searched in the knowledge base for an answer, and returned the results in natural language. Constructing and maintaining these knowledge bases became a true bottleneck and the resulting systems were brittle in nature and non-scalable. Well-known examples are the SHRDLU system (Winograd, 1972) and the LUNAR system (Woods, 1977).

Recently there has been an increase of research on text-based QA, triggered by the Text REtrieval Conference (TREC) Question Answering Track (Voorhees, 2001). In these modern approaches the knowledge base is replaced by collections of documents (mainly large corpora of newspaper articles) thereby eliminating the major problem of early QA systems.

The TREC Question Answering Track demonstrated from an early stage the deficiency of traditional IR approaches when applied to extracting answers from documents. This inadequacy of IR techniques is most visible when answers have to be found within a small window of text (50 bytes). It turned out that systems that used some form of deeper linguistic knowledge did a good job when the answer had to be localised within a small snippet of text.

Some sort of convergence appears to be emerging towards a common base architecture for text-based QA which is centred around four core components (Voorhees, 2001; Hirschman and Gaizauskas, 2001): A **Passage Retrieval** module is used to identify paragraphs (or text windows) that show similarity to the question (according to some system specific metric), a **Question Classification** module is used to detect possible answer types, an **Entity Extraction** module analyses the passages and extracts all the entities that are potential answers and finally a **Scoring** module ranks these entities against the question type, thus leading to the selection of the answer(s).

Recent QA systems use the Web as a resource. Several contributions to the QA track of TREC used the Web as a means to obtain data redundancy and avoid the need for complex linguistic analysis (Clarke et al., 2001; Brill et al., 2001). The rationale is, provided that we have enough data, there will always be some passage that explicitly shows the answer to the question using a simple pattern. The Web becomes a knowledge resource that can be accessed by crawlers and search engines and used for question answering.

In this paper we will argue that QA over technical domains cannot benefit in the same way from data-intensive approaches. Instead, the formal writing style used in these documents make them a good target object for intensive NLP techniques. The remainder of this paper is structured as follows: In Section 2, we motivate why we believe that technical texts are a good application domain for NLP-intensive approaches. In Section 3, we present ExtrAns, an answer extraction system that finds and displays answers to questions in technical domains. In Section 4, we show how we get a grip on terminology. In Section 5, we discuss how we represent the propositional content of sentences as minimal logical forms. In Section 6, we compare ExtrAns with a traditional information retrieval system. Finally, in Section 7, we conclude and summarize our experiences with NLP for answer extraction.

2 Technical Domains and Terminology

There will always be a need for technical documentation, and there will always be a need for tools that help people find the information they want from technical documentations. A Linux user may want to know how to set a symbolic link to a file or a directory. A user of Photoshop may want to know how to improve the tonal range of an image. A member of an Airbus technical maintenance crew may want to know the location of the Electronic Centralised Aircraft Monitor contactor. These technical documentations are not large when compared with the data used in the TREC Question Answering Track, and the user is unlikely to find the answer to some of these technical questions on the Web.

Approaches that rely on data redundancy do not work well in these domains for two reasons. First of all, the amount of text is not large enough and therefore problems of sparse data are likely to occur. Second, authors of technical manuals typically try to avoid redundancy, they do not want to explain the same concept more than once or twice. Trying to use data redundancy approaches in non-redundant data is a self-defeating task.

On the other hand, technical manuals become good source documents on which to apply NLP-intensive approaches. The formal writing in these texts makes it possible to write a grammar that will cover these texts. In fact, in a parser evaluation up to 90% of the sentences in a software manual were parsed by the publicly-available Link Grammar parsing system after incorporating a specific lexicon, and the evaluation was done more than 5 years ago (Sutcliffe and McElligott, 1996). Current parsing systems have improved since. It is currently possible to build the logical form of a sentence and use it in the question answering process, as the system described in Section 3 shows.

Given the non-redundant nature of technical texts, an approach that attempts to find the meaning of the text and use it for question answering is preferred to an approach that uses bags of words or collections of sentence patterns. In other words, technical texts allow and require the use of NLP-intensive approaches.

Technical domains typically use technical terms that are not defined in standard lexicons. In fact, in any technical domain the most important concepts are represented using terminology. These terms need to be properly detected and managed in order to be leveraged upon in a functioning QA system. For example in the Aircraft Maintenance Manual (AMM) different materials, parts of the aircraft, technician's tools and units of measure are so abundant that without proper identification of technical terms any NLP system would perform very poorly (Dowdall et al., 2002; Rinaldi et al., 2002).

3 ExtrAns, an Answer Extraction System

To deal with real-word problems in technical domains, we have developed and implemented ExtrAns, an answer extraction system that finds and displays precise answers in technical documents.

In contrast to other modern QA systems that operate over large collections of documents and use relatively little linguistic information, ExtrAns answers questions over technical domains exploiting linguistic knowledge from the documents and terminological knowledge about a specific domain.

The original ExtrAns system was used to extract answers to arbitrary user queries in the domain of Unix documentation files. An on-line demo of this early version of ExtrAns is available at the project web page.[1] More recently, we tackled a different domain, the Aircraft Maintenance Manual (AMM) of the Airbus A320 to prove the scalability of our approach.[2] The highly technical nature of this domain as well as an SGML-based format and a much larger size (120MB) than the Unix documentation, provide an important test-bed for the scalability and domain independence of the system. Currently we are integrating the HOWTOs from the Linux domain. These are documents that describe in detail certain aspects of configuring or using the GNU/Linux operating system.

The architecture of the ExtrAns system consists of several modules some of which are adaptations of third-party systems (Figure 1). The entire document collection is processed in an off-line stage and user queries are processed on-line. The same linguistic analysis is applied in both stages, transforming the input into a semantic representation called Minimal Logical Forms (MLFs).

The documents are first processed by the terminology extraction tool FASTR (Jacquemin, 2001) so that linguistic variations of terms can be taken into account. The linguistic analysis is done by Link Grammar (LG), a robust dependency-based parser (Sleator and Temperley, 1993). Multi-word terms are parsed as single syntactic units. This reduces the complexity of parsing (in terms of processing time and memory requirements) of the source text by as much as 50% since there is no need to compute the internal structure of such terms. Different forms of attachment ambiguities (prepositional phases, gerunds, infinitives, and *wh*-relative clauses) are resolved by an extension of Brill and Resnik's approach (Brill and Resnik, 1994). Sentence-internal pronouns are dealt with using the anaphora resolution algorithm (Lappin and Leass, 1994). From these partially disambiguated dependency structures ExtrAns derives one or more MLFs as semantic representation for the core meaning of each sentence (Mollá et al., 2000). If ExtrAns detects that a term belongs to a set of synonyms (= synset) in the terminological knowledge base, then the term is replaced by a synset identifier in the MLF. This results in a canonical form, where the synset identifier denotes the concept named by the terms in the synset (Rinaldi et al., 2003).

Unlike sentences in documents, user queries are processed on-line and the resulting MLFs are proved by deduction over MLFs of document sentences. When no direct answer for a user query can be found, the system is able to relax the proof criteria in a stepwise manner. First, synonyms are considered, then hyponyms, in a next step an overlap of logical forms is calculated, and as a last resort a bag of words approach is used (Mollá et al., 2000).

The MLFs contain pointers to the original text which allow ExtrAns to identify and highlight those words in the retrieved sentence that contribute most to a particular answer (Mollá et al., 2000). An example of the output of ExtrAns can be seen in Figure 2. When the user clicks on one of the answers provided, the corresponding document will be displayed with the relevant passages highlighted. This allows the user to check the answer in the context of the document and to verify the justification of the answer. This is especially important in the case of procedural questions where an explicit solution to a problem is required.

4 Terminology

As Answer Extraction involves a high degree of linguistic processing, terminology quickly becomes a major thorn in the side of computational efficiency. And worse, unstandardised terminology can effectively stop Answer Extraction in its tracks.

To produce a syntactic representation for each sentence the terms need to be identified as a phrasal unit. As only the word *compartment* in the term *overhead stowage compartment* inter-

[1] `http://www.cl.unizh.ch/extrans/`

[2] `http://www.cl.unizh.ch/webextrans/`

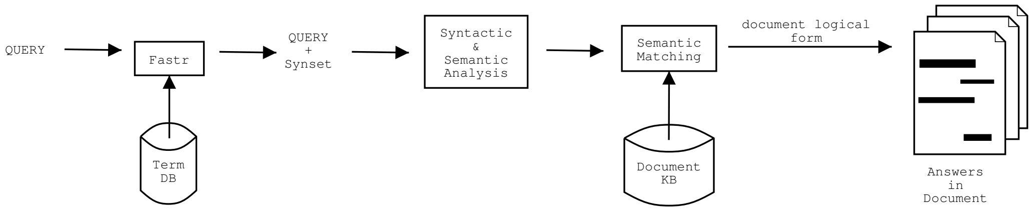
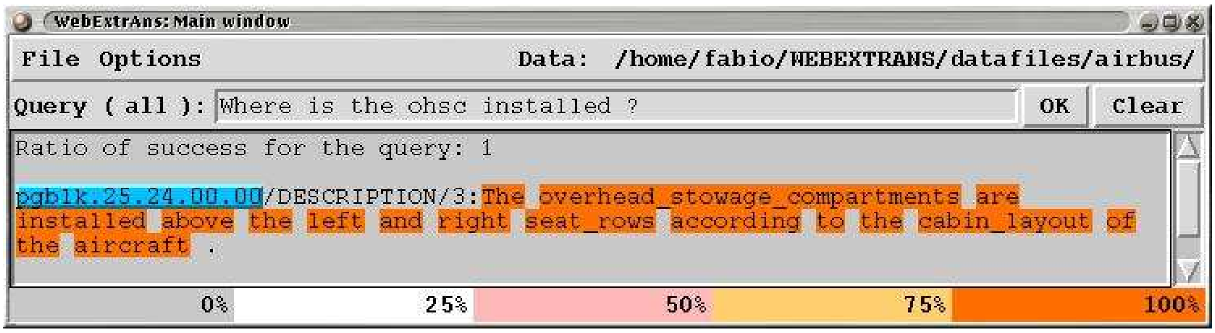


Figure 1: Schematic architecture of the ExtrAns system



Data: /home/fabio/WEBEXTRANS/datafiles/airbus/

Query (all): Where is the ohsc installed ?

```
Ratio of success for the query: 1

pgblk.25.24.00.00/DESCRIPTION/3:The overhead_stowage_compartments are
installed above the left and right seat_rows according to the cabin_layout of
the aircraft .
```

Figure 2: An example of the output of ExtrAns - query window.

acts with other sentence words, *overhead stowage* should be effectively ignored to produce the sentence parse. This is an advisable strategy as the *overhead stowage* may be plausibly combined with the wrong sentence words producing multiple possibilities for a given sentence, leaving the correct parse still to be chosen. The parser also wastes effort in assigning a structure to the term itself. Unfortunately, the internal syntactic structure of terms are notoriously idiosyncratic and resist standard parsing techniques.

To address this problem, we pre-process the document collection, extracting all the terminology. The extracted term list is organised into a hierarchy of subtypes to determine which terms are more specific kinds of other terms, and all of the different ways a single concept is refered to in the domain are gathered into synsets.

The subtype relations across the term set can easily be automatically determined. A pleasing characteristic of technical terminology is that more specific terms are formed by adding words to a more generic term. So a simple algorithm can determine that the *overhead stowage compartment* is a subtype of *stowage compartment* and the *crew member seat* is a subtype of *seat*. This hyponymy relation is comparable to the insertion variations defined by (Daille et al., 1996).

In order to find synonymous terms we have adapted the terminology extraction tool FASTR (Jacquemin, 2001). Using phrase structure rules in combination with a morphological database and WordNet, linguistic variations between two terms can be identified. This can detect simple head inversion (*water flow* ⟶ *flow of water*), morphological variations (*electric connector* ⟶ *electrical connector*) and complex morphosyntactic variations (*electrical generation equipment* ⟶ *equipment for generating electricity*).

Exploiting the WordNet synsets allows weaker synonymy relations to be discovered. Terms with synonymous heads (*bulk cargo* ⟶ *bulk load*), synonymous modifiers (*upright position* ⟶ *vertical position*) or both (*functional test* ⟶ *operational check*) are all detected.

We organise the terminology of the domain (6032 terms) in an internal structure that we call the Terminological Knowledge Base containing 2770 synsets with 1176 hyponymy links. In plainer terms, we could describe it simply as a computational thesaurus for the domain, organised around synonymy and hyponymy and stored in a database.

5 Minimal Logical Forms

The success of ExtrAns depends heavily on its use of logical forms. ExtrAns' logical forms are designed so that they are easy to build and to use, yet expressive enough for the task at hand. Not least importantly, the logical forms and associated semantic interpretation method are designed to cope with problematic sentences. This includes very long sentences, even sentences with spelling mistakes, and structures that are not recognised by the syntactic analyser.

To this end, ExtrAns uses *Minimal* Logical Forms (MLFs). The ability of these MLFs to underspecify makes them good candidates for NLP applications, specially when the applications benefit from the semantic comparison of sentences (Copestake et al., 1997; Mollá, 2001). In the case of ExtrAns, the logical forms only encode the dependencies between verbs and their arguments, plus modifier and adjunct relations. Ignored information includes complex quantification, tense and aspect, temporal relations, plurality, and modality. We have argued elsewhere that too detailed logical forms may interfere with the answer extraction mechanism and that additional information can be added incrementally (Mollá et al., 2000).

The MLFs of ExtrAns use *reification* to allow different kind of modifications, very much in the line of (Davidson, 1967; Hobbs, 1985; Copestake et al., 1997). The MLFs do not reify all predicates, as opposed to (Hobbs, 1985; Copestake et al., 1997; Mollá, 2001). In the current implementation only reification of objects, eventualities (events or states), and properties is carried out. The MLFs are expressed as conjunctions of predicates where all variables are existentially bound and have wide scope. For example, the MLF of the sentence *A coax cable connects the external antenna to the ANT connection* is:

```
holds(e1),
object(coax_cable,o1,[x1]),
object(external_antenna,o2,[x2]),
object(ant_connection,o3,[x3]),
evt(connect,e1,[x1,x2]),
prop(to,p1,[e1,x3]).
```

In other words, ExtrAns derives three multi-word terms using the Terminological Knowledge Base and translates them into objects: `x1`, a `coax_cable`, `x2` an `external_antenna`, and `x3` an `ant_connection`. The entity `e1` represents an eventuality derived from the verb involving two objects, the `coax_cable` and the `external_antenna`. The entity `e1` is used in the property derived from the prepositional phrase to assert that the eventuality happens to `x3`, the `ant_connection`.

The entities `o1`, `o2`, `o3`, and `p1` are not used in the MLF above, but other more complex sentences may need to refer to the reification of objects (required for non-intersective adjectives such as *the former pilot*) or properties (required for adjective-modifying adverbs such as *very safe*).

Reification can also be used to encode the existence of concepts. The predicate `holds(e1)` expresses that the connecting event `e1` holds in the given context.

In general, MLFs are monotonically extensible and allow us to add new constraints over entities as soon as new information becomes available without destructively rewriting the original expression. For example, the adverb in the sentence *A coax cable* **directly** *connects the external antenna to the ANT connection* changes nothing in the original MLF, but additionally asserts that `e1` is *directly*:

```
prop(direct,p2,e1)
```

Answer extraction is performed by finding those sentences whose MLFs form a superset of the MLFs of the question. To make this happen, the MLFs are translated into Prolog predicates and Prolog's theorem prover is used to find the answers. For example, the following Prolog call is generated for the question *How is the external antenna connected?*

```
?- object(external_antenna,O2,[X2]),
evt(connect,E1,[X1,X2]),
object(Anonymous_object,O1,[X1]).
```

If a sentence in the document asserts that the *external antenna* is connected to or by *something*, the Prolog query will succeed. This *something* is the `Anonymous_object` in the query. If there are no answers or too few answers, ExtrAns relaxes the proof criteria as described in Section 3.

Given that the MLFs are simplified logical forms converted into flat structures, ExtrAns may find sentences that are not exact answers but are still related to the user's question. Thus, given the question above, ExtrAns may also find sentences such as

- *The external antenna must not be directly connected to the control panel.*
- *Do not connect the external antenna before it is grounded.*
- *The external antenna is connected, with a coax cable, to the ANT connection on the ELT transmitter.*

An additional advantage of ExtrAns' MLFs is that they can be produced with minimal domain knowledge. This makes our technology easily portable to different domains. The only true impact of the domain is during the preprocessing stage of the input text and during the creation of a terminological knowledge base that reflects the specific terms used in the chosen domain, their lexical relations and their word senses.

6 Evaluation

An interesting evaluation framework is to compare the performance of ExtrAns against a traditional IR system – SMART (Salton, 1989). However, the traditional measures of *precision* and *recall* are deceptive in such a comparison due to the contrasting aims of the two systems. For the IR system these measures are of equal importance in an attempt to identify all possibly relevant documents, whereas for the AE system an increased focus on *precision* requires finding at least one answer to the question. With this in mind, a more informative measure is the Mean Reciprocal Rank (MRR) as used in the QA track of TREC (Voorhees and Tice, 1999) The rank of a given result is the position in which the first correct answer is found in the output list of the system. Over a given set of answers the MRR is calculated as the mean of the reciprocals of the ranks of all the answers.

The domain of the evaluation was the 120MB of the AMM (Rinaldi et al., 2002). By manual investigation 100 questions were devised with "known" answers in the document collection. For practical considerations an arbitrary result threshold was set at 10. If no answer was identified in the first 10 results, the case was classified as "Not Found".

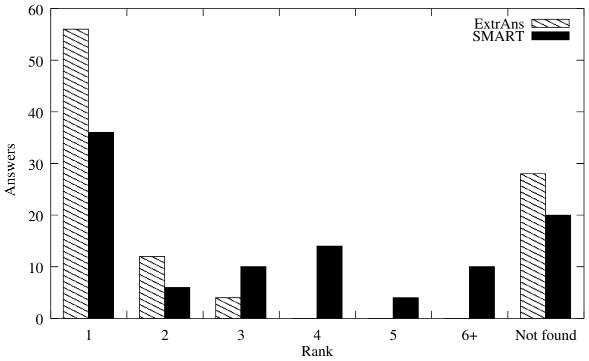


Figure 3: Answers at different ranks

Figure 3 displays the number of answers found at each rank (with 6 to 10 together). Even with the cut off point at 10, SMART clearly finds more answers than ExtrAns. However, in the majority of cases when ExtrAns does find an answer it is placed in first position. Further, in some cases ExtrAns finds more than one valid answer for the same question (possibly in the same document).

The MRR for ExtrAns was 0.63 whereas SMART achieved 0.46. As expected, ExtrAns provides far higher precision than the generic IR system, at the price of smaller recall.

7 Conclusions

We have introduced ExtrAns, an answer extraction system that uses intensive NLP techniques. Decisive factors for the success of ExtrAns include: (*i*) the explicit handling of domain-specific **terminology**; (*ii*) the integration of a full parser and semantic interpreter that include **robust technology** to cope with complex or ungrammatical sentences; (*iii*) The use of a **logical notation** that is flat and encodes the minimal semantic information required for the task at hand; and (*iv*) the integration of **displaying techniques** that help the user to find the answer in context.

Our experience with ExtrAns shows that answer extraction over technical domains is feasible and practical for real-world applications, and benefits from NLP techniques.

References

Eric Brill and Philip Resnik. 1994. A rule-based approach to prepositional phrase attachment disambiguation. In *Proc. COLING '94*, volume 2, pages 998–1004, Kyoto, Japan.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. 2001. Web reinforced question answering. In Voorhees and Harman (Voorhees and Harman, 2001).

Ann Copestake, Dan Flickinger, and Ivan A. Sag. 1997. Minimal recursion semantics: an introduction. Technical report, CSLI, Stanford University, Stanford, CA.

B. Daille, B. Habert, C. Jacquemin, and J. Royaut. 1996. Empirical observation of term variations and principles for their description. *Terminology*, 3(2):197–258.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–120. Univ. of Pittsburgh Press.

James Dowdall, Michael Hess, Neeme Kahusk, Kaarel Kaljurand, Mare Koit, Fabio Rinaldi, and Kadri Vider. 2002. Technical terminology as a critical resource. In *International Conference on Language Resources and Evaluations (LREC-2002), Las Palmas*, 29–31 May.

Lynette Hirschman and Rob Gaizauskas. 2001. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300.

Jerry R. Hobbs. 1985. Ontological promiscuity. In *Proc. ACL'85*, pages 61–69. University of Chicago, Association for Computational Linguistics.

Christian Jacquemin. 2001. *Spotting and Discovering Terms through Natural Language Processing*. MIT Press.

Shalom Lappin and Herbert J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):535–561.

Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. 2000. Extrans, an answer extraction system. *Traitement Automatique des Langues*, 41(2):495–522.

Diego Mollá. 2001. Ontologically promiscuous flat logical forms for NLP. In Harry Bunt, Ielka van der Sluis, and Elias Thijsse, editors, *Proceedings of IWCS-4*, pages 249–265. Tilburg University.

Fabio Rinaldi, James Dowdall, Michael Hess, Diego Mollá, and Rolf Schwitter. 2002. Towards Answer Extraction: an application to Technical Domains. In *ECAI2002, European Conference on Artificial Intelligence, Lyon*, 21–26 July.

Fabio Rinaldi, James Dowdall, Michael Hess, Kaarel Kaljurand, and Magnus Karlsson. 2003. The role of technical Terminology in Question Answering. In *accepted for publication at TIA 2003, Terminologie et Intelligence Artificielle, March 31 - April 1, 2003*, Strasbourg, 31 March – 1 April.

Gerard Salton. 1989. *Automatic Text Processing: the transformation, analysis, and retrieval of information by computer*. Addison Wesley, New York.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292.

Richard F. E. Sutcliffe and Annette McElligott. 1996. Using the link parser of Sleator and Temperley to analyse a software manual corpus. In Richard F. E. Sutcliffe, Heinz-Detlev Koch, and Annette McElligott, editors, *Industrial Parsing of Software Manuals*, chapter 6, pages 89–102. Rodopi, Amsterdam.

Ellen M. Voorhees and Donna K. Harman, editors. 2001. *The Tenth Text REtrieval Conference (TREC-10)*, number 500-250 in NIST Special Publication. NIST.

Ellen M. Voorhees and Dawn M. Tice. 1999. The trec-8 question answering track evaluation. In *The Eighth Text REtrieval Conference (TREC-8), Gaithersburg, Maryland*, November 17-19.

Ellen M. Voorhees. 2001. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378.

Terry Winograd. 1972. *Understanding Natural Language*. Academic Press.

W.A. Woods. 1977. Lunar rocks in natural english: Explorations in natural language question answering. In A. Zampolli, editor, *Linguistic Structures Processing*, volume 5 of *Fundamental Studies in Computer Science*, pages 521–569. North Holland.

Generating Annotated Corpora for Reading Comprehension and Question Answering Evaluation

Tiphaine Dalmas   Jochen L. Leidner   Bonnie Webber   Claire Grover   Johan Bos
Institute for Communicating and Collaborative Systems (ICCS),
School of Informatics, University of Edinburgh,
2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, UK.
t.dalmas@sms.ed.ac.uk

Abstract

Recently, reading comprehension tests for students and adult language learners have received increased attention within the NLP community as a means to develop and evaluate robust question answering (NLQA) methods. We present our ongoing work on automatically creating richly annotated corpus resources for NLQA and on comparing automatic methods for answering questions against this data set. Starting with the CBC4Kids corpus, we have added XML annotation layers for tokenization, lemmatization, stemming, semantic classes, POS tags and best-ranking syntactic parses to support future experiments with semantic answer retrieval and inference. Using this resource, we have calculated a baseline for word-overlap based answer retrieval (Hirschman et al., 1999) on the CBC4Kids data and found the method performs slightly better than on the REMEDIA corpus. We hope that our richly annotated version of the CBC4Kids corpus will become a standard resource, especially as a controlled environment for evaluating inference-based techniques.

1 Introduction

The goal of computer systems capable of simulating understanding with respect to reading a story and answering questions about it has attracted researchers since the early 1970s. We present our ongoing work on creating richly annotated corpus resources for NLQA that can provide input for a wide range of NLQA techniques and simultaneously support their evaluation and cross comparison.

2 Related Work

The challenge to computer systems of reading a story or article and demonstrating understanding through question answering was first addressed in Charniak's Ph.D. thesis (Charniak, 1972). That work showed the amount and diversity of both logical and common sense reasoning needed to link together what was said explicitly in the story or article and thereby to answer questions about it.

More recent work has stressed the value of reading comprehension exams as a research challenge in terms of (1) their targeting successive skill levels of human performance, and hence their potential to challenge automated systems to successively higher levels of performance (Hirschman et al., 1999), and (2) the existence of independently developed scoring algorithms and human performance measures, as an alternative to the special purpose evaluations developed for TREC Open Domain Question-Answering (Voorhees and Tice, 1999).

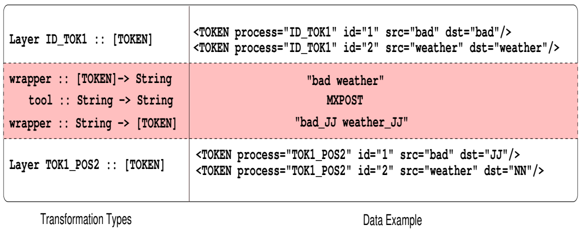
The first attempt to systematically determine the feasibility of reading comprehension tasks as a research challenge for automated systems was Deep Read (Hirschman et al., 1999). Deep Read established a baseline on a professionally-developed remedial reading comprehension test for children in grades 3-6 (ages 8-12), using a simple bag-of-words approach. Scoring essentially by word intersection with the answer key provided by the test designer, Deep Read's simple approach produced sentence-level answers that agreed with sentences supporting the answer key (a metric called **HumSent**, see below) 30% of the time. That was sufficient to establish reading comprehension tests as a tractable research problem for automated systems.[1] This work was followed in 2000 by both an ANLP-NAACL workshop on *Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*[2] and a Summer workshop on technology for reading comprehension QA at the Johns Hopkins University.[3]

3 Automatic Linguistic Annotation

Our work is driven by the following observation (Cotton and Bird, 2002): "With all the annotations expressed in the same data model, it becomes a straightforward matter to investigate the relationships between the various linguistic levels. Modeling the interaction between linguistic levels is a central concern."

The CBC4Kids corpus was developed at MITRE[4], based on a collection of newspaper stories for teenagers written for the CBC's WWW site.[5] To each article selected for inclusion in the corpus, Ferro and her colleagues added a set of 8-10 questions of various degrees of difficulty (Ferro, 2000). The corpus also includes one or more answers for each question in the form of a disjunction of a phrase or a clause (the "answer key").

Due to the wide availability of XML processing tools, we decided to define an XML DTD for the CBC4Kids corpus and to convert various automatically[6] obtained linguistic forms of annotation into XML and integrate them so as to provide a rich knowledge base for our own NLQA experiments and potential re-use by other groups. We selected a set of tools with the guiding principles of 1) public availability, 2) usefulness for our replication a Deep Read-style baseline system, and 3) quality of the automatic annotation. Because most available tools (with the exception of TTT, (Grover et al., 2000)) do not output XML, we had to develop a set of converters.



Transformation Types	Data Example
Layer ID_TOK1 :: [TOKEN]	<TOKEN process="ID_TOK1" id="1" src="bad" dst="bad"/> <TOKEN process="ID_TOK1" id="2" src="weather" dst="weather"/>
wrapper :: [TOKEN]-> String tool :: String -> String wrapper :: String -> [TOKEN]	"bad weather" MXPOST "bad_JJ weather_JJ"
Layer TOK1_POS2 :: [TOKEN]	<TOKEN process="TOK1_POS2" id="1" src="bad" dst="JJ"/> <TOKEN process="TOK1_POS2" id="2" src="weather" dst="NN"/>

Figure 1: Building a new layer of TOKEN tags.

Each sentence has three different representations: 1) the original string, 2) a list of tags labeled `TOKEN` encoding the results from linguistic tools that give information on words (POS tags, stems, etc.), 3) a list of trees (`PARSE`) corresponding to a non-terminal level, i.e. syntactic or dependency analyses. This is a compromise between redundancy and ease of use.

Because various forms of linguistic processing depend on the output of other tools, we wanted to make this processing history explicit. We devised a multi-layer annotation scheme in which an XML `process` attribute refers to a description of the input (token or tree), the output, and the tool used. Figure 1 shows how a layer of `TOKEN` is built. This annotation allows for easy stacking of mark-up for tokenization, part-of-speech (POS) tags, base forms, named entities, syntactic trees etc. (Figure 3).

Figure 4 and Figure 5 show the current status of our annotation "pipe tree" on the token and sentence levels, respectively, as described below[7].

[1] *Nota bene:* despite the name, the strand of research we report here makes no claims as to the cognitive aspects of human reading comprehension (Levelt and Kelter, 1982).

[2] `http://acl.ldc.upenn.edu/W/W00/` in which results were reported by other groups working on this same corpus

[3] `http://www.clsp.jhu.edu/ws2000/groups/reading/`

[4] The contact person for the corpus is Lisa Ferro (address see Section 7).

[5] `http://www.cbc4kids.ca`

[6] Note that the gold standard for the question answering task are the "gold answers", not perfect linguistic annotations.

[7] We call it a "pipe tree" because it represents a set of "pipe lines" with common initial sub-steps.

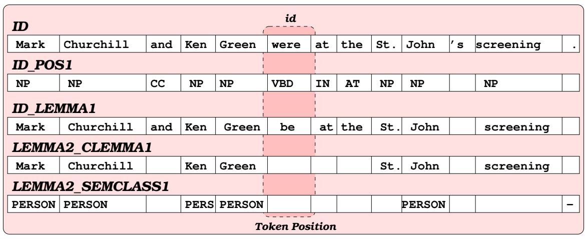

ID
id
Mark Churchill and Ken Green were at the St. John 's screening .
ID_POS1
NP NP CC NP NP VBD IN AT NP NP NP
ID_LEMMA1
Mark Churchill and Ken Green be at the St. John screening
LEMMA2_CLEMMA1
Mark Churchill Ken Green St. John screening
LEMMA2_SEMCLASS1
PERSON PERSON PERS PERSON PERSON -
Token Position

Figure 3: Multiple annotation layers.

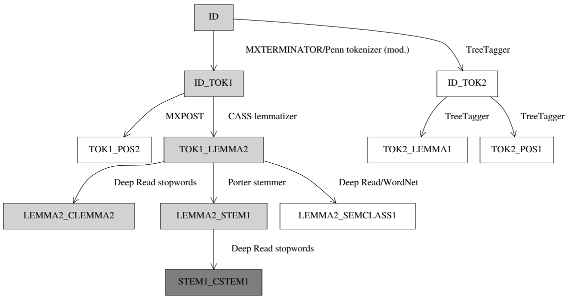


Figure 4: Annotation layers per token. The replicated Deep Read baseline system pipeline is highlighted.

Figure 2 gives an overview of our targeted annotation. A comprehensive description of the tools and structure can be found in the manual (Dalmas et al., 2003) distributed with the corpus.

The layers described here allow detailed comparisons of components' contribution for any question answering method by exploring different paths in the annotation "pipe tree".

We have implemented converters for all the tools listed (except the LTG tools, which output XML and hence do not need conversion) in Perl, and a master script that assembles the individual converters' output into a well-formed and valid XML document instance.

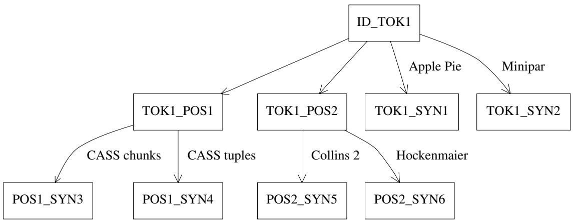


Figure 5: Annotation layers per sentence.

Difficulty	QC	R	P	AutSent	HumSent
Easy	237	0.74	0.18	0.75	0.74
Moderate	177	0.57	0.22	0.55	0.57
Difficult	67	0.49	0.19	0.43	0.43
Average	481	0.63	0.19	0.62	0.63

Table 1: Baseline evaluation using the STEM1_CSTEM1 layer according to question difficulty. QC is the number of questions.

This annotation is work in progress insofar as we are planning to include further layers featuring analyses of LT TTT, LT POS, LT CHUNK, named entity annotation using MITRE's Alembic (cf. Deep Read), the LTG MUC-7 system, as well as anaphora resolution software.

4 Baseline Results

This section describes our experiment replicating the baseline that was previously computed by Deep Read on the REMEDIA corpus, but here on the CBC4Kids data.

We began exploiting the `STEM1_CSTEM1` layer of our XML annotation scheme to get a baseline using stemmed lemmata of content words. The shaded path in Figure 4 shows these final layers we used and their ancestors in the linguistic pipeline, from token through lemma, stemming, stop-word removal, as in the Deep Read experiments.

We have implemented a batch QA system as a set of filters in the functional programming language Haskell.[8] The XML encoding of linguistic information greatly simplified the implementation part: the QA system was reduced to a program filtering a tree (the XML document containing story and questions) and computing intersection (overlap) on lists of tokens. Table 1 shows the results for the baseline using the `STEM1_CSTEM1` filter. The answers of the system are added to the XML file as a separate layer.

The evaluation metrics in Table 1 are the same as described in (Hirschman et al., 1999), namely **Recall**, **Precision**, **AutSent** and **HumSent**:[9]

[8] `http://www.haskell.org`

[9] *Cave lector:* The definitions for P and R in (Hirschman et al., 1999) appear to have been swapped.

Type	Tool	Process ID	Reference
Sentence Boundaries	MXTERMINATOR	`ID`	
Tokenization	Penn tokenizer.sed	`ID_TOK1`	
	Tree-Tagger (internal)	`ID_TOK2`	(Schmid, 1994)
	LT TTT	`ID_TOK3`	(Grover et al., 2000)
Part-of Speech	MXPOST	`TOK1_POS2`	(Ratnaparkhi, 1996)
	Tree-Tagger	`TOK2_POS1`	(Schmid, 1994)
	LT POS	`TOK3_POS3`	(Mikheev et al., 1999)
Lemmatization	CASS “stemmer”	`TOK1_LEMMA2`	(Abney, 1996)
	Tree-Tagger	`TOK2_LEMMA1`	(Schmid, 1994)
	morpha	`POS1_LEMMA3`	(Minnen et al., 2001)
Stemming	Porter stemmer	`LEMMA2_STEM1`	(Porter, 1980)
Stop-Word Filtering	Deep Read	`LEMMA2_CLEMMA2`	(Hirschman et al., 1999)
	Deep Read	`STEM1_CSTEM1`	(Hirschman et al., 1999)
Syntactic Analysis	Apple Pie Parser	`POS2_SYN1`	(Sekine and Grishman, 1995)
	Minipar relations	`TOK1_SYN2`	(Lin, 1998)
	CASS chunk trees	`POS1_SYN3`	(Abney, 1996)
	CASS dependency tuples	`POS1_SYN4`	(Abney, 1997)
	Collins parse trees	`POS2_SYN5`	(Collins, 1997)
	CCG parse trees	`POS2_SYN6`	(Hockenmaier and Steedman, 2002)
	LT CHUNK	`POS3_SYN7`	(Mikheev et al., 1999)
Named Entity Tagging	Deep Read (WordNet)	`LEMMA2_SEMCLASS1`	(Hirschman et al., 1999)
	MITRE Alembic	`TOK1_NE1`	(Aberdeen et al., 1995)
	LTG MUC-7	`SYN7_NE2`	(Mikheev et al., 1998)
Anaphora Resolution	*N.N.*	`SYN5_AR1`	*N.N.*

Figure 2: Annotation tools: Targeted list of layers.

Question Type	R	P	AutSent	HumSent
when	0.71	0.15	0.76	0.76
who/-se/-m	0.68	0.16	0.67	0.71
how	0.71	0.21	0.70	0.70
how many/much	0.62	0.08	0.63	0.67
what	0.66	0.26	0.63	0.65
which_np	0.70	0.08	0.60	0.60
where	0.58	0.14	0.56	0.56
how_att	0.56	0.15	0.56	0.56
what_np	0.59	0.18	0.56	0.56
why	0.57	0.23	0.52	0.51

Table 2: Baseline evaluation (STEM1_CSTEM1) according to question type.

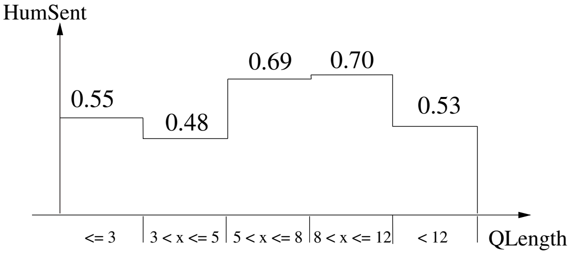


Figure 6: HumSent accuracy by length of question bag (STEM1_CSTEM1). The average bag length for QLength $\geq$ 12 is 14 words, with a maximum of 19 words.

$$\textbf{R} = |cw_{sa} \cap cw_{ha}| \;/\; |cw_{ha}|$$

$$\textbf{P} = |cw_{sa} \cap cw_{ha}| \;/\; |cw_{sa}|$$

$$\textbf{AutSent} = \#[\, sentence \mid \mathrm{R}(sentence) > 0 \,]$$

HumSent = *list of sentences considered as answers by a human annotator*

cw : content words

sa : system answer

ha : human answer (a phrase).

Sentences containing the answer picked by human and machine, respectively, are also marked up in XML. We have developed an automated evaluation program that can currently take into account three parameters: the difficulty of the answer (as annotated in the original CBC4Kids release, see below), the question type (based on the WH-word) and the length of the question bag. Table 2 and Figure 6 show some of the results.

5 Discussion

As already noted, the questions constructed for the CBC4Kids corpus are rated as to their difficulty (Ferro, 2000):

> “Easy: Uses exact wording from the text and/or the question and answer are

> close to each other in the text. [...] Moderate: Some paraphrasing from the text and/or the question and answer aren't close to each other in the text. [...] Difficult: Very or entirely different words are used in question; lots of other tempting but incorrect answers are in the story; subtle knowledge is required to answer the question."

Table 1 shows the performance of the baseline system, broken down by difficulty class. For all scoring metrics other than Precision (P), the table shows a strong correlation between the retrieval score and the class assigned according to Ferro's guidelines for Q&A writing. As for Precision, it is not really significant because human answers are phrases and our system outputs a sentence as answer. However, Precision allows us to see from Table 2 that very short answers are expected for `HOW_MANY`, `HOW_MUCH` and `WHICH_NP` questions. This is not surprising for `HOW_MANY` or `HOW_MUCH` questions, for which expected answers are very short named entities (*How many people? → twenty-five*). But for `WHICH_NP` questions, they are in fact expecting a named entity and especially a proper name (*In which city / Which two African leaders / Which U.S. states*). The length of the expected answer is not so obvious for other questions that expect named entities, such as `WHEN` questions. The main reason for this is that the corpus itself asks for a story comprehension and not for general answers as in the TREC evaluation. For example, the following `WHEN` question *When did Wilson climb onto the second-floor balcony?* expects a long answer: *when he heard the cries of Westley, Hughes, and their children.*

As already noted by Hirschman and co-workers for Deep Read, the Recall (R) and HumSent metrics behave in a similar manner. But here for `WHY` and `WHICH_NP` questions, we notice a significant difference: generally these questions contain one or two words repeated all along the story (name of the main character for instance) and therefore the possibility of a tie between possible answers becomes more important. This is particulary true when the question bag is either short (between 3 and 5 words) or very long (more than 12 words, see Figure 6).

Since an answer occurs generally only once in a story, we cannot rely on techniques using redundancy. But the advantage of a short text is also that deeper NLP techniques can be used appropriately.

We obtain significantly higher Recall scores for CBC4Kids compared to Deep Read's performance on the REMEDIA corpus, although the language used in the latter is targeted at a much younger age group. Independent experiments at MITRE have also yielded higher performance scores for CBC4Kids.[10]

One possible explanation for the overall higher scores is that the CBC4Kids questions were composed with a NLQA system in mind: for instance, question authors were told to avoid anaphoric references in the questions (Ferro, 2000), which are quite frequent in the REMEDIA questions. Another possible explanation is that the shorter sentence length due to the younger audience fragments information across sentences, thus decreasing term overlap at the given sentence granularity.[11] It remains to be investigated how much the purpose of text production impacts reading comprehension simulation results, as the REMEDIA text and questions were not authored with an informative purpose in mind. In the CBC4Kids case, the text was pre-existing and created with informative intent, but the questions were created a posteriori; hence both methods are artificial, but in different ways.

It is quite easy to carry out an error analysis once the results of the system have been encoded in XML. A simple XSL stylesheet can be sufficient for extracting questions and answers we want to analyse (Figures 7 and 8).

6 Future Work

This section describes some of the experiments we have planned for the future. These are likely to require adding further layers with linguistic annotation.

6.1 Towards Predicate/Argument Structure

Surface overlap metrics are intrinsically limited, since they cannot, for instance, distinguish be-

[10] Ben Wellner, personal communication.

[11] Lisa Ferro, personal communication.

tween *man bites dog* and *dog bites man*—they are a-semantic in nature. To overcome this, we are planning to utilize the various syntactic representations (cf. Figure 2) to obtain predicate-argument structures (`bite(man, dog)` versus `bite(dog, man)`), which allow for higher precision. One path of investigation is to induce the grammar underlying the corpus, to filter the top-*n* most likely productions and to subsequently add semantic composition rules manually. Another path worthwhile exploring is learning the mappings from chunks to predicate-argument structures in a supervised regime. Once we have more robust methods of predicate-argument structures, we will be able to explore shallow inferences for NLQA (Webber et al., 2002) in the controlled environment that CBC4Kids provides.

One path of investigation is to induce the grammar underlying the corpus, to filter the top-n most likely productions and to subsequently add semantic composition rules by hand. Another path worthwhile exploring is learning the mappings from chunks to Quasi-Logical Forms (QLF) in a supervised regime.

Once we have more robust methods of QLF extraction, we will be able to explore shallow inferences for NLQA (Webber et al., 2002) in the controlled environment that CBC4Kids provides.

6.2 Comparing Answers

Each question in the CBC4Kids corpus receives at least one answer determined by a human annotator. We would like to use this rich annotation to begin a study on detecting multiple answer cases which is part of the current roadmap for research in NLQA in the TREC community (Burger et al., 2001).

Few have so far proposed to consider the evaluation of NLQA systems retrieving complex answers, but recently (Buchholz and Daelemans, 2001) and (Webber et al., 2002) have suggested different classification sets for comparing answers. This would allow NLQA systems to provide multiple answers linked together by labels expressing their relationship, such as “P implies Q”, “P and Q are equivalent”, “P and Q are alternative answers” (exclusiveness), “P and Q provide a collective answer” (complementarity), and others (Webber et al., 2002).

One goal of this thread of research is to build a practical framework for evaluation multiple answers that allows answer comparison.

7 Conclusions

We have described the process of creating rich annotation of the CBC4Kids corpus of news for children. The chosen XML annotation architecture is a compromise that allows for multilayer annotation whilst simplifying the integration of added linguistic knowledge from heterogeneous toolsets. The architecture reduces many applications to a sequence of selections and functional mappings over the annotation layers. The application of such a scheme is by no means restricted to the corpus under consideration; we intend to reuse it, notably for textual resources from the biomedical domain.

On the basis of the resulting dataset, CBC4Kids, we have replicated an evaluation performed by (Hirschman et al., 1999), but on the CBC4Kids corpus. This will serve as a basis for our future experiments involving robust semantic construction and inference for question answering.

We do not know of any other corpus that has been automatically annotated with comparably rich strata of linguistic knowledge and believe that the corpus can be a valuable resource also for other NLQA research groups.

The corpus is distributed by MITRE, with layers as given above, including answers given by our system for the Deep Read baseline. Please contact Lisa Ferro directly for a copy.[12]

Acknowledgments. We are grateful to Lynette Hirschman and Lisa Ferro at MITRE, who provided us with the initial CBC4Kids corpus. We would also like to thank the authors of all the tools mentioned and used in this paper for making them available to the academic community. Thanks to Julia Hockenmaier, Maria Lapata, Dekang Lin, Katja Markert, Satoshi Sekine and Bill Wellner and three anonymous reviewers for helpful advice and feedback.

We would like to acknowledge the financial support of the German Academic Exchange Service (DAAD) under grant D/02/01831, of Linguit

[12] `lferro@mitre.org`

GmbH (research contract UK-2002/2), and of the School of Informatics, University of Edinburgh.

References

J. Aberdeen, D. Day, L. Hirschman, P. Robinson, and M. Vilain. 1995. MITRE: Description of the Alembic system used for MUC-6. *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 141–155.

S. Abney. 1996. Partial parsing via finite-state cascades. *Journal of Natural Language Engineering*, 2(4):337–344.

S. Abney. 1997. Part-of-speech tagging and partial parsing. *Corpus-Based Methods in Language and Speech Processing*.

S. Buchholz and W. Daelemans. 2001. Complex answers: A case study using a WWW question answering system. *Journal of Natural Language Engineering*, 7:301–323.

J. Burger, C. Cardie, V. Chaudhri, S. Harabagiu, D. Israel, Chr. Jacquemin, C.-Y. Lin, S. Mariorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weischedel. 2001. Issues, tasks and program structures to roadmap research in question and answering. *NIST*.

E. Charniak. 1972. *Toward a Model of Children's Story Comprehension*. Ph.D. thesis, Massachusetts Institute of Technology.

M. J. Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, Madrid. Association for Computational Linguistics.

S. Cotton and S. Bird. 2002. An integrated framework for treebanks and multilayer annotations. In *Proceedings of the Third International Conference on Language Resources and Evaluation*.

T. Dalmas, J. L. Leidner, B. Webber, C. Grover, and J. Bos. 2003. *Annotating CBC4Kids: A Corpus for Reading Comprehension and Question Answering Evaluation. (Technical Report)*. School of Informatics, University of Edinburgh.

L. Ferro. 2000. *Reading Comprehension Tests: Guidelines for Question and Answer Writing. (Unpublished Technical Report)*. The MITRE Corporation.

C. Grover, C. Matheson, A. Mikheev, and M. Moens. 2000. LT TTT–a flexible tokenisation tool. In *Proceedings of Second International Conference on Language Resources and Evaluation (LREC 2000)*.

L. Hirschman, M. Light, E. Breck, and J. D. Burger. 1999. Deep Read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

J. Hockenmaier and M. Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*.

W. J. M. Levelt and S. Kelter. 1982. Surface form and memory in question answering. *Cognitive Psychology*, 14:78–106.

D. Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*.

A. Mikheev, C. Grover, and M. Moens. 1998. Description of the LTG system used for MUC-7. In *Proceedings of 7th Message Understanding Conference (MUC-7)*.

A. Mikheev, C. Grover, and M. Moens. 1999. XML tools and architecture for named entity recognition. *Journal of Markup Languages: Theory and Practice*, 3:89–113.

G. Minnen, J. Carroll, and D. Pearce. 2001. Applied morphological processing of English. *Journal of Natural Language Engineering*, 7(3):207–223.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

A. Ratnaparkhi. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the Empirical Methods in Natural Language Processing Conference*.

H. Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. *International Conference on New Methods in Language Processing*.

S. Sekine and R. Grishman. 1995. A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings Fourth International Workshop on Parsing Technologies*.

E. Voorhees and Dawn M. Tice. 1999. The TREC-8 question answering track evaluation. In E. M. Voorhees and D. K. Harman, editors, *The Eighth Text REtrieval Conference (TREC 8)*, NIST Special Publication 500–246, pages 1–24, Gaithersburg, Maryland, November 17-19, 1999. National Institute of Standards and Technology.

B. L. Webber, C. Gardent, and J. Bos. 2002. Position statement: Inference in question answering. In *Proceedings of the LREC Workshop on Question Answering: Strategy and Resources*, Las Palmas, Gran Canaria, Spain.

A Sample Story from CBC4Kids

Tragedy Strikes a Northern Village
January 4, 1998

The six hundred mostly Inuit residents of the northern Quebec village of Kangiqsualujjuaq had planned to bury the bodies of nine of their friends and children in a funeral this afternoon. But the bad weather that resulted in their deaths has also delayed the funeral until Tuesday.

Kangiqsualujjuaq* is about 1,500 kilometres north of Montreal, at the mouth of the George River on Ungava Bay. This region is known as Nunavik.

An avalanche hit the town's Satuumavik school gymnasium in the Northern Quebec community early Friday morning.

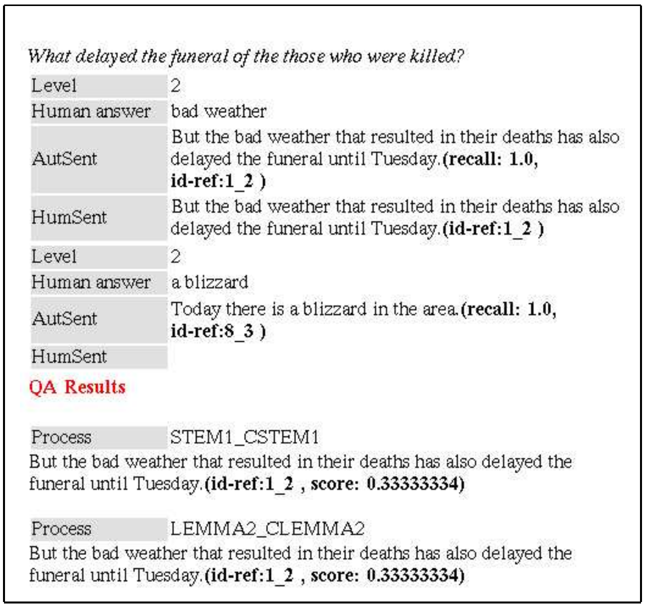
[...]

Principal Jean Leduc said an inquiry commissioned by the local school board after the earlier avalanche had recommended that fences be built. The fences were never built. Speculation on the cause of the avalanche centered on a ceremonial gun salute at midnight, 90 minutes before the snow crashed in. Villagers wondered if the shots set in motion vibrations that eventually caused the avalanche, while others wondered if music from the dance had played a role.

Police and avalanche experts will travel to the village to investigate the tragedy. Quebec Premier Lucien Bouchard announced there will be a full public inquiry into the disaster.

Questions

How far is Kangiqsualujjuaq from Montreal?
When did the avalanche hit the school?
Where was Mary Baron when the avalanche hit?
How many people were seriously injured by the avalanche?
What delayed the funeral of the those who were killed?
What could have possibly prevented the tragedy?
Who will investigate the tragedy?



What delayed the funeral of the those who were killed?

Level	2
Human answer	bad weather
AutSent	But the bad weather that resulted in their deaths has also delayed the funeral until Tuesday.**(recall: 1.0, id-ref:1_2)**
HumSent	But the bad weather that resulted in their deaths has also delayed the funeral until Tuesday.**(id-ref:1_2)**
Level	2
Human answer	a blizzard
AutSent	Today there is a blizzard in the area.**(recall: 1.0, id-ref:8_3)**
HumSent	

QA Results

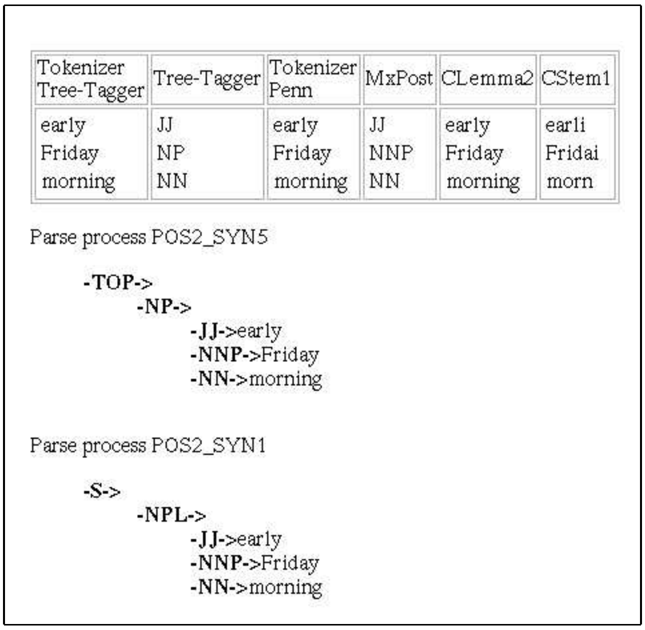
Process	STEM1_CSTEM1

But the bad weather that resulted in their deaths has also delayed the funeral until Tuesday.**(id-ref:1_2 , score: 0.33333334)**

Process	LEMMA2_CLEMMA2

But the bad weather that resulted in their deaths has also delayed the funeral until Tuesday.**(id-ref:1_2 , score: 0.33333334)**

Figure 7: HTML view for a question. The *score* given for each answer corresponds to the overlap between a candidate answer sentence and the question (WdAnsRecall).



Tokenizer Tree-Tagger	Tree-Tagger	Tokenizer Penn	MxPost	CLemma2	CStem1
early Friday morning	JJ NP NN	early Friday morning	JJ NNP NN	early Friday morning	earli Fridai morn

Parse process POS2_SYN5

```
-TOP->
    -NP->
        -JJ->early
        -NNP->Friday
        -NN->morning
```

Parse process POS2_SYN1

```
-S->
    -NPL->
        -JJ->early
        -NNP->Friday
        -NN->morning
```

Figure 8: Partial HTML view of linguistic layers for a human answer.

GETARUNS: A HYBRID SYSTEM FOR SUMMARIZATION AND QUESTION ANSWERING

Rodolfo Delmonte
Ca' Garzoni-Moro, San Marco 3417
Università "Ca Foscari"
30124 - VENEZIA
Tel. 39-041-2579464/52/19 - Fax. 39-041-2577910
E-mail: delmont@unive.it - website: project.cgm.unive.it

Abstract

Information Retrieval, Summarization and Question Answering need accurate linguistic information with a much higher coverage than what is being recently offered by currently available parsers, so we assume that the starting point of any interesting application in those fields must necessarily be a good syntactic-semantic parser. The system presented in the paper has undergone extensive testing and the parser has been trained on available testsuites and the Remedia corpus texts, one of which will be commented in some detail.

1. Introduction

In this paper we present an approach to natural language processing which we define as "hybrid", in which symbolic and statistical approaches are reconciled. In particular, the search space for syntactic analysis is inherently deterministic, in that it is severely limited by the grammar of the specific language, which in turn is constituted by a set of peripheral rules to be applied in concomitance with the more general rules of core grammar. Variations on these rules are only determined by genre, which can thus contribute a new set of peripheral rules, or sometimes just a set of partially overlapping rules to the ones already accepted by a given linguistic community. As far as parsing is concerned, we purport the view that the implementation of sound parsing algorithm must go hand in hand with sound grammar construction. Extragrammaticalities can be better coped with within a solid linguistic framework rather than without it. Our parser is a rule-based deterministic parser in the sense that it uses a lookahead and a Well-Formed Substring Table to reduce backtracking (Delmonte, 2000a). It also implements Finite State Automata in the task of tag disambiguation (Delmonte, 2000b), and produces multiwords whenever lexical information allows it. In our parser (Delmonte, 2000c), we use a number of parsing strategies and graceful recovery procedures which follow a strictly parameterized approach to their definition and implementation. Recovery procedures are also used to cope with elliptical structures and uncommon orthographic and punctuation patterns. A shallow or partial parser, in the sense of Abney(1996), is also implemented and always activated before the complete parse takes place, in order to produce the default baseline output to be used by further computation in case of total failure. In that case partial semantic mapping will take place where no Logical Form is being built and only referring expressions are asserted in the Discourse Model – but see below.

1.1 Intelligent Sentence Extraction

The use of NLP techniques in IR/IE is in our opinion mandatory as a preliminary step towards the summarization itself: Intelligent Sentence Extraction (hence ISE) is the first step to produce a summary which in our case is strongly based on the use of NLP techniques. Its main features are: full tokenization, FSA restricted multiword creation, POS tagging limited to content words. The aim of ISE is to produce an 800 words extract to be used by GETARUNS, the system for summarization proper.

In this preliminary phase we rely on statistical techniques for extracting keywords to be used as topics. However before Sentence Extraction takes place, we perform a search for Topic Density which for each most frequent wordform computes a lemma and a tag. This allows to go beyond stemming by taking into account only noun/verbs and denominal adjectives. The final Sentence Extraction step computes Dangling Sentences, those depending on some previous discourse element and tries to amend that by adding the previous sentence to the final output.

2. General Parser and Anaphora Resolution Module

GETARUNS, the system for text understanding developed at the University of Venice, is equipped with three main modules: a lower module for parsing where sentence strategies are implemented; a middle module for semantic interpretation and discourse model construction which is cast into Situation Semantics; and a higher module where reasoning and generation takes place (Delmonte, 2000c).

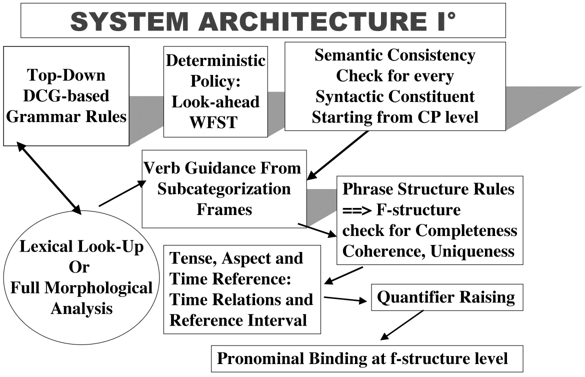
The system is based on LFG theoretical framework (see Bresnan, J. 2001) and has a highly interconnected modular structure. It is a top-down depth-first DCG-based parser written in Prolog which uses a strong deterministic policy by means of a lookahead mechanism with a WFST to help recovery when failure is unavoidable due to strong attachment ambiguity.

It is divided up into a pipeline of sequential but independent modules which realize the subdivision of a parsing scheme as proposed in LFG theory where a c-structure is built before the f-structure can be projected by unification into a DAG (Delmonte, 2002).

Syntactic and semantic information is accessed and used as soon as possible: in particular, both categorial and subcategorization information attached to predicates in the lexicon is extracted as soon as the main predicate is processed, be it adjective, noun or verb, and is used to subsequently restrict the number of possible structures to be built (see Fig.1 below).

The output of grammatical modules is fed then onto the Binding Module(BM) which activates an algorithm for anaphoric binding. Antecedents for pronouns are ranked according to grammatical function, semantic role, inherent features and their position at f-structure. Eventually, this information is added into the original f-structure graph and then passed on to the Discourse Module(DM).

Fig.1 GETARUNS' Sentence Level Modules


SYSTEM ARCHITECTURE I°
Top-Down DCG-based Grammar Rules
Deterministic Policy: Look-ahead WFST
Semantic Consistency Check for every Syntactic Constituent Starting from CP level
Verb Guidance From Subcategorization Frames
Phrase Structure Rules ==> F-structure check for Completeness Coherence, Uniqueness
Lexical Look-Up Or Full Morphological Analysis
Tense, Aspect and Time Reference: Time Relations and Reference Interval
Quantifier Raising
Pronominal Binding at f-structure level

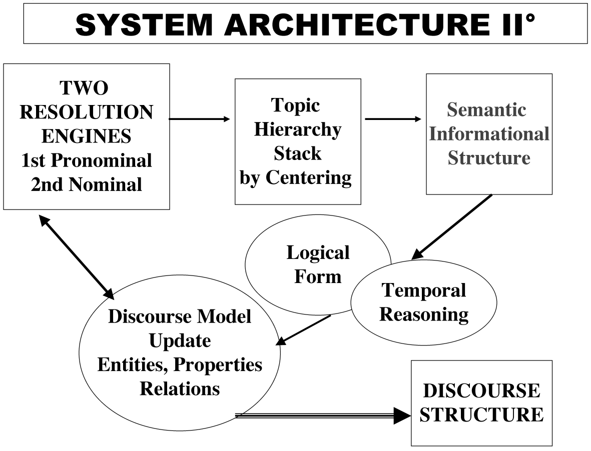
The grammar is equipped with a core lexicon containing most frequent 5000 fully specified inflected word forms where each entry is followed by its lemma and a list of morphological features, organised in the form of attribute-value pairs. However, morphological analysers for English are also available with big root dictionaries (25,000 for English) which only provide for syntactic subcategorization, though. In addition to that there are all lexical form provided by a fully revised version of COMLEX, and in order to take into account phrasal and adverbial verbal compound forms, we also use lexical entries made available by UPenn and TAG encoding. Their grammatical verbal syntactic codes have then been adapted to our formalism and are used to generate a subcategorization schemes with an aspectual and semantic class associated to it – however no restrictions can reasonably be formulated on arguments of predicates. Semantic inherent features for Out of Vocabulary Words , be they nouns, verbs, adjectives or adverbs, are provided by a fully revised version of WordNet - plus EuroWordnet, with a number of additions coming from computer, economics, and advertising semantic fields - in which we used 75 semantic classes similar to those provided by CoreLex.

2.1 The Upper Module

GETARUNS, has a highly sophisticated linguistically based semantic module which is used to build up the Discourse Model. Semantic processing is strongly modularized and distributed amongst a number of different submodules which take care of Spatio-Temporal

Reasoning, Discourse Level Anaphora Resolution, and other subsidiary processes like Topic Hierarchy which cooperate to find the most probable antecedent of coreferring and cospecifying referential expressions when creating semantic individuals. These are then asserted in the Discourse Model (hence the DM), which is then the sole knowledge representation used to solve nominal coreference. The system uses two resolution submodules which work in sequence: they constitute independent modules and allow no backtracking. The first one is fired whenever a free sentence external pronoun is spotted; the second one takes the results of the first submodule and checks for nominal anaphora. They have access to all data structures contemporarily and pass the resolved pair, anaphor-antecedent to the following modules. See Fig.2 below. Semantic Mapping is performed in two steps: at first a Logical Form is produced which is a structural mapping from DAGs onto of unscoped well-formed formulas. These are then turned into situational semantics informational units, infons which may become facts or sits. Each unit has a relation, a list of arguments which in our case receive their semantic roles from lower processing – a polarity, a temporal and a spatial location index.

Fig.2 GETARUNS' Discourse Level Modules


SYSTEM ARCHITECTURE II°
TWO RESOLUTION ENGINES 1st Pronominal 2nd Nominal
Topic Hierarchy Stack by Centering
Semantic Informational Structure
Logical Form
Temporal Reasoning
Discourse Model Update Entities, Properties Relations
DISCOURSE STRUCTURE

2.1 Building the Discourse Model

All entities and their properties are asserted in the DM with the relations in which they are involved; in turn the relations may have modifiers - sentence level adjuncts and entities may also have modifiers or attributes. Each entity has a polarity and a couple of spatiotemporal indices which are linked to main temporal and spatial locations if any exists; else they are linked to presumed time reference derived from tense and aspect computation. Entities are mapped into semantic individual with the following ontology: on first occurrence of a referring expression it is asserted as an INDividual if it is a definite or indefinite expression; it is asserted as a CLASS if it is quantified (depending on quantifier type) or has no determiner. Special individuals are ENTs which are associated to discourse level anaphora which bind relations and their arguments. Finally, we have LOCs for main locations, both spatial and temporal. If it has a cardinality determined by a number, it is plural or it is quantified (depending on quantifier type) it is asserted as a SET and the cardinality is simply inferred in case of naked plural, i.e. in case of collective nominal expression it is set to 100, otherwise to 5. On second occurrence of the same nominal head the semantic index is recovered from the history list and the system checks whether it is the same referring expression:

- in case it is definite or indefinite with a predicative role and no attributes nor modifiers nothing is done;
- in case it has different number - singular and the one present in the DM is a set or a class nothing happens;
- in case it has attributes and modifiers which are different and the one present in the DM has none, nothing happens;
- in case it is quantified expression and has no cardinality, and the one present in the DM is a set or a class, again nothing happens.

In all other cases a new entity is asserted in the DM which however is also computed as being included in (a superset of) or by (a subset of) the previous entity.

2.2 The Partial System

The system switches automatically to a Partial or Shallow modality in case of failure in the Complete modality. The Partial system has standard components. like a statistical/syntactic tagger and a cascaded shallow parser which in a final run turns syntactic constituents into functionally labelled arguments/adjuncts. The "Partial" modality when activated allows the system to relax both parsing and semantic mapping constraints thus producing a "partial" semantic mapping. Here partial refers both to the possibility that the parser output a parse which covers only parts of the input sentence, leaving out some linguistic material unparsed. It may also refer to cases in which the predicate-argument mapping does not succeed due to lack of adequate subcategorization information. Finally it may refer to the cases of missed anaphora resolution, in which the actual semantic mapping does not simply make the right inferences due to the presence of bridging expressions which are mismatched with their coreferring expression, or else for the lack of the appropriate inferential information to fire the inference in the first place. In the "Partial" modality, the list of Essential Properties of Linguistic Objects which can be derived is as follows:

1. Grammatical properties of linguistic objects
   i. Functional Features of NPs
   ii. Selectional Restrictions of NPs #
   iii. Grammatical Functions of NPs
   iv. Semantic Roles of all Phrases
2. Topic Hierarchies for referring expressions
3. Discourse Model of facts and locations
4. External World Knowledge Base

where we indicate with a grid Selectional restrictions of NPs that will only be used to induce the right adjunct/oblique semantic role in presence of a given preposition. In addition, there is no attempt at producing a full semantic interpretation, no logical form is being generated due to the uncertainty at clause structure building.

2.3 Getaruns at work

We will show how Getaruns computes the DM by presenting the output of the system for the «Maple Syrup» text made available by Mitre for the ANLP2000 Workshop(see Hirschman et al.). Here below is the original text which is followed by the DM with the Semantic Database of Entities and Relations of the Text World.

> How Maple Syrup is Made
>
> Maple syrup comes from sugar maple trees. At one time, maple syrup was used to make sugar. This is why the tree is called a "sugar" maple tree.
>
> Sugar maple trees make sap. Farmers collect the sap. The best time to collect sap is in February and March. The nights must be cold and the days warm.
>
> The farmer drills a few small holes in each tree. He puts a spout in each hole. Then he hangs a bucket on the end of each spout. The bucket has a cover to keep rain and snow out. The sap drips into the bucket. About 10 gallons of sap come from each hole.
>
> 1. Who collects maple sap? (Farmers)
> 2. What does the farmer hang from a spout?
> (A bucket)
> 3. When is sap collected? (February and March)
> 4. Where does the maple sap come from?
> (Sugar maple trees)
> 5. Why is the bucket covered?
> (to keep rain and snow out)

2.4 Discourse Model for the text organized sentence by sentence

We list here below the DM related to the most relevant sentences addressed by the QA module:

```
1.How Maple Syrup is Made
loc(infon1, id1, [arg:main_tloc, arg:tr(f2_es1)])
class(infon2, id2)
fact(infon3, Maple, [ind:id2], 1, id1, univ)
fact(infon4, inst_of, [ind:id2, class:edible_substance], 1, univ,
univ)
fact(infon5, isa, [ind:id2 , class:Syrup], 1, id1, univ)
ind(infon6, id3)
fact(infon7, inst_of, [ind:id3, class:plant_life], 1, univ, univ)
fact(infon8, isa, [ind:id3, class:Maple], 1, id1, univ)
in(infon9, id3, id2)
fact(id5, make, [agent:id2, theme_aff:id4], 1, tes(f2_es1), univ)
fact(infon13, isa, [arg:id5, arg:ev], 1, tes(f2_es1), univ)
fact(infon14, isa, [arg:id6, arg:tloc], 1, tes(f2_es1), univ)
fact(infon15, plu_perf, [arg:id6], 1, tes(f2_es1), univ)
fact(infon16, time, [arg:id5, arg:id6], 1, tes(f2_es1), univ)
fact(infon17, h ow, [arg:id5], 1, tes(f2_es1), univ)
before(tes(f2_es1), tes(f2_es1))
includes(tr(f2_es1), id1)

2.Maple syrup comes from sugar maple trees
loc(infon20, id7, [arg:main_sloc, arg:tree])
in(infon21, id8, id7)
set(infon22, id8)
card(infon23, id8, 5)
fact(infon24, sugar_maple, [ind:id8], 1, id1, id7)
fact(infon25, inst_of, [ind:id8, class:plant_life], 1, univ, univ)
fact(infon26, isa, [ind:id8, class:tree], 1, id1, id7)
ind(infon27, id9)
fact(infon28, inst_of, [ind:id9, class:plant_life], 1, univ, univ)
fact(infon29, isa, [ind:id9, class:sugar_maple], 1, univ, univ)
in(infon30, id9, id3)
class(infon32, id10)
fact(infon33, isa, [ind:id10, class:sugar], 1, univ, univ)
fact(infon34, of, [arg:id10, specif:id9], 1, univ, univ)
fact(id11, come, [actor:id2, locat:id8] , 1, tes(f1_es2), id7)
fact(infon38, isa, [arg:id11, arg:ev], 1, tes(f1_es2), id7)
fact(infon39, isa, [arg:id12, arg:tloc], 1, tes(f1_es2), id7)
fact(infon40, pres, [arg:id12], 1, tes(f1_es2), id7)
fact(infon41, time, [arg:id11, arg:id12], 1, tes(f1_es2), id7)
during(tes(f1_es2), tes(f2_es1))
includes(tr(f1_es2), univ)
```

3.At one time, maple syrup was used to make sugar

```
class(infon42, id13)
fact(infon43, inst_of, [ind:id13, class:substance], 1, univ, univ)
fact(infon44, isa, [ind:id13, class:sugar], 1, univ, id7)
in(infon45, id13, id9)
fact(id14, make, [agent:id2, theme_aff:id13], 1, tes(finf1_es3), id7)
fact(infon47, isa, [arg:id14, arg:ev], 1, tes(finf1_es3), id7)
fact(infon48, isa, [arg:id15, arg:tloc], 1, tes(finf1_es3), id7)
fact(infon49, pres, [arg:id15], 1, tes(finf1_es3), id7)
fact(infon50, time, [arg:id14, arg:id15], 1, tes(finf1_es3), id7)
fact(id16, use, [theme_aff:id2, result:id14], 1, tes(f2_es3), id7)
fact(infon51, isa, [arg:id16, arg:pr], 1, tes(f2_es3), id7)
fact(infon52, isa, [arg:id17, arg:tloc], 1, tes(f2_es3), id7)
fact(infon53, plu_perf, [arg:id17], 1, tes(f2_es3), id7)
fact(infon54, time, [arg:id16, arg:id17], 1, tes(f2_es3), id7)
fact(infon55, at_one_time, [arg:id16], 1, tes(f2_es3), id7)
before(tes(f2_es3), tes(f2_es1))
includes(tr(f2_es3), univ)
```

4.This is why the tree is called a "sugar" maple tree

```
ent(infon61, id18)
fact(infon62, prop, [arg:id18, disc_set:[id16:use:[theme_aff:id2, result:id14]]], 1, univ, id7)
ind(infon63, id19)
fact(infon64, tree, [nil:id19], 1, univ, id7)
fact(infon65, inst_of, [ind:id19, class:plant_life], 1, univ, univ)
fact(infon66, isa, [ind:id19, class:tree], 1, univ, id7)
in(infon67, id19, id8)
ind(infon68, id20)
fact(infon69, inst_of, [ind:id20, class:thing], 1, univ, univ)
fact(infon70, isa, [ind:id20, class:reason], 1, univ, id7)
fact(infon72, reason, [nil:id18], 1, univ, id7)
fact(id21, be, [prop:infon72], 1, tes(f15_es4), id7)
fact(id22, call, [actor:id19, theme:id10, prop:infon72], 1, tes(f15_es4), id7)
fact(infon73, isa, [arg:id22, arg:st], 1, tes(f15_es4), id7)
fact(infon74, isa, [arg:id23, arg:tloc], 1, tes(f15_es4), id7)
fact(infon75, pres, [arg:id23], 1, tes(f15_es4), id7)
fact(infon82, time, [arg:id22, arg:id23], 1, tes(f5_es4), id7)
during(tes(f15_es4), tes(f2_es3))
includes(tr(f15_es4), univ)
```

5.Sugar maple trees make sap

```
class(infon85, id24)
fact(infon86, inst_of, [ind:id24, class:substance], 1, univ, univ)
fact(infon87, isa, [ind:id24, class:sap], 1, univ, id7)
fact(id26, make, [agent:id8, theme_aff:id24], 1, tes(f1_es5), id7)
fact(infon92, isa, [arg:id26, arg:ev], 1, tes(f1_es5), id7)
fact(infon93, isa, [arg:id27, arg:tloc], 1, tes(f1_es5), id7)
fact(infon94, pres, [arg:id27], 1, tes(f1_es5), id7)
fact(infon98, time, [arg:id26, arg:id27], 1, tes(f1_es5), id7)
during(tes(f1_es5), tes(f15_es4))
indudes(tr(f1_es5), univ)
```

6.Farmers collect the sap

```
set(infon99, id28)
card(infon100, id28, 5)
fact(infon101, inst_of, [ind:id28, class:man], 1, univ, univ)
fact(infon102, isa, [ind:id28, class:farmer], 1, univ, id7)
fact(id29, collect, [agent:id28, theme_aff:id24], 1, tes(f1_es6), id7)
fact(infon105, isa, [arg:id29, arg:ev], 1, tes(f1_es6), id7)
fact(infon106, isa, [arg:id30, arg:tloc], 1, tes(f1_es6), id7)
fact(infon107, pres, [arg:id30], 1, tes(f1_es6), id7)
fact(infon108, time, [arg:id29, arg:id30], 1, tes(f1_es6), id7)
during(tes(f1_es6), tes(f1_es5))
includes(tr(f1_es6), univ)
```

7.The best time to collect sap is in February and March

```
ind(infon110, id32)
fact(infon111, best, [ind:id32], 1, univ, id7)
fact(infon112, inst_of, [ind:id32, class:time], 1, univ, univ)
fact(infon113, isa, [ind:id32, class:time], 1, univ, id7)
set(infon114, id33)
card(infon115, 2)
fact(infon116, inst_of, [ind:id33, class:time], 1, univ, univ)
fact(infon117, isa, [ind:id33, class:[march, February]], 1, univ, id7)
fact(id35, collect, [agent:id28, theme_aff:id24], 1, tes(finf1_es7), id7)
fact(infon118, isa, [arg:id35, arg:ev], 1, tes(finf1_es7), id7)
fact(infon119, isa, [arg:id36, arg:tloc], 1, tes(finf1_es7), id7)
fact(infon120, nil, [arg:id36], 1, tes(finf1_es7), id7)
fact(infon121, [march, February], [arg:id32], 1, univ, id7)
fact(id37, be, [prop:id35, prop:infon130], 1, tes(f1_es7), id7)
fact(infon122, isa, [arg:id37, arg:st], 1, tes(f1_es7), id7)
fact(infon123, isa, [arg:id38, arg:tloc], 1, tes(f1_es7), id7)
fact(infon124, pres, [arg:id38], 1, tes(f1_es7), id7)
fact(infon125, time, [arg:id37, arg:id38], 1, tes(f1_es6), id7)
during(tes(f1_es7), tes(f1_es6))
includes(tr(f1_es7), univ)
```

……..

9.The farmer drills a few small holes in each tree

```
class(infon163, id38)
fact(infon164, small, [ind:id38], 1, univ, id7)
fact(infon165, inst_of, [ind:id38, class:place], 1, univ, univ)
fact(infon166, isa, [ind:id38, class:holes], 1, univ, univ)
fact(id47, drill, [agent:id28, theme_aff:id38], 1, tes(f1_es9), id7)
fact(infon170, isa, [arg:id47, arg:ev], 1, tes(f1_es9), id7)
fact(infon171, isa, [arg:id48, arg:tloc], 1, tes(f1_es9), id7)
fact(infon172, pres, [arg:id48], 1, tes(f1_es9), id7)
fact(infon173, in, [arg:id45, locat:id19], 1, tes(f1_es9), id7)
fact(infon157, time, [arg:id47, arg:id48], 1, tes(f1_es9), id7)
during(tes(f1_es9), tes(f1_es8))
includes(tr(f1_es9), univ)
```

…….

12.The bucket has a cover to keep rain and snow out

```
class(infon218, id59)
fact(infon219, inst_of, [ind:id59, class:thing], 1, univ, univ)
fact(infon220, isa, [ind:id59, class:cover], 1, id53, id7)
fact(infon222, cover, [nil:id54], 1, id53, id7)
fact(id60, have, [actor:id54, prop:infon222, prop:id65], 1, tes(f1_es12), id7)
fact(infon223, isa, [arg:id60, arg:st], 1, tes(f1_es12), id7)
fact(infon224, isa, [arg:id61, arg:tloc], 1, tes(f1_es12), id7)
fact(infon225, pres, [arg:id61], 1, tes(f1_es12), id7)
fact(infon227, isa, [arg:id62, arg:rain], 1, tes(f1_es12), id7)
fact(infon228, isa, [arg:id63, arg:snow], 1, tes(f1_es12), id7)
fact(id65, keep_out, [agent:id54, theme_aff:id64], 1, tes(finf1_es12), id7)
fact(infon229, isa, [arg:id65, arg:pr], 1, tes(finf1_es12), id7)
fact(infon230, isa, [arg:id66, arg:tloc], 1, tes(finf1_es12), id7)
fact(infon231, pres, [arg:id66], 1, tes(finf1_es12), id7)
fact(infon232, time, [arg:id65, arg:id66], 1, tes(f1_es12), id7)
fact(infon233, coincide, [arg:id60, prop:id65], 1, tes(f1_es12), id7)
during(tes(f1_es12), tes(f1_es11))
includes(tr(f1_es12), id53)
```

…

```
14.About 10 gallons of sap come from each hole
set(infon259, id69)
card(infon260, id69, 10)
fact(infon261, inst_of, [ind:id69, class:nquant], 1, univ, univ)
fact(infon262, isa, [ind:id69, class:gallon], 1, id1, univ)
fact(infon263, of, [arg:id69, specif:id24], 1, id1, univ)
fact(id70, come, [agent:id69, theme_aff:id38], 1, tes(f1_es14),
id7)
fact(infon264, isa, [arg:id70, arg:ev], 1, tes(f1_es14), id7)
fact(infon265, isa, [arg:id71, arg:tloc], 1, tes(f1_es14), id7)
fact(infon266, pres, [arg:id71], 1, tes(f1_es14), id7)
fact(infon267, time, [arg:id70, arg:id71], 1, tes(f1_es14), univ)
during(tes(f1_es14), tes(f1_es13))
includes(tr(f1_es14), id53)
```

3. Question-Answering

Coming now to Question Answering, the system accesses the DM looking for relations at first then for entities : entities are searched according to the form of the focussed element in the User DataBase of Question-Facts as shown below with the QDM for the first question:

User Question-Facts Discourse Model

```
q_loc(infon3, id1, [arg:main_tloc, arg:tr(f1_free_a)])
q_ent(infon4, id2)
q_fact(infon5, isa, [ind:id2, class:who], 1, id1, univ)
q_fact(infon6, inst_of, [ind:id2, class:man], 1, univ, univ)
q_class(infon7, id3)
q_fact(infon8, inst_of, [ind:id3, class:coll], 1, univ, univ)
q_fact(infon9, isa, [ind:id3, class:sap], 1, id1, univ)
q_fact(infon10, focus, [arg:id2], 1, id1, univ)
q_fact(id4, collect, [agent:id2, theme_aff:id3], 1, tes(f1_free_a),
univ)
q_fact(infon13, isa, [arg:id4, arg:pr], 1, tes(f1_free_a), univ)
q_fact(infon14, isa, [arg:id5, arg:tloc], 1, tes(f1_free_a), univ)
q_fact(infon15, pres, [arg:id5], 1, tes(f1_free_a), univ)
```

As to the current text, it replies correctly to the all questions. As to question 4, at first the system takes « come from » to be answered exhaustively by sentence 14 ; however, seen that «hole » is not computed with a « location » semantic role, it searches the DM for a better answer which is the relation linguistically expressed in sentence 9, where «holes » are drilled «in each tree ». The « tree » is the Main Location of the whole story and « hole » in sentence 9 is inferentially linked to « hole » in sentence 14, by a chain of inferential inclusions. In fact, come_from does not figure in WordNet even though it does in our dictionary of synonyms. As to the fifth question, the system replies correctly.

1. Who collects maple sap? (Farmers)
2. What does the farmer hang from a spout?
(A bucket)
3. When is sap collected? (February and March)
4. Where does the maple sap come from?
(Sugar maple trees)
5. Why is the bucket covered? (to keep rain and snow out)

Another possible « Why » question could have been the following : « why the tree is called a "sugar" maple tree », which would have received the appropriate answer seen that the corresponding sentence has received an appropriate grammatical and semantic analysis. In particular, the discourse deictic pronoun « This » has been bound to the previous main relation « use » and its arguments so that they can be used to answer the « Why » question appropriately.
There is not enough space here to comment in detail the parse and the semantics (but see Delmonte 2000c); however, as far as anaphora resolution is concerned, the Higher Module computes the appropriate antecedent for the big Pro of the arbitrary SUBject of the infinitive in sentence n. 7, where the collecting action would have been left without an agent. This resolution of anaphora is triggered by the parser decision to treat the big Pro as an arbitrary pronominal and this information is stored at lexical level in the subcategorization frame for the name « time ».
Differently from what Schwitter et al. conclude in their paper, good linguistic processing can be achieved as long as strongly linguistically-based processing is performed. In the analysis they make of the semantic processing necessary in their opinion to account for the complexity of the task they frequently refer to the use of abductive reasoning: this is what is done in our system in order to cope with uncertain and/or insufficient information. For instance, with question n.4 the text only makes available information related to « maple syrup ». Since we start looking for relation, and the «come from » relation has a different linguistic description as SUBJect argument, what we do is to try and see whether there is some inferential link between « sap » and « syrup ». This is not the case seen that WordNet does not link the two concepts explicitly. However both are classified as « substance » thus allowing the required inference to be fired – both are also taken as synonyms in our dictionary. The final question does not constitute a problem seen that the predicate «cover» has become a semantic

relation and is no longer a noun or a verb. Also worth noting is the fact that the question is not a real passive, but a quasi-passive or an ergative construction, so no agent should be searched for. So our conclusion is that the heart of a Q/A system should be a strongly restrictive pipeline of linguistically based modules which alone can ensure the adequate information for the knowledge representation and the reasoning processes required to answer natural language queries.

3.1.1 Evaluating GETARUNS approach to QA

Totally shallow approaches when compared to ours will always be lacking sufficient information for semantic processing at propositional level: in other words, as happens with our "Partial" modality, there will be no possibility of checking for precision in producing predicate-argument structures

Most systems would use some Word Matching algorithm that counts the number of words that appear in both the question and the sentence being considered after stripping them of stopwords: usually two words will match if they share the same morphological root after some stemming has taken place. Most QA systems presented in the literature rely on the classification of words into two classes: function and content words. They don't make use of a Discourse Model where input text has been transformed via a rigorous semantic mapping algorithm: they rather access tagged input text in order to sort best match words, phrases or sentences according to some matching scoring function.

It is also common knowledge the fact that only by introducing or increasing the amount of linguistic knowledge over crude IR-based systems will contribute substantial improvements. In particular, systems based on simple Named-Entity identification tasks are too rigid to be able to match phrase relations constraints often involved in a natural language query.

First objection is the impossibility to take into account pronominal expressions, their relations and properties as belonging to the antecedent, if no head transformation has taken place during the analysis process.

Second objection is the use of grammatical function labels, like SUBJ/OBJects without an evaluation of their relevance in the utterance structure: higher level or main clause SUBJ/OBJects are more important than other SUBJects. In addition, there is no attempt at semantic role assignment which would come from a basic syntactic/semantic tagging of governing verbs: a distinction into movement verbs, communication verbs, copulative verbs, psychic verbs etc. would suffice to assign semantic roles to main arguments if present.

It is usually the case that QA systems divide the question to be answered into two parts: the Question Target represented by the wh- word and the rest of the sentence; otherwise the words making up the yes/no question and then a match takes place in order to identify most likely answers in relation to the rest/whole of the sentence except for stopwords.

However, it is just the semantic relations that need to be captured and not just the words making up the question that matter. Some system implemented more sophisticated methods (notably Hovy et al.; Litkowski): syntactic-semantic question analysis. This involves a robust syntactic-semantic parser to analyse the question and candidate answers, and a matcher that combines word- and parse-tree-level information to identify answer passages more precisely.

3.1.2 Answering Generic Question

An important issue in QA is answering generic questions on the "aboutness" of the text, questions which may be answered by producing appropriate headlines or just a title. In our system, given the concomitant work of anaphora resolution modules and the semantic mapping into predicate-argument structures, this can be made as follows. The system collapses all entities and their properties, relations and attributes at the end of the analysis, by collecting them for ontological type; each semantic id receives a score for topichood thus allowing a ranking of the entities; in addition, starred facts are inherited by the inclusion relation specified by the "in" semantic predicate, as in the case of the "specifying" relation between "sugar" and "maple". Here below we list the most relevant entities for the text reported above:

```
entity(class,id2,115,facts([
fact(infon3, 'Maple', [ind:id2], 1, T, P),
fact(infon4, inst_of, [ind:id2, class:edible_animal], 1, T, P),
fact(infon5, isa, [ind:id2, class:'Syrup'], 1, T, P),
fact(id5, make, [theme_bound:id2, agent:id4], 1, T, P),
fact(id11, come, [actor:id2, locat:id8], 1, T, P),
fact(id14, make, [agent:id2, theme_aff:id13], 1, T, P),
fact(id16, use, [theme_aff:id2, result:id14], 1, T, P)])).

entity(class,id30,77,facts([
fact(infon114, inst_of, [ind:id30, class:man], 1, T, P),
fact(infon115, isa, [ind:id30, class:farmer], 1, T, P),
fact(id39, drill, [agent:id30, theme_aff:id38], 1, T, P),
fact(id42, put, [agent:id30, theme_aff:id41, locat:id38], 1, T, P),
fact(id48, hang, [agent:id30, theme_aff:id44], 1, T, P)])).
```

In this way, an appropriate answer to the question “What is the text about” can be generated directly from the entity list by picking up relations and properties of the most relevant individuals, sets and classes (see Delmonte 2000c).

4. System Evaluation

The complete system has been built for a restricted linguistic domain and evaluated on the basis of the texts making up the domain: 3500 total words, 265 sentences. The performance is 95% correct. The system has been tested with a set of texts derived from newspapers, narrative texts, children stories summing up to 10,000 words where we got the same results: However, updating and tuning of the system is required for each new text whenever a new semantic relation is introduced by the parser and the semantic does not provide the appropriate mapping. For instance, consider the case of the constituent "holes in the tree", where the syntax produces the appropriate structure but the semantics does not map "holes" as being in a LOCATion semantic relation with "tree". In lack of such a semantic role information a dummy "MODal" will be produced which however will not generate the adequate semantic mapping in the DM and the meaning is lost.

As to the partial system, it has been used for DUC summarization contest, i.e. it has run over approximately 1 million words, including training and test sets, for a number of sentences totalling over 50K. We tested the "Partial" modality with an additional 90,000 words texts taken from the testset made available by DUC 2002 contest. On a preliminary perusal of the results, we have calculated a 85% Precision on parsing and an 80% on semantic mapping. However evaluating such results requires a manually annotated database in which all linguistic properties have been carefully decided by human annotators. In lack of such a database, we are unable to provide precise performance data.

References

Abney, S. 1996. Part-of-Speech Tagging and Partial Parsing, in Ken Church, Steve Young, and Gerrit Bloothooft, eds. *Corpus-Based Methods in Language and Speech*, Kluwer Academic Publishers, Dordrecht.

Bresnan, Joan. 2001. *Lexical-Functional Syntax*. Blackwells.

Brill, E., Lin, J., Banko, M., Dumais, S., & Ng, A. (2002). Data-Intensive Question Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001).* 122-131.

Delmonte R., 2000a. Parsing Preferences and Linguistic Strategies, in *LDV-Forum - Zeitschrift fuer Computerlinguistik und Sprachtechnologie - "Communicating Agents"*, Band 17, 1,2, pp. 56-73.

Delmonte R., 2000b. Shallow Parsing And Functional Structure In Italian Corpora, *Proc. LREC*, Athens, pp.113-119.

Delmonte R. 2000c. Generating from a Discourse Model, *Proc. MT-2000*, BCS, Exeter, pp.25-1/10.

Delmonte R. 2002. GETARUN PARSER - A parser equipped with Quantifier Raising and Anaphoric Binding based on LFG, *Proc. LFG2002 Conference*, Athens, pp.130-153, at *http://cslipublications. stanford.edu/hand/miscpubsonline.html*.

Hirschman, L. Marc Light, Eric Breck, & J. D. Buger. 1999. Deep Read: A reading comprehension system. In *Proc. A CL '99*.University of Maryland.

Schwitter R., D. Mollà, R. Fournier & M. Hess, 2000. Answer Extraction: Towards better Evaluations of NLP Systems. *In Proc. Works. Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*, Seattle, 20-27.

Hovy, E., U. Hermjakob, & C. Lin. (2002a). The Use of External Knowledge in Factoid QA. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001).* 644-652.

Litkowski, K. C. (2001). Syntactic Clues and Lexical Resources in Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Ninth Text Retrieval Conference (TREC-9).* 157-166.

Ravichandran, D. & E. Hovy. (2002). Learning Surface Text Patterns for a Question Answering System. Proceedings of the 40th ACL. Philadelphia, PA., 41-7.

Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering

Mark A. Greenwood and Robert Gaizauskas
Department of Computer Science
University of Sheffield
Regent Court, Portobello Road
Sheffield S1 4DP UK
{m.greenwood,r.gaizauskas}@dcs.shef.ac.uk

Abstract

This paper explores one particular limitation common to question answering systems which operate by using induced surface matching text patterns – namely the problems concerned with question specific words appearing within the induced answer extraction pattern. We suggest a solution to this problem by generalising the learned answer extraction patterns to include named entity classes.

1 Introduction

Many of the open-domain question answering systems developed prior to 2001 were heavily reliant on external sources of knowledge and tools for pinpointing exact answers within text. These included resources such as WordNet (Miller, 1995), named entity taggers, parsers and gazetteers. The performance of these systems varied widely with one of the best performing systems, FALCON (Harabagiu et al., 2000), being able to answer approximately 65% of the questions (Voorhees, 2000).

However, at the 10th Text REtrieval Conference, TREC 2001 (Voorhees, 2001), a system was entered that used only a single resource consisting of a large collection of surface matching text patterns which were derived using simple machine learning techniques from a set of question-answer pairs and a corpus of documents in which the answer was known to occur (Soubbotin and Soubbotin, 2001). The surprising fact, was that this system did exceptionally well, performing better than any of the other systems entered into the evaluation. The performance of what appeared to be a relatively simple system, provoked so much interest that at least one group, Ravichandran and Hovy (2002), choose to implement their own version.

In this paper we observe that overly specific patterns learned by this approach can cause problems such that it becomes extremely difficult to answer certain types of question. These problems are highlighted using our own implementation which has been developed as the basis for a more advanced question answering system.

2 The Basic Pattern Induction Approach

The following sections detail the algorithms involved in the basic approach to finding and using surface matching text patterns to answer questions.

2.1 Learning Text Patterns

Our approach to learning patterns is similar to that outlined in the paper by Ravichandran and Hovy (2002) as it also relies on suffix trees (Ukkonen, 1995) to extract patterns of an optimal length from unstructured text. The best way of describing the algorithm is through an example. The input to the algorithm is a set of questions of a specific type and their associated exact answer phrases. For comparison with the Ravichandran paper we will use questions of the form *“When was X born?”*. Given this the algorithm is as follows:

1. For each example question of the specific question type, produce a pair consisting of the question term and the answer term. For example:

   - "Abraham Lincoln" "1809"
   - "Adolf Hitler" "1889"
   - "Louisa May Alcott" "1832"
   - "Isaac Asimov" "1920"

2. For each example the question and answer terms are submitted to Google, as a single query, and the top 10 documents are downloaded[1].

3. Each document then has the question term replaced by the single token `AnCHoR` and the answer term by `AnSWeR`.

4. A tokeniser and sentence splitter are then applied to the documents.

5. Those sentences which contain both `AnCHoR` and `AnSWeR` are retained and joined together to create one single document, in which each sentence is separated by a # and the end of the created document is marked by $[2].

6. The single generated document is then used to produce a token-level suffix tree, from which the repeated substrings are then extracted.

7. Finally the list of repeated substrings is filtered to retain only those which contain both `AnCHoR` and `AnSWeR` and do not span a sentence boundary (i.e. do not contain # or $).

This produces a set of patterns for the specific question type. The following are a few of the patterns generated using this approach, for questions of the form *"When was X born?"*:

```
from AnCHoR ( AnSWeR - 1969 )
AnCHoR , AnSWeR -
- AnCHoR ( AnSWeR
from AnCHoR ( AnSWeR -
:  AnCHoR , AnSWeR -
```

Unfortunately some of these patterns are specific to one or more of the questions used to generate them (e.g. the first pattern includes a date of death, which is question specific). A further stage is therefore needed to analyse the patterns to decide which are generic enough to be used to answer unseen questions.

The algorithm used to analyse the patterns and discard those which are not generic, also allows us to associate a numerical precision with each pattern which can later be used as a measure of how confident the system is in any answers it proposes. Continuing with the same example as above, the steps in this algorithm are as follows:

1. Using a different set of question-answer pairs, only the question term is submitted to Google and the top ten documents are downloaded.

2. Each document then has the question term replaced by `AnCHoR` and the answer term (if it appears within the document) is replaced by `AnSWeR`.

3. Those sentences which contain `AnCHoR` are retained and joined together to create one single document.

4. Each of the previously generated patterns is converted to a standard regular expression designed to capture the answer text, giving expressions such as[3]:

   ```
   from AnCHoR \( ([ˆ ]+) - 1969 \)
   AnCHoR , ([ˆ ]+) -
   - AnCHoR \( ([ˆ ]+)
   from AnCHoR \( ([ˆ ]+) -
   :  AnCHoR , ([ˆ ]+) -
   ```

   These regular expressions allow us to easily retrieve the single token which `AnSWeR` in the original pattern would have matched against.

5. Each regular expression is then matched against each sentence in the generated document. Along with each pattern, P, two counts are maintained: C_a^P, which counts the total number of times this pattern has matched

[1] The documents are actually downloaded from Googles cache to guarantee that we use the version of the page indexed by Google.

[2] These separators are a necessary part of the suffix tree construction and processing but they do not appear in the resulting patterns.

[3] For those not familiar with standard regular expressions, `([ˆ ]+)` matches any sequence of one or more non-space characters and captures that part of the text in a variable for latter use.

Regular Expression	***Precision***
AnCHoR \(([^]+) -	0.967
AnCHoR \(([^]+)	0.566
AnCHoR ([^]+) -	0.263

Table 1: Regular expressions and their associated precision for questions of the form *"When was X born?"*.

against the text and C_c^P, which counts the number of matches which had AnSWeR as the extracted answer.

6. After a pattern, P, has been matched against every sentence in the generated document if C_c^P is less than five then it is discarded otherwise the precision of the pattern is calculated as C_c^P/C_a^P and the pattern is retained if its precision is greater than 0.1[4].

Using this method to produce a list of analysed patterns for the question type *"When was X born?"* gives regular expressions such as those in Table 1, which are now generic and could be applied to any other question of the same type.

2.2 Using Text Patterns to Find Answers

Using these regular expressions to find answers to questions is extremely simple. Firstly the question term is extracted from the question and submitted as a query to Google. Each document returned by Google then has the question term replaced by AnCHoR and those sentences containg AnCHoR are retained to create a single document. Each regular expression is then matched against the sentences and for each successful match the token captured by the expression is stored along with the precision of the pattern. When all the regular expressions have been applied to all the sentences, any answers found are sorted based firstly on the precision of the pattern which located them and secondly on the number of times the same answer was found.

3 The Limitations Imposed by Overly Specific Patterns

Most papers which describe systems using surface matching text patterns (Soubbotin and Soubbotin, 2001; Ravichandran and Hovy, 2002), including the current paper, explain the workings of the system through questions of the form *"When was X born?"* often using *"When was Mozart born?"* as a specific example. One or more of the analysed patterns are usually capable of extracting the answer from text such as: *"Mozart (1756-1791) was a musical genius"*. Indeed extracting the answer from sentences of this form is a sensible thing to do, due to the fact that this formulation of the answer is both precise and commonly occurring in unstructured text. This example, however, exposes a serious problem with the approach.

A similar question, which could also be answered from the example sentence, is *"When did Mozart die?"*. The problem is that generating patterns for this type of question using multiple examples will not lead to a simple generic pattern that can be applied to other questions. The extracted patterns, for a single example, will include the year of birth, such as:

```
AnCHoR (1756 - AnSWeR
AnCHoR (1756 - AnSWeR )
```

When these patterns are analysed against a second set of question-answer pairs, and their precision is calculated, they will most likely be discarded[5], due to the presence of a specific year of birth. This problem does not occur when generating patterns for questions of the form *"When was X born?"* as multiple patterns will be produced some of which contain the date of death and some of which do not, simply because the date of death usually appears after the year of birth in the answer phrases.

More generally any acquired pattern must consist of three components 1) the AnCHoR tag (which gets initialised as the question-specific anchor, e.g. Mozart), 2) the AnSWeR regular expression, and 3) literal text occurring between 1) and 2). In the basic text pattern learning approach of Section 2, component 3) cannot be generalised, i.e. cannot be a regular expression containing meta-characters, and hence can only match itself.

There are other patterns, which could be extracted, for questions of the form *"When did X*

[4] These cut-off values were adopted based on empirical observations made during development.

[5] An exception, specific to this example, would be if the same year of birth appeared in the question sets used for both inducing the answer patterns and assigning precisions to the answer patterns.

die?”. For example:

```
AnCHoR died in AnSWeR
AnCHoR was killed in AnSWeR
```

These patterns, however, are not as precise as those possible for *“When was X born?”* (for example AnSWeR could easily be a location instead of a date). In an experiment (documented in Section 5) our system failed to generate any patterns for the question type *”When did X die?”*.

4 Generalising the Answer Patterns

It is clear that for this style of question answering to be as accurate as possible a way needs to be found to generate as precise a set of patterns as possible for each question type. As we have already noted one problem is that in certain formulations of the answer phrase words specific to the question appear between AnCHoR and AnSWeR. Many of these words are dates, names and locations in fact exactly the words that can be recognised using the well understood natural language techniques of gazetteers and named entity taggers. The solution employed by our system is therefore a combination of the question answering system described in Section 2 and a gazetteer and a named entity tagger [6].

The approach taken to incorporate these NLP techniques is to substitute the text marked as a named entity by a tag representing its type, hence dates become DatE, locations become LocatioN, etc. This replacement is carried out after the question and answer text have been replaced with AnCHoR and AnSWeR respectively but before any other processing is carried out. This is the only change to the algorithms for inducing and assigning precisions to the answer patterns.

When these new patterns are used to answer questions extra work is, however, required as it is possible that an answer found by a pattern may in fact be, or may include, a named entity tag. When using the patterns not only do we replace named entities with a tag but also store the original text so that if an answer contains a tag it can be expanded back to the original text.

As was previously mentioned, the standard implementation failed to create any patterns for the questions *“When did X die?”* this extended implementation, however, produces regular expressions such as those in Table 2.

Regular Expression	***Precision***
`AnCHoR \( DatE - ([^ ]+) \) .`	1.000
`AnCHoR \( DatE - ([^ ]+) \)`	1.000
`AnCHoR DatE - ([^ ]+)`	0.889

Table 2: Regular expressions, augmented with named entity tags, and the associated precisions for questions of the form *“When did X die?”*.

Regular Expression	***Precision***
`AnCHoR \( ([^ ]+) -`	0.941
`AnCHoR \( ([^ ]+) - DatE \) .`	0.941
`AnCHoR \( ([^ ]+) - DatE \)`	0.941
`AnCHoR \( ([^ ]+)`	0.600
`AnCHoR ([^ ]+) - DatE`	0.556
`AnCHoR ([^ ]+) -`	0.263

Table 3: A selection of regular expressions, augmented with named entity tags, and the associated precisions for questions of the form *“When was X born?”*.

It is clear, from these patterns, that incorporating the NLP techniques allowed us to extract exactly the type of patterns we extended the system to handle.

This extended system can also be used to generate a new set of patterns for questions of the form *“When was X born?”*, a selection of these can be seen in Table 3.

5 Results

A set of experiments was carried out to see the effect of extending the patterns in the way suggested in the previous section. The question sets used for the experiments consisted of one hundred and forty examples, divided into three groups: twenty examples for inducing the patterns, twenty for assigning precisions to the patterns and one hundred over which to test the patterns. A selection of the analysed patterns have already been presented in Tables 1, 2 and 3.

Results are given for four experiments the combination of the original and extended systems over

[6] The gazetteer and named entity tagger used in these experiments are slightly modified versions of those which are included as part of the GATE 2 framework (Cunningham et al., 2002), available from http://gate.ac.uk.

System Number	*% Correctly Answered*	*MRR Score*	*Confidence Weighted*
1	52%	0.52	0.837
2	53%	0.52	0.843
3	0%	0.00	0.000
4	53%	0.53	0.852

Table 4: Results of using both the original and extended systems.

the two different question types. The experiments are as follows:

1. The original system answering questions of the form *"When was X born?"*.

2. The extended system answering questions of the form *"When was X born?"*.

3. The original system answering questions of the form *"When did X die?"*.

4. The extended system answering questions of the form *"When did X die?"*.

The results of these experiments can be seen in Table 4. The mean reciprocal rank (MRR) score (Voorhees, 2001) of 0.52 for system 1 is comparable to the results of similar experiments (Ravichandran and Hovy, 2002) over the same question type.

The results show that not only does the extended system allow us to achieve similar results for the questions *"When did X die?"* but also that extending the system in this way had no significant detrimental effects on the performance over question types answerable by the original system and actually produced a higher confidence weighted score (Voorhees, 2002) for all question types. The slight increase in the confidence weighted score is probably due to the greater number of overlapping high precision patterns induced for a specific question type. This leads to the same answer being extracted more often and with a higher precision than in the original system, leading to these answers being ranked higher when the answers for multiple questions are sorted.

6 Discussion and Conclusions

Although the results given in this paper cover only a small number of experiments, they show that the use of a gazetteer and named entity tagger allow the simple pattern matching question answering system to be extended to answer some questions which the original approach could not answer. Furthermore, the performance of the extended system on questions which could already be answered is improved.

Clearly more experimentation is needed before we can claim that this technique solves all the problems associated with overly specific answer patterns. This paper has shown that it successfully handles one specific question type. Experiments were also carried out for the question type *"What is the capital of X?"* in which although the extend system produced better results, than the original system, the improvement was not significant, because in most cases no question-specific text fell between the `AnCHoR` and the `AnSWeR`.

It should be clear, however, that this appraoch can be applied to any question type where question-specific text is likely to occur between the `AnCHoR` and the `AnSWeR`, such as *"When was America discovered?"* which can easily be answered by the text *"In 1492 Columbus discovered America"*, where Columbus needs to be generalised before a sensible pattern could be induced from this answer phrase.

There are other ways in which the text within an answer pattern can be generalised, and we do not claim that our solution is the only way forward, rather that it has been shown to work well over a small set of question types. More work is needed to expand not only the types of question the system can answer but also to test other methods of generalising the surface matching text patterns induced from free text.

References

H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*.

Sanda Harabagiu, Dan Moldovan, Marius. Paşca, Rada Mihalcea, Mihai Surdeanu, Răzvan Bunescu, Roxana Gîrju, Vasile Rus, and Paul Morărescu. 2000. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the 9th Text REtrieval Conference*.

George A. Miller. 1995. WordNet: A Lexical Database. *Communications of the ACM*, 38(11):39–41, November.

Deepak Ravichandran and Eduard Hovy. 2002. Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 41–47, Pennsylvania.

M. M. Soubbotin and S. M. Soubbotin. 2001. Patterns of Potential Answer Expressions as Clues to the Right Answers. In *Proceedings of the 10th Text REtrieval Conference*.

E. Ukkonen. 1995. On-line Construction of Suffix Trees. *Algorithmica*, 14(3):249–260.

Ellen M. Voorhees. 2000. Overview of the TREC-9 Question Answering Track. In *Proceedings of the 9th Text REtrieval Conference*.

Ellen M. Voorhees. 2001. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text REtrieval Conference*.

Ellen M. Voorhees. 2002. Overview of the TREC 2002 Question Answering Track. In *Proceedings of the 11th Text REtrieval Conference*. Draft version which appeared in the conference notebook.

Learning Paraphrases to Improve a Question-Answering System

Florence Duclaye
France Télécom R&D and ENST
2 avenue Marzin
22307 Lannion Cedex, France
florence.duclaye@rd.francetelecom.com

François Yvon
ENST
46 rue Barrault
75634 Paris Cedex 13, France
yvon@enst.fr

Olivier Collin
France Télécom R&D
2 avenue Marzin
22307 Lannion Cedex, France
olivier.collin@rd.francetelecom.com

Abstract

In this paper, we present a nearly unsupervised learning methodology for automatically extracting paraphrases from the Web. Starting with one single linguistic expression of a semantic relationship, our learning algorithm repeatedly samples the Web, in order to build a corpus of potential new examples of the same relationship. Sampling steps alternate with validation steps, during which implausible paraphrases are filtered out using an EM-based unsupervised clustering procedure. This learning machinery is built on top of an existing question-answering (QA) system and the learnt paraphrases will eventually be used to improve its recall. We focus here on the learning aspect of this system and report preliminary results.

1 Introduction

Question-answering systems (Voorhees, 1999) require efficient and sophisticated NLP tools, crucially capable of dealing with the linguistic variability of questions and answers, which reflects the widely acknowledged fact that the same meaning can be conveyed using a wide variety of lexico-syntactic structures (forms). This situation is by no means specific to the QA domain, this variability being a source of difficulties for most practical applications of NLP.

Part of this variability can be captured at the syntactic level, where it takes the form of regular alternations between for instance active and passive forms, or verbal and nominal expressions of a concept. A more systematic treatment however requires some form of semantic knowledge, such as the one found in semantic networks (Miller et al., 1990). The help provided by these resources is limited as (i) synonymy relationships found in such dictionaries cannot be taken at face value, for the lack of contextual information; (ii) synonymy implies a notion of paraphrasing which is far too restricted for our application: it is often the case that the answer to a question is expressed using terms which are only loosely (eg. metaphorically) related to the ones used in the question. For instance, ”X caused Y” can be considered to be semantically similar to ”Y is blamed for X” in the context of question-answering (Lin and Pantel, 2001). Rather that trying to manually complete these static resources, a virtually endless process, we have chosen to explore the benefits of a corpus-based approach and to learn such equivalences automatically. We will refer to these relationships as paraphrases, although we adopt here a rather restricted definition of paraphrase, focusing mostly on two types of linguistic phenomena: linguistic paraphrases and semantic derivations. (Fuchs, 1982) describes paraphrases as sentences whose denotative linguistic meaning is equivalent. Semantic derivations are sentences whose meaning is preserved, but whose lexico-syntactic structure is different (e.g. *AOL bought Netscape / the acquisition of Netscape by AOL*). The corpus we use for acquiring paraphrases is the Web. Using the Web as a corpus offers several clear advantages (see also (Grefenstette, 1994)): (i) it contains a great variety and redundancy: the same information is likely to occur under many guises, a property on which our learning algorithm heavily relies; (ii) contextual information is available and can be used to restrict the scope of a paraphrase relationship. Moreover, as our QA system uses the Web as its only information source, it is important to extract those formulations of a given concept which are actually frequently used on the Web. This strategy is not without its own difficulties: in particular, reducing the level of noise in the acquired data becomes a serious issue. The learning mechanism we propose is capable of automatically acquiring multiple formulations of a given semantic relationship from *one single example*. This seed data consists of one instance of the target semantic relationship, where both the linguistic expression of the relationship (formulation) and the tuple of arguments have been identified. This kind of data is directly provided by our QA system, but is also widely available in usual dictionaries. Given this positive example, our learning machinery repeatedly queries the Web, trying alternately to use the currently known formulations to acquire new argument tuples, and the known argument tuples to find new formulations. This mechanism decomposes into two steps: the search for potential paraphrases of the semantic relation and the validation of these paraphrases, which is based on frequency counts and the Expectation-Maximisation (EM) algorithm.

This paper introduces, in Section 2, some background technical work which has been influencial for our approach, as well as related research on paraphrase learning. Section 3 then gives a thorough presentation of our system, first giving a general overview of its behavior, then explaining our EM-based filtering strategy, and finally going into the details of the acquisition procedure. Before concluding, we discuss in Section 4 some experimental results that highlight the interest of our approach.

2 Background

2.1 Paraphrase learning

As paraphrases can be used in various contexts and applications, learning them is accomplished using very different methodologies. (Barzilay and McKeown, 2001) distinguish between three different methods for collecting paraphrases. The first one is manual collection, the second one is the use of existing linguistic resources, and the third one is corpus-based extraction of similar words or expressions. Of these three methods, manually collecting paraphrases is certainly the easiest one to implement, though probably the most tedious and time-consuming one.

Linguistic resources such as dictionaries can prove to be useful for collecting or generating paraphrases. For instance, (Kurohashi and Sakai, 1999) uses a manually-tailored dictionary to rephrase as verbal phrases ambiguous noun phrases. Such linguistic resources as dictionaries may be useful for disambiguation purposes, but they rarely provide linguistic information in context, so that the proper paraphrases cannot always be spotted. Moreover, they are often recognised to be poorly adapted to automatic processing (Habert et al., 1997). (Torisawa, 2001) proposes a method using the Expectation-Maximisation algorithm to select verb schemes that serve to paraphrase expressions.

Finally, some of the works in the area of corpus-based extraction of similar words or expressions rely on Harris'Distributional Hypothesis, stating that words occurring in the same context tend to have similar meanings. Relying on this assumption, (Barzilay and McKeown, 2001) and (Akira and Takenobu, 2002) work on a set of aligned texts and use contextual cues based on lexical similarities to extract paraphrases. In the same line, (Lin and Pantel, 2001) uses an unsupervised algorithm for discovering inference rules from text. Instead of applying Harris'rule to words, the authors apply it to paths in dependency trees of a parsed corpus.

2.2 Information extraction by bootstrapping

Recent work on information extraction provides us with interesting approaches that can be adapted to solving the problem of paraphrase learning. (Riloff and Jones, 1999) describes an information extraction system relying on a two-level bootstrapping mechanism. The “mutual bootstrapping” level alternatively constructs a lexicon and contextual extraction patterns. The “meta-bootstrapping” level keeps only the five best new terms extracted during a given learning round before continuing with the mutual bootstrapping. In this way, the author manages to reduce the amount of invalid terms retrieved by the application of extraction patterns.

The DIPRE technique (Dual Iterative Pattern Relation Extraction) presented in (Brin, 1998) is also a bootstrapping method, used for the acquisition of (author,title) pairs out of a corpus of Web documents. Starting from an initial seed set of examples, the author constructs extraction patterns that are used to collect (author,title) pairs. In their turn, these pairs are searched in the corpus and are used to construct new extraction patterns, and so on. Finally, (Collins and Singer, 1999) describes a method for recognising named entities with very little supervision data by building two classifiers operating on disjoint feature sets in parallel.

3 System overview

3.1 General overview of the paraphrase learning system

Our paraphrase inference algorithm learns from one single positive example, using a two-level bootstrapping mechanism. This seed example is an answer to a question, returned by our QA system. In our model, a meaning is represented as the association between the linguistic formulation f of a predicate, and its arguments tuple a. For instance, one example of the "authorship" relationship would be represented as: f="to be the author of", a=("Melville", "Moby Dick"). Identification of paraphrases relies on a probabilistic decision model, whose parameters are estimated in an almost unsupervised way. Estimation relies on an EM-based clustering algorithm presented in Section 3.2: it takes as input a matrix containing frequency data for the co-occurrence of a set of formulations F and the corresponding argument tuples A, as measured in a corpus C.

Our initial corpus C_i contains one unique "seed" example expressing the target relationship, and represented as the cooccurrence of a formulation f_i and an argument tuple a_i. Given this seed, we would like to build a new corpus C, potentially containing many more instances of the target relationship. This is done by using independently f_i and a_i to formulate queries, which are used to sample from the web. The retrieved documents are searched for new interesting formulations and arguments pairs, repeatedly used to produce new queries, which in turn will extract more arguments and formulations... During this stage, we need to be able to (i) generate queries and process the retrieved documents so as to (ii) extract new formulations and argument tuples. Details of these corpus building procedures are given in Section 3.3.

The quality of the extracted paraphrases depends critically on our ability to keep the expanding corpus *focused on the target semantic relationship*: to this end, the acquisition phases are interleaved with filtering stages, which are also based on our EM-based clustering. Filtering is indeed critical to ensure the convergence of this procedure. The overall architecture of our system is represented on figure 1.

3.2 Filtering with the Expectation-Maximisation algorithm

The filtering problem consists in sorting out incorrect paraphrases of the original relationship from valid ones. This amounts to classifying each formulation in our corpus as 1 (valid paraphrase) or 0 (not valid), based on co-occurrence data between arguments tuples and formulations. This bipartitioning problem is weakly supervised, as we initially have one positive example: the seed formulation. This is a favorable configuration for the use of EM-based clustering algorithms for cooccurrence data (Hofmann and Puzicha, 1998). We thus assume that each phrase (consisting of a formulation f and its arguments a) is generated by the following stochastic model:

$$\begin{aligned} P(f,a) &= \Sigma_{s\in S} P(f,a|s)P(s) \quad (1) \\ &= \Sigma_{s\in S} P(f|s)P(a|s)P(s) \quad (2) \end{aligned}$$

where S is the set of semantic relationships expressed by sentences in our corpus. We further assume that our corpus only contains two such relationships, whose values are defined as $S = 1$, meaning that a given sentence expresses the same relationship as the seed sentence, and $S = 0$, meaning that the sentence expresses another (unspecified) relationship.

Given this model, the reestimation formulas are easily derived (see eg. (Hofmann and Puzicha,

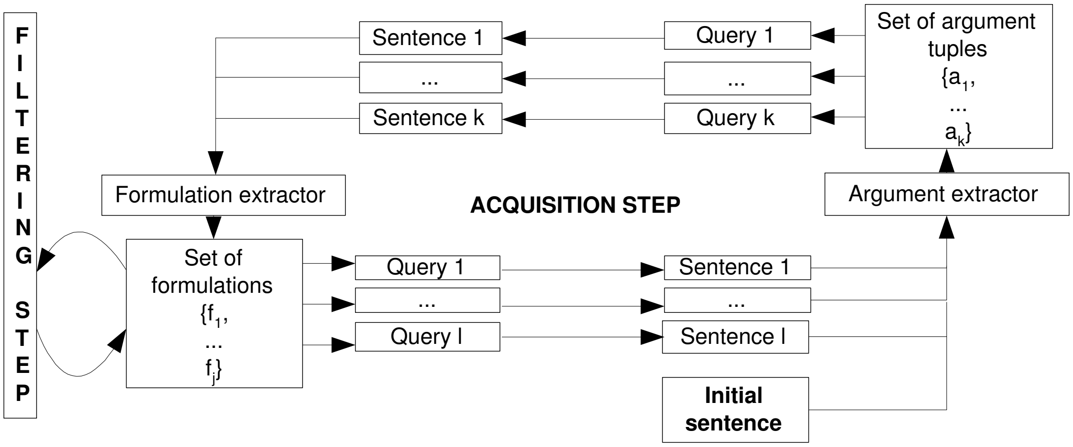


Figure 1: Paraphrase learning system

1998)); they are given in Table 1, where $N()$ denotes the count function.

E-Step

$$P(s|f,a) = \frac{P(s)P(f|s)P(a|s)}{\sum_i P(s_i)P(f|s_i)P(a|s_i)} \qquad (3)$$

M-Step

$$P(a|s) = \frac{\sum_{f\in F} N(f,a)P(s|f,a)}{\sum_{a\in A}\sum_{f\in F} N(f,a)P(s|f,a)} \qquad (4)$$

$$P(f|s) = \frac{\sum_{a\in A} N(f,a)P(s|f,a)}{\sum_{f\in F}\sum_{a\in A} N(f,a)P(s|f,a)} \qquad (5)$$

$$P(s) = \frac{\sum_{f\in F}\sum_{a\in A} N(f,a)P(s|f,a)}{\sum_{f\in F}\sum_{a\in A} N(f,a)} \qquad (6)$$

Table 1: Reestimation formulas for EM

This model enables us to incorportate supervision data during the parameter initialisation stage, where we use the following values: $P(S = 1|f_i, a_i) = 1$ and $P(S = 1|f_i, a) = 0.6, \forall a \neq a_i$ in equation (3). All the other values of $P(S|F, A)$ are taken equal to 0.5. EM is then run until convergence of the maximised parameters. In our case, this convergence generally achieved within 10 iterations.

Once the parameters have been learnt, we use this model to decide whether a formulation f is a valid paraphrase based on the ratio between $P(S = 1|f)$ and $P(S = 0|f)$, computed as: $r = \frac{P(S=1)P(f|S=1)}{P(S=0)P(f|S=0)}$. Given that $P(S = 1)$ is grossly overestimated in our corpus, we require this ratio to be greater than a predefined threshold $\theta > 1$.

3.3 The acquisition procedure

The main tool used during the acquisition step is our QA system itself, which has been adapted in order to be also used as an information extraction tool. The original system has two main components. The first one turns an input question into a Web query and performs the search. The second module analyses the retrieved pages, trying to match answers, an answer corresponding to a set of predefined extraction patterns. Both the query and the extraction patterns are derived from the original question using rules. Details regarding this QA system and the NLP components involved at each stage are given in (Duclaye et al., 2002).

In "learning" mode, we by-pass the query construction phase and enforce the use of the argument tuples (or formulations) as search keywords. The analysis phase uses very general information extraction patterns directly derived from the arguments (or formulations) being processed. Assume, for instance, that we are in the process of searching for paraphrases, based on the argument pair ["Melville", "Moby Dick"]. Both arguments will be used as keywords, and two patterns will be matched in the retrieved documents : "Melville [Verb] Moby Dick" and "Moby Dick

[Verb] Melville". In this example, a verb is required to occur between the two keywords. This verb will be considered to be a potential paraphrase of the initial formulation. For each query, only the top N documents returned by the search engine are considered.

Notwithstanding the effects of the filtering procedure, the extracted $(arguments, formulations)$ are cumulated, round after round, in a corpus C, from which statistics are then estimated. This iterative process of acquiring formulations and argument tuples, combined with the validation process at the end of every iteration, converges and ends up when no new formulation is found.

4 Experimental results

The experiments described in this section were conducted on 18 initial sentences, representing 12 different semantic relationships (e.g. purchase of a company, author of a book, invention of something, ...). See table 2 for examples of formulations and argument tuples. For each of these sentences, the learning procedure described in Section 3 was run on one iteration. The results presented here were obtained by searching for French documents on the Web and taking the first N=1000 previews returned by the search engine.

The extracted paraphrases were manually checked and classified as valid or invalid by ourselves. In this application, success is only measured as the average precision of the extracted paraphrases which should eventually be fed into the QA system. Recall, in comparison, is unimportant, as long as we can find the most frequently used paraphrases. The selection ratio represents the percentage of formulations classified as valid paraphrases by our system. The decision to classify a formulation as a valid or invalid paraphrase is based on the ratio between $log(P(S = 1|f))$ and $log(P(S = 0|f))$, called θ. The selection ratios and precision results for various filtering thresholds θ are reported in table 3.

In these experiments, the best average precision achieved is 66.6%, when $\theta = 186$. Performed on several relationships, these experiments showed that the precision rate may vary importantly from one semantic relationship to another : it can be as high as 100% for certain relationships, and as low as 6% for others. These results may seem to be low. This is partly due to the varying amount of data extracted from the Web for the semantic relationships. Applying the same threshold θ to all relationships may not be the best method, as the system extracts a huge quantity of formulations for certain relations, and a small one for others. Moreover, the majority of the formulations wrongly classified as good paraphrases are thematically related to the seed formulation (e.g. for the purchase relationship : to own, to belong, to merge, ...).

theta	selection ratio	precision
7	44.0%	42.9%
25	29.8%	47.3%
48	23.9%	47.3%
117	14.2%	54.9%
186	10%	66.6%
232	9.4%	65.4%

Table 3: Experimental results

As indicated in table 3, the increasing values of θ cause the selection ratios to decrease and the precision to increase. The general tendency is that as θ gets bigger and bigger, the amount of formulations classified as bad paraphrases increases, so that eventually only the seed formulation is kept as valid. Increasing θ is thus insufficient to improve the average precision of the extracted paraphrases. A balance needs to be found between the selection ratio and the precision of the extracted paraphrases.

Let us point out that the results shown in table 3 only reflect the first learning iteration. Other experiments were conducted on several learning iterations, which lead us to believe that precision should increase with the number of iterations. Table 4 shows the results obtained on the purchase relationship, after five learning iterations. The filtering strategy was different from the one detailed in Section 3.2. Instead of keeping the formulations according to the ratio between $P(S = 1|f)$ and $P(S = 0|f)$, we decided to only keep the five best formulations at each learning iteration.

purchase of	”acheter” (*to buy*)	AOL; Netscape
author of	”écrire” (*to write*)	Melville; Moby Dick
inventor of	”inventer” (*to invent*)	Gutenberg; imprimerie (*printing machine*)
assassination of	”assassiner” (*to murder*)	Oswald; Kennedy

Table 2: Some exemplary relationships and formulations

Iter.	Formulations classified as valid paraphrases
1	racheter (*to buy out*), acqurir (*to acquire*), acheter (*to buy*), utiliser (*to use*), recevoir (*to receive*)
2	racheter, acqurir, acheter, reprendre (*to take back*), absorber (*to take over*)
3	racheter, acheter, acqurir, qui racheter (*[which] buy out*), devenir (*to become*)
4	racheter, acheter, acqurir, absorber, grouper (*to gather*)
5	racheter, acheter, reprendre, devenir, acqurir

Table 4: Results of five learning iterations on the purchase relationship

5 Conclusions and future work

In this paper, we have presented a nearly unsupervised methodology for learning paraphrases automatically, starting with one single positive learning example. Using an EM-based validation strategy, we are able to filter out the invalid potential paraphrases extracted during the acquisition steps.

Not only are these paraphrases useful to improve the results of our question answering system, but the acquired argument tuples could also be used for other purposes than paraphrase learning, such as the construction of semantic lexicons. In fact, the filtering step could as well be applied on the acquired argument tuples.

Beyond its promising experimental results, the adaptability of our approach brings to the fore the multiple practical applications of this work. Focused on the acquisition and validation steps, various improvements are presently under investigation. Concerning the acquisition step, we are planning to learn multilingual paraphrases, as well as more complex extraction patterns (involving nominalisations). We are also considering using automatically learnt contextual information to refine the quality of the queries we use to sample the Web. Future improvements of the filtering / validation step will aim at testing other filtering strategies.

Based on a language-independent learning strategy, our paraphrase learning system will be integrated into the multilingual question-answering system. Our system will act as an offline component which will learn paraphrases of answers returned by the QA system. Its integration will not require many developments, as the QA system already takes into account manually-entered paraphrasing rules. We will thus have to automate this process of entering paraphrasing rules into the QA system. This integration will enable us to evaluate our methodoloy and to measure the improvements incurred by this paraphrase learning module.

References

Terada Akira and Tokunaga Takenobu. 2002. Automatic disabbreviation by using context information. In *Proceedings of the NLPRS Workshop on Automatic Paraphrasing : Theories and Applications*.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceeding of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 50–57, Toulouse.

Sergei Brin. 1998. Extracting patterns and relations from the world wide web. In *Proceedings of WebDB Workshop at EDBT*.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Workshop on Empirical Methods for Natural Language Processing*.

Florence Duclaye, Pascal Filoche, Jerzy Sitko, and Olivier Collin. 2002. A polish question-answering system for business information. In *Proceedings of the Business Information Systems Conference*, Poznan.

Catherine Fuchs. 1982. *La Paraphrase*. Presses Universitaires de France.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Boston.

Benoît Habert, Adeline Nazarenko, and André Salem. 1997. *Les linguistiques de corpus*. Armand Colin, Paris.

Thomas Hofmann and Jan Puzicha. 1998. Statistical models for co-occurrence data. Technical Report AI. 1625, MIT, AI Lab.

Sadao Kurohashi and Yasuyuki Sakai. 1999. Semantic analysis of japanese noun phrases : a new approach to dictionary-based understanding. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 481–488.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. In *Natural Language Engineering*, volume 7, pages 343–360.

George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1990. Introduction to wordnet: An on-line lexical database. In *Journal of Lexicography*, volume 3, pages 234–244.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence*.

Kentaro Torisawa. 2001. A nearly unsupervised learning method for automatic paraphrasing of japanese noun phrases. In *Proceedings of the NLPRS 2002 workshop on Automatic Paraphrasing : Theories and Applications*, Tokyo.

Ellen Voorhees. 1999. The TREC-8 question answering track report. In *Proceedings of TREC-8*.

Selectively Using Relations to Improve Precision in Question Answering

Boris Katz and **Jimmy Lin**
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139
{boris,jimmylin}@ai.mit.edu

Abstract

Despite the intuition that linguistically sophisticated techniques should be beneficial to question answering, real gains in performance have yet to be demonstrated empirically in a reliable manner. Systems built around sophisticated linguistic analysis generally perform worse than their linguistically-uninformed cousins. We believe that the key to effective application of natural language processing technology is to selectively employ it only when helpful, without abandoning simpler techniques. To this end, we identify two linguistic phenomena that current information extraction driven systems have difficulty with, and demonstrate how syntactic processing can help. By indexing syntactic relations that can be reliably extracted from corpus text and matching questions with documents at the relation level, we demonstrate that syntactic analysis enables a question answering system to successfully handle these phenomena, thereby improving precision.

1 Introduction

Most current question answering systems utilize a combination of information retrieval and information extraction techniques to find short answers to fact-based questions such as "Who killed Lincoln?" The dominant approach, driven by IE technology, is to first find a set of potentially relevant passages and then "pinpoint" the exact location of the answer by searching for an entity whose semantic type matches the question type.

Although respectable performance can be achieved with this relatively simple two-stage process, there exist empirical limits on the effectiveness of this approach. By analyzing a subset of TREC-9 and CBC questions, Light et al. (2001) established an expected upper bound on the performance of a question answering system with perfect passage retrieval, named-entity detection, and question classification at around 70%. The primary reason for this limit is that many named entities of the same semantic type often occur close together, and a QA system, without the aid of any additional knowledge, would be forced to choose randomly.

Although we are still years away from systems that can provide accurate semantic analysis on open-domain text, significant headway has been made in the syntactic analysis of documents. Matching questions and passages based on syntactically-derived relations offers an interim solution for overcoming the limitation of IE-based question answering systems. Although previous systems have employed similar techniques (to be discussed further in Section 2), they generally did not perform as well as systems that utilize linguistically-uninformed techniques. We attribute these results to the reliance on NLP techniques as the fundamental machinery for question answering, despite their brittleness. Instead, we suggest a more pragmatic approach: continue to use linguistically-uninformed techniques as the foundation of a question answering system, and apply sophisticated NLP approaches *only when they are known to improve performance.* To this end, we identify two linguistic phenomena that current IE-driven systems have difficulty with, and

(1) The bird ate the snake.
(1′) The snake ate the bird.
(2) the largest planet's volcanoes
(2′) the planet's largest volcanoes
(3) the house by the river
(3′) the river by the house
(4) The Germans defeated the French.
(4′) The Germans were defeated by the French.

Figure 1: Semantic differences that cannot be captured by lexical content alone

demonstrate how syntactic processing can help. Additionally, to overcome the usual brittleness associated with syntactic parsing, we utilize syntactic relations, captured in terms of ternary expressions, that can be reliably and easily extracted from complex real-world text.

The fragment pairs in Figure 1 illustrate the elusive nature of "meaning"; although fragments in each pair are nearly indistinguishable in terms of lexical content, their meanings are vastly different. Naturally, because one text fragment may be an appropriate answer to a question while the other fragment may not be, a question answering system seeking to achieve high precision must differentiate the semantic content of the pairs. Ideally, question answering should be based on the semantics of questions and documents, but unfortunately, full semantic analysis is beyond current technology in anything but highly-restricted domains. Instead, as a compromise, we propose to match questions and answers at the syntactic level.

2 Hasn't this been tried before?

The concept of marrying NLP techniques with large-scale IR is not new, but effective integration of the two remains an open research question. Fagan (1987) experimented with indexing noun phrases and prepositional phrases. More recently, various researchers have experimented with indexing syntactically derived word pairs (Strzalkowski et al., 1996; Zhai et al., 1996; Arampatzis et al., 1998); the types of constructions examined in the context of indexing include linguistically motivated pairs such as head/modifier and adjective/noun. In addition, full linguistic trees (Smeaton et al., 1994) and case frames (Croft and Lewis, 1987) as units of indexing have been tried. However, none of these experiments resulted in dramatic improvement in precision or recall, and often even resulted in degraded performance. In all of theses studies, the word-level index was directly augmented with linguistically-derived representations. Often, this caused performance issues because the creation of an index is limited by the speed of the parser, and because sophisticated linguistic representations were not amenable to large-scale indexing.

The current generation of question answering systems that employ NLP alleviate performance problems by delaying linguistic analysis until the corpus has been narrowed down to a small set of candidate documents or passages. The MURAX System (Kupiec, 1993) is an early example of such an approach. More recently, Litkowski (1999) described a system that utilizes a combination of syntactic relations, e.g., subject-verb-object, and some semantic relations, e.g., time and location. After initially retrieving a set of candidate documents, the system then parses both the question and the passages and attempts matching at the relation level. Unfortunately, this and similar techniques that depend heavily on syntactic analysis, e.g., PIQASso (Attardi et al., 2001), yielded relatively poor performance. A drawback of this two-step paradigm is low recall: if the keyword-based document retrieval system does not return *any* relevant documents due to such problems as synonymy, anaphora, or argument alternations, any amount of additional processing is useless. The current work-around to this problem is to implement feedback loops that relax the query set if the results are too restrictive (Moldovan et al., 2002). Not only does this introduce complex dependencies in a system's architecture, but it also necessitates the addition of new modules to assess the quality of the result sets.

In the domain of information access, we attribute the mediocre track record of sophisticated linguistic techniques not to the impotence of NLP technology in general, but rather to the manner in which it has been applied. Results appear to demonstrate that the current level of natural language technology is still too brittle to be applied in all situations. Because existing linguistically-impoverished methods have proven to be robust and capable of delivering useful levels of performance, we propose a more pragmatic approach: recognize situations where linguistic techniques would help and employ them only when necessary.

With this approach, the critical question becomes: under what circumstances can natural language processing techniques improve question answering? In reply, we describe two broad linguistic phenomena that are difficult to handle with the information extraction driven paradigm. Within these two areas, the use of syntactic relations results in a dramatic improvement in the precision of question answering systems.

3 Two Phenomena

We have identified two broad phenomena that cannot be easily handled by linguistically uninformed question answering systems: *semantic symmetry* and *ambiguous modification*. Examples representing typical results from current QA systems (Figure 2) help illustrate the phenomena.

The first example (Q1) demonstrates the problem of semantic symmetry: although the questions "What do frogs eat?" and "What eats frogs?" are similar at the word level, they have very different meanings and should be answered differently. The second example (Q2) demonstrates the problem of ambiguous modification: adjectives like *largest* and prepositional phrases such as *in the Solar System* can modify a variety of different head nouns. Potential answers may contain the correct entities, but they may not be in the correct syntactic relations with each other, e.g., *the largest planet* instead of *the largest volcano*. Both these phenomena could benefit from a more detailed linguistic treatment to pinpoint more precise answers.

Semantic symmetry occurs when the selectional restrictions of different arguments of the same head overlap; for example, the selectional restriction for the subject of *eat* is *animate* and the selectional restriction for the object is *edible*, so semantic symmetry occurs whenever the subject and object are both animate and edible. In these cases, lexical content is insufficient to determine the meaning of the sentence—syntactic analysis is required to discover head-arguments relations.

(Q1) What do frogs eat?

(A1) Adult *frogs eat* mainly insects and other small animals, including earthworms, minnows, and spiders.

(A2) Alligators *eat* many kinds of small animals that live in or near the water, including fish, snakes, *frogs*, turtles, small mammals, and birds.

(A3) Some bats catch fish with their claws, and a few species *eat* lizards, rodents, small birds, tree *frogs*, and other bats.

(Q2) What is the largest volcano in the Solar System?

(B1) Mars boasts many extreme geographic features; for example, Olympus Mons, the *largest volcano in the solar system*.

(B2) The Galileo probe's mission to Jupiter, the *largest* planet *in the Solar system*, included amazing photographs of the *volcanoes* on Io, one of its four most famous moons.

(B3) Even the *largest volcanoes* found on Earth are puny in comparison to others found around our own cosmic backyard, *the Solar System*.

(B4) Olympus Mons, which spans an area the size of Arizona, is the *largest volcano in the Solar System*.

Figure 2: Examples illustrating semantic symmetry and ambiguous modification (emphasis added)

Ambiguous modification occurs when an argument's selectional restrictions are so *un*restrictive that the argument can belong to more than one head in a particular context. Since nearly anything can be *large* or *good*, syntactic analysis is necessary to pin down which head this argument actually belongs to.

In order to define the phenomena described above more formally, we shall adopt a first order predicate logic formalism. In our description, sentences are parsed into logical forms consisting of relations (*n*-ary predicates) with words as their arguments. The semantics of the predicate logic can be modeled as constraints on the domain of the arguments: a logical expression is semantically valid, or "makes sense," if and only if the arguments of every predicate type-check with constraints imposed by that predicate. For example, if R is a one place predicate whose argument is constrained on the set $s = \{a, b, c\}$, then $R(d)$ is not a semantically valid expression. Given this definition of semantic validity, we can define a function $\mathcal{S}$ on any logical expression e:

$$\mathcal{S}(e) = \begin{cases} 1 & \text{if } e \text{ is a semantically valid} \\ 0 & \text{otherwise} \end{cases}$$

Using this framework, we can then formally define semantically symmetric relations:

> **Semantic Symmetry**
> A relation is *semantically symmetric* iff there exists w, w_1, and w_2 such that $\mathcal{S}(R(w, w_1)) = \mathcal{S}(R(w_2, w)) = 1$.

A typical example of semantically symmetric relations involves sentences where one can swap the subject and object and still end up with a sentence that "makes sense," with respect to $\mathcal{S}$. This occurs when the domains of the arguments of a particular predicate (as determined by the selectional restrictions of that particular head) have a non-null intersection, e.g., some questions involving predators and prey:

$eat(x, y)$
$x \subset$ animate-agent
$y \subset$ animate-agent, inanimate-object . . .

Thus, the difficulty with "What do frogs eat?" is that the question seeks entities that fulfill a certain relation, namely, all x such that $eat(frog, x)$ is true. However, statements that answer the question "What do frogs eat?" and "What eats frogs?" are likely to contain both the relevant keywords *frog* and *eat*. Since *eat* is a semantically symmetric relation, both $eat(frog, x)$ and $eat(x, frog)$ are likely to be found within the corpus.

The phenomenon of semantically symmetric relations observed above is by no means an isolated instance. Examples of such verbs abound in English, e.g., *visit*, *meet*, *defeat*, *kill*, *love*, *see*, and many, many more. Together, all the verbs in this class of semantically symmetric relations present a problem for any non-linguistically informed QA system.

Ambiguous modification represents another phenomenon that linguistically-uninformed QA systems have difficulty handling:

> **Ambiguous Modification**
> A word w, involving a relation R, is an ambiguous modifier iff there exist at least two words, w_1 and w_2, in the same local context as w, such that $\mathcal{S}(R(w, w_1)) = \mathcal{S}(R(w, w_2)) = 1$.

A phrase like *the planet's largest volcanoes* illustrates the ambiguous modification phenomenon. For example, the adjective *largest*, involved in an adjective-noun modification relation, is not very constrained in its possible choices of head nouns, and hence is free to "float" among nouns in its local context.[1] This means that given passages with similar lexical content containing the adjective *largest*, it is difficult to determine exactly which head noun the adjective is modifying without syntactic analysis.[2] Hence, if the relation under consideration is crucial to answering the question,[3] syntax is required to precisely pin down relations from both the question and the corpus to ensure that the relations match satisfactorily. The possessive relation involving *planet* and *volcano* is another instance of the ambiguous modification phenomenon because there are other potential choices for modifiers and modifiees, e.g., *the planet's ocean* or *the island's volcano*.

In the example (Q2) in Figure 2, there are two ambiguous modifiers: *largest*, involved in an adjective-noun modification relation, and *in the Solar System*, involved in a location relation. Sentences (B2) and (B3) have the correct lexical content, but only some of the correct syntactic relations. In (B2), both *largest* and *in the Solar System* modify the incorrect head noun. In (B3), *in the Solar System* does not modify the correct head noun.

[1] By local context, we mean the set of head nouns surrounding the ambiguous modifier in question. The size of this local context is related to the granularity of the information retrieval process we are comparing against. For example, in document retrieval, where all the words in a document are considered, an ambiguous modifier can potentially modify a head noun anywhere in the document.

[2] Researchers have attempted to work around this problem by indexing consecutive word pairs, but there are simple examples in which this technique would not help, e.g., "the brown apple" vs. "the apple brown from bruising," "John's house" vs. "house of John."

[3] True in this case, because we are looking for a planet with the *largest* volcano, and not, for example, the largest planet that possesses a volcano.

Adjectives are often ambiguous modifiers: given a context with a pool of adjectives and nouns, any particular adjective could potentially modify many nouns. Under such circumstances, a question answering system cannot achieve high precision without exactly identifying the particular relation between words through detailed syntactic analysis.

4 Ternary Expressions

In the previous section, we identified two natural language phenomena that pose difficulties to traditional IE-driven QA systems, difficulties which are alleviated by endowing a system with the ability to perform syntactic analysis. To do so, we need a syntactic representation that can capture the important relations between words in text, yet is amenable to rapid processing and matching.

Although parse trees capture syntactic relations, they are difficult to generate, store, and manipulate rapidly. In a particular set of experiments, Smeaton et al. (1994) lamented the awkwardness and slow speed of processing full parse trees. Indeed, generating detailed parses is often time-consuming, and much of the parse information is not directly relevant to question answering anyway.

Similarly, logical form is not the best representation for our purposes, despite its utility in precisely and formally delineating problems. Manipulation of logical forms requires significant computational machinery, e.g., unification mechanisms, a general theorem prover, etc. Furthermore, using logic as the paradigm for matching relations requires careful and exact formulation of all axioms and allowed inferences *a priori*, which is not practical due to the ambiguous nature of language.

We believe that a more pragmatic solution to capturing the relations relevant for question answering is to distill natural language text into ternary (three-place) expressions (Katz, 1988). Ternary expressions may be intuitively viewed as subject-relation-object triples, and can easily express many types of relations, e.g., subject-verb-object relations, possession relations, etc. The START Question Answering System (Katz, 1997) has been employing such representations effectively in question answering for the last two decades, and we find that they are a good compromise between expressiveness and simplicity.

Using ternary expressions, the semantic differences between the text fragments presented in Figure 1 can be distinguished at the syntactic level:

(1) [bird eat snake]
(1′) [snake eat bird]
(2) [largest `adjmod` planet]
[planet `poss` volcano]
(2′) [largest `adjmod` volcano]
[planet `poss` volcano]
(3) [house by river]
(3′) [river by house]
(4) [Germans defeat French]
(4′) [French defeat Germans]

5 Initial Experiments

In order to demonstrate our ideas, we have implemented Sapere, a prototype natural language question answering system that retrieves answers by matching ternary expressions derived from the question with those derived from the corpus text.

As a baseline for comparison, we implemented a simple boolean retrieval engine that uses a standard inverted keyword index to index documents at the sentence level. All stopwords are discarded, and all content words are stemmed. For the baseline, a conjunctive query of all non-stopwords from the query is issued to the boolean retrieval engine; the resulting set of sentences is ranked by the number of non-stopwords that were found in each sentence. Although boolean keyword search systems do not perform as well as state-of-the-art IR engines, we believe that they serve as an adequate baseline for comparison since there is substantial empirical evidence that boolean-based passage retrieval techniques are sufficient to obtain reasonable performance in question answering tasks (Light et al., 2001).

Sapere is primarily a relations-indexing engine; it stores and indexes ternary expressions extracted from the corpus text and performs matching at the relation level between questions and sentences stored in its index. Ternary expressions are generated from text by postprocessing the results of Minipar (Lin, 1993), a fast and robust functional dependency parser. Currently, Sapere detects the following types of relations: subject-verb-object (including passive constructions), adjective-noun modification, noun-noun modification, possessive

What countries has Japan invaded?
What eats snakes?
Who defeated the Spanish Armada?
When do lions hunt?
What is the largest planet?

Figure 3: Sample questions used in the user study

relations, predicate nominatives, predicate adjectives, appositives, and prepositional phrases.

Ternary expressions are similarly derived from the question, with the *wh*-entity left as an unbound variable. Sapere attempts to match relations in the question with those found in the corpus text, thereby binding the unbound variable in the question with the actual answer. If such a match occurs, the candidate sentence is returned.

The test corpus used in our experiments was an electronic version of the Worldbook Encyclopedia, which contains approximately 20,000 articles. The entire corpus was parsed and relations extracted from it were indexed by Sapere.

The test set consisted of 16 hand-selected questions that illustrate the two linguistic phenomena discussed previously; some of these questions are shown in Figure 3. For example, "Who defeated the Spanish Armada?" probes semantically symmetric relations; "What is the largest planet?" tests ambiguous modification.

Results from both the baseline system and Sapere were collected and manually judged to be either relevant or irrelevant. The comparison between the baseline and Sapere can be seen in Table 1. For the sixteen questions in this particular test set, indexing and matching relations (Sapere) achieved a precision of 0.84 ± 0.11, while basic keyword matching (baseline) achieved only a precision of 0.29 ± 0.11. In addition, Sapere returned far fewer results, reducing the amount of reading the user must perform to obtain the correct answer.

A sample output from the baseline keywords indexer is shown in Figure 4. After removing stopwords from the query, our simple keyword search engine returned 32 results that contain the keywords *frog* and *eat*. Of all the sentences returned, only (C4) correctly answers the user query.

	Relations	Keywords
Avg. # of sentences returned	4	43.88
Avg. # of correct sentences	3.13	5.88
Avg. precision	0.84	0.29

Table 1: Comparison between relations and keyword indexing.

(Q3) What do frogs eat?
(C1) Alligators eat many kinds of small animals that live in or near the water, including fish, snakes, frogs, turtles, small mammals, and birds.
(C2) Some bats catch fish with their claws, and a few species eat lizards, rodents, small birds, tree frogs, and other bats.
(C3) Bowfins eat mainly other fish, frogs, and crayfish.
(C4) Adult frogs eat mainly insects and other small animals, including earthworms, minnows, and spiders.
(C5) Kookaburras eat caterpillars, fish, frogs, insects, small mammals, snakes, worms, and even small birds.
(C6) . . .

Figure 4: Sample output from the baseline keyword indexer

By comparison, our relations matcher returns only (C4) as the correct answer.

Syntactic analysis of the question can distinguish the ternary expression derived from "What do frogs eat?" ([frog eat $?x$]) from the ternary expression representing "What eats frogs?" ([$?x$ eat frog]) and respond to the former question with sentences containing such relations as [frog eat insect] instead of sentences containing relations like [alligator eat frog], despite the similar lexical content of all the sentences.

6 Discussion

Large precision gains were achieved in our experiments because the test set was engineered to illustrate the two phenomena that we identified: semantic symmetry and ambiguous modification. By exactly pinpointing the areas in which natural language techniques are helpful, Sapere was able to exploit syntactic parsing to dramatically increase precision. Instead of relying exclusively on sophisticated linguistic techniques, we suggest that simpler linguistically-uninformed techniques

(Q1003) What is the highest dam in the U.S.?

(D1) Extensive flooding was reported Sunday on the Chattahoochee River in Georgia as it neared its crest at Tailwater and George *Dam*, its *highest* level since 1929. (AP900319-0047)

(D2) A swollen tributary the Ganges River in the capital today reached its *highest* level in 34 years, officials said, as soldiers and volunteers worked to build *dams* against the rising waters. (AP880902-0066)

(D3) Two years ago, the numbers of steelhead returning to the river was the *highest* since the *dam* was built in 1959. (SJMN91-06144185)

Figure 5: Sample answers from TREC that illustrate problems with ambiguous modification.

should not be abandoned. The resulting combined system will see more modest, but still valuable, gains in precision.

Because Sapere currently derives relations from Minipar, the quality of the relations ultimately depends on the quality of the parser. Despite parse errors, indexing syntactic representations boosts the performance of our question answering system because the relations we chose to index are generally ones that can be reliably and accurately extracted from text, e.g., adjective-noun modification, simple subject-verb-object relations from the matrix clause, etc. However, deriving relations using off-the-shelf parsers, while convenient for current experimental purposes, might not be the ideal situation in the longer term. A custom-built lightweight parser specifically designed to extract relations might be faster and more accurate.

A quick analysis of the TREC corpus reveals that instances of the phenomena we've described occur frequently. Many real-word user questions crucially depend on particular relations between entities, e.g., "When were William Shakespeare's twins born?", taken from Q1002 at TREC-2001 QA Track. The vast majority of incorrect answers reported the birthday of Shakespeare himself, because the crucial possessive relation between the poet and his twins was neglected. Shown in Figure 5 are a few more incorrect answers returned by actual systems in TREC-2001 QA Track, attributed to ambiguous modification.

A distinct feature of Sapere is that it indexes all relations extracted from the corpus, instead of post-processing a set of candidate documents or passages from an IR system. We believe that this strategy overcomes the recall problem associated with the standard two-step approach: if the IR system doesn't produce *any* relevant documents, further processing (linguistic or otherwise) is useless. Take the example shown in Figure 5: since *highest* and *dam* co-occur frequently, it is possible that irrelevant results will "swamp out" relevant documents, so that relevant documents might not even be considered in later processing stages of a system. This problem is especially severe if the answer is only mentioned in one document from the entire corpus. Indexing relations from the ground up is a potential solution to this problem. Unfortunately, this strategy is also very computationally intensive, limiting our initial experiments to a smaller corpus. Our next goal is to apply our relations-indexing technique to the much larger TREC corpus.

Sapere's relation matcher currently attempts to match relations derived from the question against relations derived from the corpus. It does not, however, verify the absence of particular relations that may "invalidate" a response. The simplest example of this is explicit negation, although other adverbial modifiers (e.g., *never*, *barely*, *unlikely*), modals, verbs that take sentential complements (e.g., *deny*, *hallucinate*), and even certain adjective (e.g.,*former*, *non-existent*) can have the same effect. We are currently building a more sophisticated relations matcher that will take these effects into account. An even more troublesome issue is that of implied relations in a user query. When a user asks "What is the tallest mountain?", the qualification "in the world" is generally assumed. The implied relation is a matter of common sense, based on shared world knowledge. Without this knowledge, a system like Sapere might return the tallest mountain *in the world*, the tallest mountain *in Asia*, the tallest mountain *in Europe*, etc. Although these are arguably all correct answers (in some sense), it might not be appropriate to list the tallest mountains in every country. Short of "solving" common sense, the best solution is to build better interfaces that allow users to iteratively refine queries and supply missing information.

Another important issue remains to be resolved: how do we classify questions into classes that exhibit either semantic symmetry or ambiguous modification? It should be possible to automatically construct a relational "lexicon" of symmetric relations and ambiguous modifiers. Semantically symmetric relations can be recognized by considering the domains of their arguments; if significant intersection exists between the arguments, then the relation exhibits semantic symmetry. Ambiguous modifiers can be automatically acquired in a similar way; given a relation $R(m, w)$ between modifier m and head w (extracted from the question), find all heads $c_1 \ldots c_n$ that co-occur with w; if the relation $R(m, c_i)$ holds for any of the c's, then we can conclude that the modifier m is ambiguous.

A future research direction is to expand on our catalog of linguistic phenomena; we have presented two here, but we believe that there are additional opportunities where syntactic analysis could benefit question answering. By selectively using natural language processing technology only when they are known to be beneficial, a system like Sapere can significantly boost question answering performance. Naturally, if matching user queries and corpus documents fails at the syntactic relations level (i.e., produces no matches), a question answering system should fall back on less linguistically-informed approaches.

7 Conclusions

Many members of the IR and NLP community believe that question answering is an application in which sophisticated linguistic techniques will truly shine. However, this belief has yet to be empirically verified, as linguistically-impoverished systems have generally outperformed those that attempt syntactic or semantic analysis. In support of those techniques, we have categorized and empirically verified two phenomena, semantic symmetry and ambiguous modification, in which syntactic relations prove to be extremely effective. By first identifying, and then selectively applying linguistically-sophisticated techniques, we can overcome the limitations of present-day natural language technology and increase the performance of question answering systems.

Acknowledgements

We would like to thank Sue Felshin, Gregory Marton, and Stefanie Tellex for their valuable suggestions and comments on earlier drafts of this paper. We would also like to thank the anonymous reviewers for their comments and suggestions. This research is funded by DARPA under contract number F30602-00-1-0545 and administered by the Air Force Research Laboratory. Additional funding is provided by the Oxygen Project.

References

A. Arampatzis, Th.P. van der Weide, C.H.A. Koster, and P. van Bommel. 1998. Phrase-based information retrieval. *Information Processing and Management*, 34(6):693–707, December.

G. Attardi, A. Cisternino, F. Formica, M. Simi, A. Tommasi, and C. Zavattari. 2001. PIQASso: Pisa question answering system. In *TREC 2001*.

B. Croft and D. Lewis. 1987. An approach to natural language processing for document retrieval. In *SIGIR-1987*.

J. Fagan. 1987. *Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparisons of Syntactic and Non-Syntactic Methods*. Ph.D. thesis, Cornell University.

B. Katz. 1988. Using English for indexing and retrieving. In *RIAO'88*.

B. Katz. 1997. Annotating the World Wide Web using natural language. In *RIAO'97*.

J. Kupiec. 1993. MURAX: A robust linguistic approach for question answering using an an on-line encyclopedia. In *SIGIR-1993*.

M. Light, G. Mann, E. Riloff, and E. Breck. 2001. Analyses for elucidating current question answering technology. *Journal of Natural Language Engineering*, Fall–Winter.

D. Lin. 1993. Principle-based parsing without overgeneration. In *ACL-1993*.

K. Litkowski. 1999. Question-answering using semantic relation triples. In *TREC-8*.

D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. 2002. Performance issues and error analysis in an open-domain question answering system. In *ACL-2002*.

A. Smeaton, R. O'Donnell, and F. Kelledy. 1994. Indexing structures derived from syntax in TREC-3: System description. In *TREC-3*.

T. Strzalkowski, L. Guthrie, J. Karlgren, J. Leistensnider, F. Lin, J. Perez-Carballo, T. Straszheim, J. Wang, and J. Wilding. 1996. Natural language information retrieval: TREC-5 report. In *TREC-5*.

C. Zhai, X. Tong, N. Milic-Frayling, and D. Evans. 1996. Evaluation of syntactic phrase indexing—CLARIT NLP track report. In *TREC-5*.

Author Index

THIS IS A BLANK PAGE PLEASE IGNORE

Author Index