Knowledge and Reasoning for Answering Questions

Workshop associated with IJCAI05

Edinburgh, July 30th 2005

Proceedings edited by Farah Benamara, Marie-Francine Moens and Patrick Saint-Dizier

Programme

Session 1 8.30-10.30
Being Erlan Shen: Identifying Answerable Questions,
  H. Yu, C. Sable, USA.
Reasoning over Depedency Relations for QA,
  G. Bouma, J. Mur, G. Van Noord, the Netherlands.
Towards Answering procedural Questions,
  F. Aouladomar, France.
Towards a Framework for the Summarization of Help-Desk Responses
  Y. Marom, I. Zukerman, Australia.

10h30-11h00 Coffee break

11h00 – 11h45 ***Invited talk*** by Johan Bos, UK

11h45-12h15 ***Short papers and posters***
Toward Question Answering for Simulation,
  M. Core et ali. USA
A Question-Answering System for Portuguese,
  C. Prolo et ali., Brasil and Portugal
Semantic Knowledge in Question-Answering Systems,
  V. Barbier et ali., France
A Typology and Feature Set for Questions,
  L. Aunimo, Finland
Recognition of Alternation Paraphrases: a robust and exhaustive symbolic approach,
  M. Amoia and C. Gardent, France.

12h15-13h30 lunch break and poster visits

Session 2 13h30-15h00
On the Effective Use of Cyc in a Question-answering System,
  J. Curtis et ali., USA.
An Inference Model for Semantic Entailment and Question Answering
  R. de Salvo Braz et ali., USA.
Using Information Fusion for Open-Domain Question Answering,
  T. Dalmas, B. Webber, UK.

15h00-15h30 Coffee Break

Session 3 15h30-16h45
Supervised Machine learning Techniques for Question Answering,
  I. Zukerman et ali., Australia.
Invited talk : Marie-Francine Moens, Belgium (45 mns)

Panel Session: 17h00 – 18h00
Moderators: M. Minock (Sweden) and T. Poibeau (France)

Table of Contents

Foreword

The introduction of reasoning capabilities in question-answering (QA) systems appeared in the late 70s. A second generation of QA systems, aimed at being cooperative, emerged in the late 80s - early 90s. In these systems, quite advanced reasoning models were developed on closed domains to go beyond the production of direct responses to a query, in particular when the query has no response or when it contains misconceptions. More recently, systems such as JAVELIN, Inference WEB or Cogex, operating over open domains, integrate gradually inferential components, but not as advanced as those of the 90s. Performances of these systems in the recent TREC-QA tracks show that reasoning components do improve the response relevance and accuracy. They can also potentially be much more cooperative. However, there is still a long way before being able to produce accurate, cooperative and robust QA systems.

Recent foundational, methodological and technological developments in knowledge representation (e.g. ontologies, knowledge bases incorporating various forms of incompleteness or uncertainty), advanced reasoning forms (e.g. relaxation, intensional calculus, data fusion), not necessarily based on unification, advanced language processing resources and techniques (for question processing as well as for generating responses), and recent progress in HLT make it possible to foresee the elaboration of much more accurate, cooperative and robust systems dedicated to answering questions from textual data, from e.g. online texts or web pages, operating either on open or closed domains.

The workshop will be organized around a few major questions of interest to a number of AI, NLP, HLT and pragmatics people. One main question is the characterization of those reasoning procedures that need to be developed to answer questions, either on closed or on open domains. Then, are enhancing reasoning procedures and accuracy of knowledge representation sufficient conditions to improve responses ? If not, what is the role of language processing and what are the relevant paradigms (e.g. lexical inference) ? How do language and reasoning interact ? Next, what are the language models and techniques appropriate for producing responses which sound natural for the user (relevant, fluid, of an appropriate granularity, with terms the user understands, etc.). Another perspective is the role of pragmatics as a means, for example, to better capture the user's goals and intentions from his query, and therefore to better organize the response. Pragmatics is also of importance to better analyse the potential implicatures the user may draw from NL responses, in particular when the response is not direct.

This relatively new area of research includes the following topics, a number of which are addressed in the papers presented hereafter:

- Methodologies for intelligently answering questions,
- New types of questions and related KR, pragmatic and linguistic paradigms: procedural questions (how), causal questions (why), questions with comparative expressions, questions with negation, etc.
- Reasoning aspects:
  * information fusion,
  * search criteria expansion models (e.g. relaxation techniques),
  * summarization and intensional answers,
  * reasoning under uncertainty or with incomplete knowledge,

* Detecting and resolving query failure (due to e.g. incomplete data, misconceptions or false presuppositions)
- Knowledge representation and integration:
  * levels of knowledge involved (e.g. ontologies, domain knowledge),
  * knowledge extraction models and techniques to optimize response accuracy,
  * coherence and integration.
- Flexible and interactive systems possibly including a user model,
- Pragmatic dimensions of intelligently answering questions:
  * user intentions, plans and goals recognition in questions,
  * conversational implicatures in responses,
  * principles for the design of cooperative systems.
- Language processing:
  * question processing : parameters of interest for response production,
  * response generation (e.g. lexical choice, templates),
  * use of language resources for reasoning in question-answering,
  * explanation production (showing sources and inferences, reporting data incompleteness, etc.)
- evaluation
  * End-to-end evaluation of complex question types,
  * Intrinsic evaluation of inference methods,
  * Data-intensive vs knowledge-intensive methods,
  * portability techniques for closed domains.

The programme committee was the following, we warmly thank all its members whose contribution helped improve the papers presented here.

Farah Benamara, IRIT, France
Johan Bos, University of Edinburgh, UK
Sanda Harabagiu, University of Texas, USA
Eduard Hovy, ISI, USA
Daniel Kayser, LIPN, France
Mark Maybury, The MITRE Corp., USA
Michael Minock, University of Umea, Sweden
Marie-Francine Moens, KUL, Belgium
Jacques Moeschler, Geneva university, Switzerland
Dan Moldovan, University of Texas, USA
John Prager, IBM, USA
Ehud Reiter, University of Aberdeen, UK
Maarten de Rijke, University of Amsterdam, The Netherlands
Gérard Sabah, LIMSI, CNRS, France
Patrick Saint Dizier, IRIT, CNRS, France
Manfred Stede, University of Potsdam, Germany
Mathiew Stone, Center of Cognitive Science, Rutgers, USA
Kees Van Deemter, University of Aberdeen, UK
Ellen Voorhees, NIST, USA
Bonnie Webber, University of Edinburgh, UK

The workshop organizers,
Farah Benamara, Marie-Francine Moens, Patrick Saint-Dizier

Being *Erlang Shen*: Identifying Answerable Questions

Hong Yu
Columbia University
Department of Biomedical Informatics
622 West, 168th Street, VC-5, NY, NY 10032
yuh9001@dbmi.columbia.edu

Carl Sable
Cooper Union
Department of Electrical and Computer Engineering
51 Astor Place, NY, NY 10003
sable2@cooper.edu

Topics: language processing, reasoning aspects, knowledge representation and integration

Abstract

Research has shown that answers do not exist in biomedical corpora for many questions posed by physicians. We have therefore developed a *question filtering* component that determines whether or not a posed question is answerable. Using 200 clinical questions that have been annotated by physicians to be answerable or unanswerable, we have explored the use of supervised machine-learning algorithms to automatically classify questions into one of these two categories. We also have incorporated semantic features from a large biomedical knowledge terminology. Our results show that incorporating semantic features in general enhances the performance of question classification and the best system is a probabilistic indexing system that achieves an 80.5% accuracy. Our analysis also shows that stop words may play an important role for separating *Answerable* from *Unanswerable*.

1 Introduction

Chinese myth has long portrayed a powerful god *Erlang Shen*, who has a magical third eye in the middle of his forehead that sees truth. The real world mixes truth and falsehood and questions may be answerable or unanswerable. In the field of automatic question answering (QA), most QA systems implicitly assume that all questions are answerable. This study presents what we believe is the first attempt to separate answerable questions from unanswerable ones. We are essentially aiming to create the keen third eye to filter out unanswerable questions. The answerable questions can then be further processed for answer extraction and generation; the unanswerable questions may be further analyzed to determine the user's intentions.

Automatic question answering applies artificial intelligence and natural language processing techniques to extract information from corpora or databases in order to answer a user's question. Since no corpora or databases, no matter how large, can incorporate the entire universe of knowledge, they will not contain answers to certain questions. For example, research (Jacquemart and Zweigenbaum 2003) has found that the largest text collection, the World Wide Web, is not a good source for answering medical, domain-specific questions. On the other hand, biomedical literature and reputed online medical databases are useful for this task (Sackett et al. 2000, Straus and Sackett 1999). However, these same biomedical resources can not answer the question "What is causing her hives?"; this question was posed by a family physician (Ely et al. 2002). This study explores the use of supervised machine-learning approaches to automatically identify whether or not a question is answerable using biomedical corpora and databases.

Determining whether or not a question is answerable is a first step towards question answering. A question answering system needs first to identify a user's intentions, and then to generate a useful answer. Previously, studies have proposed models to offer explanations for failed queries or the results of the queries that are "unknown" (Chalupsky and Russ 2002). In this application, when a question is not answerable, the question answering system may further evaluate the question. For example, if the unanswerable question is not related to the medical domain, a system might return the question to user and provide the justification that the system only handles medical questions. If the unanswerable question is ambiguous, a system could use disambiguation to generate a list of non-ambiguous questions, from which the user can identify one or more according to his/her intentions. The efforts on identifying a user's intentions have been addresses in earlier work (Chalupsky and Russ 2002, Harabagiu et al. 2004, Gaasterland et. al. 1994, Grice 1975).

2 Related Work

Research on identifying a user's intentions starts with maxims of cooperative conversation (Grice 1975). A review is given by (Gaasterland et. al. 1994), who have analyzed cooperative answering as a specific application of Grice's maxims of cooperative conversation. According to these maxims, answers (and other contributions to a conversation) should not only be correct, but in addition, they should be useful, they should not be misleading, and they should not contain too much information. The overview provided by Gaasterland and his colleagues discusses how these maxims might be applied to query/answer systems, which they define to include not only question answering systems as defined in this paper, but also database systems and deductive databases that accept logical queries.

One interesting discussion in the work of Gaasterland and his colleagues involve general categories of reasons that a query or question might fail to have an answer. For example, the wording of a question might contain a false presupposition. An example in the medical domain might be, "What drug can fight the disease blindness?" The response "None" would be incorrect, since it seems to validate the false presupposition that blindness is a disease. A good response might be "Blindness is not a disease." Another interesting case involves questions with misconceptions, which are more general than false presuppositions. Questions with misconceptions can have correct answers that are still misleading. An example in the medical domain might be "What drug can a therapist prescribe to fight depression?" In this case, the answer "None" would technically be correct but misleading; a better response would be "Therapists can not prescribe drugs."

Chalupsky and Russ (2002) propose to provide a list of plausible answers or explanations when exact answers cannot be found in a database in response to a user's query. Possible explanations deal with missing knowledge, limitations of resources, user misconceptions, and bugs in the system. Chalupsky and Russ have created a system called *WhyNot*, which accepts queries to the general knowledge base *Cyc*, and attempts to provide what they call partial proofs for failed queries. An example provided by the authors involves the question, "Is it true that anthrax lethally infects animals?" The answer, according to the system, is unknown, but WhyNot also determines that the answer would be known if an animal is a kind of mammal. *WhyNot* was built on a relational database and does not handle ad hoc questions.

Harabagiu and her associates (2004) have proposed methods to combine semantic and syntactic features for identifying a user's intentions. As stated in their paper, if a user asks "Will Primer Minister Mori survive the crisis?", the method detects the user's belief that the position of the Prime Minister is in jeopardy, since the concept DANGER is associated with the words "survive" and "crisis". In addition, they propose that the predicate-argument structures of a question can be used to coerce a user's intention when there exist questions with known intentions. The work discussed in (Harabagiu et al. 2004) derives intentions only from the questions, and does not involve human-computer dialogue.

Many research groups have developed either rule-based (Hughes 1986) or machine-learning approaches (Hermjakob 2001, Zhang and Lee 2003) to automatically classify questions into predefined question types (e.g., definitional questions such as "What is X"?) for the purpose of answer generation. However, they all assume that all questions can be answered. Our study presents a different dimension that demonstrates that not all questions can be answered, and that unanswerable questions can be automatically identified.

This study is a part of our ongoing effort involving the development of a domain-specific QA system, BioMedQA, which will automatically generate answers to questions posed by physicians and biomedical researchers. In the following sections, we first describe QA in general, as well as particular considerations relevant to the development of a domain-specific QA system. Next we describe our question collection and our approaches of classifying questions as *Answerable* or *Unanswerable*. We then present and evaluate our results. We close our paper with discussion, conclusions, and future work.

3 Question Answering

Question answering is an advanced form of information retrieval in which focused answers are generated for either user queries or ad hoc questions. Most research development in the area is in the context of open-domain, collection-based or web-based QA. Largely driven by the Text REtrieval Conference (TREC) QA track[1], technologies have been developed for generating short answers to factual questions (e.g., "Who is the president of the United States?"). Recently, the Advanced Research and Development Activity (ARDA)'s Advanced Question & Answering for Intelligence (AQUAINT) program[2] has supported QA techniques that generate long answers for scenario questions (e.g., opinion questions such as "What does X think about Y?" (Yu and Hatzivassiloglou 2003)). Most QA systems leverage techniques from several fields including *information retrieval* (Rigsbergen 1979), which generates query terms relevant to a question and selects documents that are likely candidates to contain answers; *information extraction,* which locates portions of a document (e.g., phrases, sentences, or paragraphs) that contain the specific answers; and *summarization* and *natural language generation*, which are used to generate coherent, readable answers.

Recently there has been growing interest in domain-specific question answering. For example, ACL 2004 dedicated a workshop to QA within restricted domains. Domain-specific, biomedical QA can differ from open-domain QA in at least two important ways. For one, it might be possible to have a list of question types that are likely to occur, and separate answer strategies might be developed for each one. Secondly, domain-specific resources such as knowledge bases and tools exist with a level of detail that might allow a deeper processing of questions than is not possible for open-domain questions.

4 Question Collection and Annotation

Ely and his colleagues (Ely et al. 1999, Ely et al. 2000) have collected thousands of clinical questions from more than one hundred family doctors. They have excluded requests for facts that could be obtained from the medical records (e.g., "What was her blood potassium concentration?") or from the patient (e.g., "How long have you been coughing?"). The National Library of Medicine has made available a total of 4,653 clinical questions[3] over different studies (Alper et al. 2001, D'Alessandro et al. 2004, Ely et al. 1999, Ely et al. 2000, Gorman et al. 1994, Niu et al. 2003).

Although physicians tend to ask many questions when caring for patients, studies have found that many physicians cannot find satisfactory answers for their questions. Ely and his colleagues have identified

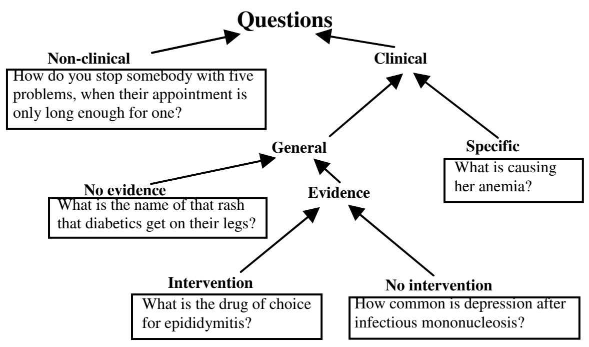

Questions
Non-clinical
How do you stop somebody with five problems, when their appointment is only long enough for one?
Clinical
General
Specific
What is causing her anemia?
No evidence
What is the name of that rash that diabetics get on their legs?
Evidence
Intervention
What is the drug of choice for epididymitis?
No intervention
How common is depression after infectious mononucleosis?

Figure 1: "Evidence taxonomy" created by Ely and his colleagues (Ely et al. 2002) with examples.

59 obstacles that prevent physicians from finding answers to some of those questions (Ely et al. 2002). They found that the most common class of obstacle preventing physicians from getting answers to their clinical questions is that the information resources do not always contain the answers. For example, biomedical information resources can not answer non-clinical questions such as "How do you stop somebody with five problems, when their appointment is only long enough for one?"

In addition, in the medical domain, physicians are urged to practice *Evidence Based Medicine* when faced with questions about how to care for their patients (Gorman et al. 1994, Straus and Sackett 1999, Bergus et al. 2000). Evidence based medicine refers to the use of the best evidence from scientific and medical research to make decisions about the care of individual patients. The needs of evidence based medicine have also driven biomedical researchers to provide evidence in their research reports. With this in mind, Ely and his colleagues have created an "evidence taxonomy" to organize medical questions into five hierarchical categories (shown in Figure 1).

In addition, Ely and his colleagues have manually annotated 200 clinical questions, placing them into the five leaf categories shown in Figure 1. Those 200 questions were randomly selected from the thousands that they collected (Ely et al. 2002). After searching for answers to these questions in biomedical literature and online medical databases, Ely and his colleagues have concluded that the *Non-clinical*, *Specific,* and *Non-evidence* questions are not answerable, while both subcategories of *Evidence* (i.e., *Intervention* and *No-intervention* questions) are potentially answerable with evidence. *Non-clinical* questions do not deal with the specific domain, *Specific* questions require information from a patient's record, and *Non-evidence* questions are questions for which the answer is generally unknown. This results in a total of 83 unanswerable questions and 117 answerable questions. These 200 questions have been used in our

[1] http://trec.nist.gov/
[2] http://www.informedia.cs.cmu.edu/aquaint/
[3] http://clinques.nlm.nih.gov/JitSearch.html

study to automatically classify a question as either *Answerable* or *Unanswerable*.

5 Supervised Machine-Learning

Separating *Answerable* from *Unanswerable* is a task of document categorization. We have explored supervised machine-learning approaches to automatically classify a question into one of these two categories. In the following subsections, we will describe the machine-learning systems, the learning features, the cross-validation methodology, and the evaluation metrics used for our classification.

5.1 Systems

We have applied seven text categorization systems using a variety of approaches. Five of the seven systems comprise the publicly available Rainbow package (McCallum 1996). The approaches used by these systems are Rocchio/TF*IDF, K-nearest neighbors (kNN), maximum entropy, probabilistic indexing, and naïve Bayes. All of these approaches have been used successfully for text categorization tasks (Sebastiani 2002). We have also applied support vector machines[4] because it has shown to be successful for text categorization tasks (Yang and Liu 1999, Sebastiani 2002). Additionally, we have explored the machine-learning system, BINS (Sable and Church 2001), which is a generalization of Naive Bayes. Brief descriptions of these approaches are given in the following subsections; see (Sable 2003) for more detailed descriptions of these machine-learning algorithms.

Rocchio/TF*IDF

A Rocchio/TF*IDF system (Rocchio 1971) adopts TF*IDF, the vector space model typically used for information retrieval, for text categorization tasks. Rocchio/TF*IDF represents every document and category as a normalized vector of TF*IDF values. The term frequency (TF) of a token (typically a word) is the number of times that the token appears in the document or category, and the inverse document frequency (IDF) of a token is a measure of the token's rarity (usually calculated based on the training set). For test documents, scores are assigned to each potential category by computing the similarity between the document to be labeled and the category, often computed to be the cosine measure between the document vector and the category vector; the category with the highest score is than chosen.

K-Nearest Neighbors (kNN)

A K-nearest neighbors system determines which training questions are the most similar to each test question, and then uses the known labels of these similar training questions to predict a label for the test question. The similarity between two questions can be computed as the number of overlapping features between them, as the inverse of the Euclidean Distance between feature vectors, or according to some other measure. The kNN approach has been successfully applied to a variety of text categorization tasks (Sebastiani 2002, Yang and Liu 1999).

Naïve Bayes

The naïve Bayes approach is commonly used for machine learning and text categorization tasks. Naïve Bayes is based on Bayes' Law and assumes conditional independence of features. For text categorization, this “naive” assumption amounts to the assumption that the probability of seeing one word in a document is independent of the probability of seeing any other word in a document, given a specific category. Although this is clearly not true in reality, Naive Bayes has been useful for many text classification and other information retrieval tasks (Lewis 1998). The label of a question is the category that has the highest probability given the “bag of words” in the document. To be computationally feasible, log likelihood is generally maximized instead of probability.

Probabilistic Indexing

This is another probabilistic approach that chooses the category with the maximum probability given the words in a document. Probabilistic indexing stems from Fuhr’s probabilistic indexing paradigm (Fuhr 1988), which was originally intended for relevance feedback and was generalized for text categorization by Joachims (Joachims 1997), who considered it a probabilistic version of a TF*IDF classifier, although it more closely resembles naïve Bayes. Unlike naïve Bayes, the number of times that a word occurs in a document comes into play, because the probability of choosing each specific word, if a word were to be randomly selected from the document in question, is used in the probabilistic calculation. Although this approach is less common in the text categorization literature, one author of this paper has seen that it is very competitive for many text categorization tasks (Sable 2003).

Maximum Entropy

This is another probabilistic approach that has been successfully applied to text categorization (Nigam et. al. 1999). A maximum entropy system starts with the initial assumption that all categories are equally likely. It then iterates through a process known as improved iterative scaling that updates the estimated probabilities until some stopping criterion is met. After the process is complete, the category with the highest probability is selected.

[4] We have applied Libsvm, which is available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/

Support Vector Machines (SVMs)
A support vector machine system is a binary classifier that learns a hyperplane in a feature space that acts as an optimal linear separator which separates (or nearly separates) a set of positive examples from a set of negative examples with the maximum possible margin (the margin is defined as the distance from the hyperplane to the closest of the positive and negative examples). SVMs have been widely tested to be one of the best machine-learning classifiers, and previous studies have shown that SVMs outperform other machine learning algorithms for open-domain sentence classification (Zhang and Lee 2003) and other text categorization tasks (Yang and Liu 1999, Sebastiani 2002).

BINS
The BINS system (Sable and Church 2001) uses a generalization of Naive Bayes. BINS places words that share common features into a single bin. Estimated probabilities of a token appearing in a document of a specific category are then calculated for bins instead of individual words, and this acts as a method of smoothing which can be especially important for words with scarce evidence. BINS has proven to be very competitive for many text categorization tasks (Sable 2003, Yu and Sable 2005).

5.2 Learning Features

We have explored bag of words as learning features. Since our collection consists of biomedical, domain-specific questions, we have also incorporated concepts and semantic types from the largest biomedical knowledge resource Unified Medical Language System (UMLS), as additional learning features for question classification. Including the UMLS features represents a method of *class-based* smoothing (Resnik, 1993) where the probabilities of individual or sparse words are smoothed by the probabilities of larger or less sparse semantic classes. In the following subsection, we will describe UMLS concepts and semantic types.

5.3 The Unified Medical Language System

The National Library of Medicine (NLM) has created the Unified Medical Language System (UMLS)[5] (Humphreys and Lindberg 1993) to aid in the development of computer systems that process text in the biomedical domain. The UMLS includes the Metathesaurus, a large database that incorporates more than one million biomedical concepts plus synonyms and concept relations. For example, the UMLS links the following synonymous terms as a single concept: *Achondroplasia*, *Chondrodystrophia, Chondrodystrophia fetalis*, and *Osteosclerosis congenita.*

[5] http://www.nlm.nih.gov/research/umls/

The UMLS also consists of the Semantic Network, which contains 135 semantic types; each semantic type represents a more general category to which certain specific UMLS concepts can be mapped via is-a relationships (e.g., *Pharmacologic Substance*). The Semantic Network also describes a total of 54 types of semantic relationships (e.g., hierarchical *is-a* and *part-of* relationships). Each specific UMLS concept in the Metathesaurus is assigned one or more semantic types. For example, *Arthritis* is assigned to one semantic type, *Disease or Syndrome*; *Achondroplasia* is assigned to two semantic types, *Disease or Syndrome* and *Congenital Abnormality*.

The National Library of Medicine makes available MMTx[6], a programming implementation of MetaMap (Aronson 2001), which maps free text to UMLS concepts and their associated semantic types. The MMTx program first parses text, separating the text into noun phrases. Each noun phrase is then mapped to a set of possible UMLS concepts, taking into account spelling and morphological variations, and each concept is weighted, with the highest weight representing the most likely mapped concept. The UMLS concepts are then mapped to semantic types according to definitive rules as described in the previous paragraph. MMTx can be used either as a standalone application or as an API that allows systems to incorporate its functionality. In our study, we have applied MMTx to map terms in a question to appropriate UMLS concepts and semantic types; we have added the resulting concepts and semantic types as additional features for question classification.

5.4 Cross-Validation

To evaluate the performance of each system, we have performed four-fold cross-validation. Specifically, we have randomly divided our corpus into four subsets of 50 questions each for four-fold cross-validation experiments; i.e., we train on 150 questions and test on the other 50, and perform four such experiments with each of the text-categorization system that we have tested. We have performed these experiments using bag of words alone as well as bag of words plus combinations of the other features discussed in the previous subsection.

5.5 Evaluation Metrics

Results are reported according to two metrics. The first metric is overall accuracy, which is simply the percentage of questions that are categorized correctly (i.e., they are correctly labeled as *Answerable* or *Unanswerable*). A simple baseline system that

[6] http://mmtx.nlm.nih.gov/

automatically categorizes all questions as *Answerable* (something that most automatic QA systems assume anyway) would achieve an overall accuracy of 117/200 = 58.5%.

The second metric is the F1 measure (Rigsbergen 1979) for the *Answerable* category. The F1 measure combines the precision (P) for the category (the number of documents correctly placed in the category divided by the total number of document placed in the category) with the recall (R) for the category (the number of documents correctly placed in the category divided by the number of documents that actually belong to the category). The metric is calculated as F1 = (2 * P * R) / (P + R); the result is always in between the precision and the recall but closer to the lower of the two, thus requiring a good precision and recall in order to achieve a good F1 measure.

6 Results

Since we have applied MMTx for identifying appropriate UMLS concepts and semantic types for each question, which are then included as features for question classification, we have evaluated the precision of MMTx for this task. One of the authors (Dr. Carl Sable) has manually examined the 200 questions comprising our corpus as described in Section 3. MMTx assigns 769 UMLS Concepts and 924 semantic types to the 200 questions (remember that some UMLS concepts are mapped to more than one semantic type, as described in Section 5.3). Our analysis has indicated that 164 of the UMLS Concept labels and 194 of the semantic type labels are wrong; this indicates precisions of 78.7% and 79.0%, respectively. An example of a case that MMTx gets wrong is the abbreviation "pt", which, in this corpus, is often used as an abbreviation for "patient"; MMTx typically assigns this to the UMLS concept *pint* and the semantic type *Quantitative Concept*. Note that manually estimating the recall of MMTx would be difficult, since it would require an expert that is familiar with all possible UMLS concepts and the ways to express them.

We have compared the performance of the machine-learning systems specified in Section 5.1 used to label questions as *Answerable* or *Unanswerable* with feature combinations described in Sections 5.2 and 5.3. Table 1 shows the results of all systems tested using the cross-validation procedure explained in Section 5.4. For four of the six feature combinations, the system that achieves the best performance is the Probabilistic Indexing system; the overall accuracy is as high as 80.5% and the F1 measure for the *Answerable* category is as high as 83.0%. We have also found that incorporating UMLS concepts or semantic types often improves performance compared to using bag-of-words only.

Table 2 lists six questions that are predicted incorrectly by the best machine-learning classifier (i.e., probabilistic indexing with bag-of-words and UMLS concepts as features). Questions are presented exactly as they were expressed by physicians, including bad grammar and incorrect spellings. Since we can not control what physicians will type, these represent complexities that will have to be dealt with by a real-world system.

Answerable: 1) What is best time to get OB ultrasound for dating and to see other things? 2) What are long-term options for hemorrhagic gastritis beyond H2 blockers? 3) Does Zoloft cause stomach upset?
Unanswerable: 4) What is the cause of this patient's tremor? 5) What dose the HMO formulary say I can use for this patient's nasal condition? 6) How long shall I treat knee injury w conservative measures before referring?

Table 2: Three *Answerable* and three *Unanswerable* questions that the classifier predicts incorrectly.

In order to examine useful features for the classification, we have calculated log likelihood ratios of word occurrences in each of our two categories (i.e., *Answerable* and *Unanswerable*). For each word/category pair, the level of indication of the word for the category is computed as the log likelihood of seeing the word in a question of the specified category minus the log likelihood of seeing the word in the other category. Thus, the strength of the word for a category will only be positive if it is the more likely category of the two, given the word, and the magnitude of the strength will depend on the likelihood of the other category. For each question, the strength of all words in the question have been computed for both categories based on evidence from the other questions (one category will have a positive strength and the other category will have a negative strength for each word), and the top words for both categories have been examined. For example, consider the following *Answerable* question:

"How soon should you ambulate a patient with a deep vein thrombosis?"

The top three words indicating the *Answerable* and *Unanswerable* categories, with scores calculated as described above (higher scores representing stronger indications of a category), are:

Answerable: you (1.8), should (1.0), how (0.5)
Unanswerable: a (1.6), patient (0.2), with (-0.2)

ML Approach	Performance Using Features (C means UMLS Concepts, ST means semantic types)					
	Bag of Words	Words+C	Words+ST	Words+C+ST	C only	ST only
*Rocchio/TF*IDF	74.0 (77.4)	72.5 (75.8)	74.5 (77.5)	74.0 (77.2)	67.6 (70.3)	65.0 (68.5)
*kNN	68.5 (71.7)	69.0 (73.5)	65.5 (69.9)	65.5 (70.1)	65.0 (66.0)	61.5 (61.6)
*MaxEnt	66.0 (69.6)	68.0 (73.1)	70.5 (76.1)	69.5 (74.9)	65.0 (67.6)	65.5 (70.9)
*Prob Indexing	78.0 (81.7)	80.5 (83.0)	80.0 (82.9)	79.0 (82.1)	70.0 (70.8)	66.5 (70.0)
*Naïve Bayes	68.0 (74.8)	74.5 (77.9)	73.5 (77.6)	73.0 (76.7)	71.0 (76.0)	64.0 (69.2)
**SVMs	67.5 (74.9)	68.0 (74.6)	69.0 (75.4)	67.0 (73.6)	62.5 (70.1)	67.0 (69.8)
BINS	72.0 (74.5)	72.0 (75.2)	68.5 (72.2)	66.5 (69.1)	66.0 (70.7)	58.5 (64.4)

Table 1: Percentages for overall accuracy and F1 scores (in parentheses) of machine-learning systems with different combinations of learning features for classifying *Answerable* versus *Unanswerable* biomedical questions.

"*" indicates Rainbow implementation
"**" indicates libsvm implementation

Note that the word "with" has a negative weight; this means that it is really an indicator of an *Answerable* question. So this question contains only two words that are indicative of an *Unanswerable* question. Note that the words "ambulate" and "thrombosis" are infrequent and do not show up in either list; it is likely that these words do not occur in any other question, in which case there is no evidence for them and their scores would be 0 for both categories.

We have observed that many stop words have high scores and we have therefore hypothesized that stop words may play an important role for this classification task. Studies have found that for some non-content based categorization tasks, stop words, have proven to be useful; one example is authorship attribution (Mosteller and Wallace 1963). Table 3 shows the change in classification performance when we remove the stop words from the questions. (These results have only been computed for the Rainbow systems, which provide a simple mechanism to do this.) Our results show that when we exclude stop words, this tends to decrease performance, and in particular this is true for the naïve Bayes and probabilistic indexing systems. These results provide evidence that stop words may play an important role for classifying a question posed by a physician as either *Answerable* or *Unanswerable*.

7 Discussion

Based on overall accuracy results, all systems beat random guessing (50.0%) and the simple baseline system that is described in Section 5.5. (58.5%). Furthermore, the F1 measure for the *Answerable* category is higher than the overall accuracy for each system; this indicates that all systems have a slight disposition towards the *Answerable* category (based on the training documents). Compared to typical text categorization tasks, our task is more challenging because our data set is small (only 150 short questions are used for training at one time) which leads to a small feature space. Nevertheless, most systems achieve reasonable performance with several feature combinations, and the probabilistic indexing system achieves and overall accuracy that is up to 22.0% higher than the simple baseline system.

A manual inspection of the questions that are classified incorrectly reveals the problem of data sparseness; for most of these questions, the majority of words do not occur in any other question in the data set. For example, in Table 2, the word "hemorrhagic" in question 2 does not appear in any other question. We speculate that a larger training set could potentially alleviate this problem and boost our results. We have also found that some questions may be mislabeled. For example, the question "Would it be better to put her on a potassium sparing diuretic or just potassium" has been labeled as *Answerable*, but it seems to us that this is a patient-specific question that should be labeled as *Unanswerble.*

Our results show a moderate increase of performance when including the UMLS features. We have observed that many UMLS concepts in these questions, when labeled correctly, represent information that was already present in the bag of words representation. We also found that some semantic types tend to be very general, appearing in both *Answerable* and *Unanswerable* questions commonly. For example, the semantic type *Disease or Syndrome* occurs 44 times in 37 *Answerable* questions and 30 times in 26 *Unanswerable* questions. Therefore, these tokens will not play an important role for classification. However, we believe that this same information will be indispensable for potential future work discussed in Section 8.

8 Conclusions and Future Work

This paper describes what we believe is the first attempt in the field of question answering to automatically identify answerable questions, i.e., the questions for which answers can be found in the

ML Approach	Performance Difference when Stop Words are Excluded			
	Bag of Words	Words+C	Words+ST	Words+C+ST
*Rocchio/TF*IDF	-3.0 (-3.1)	-6.5 (-6.4)	-5.5 (-4.2)	-4.5 (-3.4)
*kNN	+1.5 (+1.4)	-1.0 (-2.1)	-1.5 (-1.2)	-3.0 (-3.1)
*MaxEnt	+0.5 (-2.2)	-7.5 (-7.9)	-2.5 (-1.5)	-2.0 (-0.8)
*Prob Indexing	-3.0 (-4.4)	-6.5 (-7.5)	-7.5 (-6.7)	-4.0 (-3.5)
*Naïve Bayes	-6.0 (-3.7)	-9.5 (-7.8)	-5.0 (-5.4)	-6.5 (-7.6)

Table 3: Increase (+) or decrease (-) of overall accuracy and F1 scores (in parentheses) after we remove stop words for classifying *Answerable* versus *Unanswerable* biomedical questions.

"*" indicates Rainbow implementation

available corpora. Our results are promising; the best system achieves an 80.5% overall accuracy for separating *Answerable* from *Unanswerable* questions based on a small training set. We consider this result to represent an important proof-of-concept. In the future, we expect that biomedical QA systems such as BioMedQA will be able to accurately distinguish *Answerable* from *Unanswerable* questions relying on more advanced processing described in the following paragraph.

We believe that it will eventually be possible to automatically decompose biomedical questions according to component question types, which are described in (Ely et al. 1999); for example, "What are the affects of <*drug*> on <*disease*>?". We speculate that the recognition of UMLS concepts and semantic types using tools such as MMTx will play a key role in this type of question classification. If questions can be accurately mapped to component question types, then the filtering of unanswerable questions will become straight-forward; an even greater benefit will be that specific answer strategies could be developed for each answerable component question type.

Acknowledgements

We thank Dr. John Ely, Dr. Jerry Osheroff, and Dr. James Cimino for valuable discussion and for making their annotation available to us. This project is supported by JDRF 6-2005-835. Any opinions, findings, or recommendations are those of the authors and do not necessarily reflect JDRF's views.

References

Allen, J.F. and C.R. Perrault. 1986. Analyzing intention in utterances. In B.J. Grosz, K.S. Jones, and B.L. Weber, editors, Readings in Natural Language Processing, Pages 441-458. Morgan Kaufmann Publishers, Inc., Los Altos, California, 1986.

Alper, B., J. Stevermer, D. White, and B. Ewigman. 2001. Answering family physicians' clinical questions using electronic medical databases. *J Fam Pract* 50: 960-965.

Aronson, A. 2001. Effective Mapping of Biomedical Text to the UMLS Metathesaurus: The MetaMap Program. *American Medical Information Association*.

Bergus, G.R., Randall, C.S., Sinift, S.D. and D.M. Rosenthal. 2000. Does the structure of clinical questions affect the outcome of curbside consultations with specialty colleagues? Arch Fam Med. 9(6): 541-7.

Chalupsky, H. and T.A. Russ. 2002. WhyNot: Debugging Failed Queries in Large Knowledge Bases. In Proceedings of the fourteenth innovative applications of artificial intelligence, pages 870-877, AAAI Press.

D'Alessandro, D.M., Kreiter, C.D., and M.W. Peterson. 2004. An evaluation of information seeking behaviors of general pediatricians. Pediatrics 113: 64-69.

Ely, J., J. Osheroff, M. Ebell, G. Bergus, B. Levy , M. Chambliss, and E. Evans. 1999. Analysis of questions asked by family doctors regarding patient care. *BMJl*: 358-361.

Ely, J., J. Osheroff, M. Ebell, M. Chambliss, D. Vinson, J. Stevermer, and E. Pifer. 2002. Obstacles to answering doctors' questions about patient care with evidence: qualitative study. *BMJ* 324: 710-713.

Ely, J., J. Osheroff, P. Gorman, M. Ebell, M. Chambliss, E. Pifer, and P. Stavri. 2000. A taxonomy of generic clinical questions: clasification study. *BMJ* 321: 429-432.

Fuhr, N. 1998. Models for retrieval with probabilistic indexing. *Information Processing and Management* 25(1):55-72.

Gaasterland, T., P. Godfrey, and J. Minker. 1994. An overview of cooperative answering. In *Nonstandard Queries and Nonstandard Answers*, pages 1-40, Clarendon Press.

Gorman, P., J. Ash, and L. Wykoff. 1994. Can primary care physician's questions be answered using hte medical journal literature? *Bull Med Libr Assoc* 82: 140-146.

Grice, H. 1975. Logic and conversation. In *Syntax and Semantics*, Academic Press.

Harabagiu, S.M., Maiorano, S.J., Moschitti, A, and C.A. Bejan. 2004. Intentions, implicatures and processing of complex questions. In *HLT-NAACL Workshop on Pragmatics of Question Answering.*

Hermjakob, U. 2001. Parsing and question classification for question answering. In *Proceedings of ACL Workshop on Open-Domain Question Answering.*

Hovy, E., Gerber, L., Hermjakob, U., Junk, M., and C.Y. Lin. Question answering in Webclopedia. In *Proceedings of the TREC-9 Conference.*

Hughes, S. 1986. Question classification in rule-based systems. In *Annual Technical Conference of the British Computer Society Specialist Group on Expert Systems.*

Humphreys, B. L., and D. A. Lindberg. 1993. The UMLS project: making the conceptual connection between users and the information they need. *Bull Med Libr Assoc* 81: 170-7.

Jacquemart, P., and P. Zweigenbaum. 2003. Towards a medical question-answering system: a feasibility study. *Stud Health Technol Inform* 95: 463-8.

Joachims, T. 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In Proceedings of the 14th International Conference on Machine Learning.

Lewis, D. 1998. Naive (Bayes) at forty: the independence assumption in information retrieval. In Proceedings of the European Conference on Machine Learning.

McCallum, A. 1996. A toolkit for statistical language modeling, text retrieval, classification, and clustering. *http://www.cs.cmu.edu/~mccallum/bow.*

Mosteller, F. and D. Wallace. 1963. Inference in an authorship problem. Journal of the American Statistical Association 58:275-309.

Nigam, K.; Lafferty, J.; and McCallum, A. 1999. Using maximum entropy for text classification. In Proceedings of the IJCAI-99 workshop on machine learning for information filtering.

Niu, Y., G. Hirst, G. McArthur, and P. Rodriguez-Gianolli. 2003. Answering clinical questions with role identification. *ACL workshop on natural language processing in biomedicine*.

Resnik, P. 1993. Selection and information: A class-based approach to lexical relationships. PhD thesis. Department of Computer and Information Science, University of Pennsylvania.

Rigsbergen, V. 1979. *Information Retrieval, 2nd Edition*. Butterworths, London.

Rocchio, J. 1971. Relevance feedback in information retrieval. In The Smart Retrieval System: Experiments in Automatic Document Processing, pages 313-323, Prentice Hall.

Sable, C. 2003. Robust Statistical Techniques for the Categorization of Images Using Associated Text. Columbia University, New York.

Sable, C., and K. Church. 2001. Using Bins to empirically estimate term weights for text categorization. *EMNLP*, Pittsburgh.

Sackett, D., S. Straus, W. Richardson, W. Rosenberg, and R. Haynes. 2000. *Evidence-Based Medicine: How to practice and teach EBM*. Harcourt Publishers Limited, Edinburgh.

Sebastiani, F. 2002. Machine learning in automated text categorization. *ACM Computing Surveys.* 34: 1-47.

Straus, S., and D. Sackett. 1999. Bringing evidence to the point of care. *Journal of the American Medical Association* 281: 1171-1172.

Yang, Y., and X. Liu. 1999. A re-examination of text categorization methods. In Proceedings in the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.

Yu, H., and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. *EMNLP*.

Yu, H., and C. Sable, and H. R. Zhu. 2005. Classifying medical questions based on an evidence taxonomy. Forthcoming.

Zhang, D. and Lee, WS. 2003. Question classification using support vector machines. In *Proceedings of the 26th Annual International ACM SIGIR conference, pages 26-32.*

Reasoning over Dependency Relations for QA[*]

Gosse Bouma and **Jori Mur** and **Gertjan van Noord**
Information Science
Rijksuniversiteit Groningen
Postbus 716, 9700 AS Groningen
{gosse,mur,vannoord}@let.rug.nl

Abstract

We present a QA system in which question analysis, off-line answer extraction and reranking and identification of potential answers from an IR all make use of syntactic dependency relations. We show that the addition of equivalence rules over patterns of dependency relations, which capture cases where different syntactic patterns express the same semantic relationship, improves the performance of various modules of the system.
Keywords: Reasoning, Language Processing.

1 Introduction

English QA systems can make use of robust wide-coverage parsers [Lin, 1994; Collins, 1999; Briscoe and Carroll, 2002; Clark and Curran, 2004], which either produce dependency relations directly (i.e. tuples of the form ⟨`Head, Rel, Dep`⟩, where `Head` is the head word, `Rel` the name of a dependency relation, and `Dep` the head word of the dependent), or which can be used to derive such tuples. For question analysis (i.e. the task of determining, among others, what the category of the answer to a question is), the use of dependency relations is relatively wide-spread as it tends to produce more accurate results than regular expressions.

The use of dependency information for answer identification (i.e. the task for finding an exact answer in a list of passages returned by an IR system) is much less common. One reason is the fact that parsing of large amounts of text remains a challenge, even for English. Most QA systems make use of an IR step to retrieve a reasonably sized subset of text fragments relevant to the question from the full document collection. Full parsing of these fragments is often not attempted, however. Instead, keyword matching, regular expressions, part-of-speech tagging, and recognition of base constituents is used for finding the exact answer. Notable exceptions are [Katz and Lin, 2003], who processed a text collection (of about 20,000 articles) exhaustively using Minipar, and extracted specific relation tuples from the resulting dependency tuples, and [Litkowski, 2004], who describes a QA system based on a fully parsed corpus, stored in XML. A second issue which has been addressed by several researchers is how to use dependency relations exactly. Several researchers require an exact match between dependency tuples derived from the question and the answer. This is true for the QA systems of both [Katz and Lin, 2003] and [Litkowski, 2004]. The latter uses XPath to generate queries which XML documents must match. Generic metrics, applicable to all question types, have been proposed as well. [Punyakanok *et al.*, 2004] compute the edit distance between the dependency trees of the question and answer, and select answers from sentences which minimize this distance. Comparing trees in a principled way is difficult, and most researchers therefore turn dependency trees (as produced by a parser) into sets of dependency tuples. The PiQASso system [Attardi *et al.*, 2002] and AnswerFinder [Mollá and Gardiner, 2005] compute the match between question and answer using a metric which basically computes the overlap in dependency relations between the two.

Although dependency relations eliminate many sources of variation that systems based on surface strings have to deal with, it is also true that the same semantic relations can sometimes be expressed by several dependency relation patterns. [Lin and Pantel, 2001] show how dependency paths expressing the same semantic relation can be acquired from a corpus automatically. [Rinaldi *et al.*, 2003] argue that such equivalences, or paraphrases, can be especially useful for QA in technical domains.

In this paper, we present a QA system for Dutch, which makes extensive use of dependency relations. We show that an existing, wide-coverage, parser for Dutch can be used effectively for QA by incoporating Named Entity classification and by using a disambiguation model trained on a representative corpus fragment. The use of dependency relations in various parts of the QA system is supported by a module which allows dependency patterns to be (partially) matched against dependency relations. Furthermore, syntactic variation is accounted for by formulating equivalence relations over dependency patterns. We show that both the use of dependency matching in general, and the addition of equivalence rules, improves the performance of our QA system.

[*]This research was carried out as part of the research program for *Interactive Multimedia Information Extraction*, IMIX, fi nanced by NWO, the Dutch Organisation for Scientifi c Research.

2 In-depth Dependency Analysis for QA

The Alpino system is a linguistically motivated, wide-coverage, grammar and parser for Dutch. The constraint-based grammar follows the tradition of HPSG [Pollard and Sag, 1994]. It currently consists of over 500 grammar rules (defined using inheritance) and a large and detailed lexicon (over 100.000 lexemes). To ensure coverage, heuristics have been implemented to deal with unknown words and ungrammatical or out-of-coverage sentences (which may nevertheless contain fragments that are analyzable). The grammar provides a 'deep' level of syntactic analysis, in which WH-movement, raising and control, and the Dutch verb cluster (which may give rise to 'crossing dependencies') are given a principled treatment. The output of the system is a dependency graph, compatible with the annotation guidelines of the Corpus of Spoken Dutch.

A left-corner chart parser is used to create the parse forest for a given input string. A manually corrected treebank of 140.000 words was used to train a maximum entropy disambiguation model. Beam-search is used as a heuristic to extract the most probable parse from the parse forest effeciently. [Malouf and van Noord, 2004] show that the accuracy of the system, when evaluated on a test-set of 500 newspaper sentences, is over 88%, which is in line with state-of-the-art systems for English.

For the QA task, the disambiguation model was retrained on a corpus which contained additional material consisting of the (manually corrected) dependency trees of 650 quiz questions.[1] The retrained model achieves an accuracy on 92.7% on the CLEF 2003 questions and of 88.3% on CLEF 2004 questions.

A second extension of the system for QA, was the inclusion of a Named Entity Classifier. The Alpino system already includes heuristics for recognizing proper names. Thus, the classifier needs to classify strings which have been assigned a NAME part of speech by grammatical analysis, as being of the subtype PER, ORG, GEO or MISC.[2] To this end, we collected lists of person names (120K), geographical names (12K), organization names (26k), and miscalleneous items (2K). The data are primarily extracted from the Twente News Corpus, a collection of over 300 million words of newspaper text, which comes with annotation for the names of people, organizations, and locations, involved in a particular news story. For unknown names, a maximum entropy classifier was trained, using the Dutch part of the shared task for CONLL 2003.[3] The accuracy on unseen CONLL data of the resulting classifier (which combines dictionary look-up and a maximum entropy classifier) is 88.2%.

We have used the Alpino-system to parse the full text collection for the Dutch CLEF QA-task. To this end, the text collection was tokenized (into 78 million words) and segmented into (4.1 million) sentences. Parsing this amount of text takes well over 500 CPU days. We used a Beowulf Linux cluster of 128 Pentium 4 processors[4] to complete the process in about three weeks. The dependency trees are stored as (25 Gb of) XML. Fortunately, the analyzed material is not only useful for our QA system. It has been used to improve the coverage of the lexicon by error-mining [van Noord, 2004], and is a rich source of data for corpus linguistics. It has also been used by other research groups.[5]

[1]From the *Winkler Prins spel*, a quiz game. The material was made available to us by the publisher, *Het Spectrum, bv.*

[2]Various other entities which sometimes are dealt with by NEC, such as dates and measure phrases, can be identifi ed using the information present in POS tags and dependency labels.

[3]`http://cnts.uia.ac.be/conll2003/ner/`

[4]which is part of the High-Performance Computing centre of the University of Groningen

[5]The Amsterdam system for CLEF 2004 uses information extracted from the parsed corpus. The University of Nijmegen has used it to obtain realistic data for psycholinguistic experiments.

3 Equivalences over Dependency Relations

Several components of our QA system make use of dependency relations. All of these components need to check whether a given sentence satisfies a certain syntactic pattern. We have developed a seperate module for dependency pattern matching, which also accounts for syntactic variation.

The dependency analysis of a sentence gives rise to a set of dependency relations of the form ⟨ `Head/HIx, Rel, Dep/DIx` ⟩, where `Head` is the root form of the head of the relation, and `Dep` is the head of the constituent that is the dependent. `HPos` and `DIx` are string indices, which distinguish repeated occurrences of the same token in a string, and `Rel` is the name of the dependency relation. For instance, the dependency analysis of sentence (1-a) is (1-b).

(1) a. Mengistu kreeg asiel in Zimbabwe (*Mengistu was given asylum in Zimbabwe*)

b. $\left\{ \begin{array}{l} \langle\texttt{krijg/2, su, mengistu/1}\rangle, \\ \langle\texttt{krijg/2, obj1, asiel/3}\rangle, \\ \langle\texttt{krijg/2, mod, in/4}\rangle, \\ \langle\texttt{in/4, obj1, zimbabwe/5}\rangle \end{array} \right\}$

A dependency pattern is a set of (partially underspecified) dependency relations:

(2) $\left\{ \begin{array}{l} \langle\texttt{krijg/K, obj1, asiel/A}\rangle, \\ \langle\texttt{krijg/K, su, Su/S}\rangle \end{array} \right\}$

The following Prolog program defines when a pattern matches a set of dependency relations:

```
match([],_).
match(Pat,Rels) :-
  equivalent(LHS,RHS),
  partition(Pat,LHS,PatRest),
  partition(Rels,RHS,RelsRest),
  match(PatRest,RelsRest).

equivalent(Pat,Pat).
equivalent(LHS,RHS) :- eq(LHS,RHS).
equivalent(LHS,RHS) :- eq(RHS,LHS).
```

The relation `partition(Union,Set1,Set2)` holds if `Set1` ∪ `Set2` = `Union`, and `Set1` and `Set2` are disjoint. The pattern in (2) matches with the set in (1-b) and would, among others, instantiate the variable `Su` as `mengistu`.

Equivalences can be defined to account for the fact that in some cases we want a pattern to match a set dependency relations that slightly differs from it, but nevertheless expresses the same semantic relation. For instance, the subject of an active sentence may be expressed as a PP-modifier headed by *door* (*by*) in the passive:

(3) a. Zimbabwe verleende asiel aan Mengistu (*Zimbabwe gave asylum to Mengistu*)
  b. Aan Mengistu werd asiel verleend door Zimbabwe (*Mengistu was given asylum by Zimbabwe*)

The following equivalence rule accounts for this:

$$\texttt{eq}(\{\langle\texttt{Vb/V,su,Su/S}\rangle\}, \left\{\begin{array}{l}\langle\texttt{word/W,vc,Vb/V}\rangle,\\ \langle\texttt{Vb/V,mod,door/D}\rangle,\\ \langle\texttt{door/D,obj1,Su/S}\rangle\end{array}\right\})$$

Here, the verb *word* is (the root form of) the passive auxiliary, which takes a verbal complement headed by the verb `Vb`.

We have implemented 13 additional equivalence rules, to account for, among others, coordination, relative clauses, possessive relations expressed by the verb *hebben* (*to have*) as well as the following phenomena:

(4) a. de bondscoach van Noorwegen, Egil Olsen ⇔ Egil Olsen, de bondscoach van Noorwegen (*the coach of Norway, Egil Olsen ⇔ Egil Olsen, the coach of Norway*)
  b. Australië's staatshoofd ⇔ staatshoofd van Australië (*Australia's head of state ⇔ head of state of Australia*)
  c. president van Rusland, Jeltsin ⇔ Jeltsin is president van Rusland (*president of Russia, Jeltsin ⇔ Jeltsin is president of Russia*).
  d. Moskou heeft 9 miljoen inwoners ⇔ de 9 miljoen inwoners van Moskou (*Moskow has 9 million inhabitants ⇔ the 9 million inhabitants of Moskow*).
  e. Swissair en Austrian Airlines hebben vluchten naar Kroatië ⇔ Swissair heeft vluchten naar Kroatië (*Swissair and AA have flights into Croatia ⇔ Swissair has flights into Croatia*)
  f. Ulbricht liet de Berlijnse Muur bouwen ⇔ Ulbricht, die de Berlijnse Muur liet bouwen. (*Ullbricht had the Berlin Wall be built ⇔ Ullbricht, who had the Berlin Wall be built*)

The equivalence rules we have implemented so far express linguistic equivalences, and thus are both general adn domain independent.

Note that both the `partition` and `equivalent` relation in general will have multiple solutions, and thus, backtracking may be required in order to determine whether a pattern matches a set of dependency relations. Also note that, at least for the moment, for efficiency reasons we do not allow recursive application of equivalence rules. That is, given a solution for `equivalent(LHS,RHS)`, we require that `RHS` is a subset of the set of dependency relations `Rels`, and we do not try to find an equivalent $\texttt{RHS}'$ which is part of `Rels`.

4 Applications in QA

Question analysis. Question analysis is the task of assigning a specific class (`person, location, date, ...`) to a question. Our QA system often assigns an additional argument to the class, i.e. `location(della_alpi_stadion)` asks for the location of the *Della Alpi stadium*. Question analysis requires a combination of syntactic analysis and ontological resources. Syntactic analysis helps to determine the question stem in complex WH-phrases (*with which Palestinian organization...*) and can help to identify additional properties of the question (i.e. *Give the name of a Japanese city that was struck by an earthquake* asks for the name of city, not of an earthquake. Lexical semantic knowledge is required to recognize that *Which region in the US has ...* asks for a geographical named entity, whereas *Which car factory was bought by ...* as for a organization named entity.

The advantage of using dependency relations for this task has been widely recognized. The incorporation of equivalences over dependency relations means that patterns will automatically cover a certain amount of syntactic variation. The pattern in (5-a), for instance, ensures that the questions in (5-b) and (5-d) are both classified as `cause(Effect)`, with `Effect` instantiated as `rsi`, in spite of the fact that their word order and syntactic structure differs.

(5) a. $\left\{\begin{array}{l}\langle\texttt{ontsta/O, mod, waardoor/W}\rangle,\\ \langle\texttt{ontsta/O, su, Effect/E}\rangle\end{array}\right\}$
  b. Waardoor ontstaat RSI? (*What causes RSI?*)
  c. $\left\{\begin{array}{l}\langle\texttt{waardoor/1, wh, ontsta/2}\rangle,\\ \langle\texttt{ontsta/2, mod, waardoor/1}\rangle,\\ \langle\texttt{ontsta/2, su, rsi/3}\rangle\end{array}\right\}$
  d. Waardoor kan RSI ontstaan? (*What can cause RSI?*)
  e. $\left\{\begin{array}{l}\langle\texttt{waardoor/1, wh, kan/2}\rangle,\\ \langle\texttt{kan/2, su, rsi/3}\rangle,\\ \langle\texttt{ontsta/4, mod, waardoor/1}\rangle,\\ \langle\texttt{ontsta/4, su, rsi/3}\rangle\end{array}\right\}$

The modifier *waardoor* is analyzed as a modifier of the verb *ontstaan* in both cases, and the dependency analysis also makes explicit the fact that the subject of the modal verb *kan* also functions as subject of the verbal complement. A regular expression pattern would at least have to deal with the fact that the subject follows the verb *ontstaan* in (5-b), whereas it precedes the verb in (5-d).

Note also that the use of variables for dependents in a pattern allows identification of specific syntactic arguments in the question. The pattern in (5-a) picks up the subject as the `Effect` for which a cause is asked. This information is used in the answer extraction process.

Off-line answer extraction. Off-line methods have proven to be very effective in QA ([Fleischman *et al.*, 2003], [Jijkoun *et al.*, 2004]). Before actual questions are known, a corpus is exhaustively searched for potential answers to specific question types (`capital, abbreviation, inhabitants, year of birth, ...`). The answers are extracted from the corpus off-line and stored in a semi-structured table for quick and easy access.

One of the advantages of having the corpus analyzed in full is the fact that it opens up the possibility of off-line answer extraction based on dependency relations. [Jijkoun *et al.*, 2004] have shown for English that using dependency relations for this task can lead to significant improvements in recall over systems based on regular expression pattern matching. Dependency relations often allow patterns to be stated which are hard to capture using regular expressions.

The sentences in (6), for instance, all contain information about organizations and their founders.

(6) a. **Minderop** richtte **de Tros** op toen .... (*Minderop founded the Tros when...*)
 b. Op last van generaal De Gaulle in Londen richtte verzetsheld **Jean Moulin** in mei 1943 **de Conseil National de la Résistance (CNR)** op. (*Following orders of general De Gaulle, resistance hero Jean Moulin founded in May 1943 the Conseil National de la Résistance (CNR)*)
 c. **Het Algemeen Ouderen Verbond** is op 1 december opgericht door de nu 75-jarige **Martin Batenburg**. (*The General Pensioners Union was founded on Dec, 1, by, now 75-year old, Martin Batenburg.*)
 d. **Kasparov** heeft **een nieuwe Russische Schaakbond** opgericht en... (*Kasparov has founded a new Russian Chess Union and...*)
 e. ... toen **de Generale Bank** bekend maakte met de Belgische Post **een "postbank"** op te richten. (*when the General Bank announced to found a "postal bank" with the Belgian Mail*).

The verb expressed to relation (*oprichten*, to found) can take on a wide variety of forms (active, with the particle *op* split from the root, participle, and infintival, either the founder or the organization can be the first constituent in the sentence, in passives the founder may be part of a *door* (*by*) phrase, and in control constructions the founder may be found as the subject of a governing clause. In all cases, modifiers may intervene between the relevant constituents. Such variation is almost impossible to capture accurately using regular expressions, whereas dependency relations can exploit the fact that in all cases the organization and its founder can be identified as the object and subject of the verb with the root form *oprichten*. The pattern in (7) suffices to extract this relation from all of the examples above.[6]

(7) $$\left\{ \begin{array}{l} \langle \texttt{richt\_op/R, su, Founder/S} \rangle, \\ \langle \texttt{richt\_op/V, obj1, Founded/O} \rangle \end{array} \right\}$$

Equivalence rules can be used to deal with other forms of syntactic variation. For instance, once we define a pattern to extract the country and its capital from (8-a), the equivalence rules illustrated in (4-a), (4-b), and (4-c) can be used to match this single pattern against the alternative formulations in (8-b)- (8-d) as well.

[6]The fact that the by-phrase in passives acts as the logical subject of the participle is accounted for by means of an equivalence rule.

(8) a. de hoofdstad van Afghanistan, Kabul (*the capital of Afghanistan, Kabul*)
 b. Kabul, de hoofdstad van Afghanistan (*Kabul, the capital of Afghanistan*)
 c. Afghanistans hoofdstad, Kabul (*Afghanistan's capital, Kabul*)
 d. Kabul is de hoofdstad van Afghanistan (*Kabul is the capital of Afghanistan*)

The table below illustrates the effect of using equivalence rules when applying the extraction patterns to the full CLEF corpus:

Table	-Equiv		+Equiv		Incr (%)	
	tuples	uniq	tuples	uniq	tpls	uniq
Abbreviation	21.170	8.405	21.497	8.543	1	1
Age	15.981	13.716	22.143	18520	38	35
Born Date	1.545	1.356	2356	1.990	54	47
Born Loc	371	351	937	879	152	150
Capital	1.940	406	2.146	515	10	27
Currency	3.111	124	6.619	222	113	80
Died Age	522	379	1.127	834	116	120
Died Date	374	349	583	544	56	55
Died Loc	364	332	664	583	82	76
Founded	604	559	1.021	953	69	70
Function	54.016	28.543	77.028	46.589	43	63
Inhabitants	529	473	708	633	34	34
Nobel Prize	75	67	169	141	125	110

For all relations, except abbreviations, which are found in a single syntactic environment, the overall number of extracted tuples, as well as the number of unique tuples increases considerably. Of course, for each relation,the number of extracted relations could have been increased by a similar amount by expanding the number of patterns for that relation. The interesting point here is that in this case this was achieved by adding a single, generic component.

The development and maintenance of extraction patterns is further facilitated by the fact that multiple dependency relations may be combined into paths (i.e. $\langle$`City, mod, tellend, me, Inhabitants`$\rangle$ refers to the pair $\langle$`City, mod, tellend`$\rangle$ and $\langle$`tellend, me, Inhabitants`$\rangle$), by providing developers with tools which support visualization of dependency relations as syntactic trees.

Answer reranking and extraction. For those questions that are not answered by the off-line method (i.e. either because no table exists for the given question type, or because no matching table entry was found), the QA system passes a set of keywords extracted from the question to the IR engine. IR returns a set of relevant paragraphs. Within this set, we try to identify the sentence which most likely contains the answer, and we try to extract the answer from the sentence.

For each sentence A, we compute its `a-score` relative to the question Q as follows:

$$\begin{aligned} \texttt{a-score}(Q, A) = \quad & \alpha.\texttt{d-score}(Q, A) \\ + \quad & \beta.\texttt{type-score}(Q, A) \\ + \quad & \gamma.\texttt{IR}(Q, A) \end{aligned}$$

Here, `d-score` expresses to what extent the dependency relations of Q and A match, `type-score` expresses whether a constituent matching the question type of Q could be found (in the right syntactic context) in A, and IR combines the score assigned by the IR engine and a score which expresses to what extent the proper names, nouns, and adjectives in Q and A plus the sentence immediately preceding A overlap. α, β and γ are (manually set) weights for these scores.

The d-score computes to what extent the dependency structure of question and answer match. To this end, the set of dependency relations of the question is turned into a pattern Q, by removing the dependency relations for the question word, and then substituting all string positions by variables. The d-score is the cardinality of the largest subset Q' of Q divided by $|Q|$, such that $\texttt{match}(Q', A)$ holds (where A is the set of dependency relations of the answer):

$$\texttt{d-score}(Q,A) = \frac{|\underset{Q'}{\arg\max}\ \{Q' | Q' \subset Q \wedge \texttt{match}(Q', A)\}|}{|Q|}$$

Note that dependency matching is considerably more subtle than keyword matching. A case in point are Q/A-pairs such as the following:

(9) a. Wie is voorzitter van het Europese Parlement? (*Who is chair of the European Parliament?*
b. Karin Junkers (SPD), lid van het Europese Parlement en voorzitter van de vereniging van sociaal-democratische vrouwen in Europa...(*Karin Junkers (SPD), member of the European Parliament and chair of the society of social-democrat women in Europe...*)

Here, (9-b) does not contain the correct answer in spite of the fact that it contains all keywords from the question. In fact, even most of the dependency relations of the question are present in the answer, but crucially, there is no substitution for W that would satisfy:

$$\texttt{match}(\left\{ \begin{array}{l} \langle\texttt{voorzitter/V,mod,van/W}\rangle, \\ \langle\texttt{van/W,obj1,parlement/X}\rangle \end{array} \right\}, Q)$$

The type-score indicates whether a suitable answer type was found in the sentence (i.e. a date question requires an answer containing a dependent with Part-of-Speech tag noun(date)). Higher scores are assigned to potential answers where the phrase that matches the question type is also in some syntactic dependency relation to the topic of the question. I.e. given the question in (10-a), classified as date(hereniging), the date phrase *in oktober 1990* in (10-b) receives a higher score than the date phrase *in 1962* in (10-c), as the first is a dependent of *hereniging,* whereas the second is not.

(10) a. Wanneer vond de Duitse hereniging plaats? (*When did the German unification take place?*)
b. Sinds de Duitse hereniging in oktober 1990... (*Since the German unification in October 1990...*)
c. Al in 1962 voorspelde hij de Duitse hereniging. (*As early as 1962 he predicted the German unification.*)

Merging off-line and IR-based techniques. One of the advantages of our approach is that the technology used for off-line and IR-based answer extraction becomes virtually identical. The only real difference is that the latter relies on IR to make a first selection of relevant paragraphs, whereas the off-line method performs an exhaustive search. Consequently, the metric used for selecting the best answer from a list of results provided by IR can be used for reranking the results of table look-up as well. Cases where this is useful, are questions like (11-a) and (11-b):

(11) a. Wie is de Duitse minister van Economische Zaken? (*Who is the German minister of Economy?*)
b. Wie was president van de VS in de Tweede Wereldoorlog? (*Who was president of the US during the second World War?*)

These are hard to account for by off-line methods. Question (11-a) and (11-b) would be classified as function(minister,duits) and function(president,vs), respectively, by question analysis, and thus, in principle can be answered by consulting the functions-table. However, this ignores the modifiers *van Economische Zaken* and *in de Tweede Wereldoorlog*, which, in this case, are crucial for finding the correct answer.

Applying the same scoring technique to facts extracted from the table, as to on-line extracted facts, can help to overcome this problem. In particular, the d-score between the question and the sentence on which a table entry is based, can be used as an additional factor (in conjunction with frequency) in determining whether a table answer is correct. For instance, for question (11-a) classified as function(minister,duits), there are several candidate answers, some of which are:

Answer	Freq	Answer	Freq
Klaus Kinkel	54	Volker Rühe	15
Theo Waigel	36	Günter Rexrodt	11

In this case, using frequency only to determine the correct answer, would give the wrong result, whereas a score that combines frequency and d-score (based on (12), on which one of the table entries was based) returns the correct answer.

(12) De Duitse minister van economische zaken, Günter Rexrodt, verwelkomde het rapport. (*The German minister of economy, Günter Rexrodt, welcomed the report.*)

5 Evaluation

We evaluated on 572 questions that were used in the Dutch monolingual task of the CLEF 2003 and 2004 QA track. We give the mean reciprocal rank (mrr) over the first 5 answers found by the system. The *off-line* column contains the results for questions answered by table look-up. The *on-line* column contains the score for the remaining questions, and the third column reports the overall score.

CLEF 03	off-line		on-line		total	
	# q	mrr	# q	mrr	# q	mrr
baseline	85	0.71	287	0.36	372	0.44
+d-score	85	0.71	287	0.40	372	0.47
+equiv	89	0.77	283	0.40	372	0.49

CLEF 04	off-line		on-line		total	
	# q	mrr	# q	mrr	# q	mrr
baseline	61	0.56	139	0.42	200	0.46
+d-score	61	0.57	139	0.45	200	0.48
+equiv	71	0.73	129	0.49	200	0.58

The baseline is a system which does not use `d-score` to rerank answers, and which does not use equivalences over dependency relations. It should be noted, however, that the baseline still makes use of dependency relations for question analysis and answer extraction. Adding `d-score` as a factor in reranking answers improves performance. Adding equivalences has a positive effect on the performance of the off-line method in particular. Since more tuples are extracted, the number of questions that can be answered by the off-line method increases. In addition, the accuracy of the overall system improves. The difference between the baseline system and the system using both `d-score` and equivalences is statistically significant (with 95% confidence) in both evaluations according to the paired t-test.

6 Conclusions and Future Work

We have shown that in-depth syntactic analysis in combination with a generic method for matching patterns against dependency relations and dealing with syntactic variation can be used to improve the performance of various components of a QA system. An advantage of this approach is that it opens up the possibility of using similar techniques for off-line and IR-based question answering. In future work, we would like to explore the possibility of integrating equivalence rules based on lexical equivalence (i.e. synonyms, term variants, and paraphrases acquired using the technique described in [Lin and Pantel, 2001]) and coreference.

References

[Attardi *et al.*, 2002] Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi, and Alessandro Tommasi. Piqasso: Pisa question answering system. In *Text REtrieval Conference (TREC) 2001 Proceedings*, pages 633–642, Gaithersburg, ML, 2002.

[Briscoe and Carroll, 2002] Ted Briscoe and John Carroll. Robust accurate statistical annotation of general text. In *Proceedings the third international conference on Language Resources and Evaluation (LREC 2002)*, pages 1499–1504, Gran Canaria, 2002.

[Clark and Curran, 2004] Stephen Clark and James R. Curran. Parsing the wsj using ccg and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain, 2004.

[Collins, 1999] Michael Collins. *Head-driven Statistical Models for Natural Language Processing*. PhD thesis, University of Pennsylvania, 1999.

[Fleischman *et al.*, 2003] Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proc. 41st Annual Meeting of the Association for Computational Linguistics*, pages 1–7, Sapporo, Japan, 2003.

[Jijkoun *et al.*, 2004] Valentin Jijkoun, Jori Mur, and Maarten de Rijke. Information extraction for question answering: Improving recall through syntactic patterns. In *Coling 2004*, pages 1284–1290, Geneva, 2004.

[Katz and Lin, 2003] Boris Katz and Jimmy Lin. Selectively using relations to improve precision in question answering. In *Proceedings of the workshop on Natural Language Processing for Question Answering (EACL 2003)*, pages 43–50, Budapest, 2003. EACL.

[Lin and Pantel, 2001] Dekan Lin and Patrick Pantel. Discovery of inference rules for question answering. *Natural Language Engineering*, 7:343–360, 2001.

[Lin, 1994] Dekan Lin. Principar—an efficient, broad-coverage, principle-based parser. in proceedings of coling-94. In *Proceedings of COLING-94*, pages pp.42–48, Kyoto, Japan., 1994.

[Litkowski, 2004] Kenneth C. Litkowski. Use of metadata for question answering and novelty tasks. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the eleventh Text Retrieval Conference (TREC 2003)*, pages 161–170, Gaithersburg, MD, 2004.

[Malouf and van Noord, 2004] Robert Malouf and Gertjan van Noord. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop Beyond Shallow Analyses - Formalisms and statistical modeling for deep analyses*, Hainan, 2004.

[Mollá and Gardiner, 2005] D. Mollá and M. Gardiner. Answerfinder - question answering by combining lexical, syntactic and semantic information. In *Australasian Language Technology Workshop (ALTW) 2004*, Sydney, 2005.

[Pollard and Sag, 1994] Carl Pollard and Ivan Sag. *Head-driven Phrase Structure Grammar*. Center for the Study of Language and Information Stanford, 1994.

[Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, and W. Yih. Mapping dependency trees: An application to question answering. In *The 8th International Symposium on Artificial Intelligence and Mathematics (AI&Math 04)*, Fort Lauderdale, FL, 2004.

[Rinaldi *et al.*, 2003] Fabio Rinaldi, James Dowdall, Kaarel Kaljurand, Micahel Hess, and Diego Mollá. Expliot-ing paraphrases in a question answering system. In *The Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications*, pages 25–32, Sapporo, Japan, 2003.

[van Noord, 2004] Gertjan van Noord. Error mining for wide-coverage grammar engineering. In *Proceedings of the ACL 2004*, Barcelona, 2004.

Towards Answering Procedural Questions

Farida Aouladomar
IRIT
118 route de Narbonne
31062 Toulouse Cedex France
aouladom@irit.fr

Abstract

In this paper, we first present an analysis of procedural question structure. Next, we investigate the structure of procedural texts and of the relevant rhetorical relations of interest for answering questions. We then show how, from a linguistic point of view, questions and procedural text fragments can match in order to produce responses.

1 Introduction

Procedural questions, sometimes called 'How-questions', are questions whose induced response is typically a fragment, more or less large, of a procedure, i.e., a set of coherent instructions designed to reach a goal. Procedural questions form a large subset of questions typically introduced by 'How'. Recent statistics elaborated from queries to Web search engines show that procedural questions is the second largest set of queries after factoid questions (de Rijke, 2005). This is confirmed by another detailed study carried out by (Yin, 2004). Procedural question-answering systems are of much interest both to the large-public via the Web, and to more technical staff, for example to query large textual databases dedicated to various types of procedures.

Procedural texts explain how to execute procedures. In our perspective, procedural texts range from apparently simple cooking receipes to large maintenance manuals (whose paper versions are measured in tons e.g. for aircraft maintenance). They also include documents as diverse as teaching texts, medical notices, social behavior recommendations, directions for use, assembly notices, do-it-yourself notices, itinerary guides, advice texts, savoir-faire guides etc. Procedural texts adhere more or less to a number of structural criteria, which may depend on the author's writing abilities and on traditions associated with a given domain. For example, (Schwitter et al., 2004) show that procedural knowledge in technical documents obeys in general to strict style guidelines. Typographical conventions such as the use of hyphens, bullets or other forms of numbering in front of each of the enumerated instructions are often imposed to writers via e.g. style files. The same is observed for a number of editing recommendations: language level, size of sentences, pronominal references, etc. There are also prototypical structures that depend on the domain (e.g. cooking receipes), but these are sometimes more conceptual than stylistic. In fact authors are not necessarily professionals: they may just be basic Web users that post their favorite cooking receipes.

Procedural texts explain how to realize a certain goal by means of actions which are at least partially temporally organized. Procedural texts can indeed be a simple, ordered list of instructions to reach a goal, but they can also be less linear, outlining different ways to realize something, with arguments, advices, conditions, hypothesis, preferences. They also often contain a number of recommendations, warnings, and comments of various sorts. The organization of a procedural text is in general made visible by means of linguistic and typographic marks. Another feature is that procedural texts tend to minimize the distance between language and action. Plans to realize a goal are made as immediate and explicit as necessary, the objective being to reduce the inferences that the user will have to make before acting. Texts are thus oriented towards action, they therefore combine instructions with icons, images, graphics, summaries, preventions, advices, etc.

Research on procedural texts was initiated by works in psychology, cognitive ergonomics, and didactics. Several facets, such as temporal and the argumentative structures have then been subject to general purpose investigations in linguistics, but they need to be customized to this type of text. There is however very little work done in Computational Linguistics circles.

Our work is primarily based on French. The typical French interrogative pronoun for procedural questions is *Comment ?*. This pronoun has a slightly narrower set of uses than How in English or Wie in German, it corresponds quite well to the Spanish *Cómo*. In a first stage, corpora observations are also mainly based on French, but we keep an eye on realizations in other languages. While most of the structures we have elaborated are aimed at being language independent, it is clear that each language has its own set of linguistic marks and possibly style for procedural texts. For example, (Delin et al, 94) identified different grammatical forms representing and expressing enablement and generation relations in French, English and Portuguese procedural texts, used to implement a system for generating multilingual instructions drafts for software use or for carrying out administrative procedures.

From a methodological point of view, our approach is based on (1) a conceptual and pragmatic analysis of the notion of procedure and (2) a mainly manual corpus-based analysis, whose aim is to validate and enrich the former. Procedural texts being quite complex, we feel that a symbolic perspective is appropriate to integrate corpus analysis as well as more abstract considerations (e.g. the temporal structure of instructions, the argumentative dimension, conditionals, etc.).

In this paper, we first briefly survey previous works on procedural texts and question answering. Then we propose a general typology of procedural questions in section 3. In section 4, we develop an analysis of procedural texts, from the point of view of their discursive structure and the rhetorical relations that hold between discursive elements. In section 5, We show and evaluate the adequacy of this analysis for answering How-questions. For that purpose, we introduce different notions, among which the notion of the $questionability$ of a text, i.e. its ability to be used to answer How-questions. Finally, in section 6, we sketch out a few procedures to retrieve relevant sets of instructions from How-questions .

2 State of the art

Procedural texts have been studied in psycholinguistic, linguistics and didactic circles. We briefly survey various approaches here, outlining elements of interest for our objectives.

2.1 General typology

Under the heading of procedural texts, there is a quite large diversity of texts. (Adam, 2001) notes the variability of judgements in procedural text categorization, depending on the text main objectives and style. We have, for example:

- regulatory texts (Mortara et al., 1988) that characterize expected behaviours,
- procedural texts (Longacre, 1982) defined as rather linear sets of instructions,
- 'programmatory' texts which include receipes, musical scores and architectory plans. (Greimas, 1983) identifies how knowledge from an expert is transfered via these texts to users who are expected to follow strictly the instructions given,
- instructional-prescriptive texts (Werlich, 1975), where a quite detailed analysis of temporal and event structures is carried out,
- injunctive texts, where (Adam, 1987) shows the form and style used in short notices for, e.g., fire instructions, security measures, etc.,
- advice texts (Luger, 1995), which include advice texts of various sorts, such as those found in large public magazines.
- receipe texts (Qamar, 1996), which is a domain quite well-studied, for example in language generation.

All these forms share common structures: specification of goals, description of lists of pre-requisites to reach goal, and description of sequences of instructions. They also share common stylistic forms, e.g. preferences for imperative forms, and a number of typographic elements such as enumerations.

From the analysis, mainly psychological or cognitive, of the different forms of procedural texts mentioned above, we classify them into three main categories, which will be considered in our project:

- Procedures, e.g.: receipes, maintenance and construction manuals, some medical texts, didactic texts, etc.
- Injunctions, e.g.: orders, regulations, game rules, security measures, etc.
- Advices, e.g.: beauty advices, ways : to fill in forms, to behave in certain environments, or to manage a meeting, etc.

2.2 Related works

Two works will be used as the starting point of the development of the discursive structure of procedural texts that we have elaborated, with in view to respond to How-questions. (Bieger et al., 1984-85) propose a taxonomy of the contents of instructions in 9 points: inventory (objects and concepts used), description (of objects and concepts), operational (information that suggest the agent how to realize an action), spatial (spatial data about the actions), contextual, covariance (of actions, which evolve in conjunction), temporal, qualificative (manners, limits of an information), emphatic (redirects attention to another action).

One of the main works in Computational Linguistics is due to (Kosseim, 1996). She isolated 9 main structures or operations, called *semantic elements* from corpus analysis:

1. sequential operations: a necessary action that the agent must realize,
2. object attribute: description meant to help understand the action to realize,
3. material conditions: environment in which an action must be carried out,
4. effects: consequences of the realization of a group of operations on the world,
5. influences: explain why and how and operation must be realized,
6. co-temporal operations: express operation synchronization,
7. options: optional operations,
8. preventions: describe actions to be avoided,
9. possible operations: possible operations to do in the future.

In a different range of ideas, and applied to Japanese, (Takechi and al., 2003), show that it is possible to isolate effective features used to categorize lists of Web pages as being procedural or not, in a QA perspective. They show that techniques other than in standard text categorization need to be developed. List of procedural expressions in the computer domain have been extracted with a quite high accuracy.

3 A typology of How-Questions

Let us now investigate the structure of procedural questions. Besides an introspective analysis, the work reported below is largely based on corpora studies. We considered in particular FAQ which abound in procedural questions, questions in the TREC and AnswerBus frameworks, and quite comprehensive inventories built from queries submitted to search engines over the past month(s) composed of keywords, found at: http://inventory.overture.com/d/searchinventory/suggestion/?mkt=fr. We constructed our procedural question corpora from different domains : health, education, tourism, social behavior (savoir-faire and others), computer sciences, maintenance domain.

In this section we first identify the nature of procedural questions, since they cannot just be identified by the interrogative pronoun *how* that introduces them, we then briefly present useful aspects of the syntactic structure of these questions followed by a semantic categorization of procedural questions, depending on the main verb of the query. We end the section with a semantic representation proposal, which will be the entry point of the response retrieval and construction procedure.

3.1 What is a procedural question ?

A large number of procedural questions are introduced by the interrogative pronoun *Comment* in French (How in English, Wie in German, Cómo in Spanish, etc.). However, $Comment$, similarly to How, has several uses which are not all related to procedural questions. Let us review them below in order to introduce a first set of restrictions in our analysis:

- The boolean How: as in *How are you, Cómo estas ? and Comment vas tu?* is not a procedural use. In the same range of ideas we have the evaluative How in Germanic languages: *How expensive is this?, wie teuer ist das?.*
- The nominal How: often included in a structure such as Comment + identification verb, as in: *comment surnomme-t-on le Mississippi?, comment dit-on maison en espagnol ?* (gloss: what is the other name for the Mississippi river? How do you say 'house' in Spanish?). This class does not characterize procedural questions, it is a kind of factoid question.
- The causality How: used to know the causes or the circumstances of a certain event: *How did John died?*, this use does not characterize a priori procedural questions when the response is just an NP. However, if elaborated, it can be viewed as being indirectly related to a kind of procedure, but we are closer here to narration than to procedural information.
- The instrumental or manner How: is used to learn about means or instruments as in *How is couscous eaten in Morocco?, response: by hand*. Once again, a procedural answer can be given to explain how to use fingers.
- The choice list How, responds to questions such as: *How can I pay my air ticket?*. The expected response is in general a list of choices: credit card, cash, etc. But it can also be associated with procedural elements: *cheque with a piece of ID for amounts less than 100 Euros*.
- The instructional How: which fully corresponds to procedural responses, characterized by an organized set of instructions designed to reach a goal: *how to change my car wheel?*.

Only the last type of How questions in this analysis is typical of procedural How questions. The instrumental, manner and choice list 'How-questions' may also contain some forms of procedures.

Procedural questions are often introduced by How, but there are several other forms that we survey below:

- Forms in 'Que + Faire' (gloss: what to do to...), as in *what should I do to get a visa for India ?* or forms with 'Quel + ETRE + proposition' (which/what + BE + prop) as in *what are the steps to follow to get a visa for India?* . We sometimes find directly constructions using the noun *procedure* to express procedural queries as in *What is the procedure for disconnecting my external hard drive?*.
- Elliptical use of How: abound in cases where just two or three keywords are used instead of a full, well-formed, natural language sentence: *changing wheel, add RAM, get boy-friend, assemble computer*. The verb or the deverbal used in the query allows for the identification of the type of question. More complex elliptical forms encountered in our corpora include e.g. the need of inference to identify a goal from a problematic situation as in: *PC down* which must be reformulated as *how to repair my PC?*
- questions in *is it possible to, can I, etc.* + VP, as in *is it possible to create a directory in Php ?*, have a direct response which is a priori just yes or no, but, in the case of a cooperative response, which is our perspective, the response is often a set of instructions, answering the question viewed as a goal to reach.

In our system, query processing is based on the QRISTAL system (developed by Synapse Toulouse: http://www.qristal.fr), which is not perfect, but which allows us to get the distinctions presented above and to construct an adequate representation.

3.2 A conceptual categorization of procedural questions

Let us now introduce a simple conceptual categorization for procedural questions. The four categories presented below cover about 90% of the cases found in our corpora. They correspond to different types of procedural texts. Our analysis is based on the main predicate (often a verb or a deverbal) in the query. It has been carried out with the use of the TROPES system (available at www.acetic.fr).

To have an analysis which is simple and easy to adapt to other languages, we have considered basic verb categories, as developed in WordNet (Fellbaum, 1998), and adapted to French in (Saint-Dizier, 1998). The categories considered are the following:

- action: this category is characterized by verbs of change, of creation and destruction, and maintenance, i.e. verbs such as e.g.: *construct, revise, mount, dismount, assemble, repair, change*, etc. In general, corresponding procedural texts include receipes, do-it-yourself guides, maintenance manuals, construction manuals, etc.

- communication: this class is characterized by verbs of social interactions and, possibly, some psychological verbs: *contact, convince, please, negotiate, manage,* etc., as in *how to manage a meeting ?*. Corresponding procedural texts include: practical advice notes, savoir-vivre, horoscopes, management guides, etc.
- knowledge acquisition: this class is characterized mainly by verbs of the cognition family and verbs that express forms of transfer of knowledge, such as: *know, learn, resolve, improve,* etc. Corresponding procedural texts include didactic texts, encyclopedia, etc.
- itinerary: this latter class is characterized by verbs of movement, among which: *go, reach, access,* etc. as in *How can I go to Toulouse Blagnac airport ?*.

3.3 Representing procedural questions

To represent procedural questions, given our cooperative answering framework (Benamara et al. 2004), we use a notation which is close to the form adopted in WebCoop, with the difference that the body of the query is not translated into first-order logic a priori, since this is not really useful at this stage for answering procedural questions. Furthermore, in our first set of experimentations, we only consider short procedural questions, i.e. questions essentially with one predicate. Our first objective is not to develop an analysis of complex procedural questions, which involve, for example, hypothesis, conditionals, but, rather, to study response retrieval and response generation.

The general representation format is the following:
`question(procedural(type), focus, constraints).`
where `type` is one of the four types presented in the section just above, `focus` is in general the VP: the predicate and its arguments, which does characterize the goal itself, and `constraints` is composed of restrictions which are realized as predicate adjuncts. This latter set may be empty. Focus and constraints are represented as parts of speech tagged words, using the TreeTagger (Schmid, Stuttgart University). To this tagging we add, when possible, the semantic type of the nouns so that the response search can be made more flexible (see section 5). Semantic typing is always a delicate task. We mainly consider here simple ontologies, such as those now available under Google.

As an example, the question: *how to reserve a flight on the Web?* is represented as follows, simplifying the annotation format: `question(procedural(action), [reserve(verb, morpho), flight(noun, transportation)], [on(prep,means), web(noun, communication)]).`

4 An analysis of the structure of procedural texts

In this section, we introduce our analysis of the discursive and rhetorical structures of procedural texts, with the view (in terms of topics and granularity) of answering how-questions.

Our analysis is based on the previous classifications presented in section 2. We feel however that these classifications are a little bit too vague or with a too heavy pragmatic perspective to be used directly to accurately respond to How-questions. There are also a few confusions between structural elements and rhetorical functions.

4.1 A Discursive analysis of procedural texts

Here is, represented by means of a grammar, the structure we have elaborated for procedural texts. The structures reported below correspond essentially to the organization of the informational contents. Elements concerning the layout (e.g. textual organizers such as: titles, enumerations, etc.), and linguistic marks of various sorts are used as triggers or delimiters to implement this grammar.

In what follows, parentheses express optionality, + iteration, / is an or, the comma is just a separator with no temporal connotation a priori, and the operator $<$ indicates a preferred precedence (i.e. the elements usually appear following the elements order given in the grammar nodes). Each symbol corresponds to an XML-tag, allowing us to annotate procedural texts,.

The top node is termed **objective**:
objective $\rightarrow$ title, (summary), (warning), (pre-requisites), (picture)+ $<$ instruction sequences.

summary $\rightarrow$ title+. Summary describes the global organisation of the procedure, it may be useful when procedures are complex (summary can be a set of hyper-links, often pointing to titles).

warning → text , (picture)+, (pre-requisites). Warnings represent global precautions or preventions associated with actions or objectives(e.g. switch off electricity prior to any action): they may have a complex rhetorical and modal structure, and should be studied in more depth. At the moment, we consider a warning just as a text, which sounds sufficient in most question-answering (QA) situations. Warnings are in fact of interest for answering Why? questions.

pre-requisites $\rightarrow$ list of objects, instruction sequences. Pre-requisites describe all kinds of equipments needed to realize the action (e.g. the different constituents of a receipe) and preparatory actions.

picture describes a sequence of charts and/or schemas of various sorts. They often interact with instructions by e.g. making them more clear. Analyzing this phenomena is outside the scope of this paper.

Instruction sequences is structured as follows:
instruction sequences $\rightarrow$ instseq $<$ discursive connectors $<$ instruction sequences / instseq.

instseq is then of one of four main types below:
instseq → (goal), imperative linear sequence / (goal), optional sequence / (goal), alternative sequence / (goal),

imperative co-temporal sequence.

Each type of instruction sequence is defined as follows:

imperative linear sequence → instruction < (temporal mark), imperative linear sequence/ instruction. (e.g. *cook peeled potatoes and reduce them out of mashed potatoes*) An imperative linear sequence is the kind of most common instruction sequence in procedural texts. It can be composed of one or several instructions.

optional sequence → conditional expression, imperative linear sequence. (e.g. *if you prefer a stronger flavor, add curry powder and cream.*)

alternative sequence → (conditional expression), imperative linear sequence, (alternative-opposition mark) < instseq / (conditional expression, instseq)+. (e.g. *peel potatoes, or leave the peel on if it is thin*).

imperative co-temporal sequence → imperative linear sequence < co-temporal mark < imperative co-temporal sequence / instruction.
A co-temporal sequence relates instructions which must be realized at the same time, or more generally non-sequentially (e.g. *mash tomatoes while mixing with garlic and olive oil*)

Finally, Instruction is the lowest level and has the following structure, with recursion on objective:

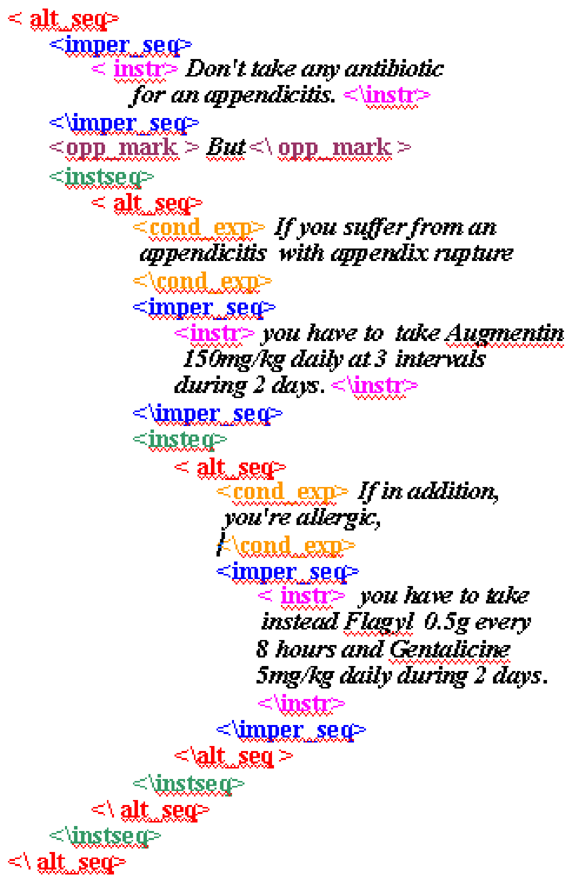
instruction → (iterative expression), action, (goal)+, (reference)+, (manner)+, (motivation), (limit), (picture)+, (warning) / objective.
Instructions can be complex since they may contain their own goals, warnings and pictures. If an instruction is complex it is analyzed as an objective.

At this level, it is most important to note reference phenomena of various sorts, pointing to already described instructions, to instructions to be described later (e.g. *see "preparation" instructions for ...*), or to external data via hyper-links. Let us also note that we have observed almost no restatements (other ways to describe an instruction if it is difficult to understand) and no summaries synthesising instructions (not to be confused with the generic summary of an objective, which is a kind of table of contents).

As an illustration, the following text, which contains embedded conditions, is analysed as an alternative sequence: *Ne prenez aucun antibiotique pour une appendicite. si vous souffrez d'une appendicite avec rupture de l'appendice, vous prendrez de l'Augmentin 150 mg / kg / jour en 3 prises pendant 2 jours. Si vous tes en plus allergique, vous prendrez la place du Flagyl 0.5g / 8 heures et de la Gentalicine 5mg / kg / jour pendant 2 jours.* See the annotation of the english translation of this exemple below.

4.2 Rhetorical structures

Rhetorical structures play several roles in our approach. They first give a semantics to the discursive structure syntax given above. They also contribute to enhancing the production of



```
< alt_seq>
    <imper_seq>
        < instr> Don't take any antibiotic
            for an appendicitis. <\instr>
    <\imper_seq>
    <opp_mark > But <\ opp_mark >
    <instseq>
        < alt_seq>
            <cond_exp> If you suffer from an
             appendicitis  with appendix rupture
            <\cond_exp>
            <imper_seq>
                <instr> you have to  take Augmentin
                 150mg/kg daily at 3  intervals
                during 2 days. <\instr>
            <\imper_seq>
            <insteq>
                < alt_seq>
                    <cond_exp> If in addition,
                     you're allergic,
                    <\cond_exp>
                    <imper_seq>
                        < instr>  you have to take
                         instead Flagyl  0.5g every
                        8 hours and Gentalicine
                        5mg/kg daily during 2 days.
                        <\instr>
                    <\imper_seq>
                <\alt_seq >
            <\instseq>
        <\ alt_seq>
    <\instseq>
<\ alt_seq>
```

Figure 1: An example of an annotated alternative sequence

well-designed responses (Kosseim, 1995). They are also useful, as shall be seen below, to allow for the integration of procedural texts dealing with similar objectives or goals, but this is an extremely difficult task. Finally, they are used to answer questions with a higher accuracy by clearly identifying e.g. instruments (for the instrumental how), risks (via the warnings) and equipment needed (via the prerequisites).

The RST (Mann et al., 1988) is a descriptive theory that specifies 23 possible relations showing how two portions of a text are linked. Previous work on procedural texts (Kosseim, 1995) (Vander Linden, 1993) (Rosner et al., 1992) used limited RST relations and suggested additional relations that fit procedural texts, which we use for our own analysis (limit, alternative, concurrence). We identified 17 relations from our corpora analysis, among which we introduced five new relations: reference, prevention, pre-requisite, option and co-occurrence. Here are the relations we use, with their definitions, possibly slightly altered from their original use:

- **Sequence**: is a multinucleic relations where Nucleuses are linked up by a succesion relation (*Disconnect the grinder, install the Standard Abrasives Quick Connect, 2-in. holder pad into the chuck and tighten the nut*).
- **Result**: specifies that an action cannot start before a de-

sired result from a previous action is reached (*once all the screw and the plugs positionned, assemble the prefabricated sections*).

- **Purpose**: occurs between a goal and the action meant to reach it (*clean the inside surfaces of the engine block to improve oil return*).
- **Evaluation**: is a little different from the purpose and the result relations because it is possible to evaluate whether the action was made correctly or not (*keep stirring in order to get an unctuous cream*).
- **Limit**: links up an action with a satellite representing a breakpoint (*reduce the sauce by stirring it until the liquid disappear*).
- **Alternative**: links 2 alternative actions, the choice can depend on the subject will or on the situation itself (*if it is screwed in place, remove the screws with a screwdriver, or If the panel hangs on hooks, pull the panel out and swing it up to remove it from the hooks*).
- **Means**: the nucleus (the action itself) is linked with the segments presenting manners of doing an action or instruments needed to realize the action (*you must reduce the air grinder's speed by using the regulator*).
- **Reference**: holds between an action and a segment which provides its procedure localization, in the text or in related texts via hyperlinks (*remove the reductor (see page 18)*).
- **Prerequisites**: occur between an action or an objective and a list of instruments or a set of actions without which the action or the objective cannot be realized (*changing a car wheel : to change a wheel is not difficult, with the proviso of having in one's car the good tools : wheel brace, jack, clean rag, torch (if dark), warning triangle*).
- **Option**: is considered when an action depends on the realization of a conditional situation. Notice that this relation can also link two sequential actions, where one is compulsory and the other depends on the subject will or on the situation (*steam the fish for 10 minutes and pass it 5 minutes in the oven if you want it to turn golden*).
- **Prevention**: is usually a relation between an action and its warnings. Satellites include expressions such as: be careful not to ..., and 'don't' expressions (*cut the wood planks, don't draw any line!*).
- **Condition**: appears when the action results from the occurrence of a conditioning situation (*if you can't do it by yoursefl, ask the joiner to cut the frames*).
- **Co-occurence**: is the word we use for Vander Linden's concurrence relation: where nuclei are linked by a co-temporal relation (*simultaneously to baking the meat, prepare the vegetables*).
- **Concurrence**: occurs between two rivals co-temporal actions (*to choose the best computer, run the program A on Mac, at the same time run the program B on PC. If Mac detects the component before the PC, then use Mac, otherwise use PC*).
- **Motivation**: occurs when the information given in the satellite intends to increase the readers desire to perform the action. Enablements are also part of this category (*you've almost come to the end, now you only have to wait till the flowerings*).

The following chart summarizes, for the rhetorical relations we use, the elements in our grammar which are involved.

Rhetorical relations	kernel-sattelite or multi-kernel pairs
Sequence	Instruction-imperative linear sequence Instruction sequence - instseq
Result	Goal-imperative linear sequence Goal-optional sequence Goal-alternative sequence Goal-imperative co-temporal sequence Instruction-imperative linear sequence Goal-action
Purpose	Imperative linear sequence - goal Optional sequence - goal Alternative sequence- goal Imperative co-temporal sequence - goal Action - goal-
Evaluation	Goal-imperative linear sequence Goal-optional sequence Goal-alternative sequence Goal-imperative co-temporal sequence Goal-action
Limit	Action - limit
Alternative	Imperative linear sequenc - instruction sequence Instruction sequence - instseq
Means	Action - manner
Reference	Action - reference
Prerequisites	Title-prerequisites
Option	Optional expression-imperative linear sequence Instruction sequence - instseq
Prevention	Title-warning Action - warning
Condition	Imperative linear sequence - optional expression Imperative linear sequence - conditional expression Instruction sequence - instseq
Concurrence	Imperative linear sequence - imperative co-temporal sequence
Co-occurrence	Imperative linear sequence - imperative co-temporal sequence
Motivation	Action - motivation

4.3 Linguistic marks

In this section, we briefly present the different types of marks that allow for the identification of the different elements presented in the grammar above. Besides conventional temporal, causal, conditional or argumentative marks, whose nature and scope are not always easy to interpret, procedural texts are particularly rich in easy to interpret typographic marks. This is particularly the case for those texts which are well-written, i.e. those we are interested in to construct responses.

Besides marks, we also present the main marks used to define the boundaries of each instruction or set of instructions. It is important to note that these texts originate from the Web. They have therefore a form proper to html coding, with some predefined know-how norms that most authors tend to use.

Discursive marks

Temporal marks include all the 'classical' marks in terms of precedence, overlap, inclusion, parallelism, etc. (Allen, 1984). They are mainly realized by means of adverbs, prepositions, conjunctions, aspectual verbs and propositions describing the realization of an event (*Once the four screws properly positioned, then assemble...*). We use the system developped by (Muller et al., 2004) to annotate temporal marks.

Causal marks are particularly rich and diverse. They are used to relate a goal to a set of instructions, or to specify within an instruction its aim; causal marks are also used to identify objectives, warnings and various forms of prevention, consequences and some forms of conclusions. They are mainly realized by means of causal verbs, prepositions and conjunctions. This is investigated in more depth in (Aouladomar et al., 2005), where a typology of natural arguments and their structure is presented for the different facets of procedural texts.

Besides temporal and causal connectors, we have identified five other types of connectors:

- restrictions and concessives: *assemble all the items except for the blue ones*,
- conditions, hypotheses: *if you do not wish to cut the frames yourself, ask....*,
- alternatives: *you can either use a hammer, or just push...*,
- consequences and conclusions: *paste the shelf so that it fits ...*,
- comparison or similitude: *use your pencil as a compass....*

These marks have been studied in various linguistic and conceptual frameworks. Marks proper to these connectors are often prepositions or semantically closely related to the semantic typology specific of prepositions. To identify and interpret them, we use the PrepNet framework (Saint-Dizier, 2005) (www.irit.fr/recherches/ILPL/prepnet.html).

Marks for instruction localization

Typographic criterion: the next point is to isolate basic instructions, called 'instructions' in the above grammar, and within these instructions the 'action' itself. The problem is essentially to identify simple marks which are delimiters of the beginning and the end of an instruction. Inter-instruction marks organize instructions. They are in general quite simple to identify in procedural texts (Takeshi et al, 2004). The beginning of an instruction is often the start of a sentence which can be introduced by various typographic marks proper to enumerations (intended lines, bullets etc.) (Luc et al, 1999). These correspond also in general to an instruction. The end of an instruction is either a punctuation mark, usually the dot, sometimes the semicolon or the comma, or typographic marks introducing the next instruction. The array below summarizes our observations on the use of layout to easily identify the instructions within a procedural text.

fig. 1 - (1) percentage of instructions or set of instructions introduced by typographic marks such as hyphens, bullets and other numbering forms, line breaks. (2) number of instructions considered in our sample.

Domains	(1)	(2)
maintenance, assembly	78%	279
receipes	89%	151
communication	63%	206
average	77%	636

Semantic criterion: within an instruction, the action is in general organized around the action verb and its arguments. Goals, references, manners, limits, are all adjuncts which appear in various orders. Goals contain specific verbs while manners are often nominal. Using the same corpora as above, we have the following verb distribution. It is elaborated with the TROPES software.

fig. 2 - (1) factive verb, (2) stative verb, (3) declarative verb, (4) performative verb.

Domains	(1)	(2)	(3)	(4)
maintenance/assembly	65%	23%	12%	
receipes	85%	13%	2%	
procedural QA pairs	67%	11%	22%	
communication	52%	26%	22%	
average	67%	18%	15%	
non-procedural texts	41%	35%	23%	1%

As can be noted, procedural texts have a much higher rate of factive verbs, and much less stative verbs. declarative verbs are about the same as in other types of texts. This verb type discrimination criterion is not precise enough for procedural text categorization, in particular to discriminate communication procedural texts from non-procedural texts.

Morphological criterion: another criterion is the morphology of the verbs encountered in procedural text. Instruction verbs are usually at the imperative, infinitive and or gerundive form. The more these forms are found in texts the more procedural these texts are. The table below presents the results obtained over a large list of verbs for procedural text and non-procedural texts.

fig. 3 - (1) number total of the verbs in texts, (2) number of verbs in the imperative/infinitive/gerundive forms, (3) percentage w.r.t. the total number of verbs.

Texts	(1)	(2)	(3)
maintenance/assembly	826	523	63%
receipes	434	386	89%
procedural QA pairs	495	244	49%
communication	315	156	49%
total average for procedural texts	2070	1309	63%
non-procedural texts	776	200	26%

Our goal is not just to use an instruction to respond to a How-question. It is often necessary to consider the level of the 'instseq' or higher, where quite generic goals, those frequently encountered in How-questions, are found. 'Instseq' are in general delimited by the expression of goals, which may have various forms, and by typographic elements (e.g. starting a new paragraph). Goals may be titles, well-identified from a typographic point of view, or they may have the form of a proposition introduced by a causal mark: *to clean the oil filter, ...*

The parser that recognizes the structure of procedural texts along the lines of the grammar given in the previous section has been implemented in Prolog for fast prototyping. So far it is quite simple, and still needs to be enriched in various marks and lexical data. Integrated into this parser, besides the closed-class elements (prepositions, connectors), we have included a typology of typographic marks, as found in html texts, a list of French verbs, classified along WordNet criteria (Saint-Dizier, 1998), a simple temporal grammar from an annotator, and a part of speech tagger. The parser runs bottom-up, using a shallow parsing strategy so that it can at least recognize parts of procedural texts. Its evaluation is still ongoing.

5 Questionability of a text

We use the term *questionability* (term due to J. Virbel, 2004) to express the ability or the relevance of any text, in our case found on the Web, to respond to How-questions. The primary goal is to have criteria to identify texts which are procedural among those obtained from a search engine. The second goal is to consider those procedural texts which are the most appropriate for responding to procedural questions. This means to be able to compare texts clearly identified as procedural texts, in terms of their level of detail, informativity, readability, conciseness, illustrations, number of links to other pages or to other parts in that same text, prevention on actions, etc.

The evaluation of the questionability of a text can be made a priori, independently from any particular query, or in relation with a query since some goals may be easier to identify than others in given texts. In this section, we establish a compromise between these two views which are of much interest. For the moment, a response to a how-question is found in a single text, using relevance, clarity, and informativity criteria. In a second stage, it would be of much interest to select a text depending on the user profile (casual user or professional) and to be able to merge or to integrate texts when they complement each other. These objectives are obviously very difficult to implement.

Let us now present the different criteria we consider to measure the questionability of a text. This measure is decomposed into two stages. The first stage aims at selecting those texts which should be a priori procedural texts. It is essentially based on surface marks to guarantee a certain efficiency. The second stage concentrates on the query, and introduces several relevance measures correlated with the query to answer. It is presented in section 6.

The first stage, called the "CATEG", selects the subset of texts returned by a search engine which should be procedural texts according to the three 'surface' criteria below; measures are all relative to text size:

- typographic forms (noted as TF) of various kinds that measure the architectural quality of the text. These forms include those given in Fig. 1 of section 4.2.
- morpho-syntactic marks, (noted as MSM): in procedural texts, we observed (cf. Fig. 3, in 4.2) that most verbs are either in the infinitive or in the imperative form, there are also marks that motivate the user to go further, such as *you must, you just have to,* followed by an action verb, or marks that indicate a task to realize: *the next stage, the next step, proceed as follows, care about, do not forget to, etc.* which abound in procedural texts,
- the presence of a large number of articulatory marks, (noted as AM): temporal, argumentative, causal marks to cite the most important ones (section 4.2).

Since it is quite difficult to assign relative weights to each of these three criteria, we consider they have an equivalent weight in the selection of procedural texts. Each counts for a third of the decision. Given a set of n texts, we evaluate for each text TF, MSM and AM. For example, TF_i is the ratio: number of typographic forms divided by the size of the text in number of words in the text i. Then, for each criterion, the average frequency is computed:
$TF_{average} = (\Sigma_{i=1,n} TF_i)/n,$
and similarly for the other two criteria. We can now define the CATEG for text i w.r.t. the set of texts considered:
$CATEG_i = TF_i/TF_{average} + MSM_i/MSM_{average} + AM_i/AM_{average}.$

The second stage, the "QUEST", investigates in more depth the questionability of the text. The objective is to evaluate the number of areas which can potentially match with How-questions. This is carried out by identifying those areas in the text on which the matching with questions should potentially be realized. Via our corpora analysis, it turns out that those areas are essentially:

- the number of titles identified, under objectives and in the summary (noted as TIT),
- the presence of a large number of action verbs (noted as AV), (cf. Fig. 2, in 4.2),
- the number of goals identified, (1) associated with instruction sequences or (2) within basic sequences, associated with the action to realize, (noted as GOA), and, finally
- manners found in instructions (noted as MAN).

Different linguistic marks allow for the identification of the goals and manner : the causal and manner connectors(e.g. in order to, so, by + gerundive verb, with, etc.).

Similarly as above, we can define the QUEST rate for a given text i in a collection of n texts. $QUEST_i = TIT_i/TIT_{average} + AV_i/AV_{average} + GOA_i/GOA_{average} + MAN_i/MAN_{average}.$

We can then compute an estimate of the overall questionability of a text i in a collection of n texts as follows:
$CATEG_{average} = (\Sigma_{i=1,n} CATEG_i)/n,$
$QUEST_{average} = (\Sigma_{i=1,n} NN_i)/n,$

$questionability_i = CATEG_i/CATEG_{average} + QUEST_i/QUEST_{average}$.

6 Responding to How-questions

The main aim of this project is to adequately and cooperatively respond to How-questions. An accurate and relevant analysis of the structure of How-questions, of procedural texts and of the notion of questionability establishes a basis for associating a query to a response which is as adequate as possible.

This task has several aspects, which entail investigations on the long term. We report here the main organization of our project and the results obtained so far. As advocated above, we introduced, in a first stage, a few restrictions: How-questions are short and relatively simple and a single procedural text, or a fragment of it, is selected as a response. We do not attempt, in this first stage, to merge texts, or to adapt the results to the user profile. However, for evaluation purposes, we list the four best candidates. We then make them accessible to the user, so that he/she can select the one he prefers.

Within our present perspective, responding to how-questions involves the following tasks:

- selecting the procedural texts which have the best questionability rate. Since, at this level, the matching with the query has not yet been done, we keep the 20 best texts based on the metrics given above,
- matching the question body with 'questionable zones' of procedural texts, hierarchically organized as: titles, goals, manners, and defining the best match,
- extracting the relevant portion of the text and returning it to the user in a user-friendly way.

The first step is realized as explained in the previous section. The second step, matching the question body with texts, is the most important part of the response construction. For that purpose, we follow the principles elaborated on in WebCoop (Benamara et al., 2004) concerning concept query relaxation. The levels of relaxation remain however quite limited because, for example, generalizations are in general not very relevant. In order to organize the matching procedure, we used the Qristal software, developed by Synapse. W.r.t. our aims, Qristal parses queries and introduces some flexibility in term matching. Terms derived from the query terms are specified in the response provided by this software. Results provided by Qristal are not very good by themselves, nevertheless, they provide us with a good basis to analyze the correspondences between the terms used in a query and those found in the response. We also use the quite detailed ontology provided in Qristal to find synonyms, parts, generic elements, and morphological variants, as described below.

Let us now report our experiments concerning the matching: query-procedural text. We carried out an experiment on a set of 80 questions from various domains. For each question, we examined the first 40 responses, identified a priori as the best ones by Qristal. Among these 40 responses, an average of 10 responses at least partly respond to the question and are potential candidates according to the criteria presented in the previous section. Considering just the matching procedure, results can be summarized as follows, we make a distinction between verbs, common nouns and proper nouns, prepositions are ignored:

- direct match, modulo flectional variants and abbreviations (for proper nouns): verbs: 42%, common nouns: 51%, proper nouns: 83%,
- match via synonyms and terms found via derivational morphology (e.g. deverbals): verbs: 37%, common nouns: 31%, proper nouns: 17%,
- match via part-of relation (wholes or parts): common nouns: 10%,
- match via a more generic term: verbs: 17%, nouns: 7%.

Although results returned by Qristal are not perfect, they nevertheless suggest the different steps of a matching strategy, which can be combined with the criteria we introduced in the previous section.

The main lines of the matching procedure is then as follows:

1. direct match (noted as DM) modulo flectional morphology: global weight of this criterion: 0.55 (weighted average of the figures given above),
2. match via synomyms or derivational morphology (noted as SDM): global weight: 0.30
3. match via part-of or more generic terms (noted as PGT), global weight: 0.15.

We can then define the questionability of a text i w.r.t. a precise query $Quest(Q)$. It is based on the number of terms of the question found in the titles, goals, action verbs and manners that match with the query. There may be overlaps between titles and goals for example, as in QUEST, but no discrimination is made at this level. Let $NW_{Q,i}$ be this total, it is defined as follows:
$NW_{Q,i} = 0.55 \times DM + 0.30 \times SDM + 0.15 \times PGT$.
Given the n texts (see section above), we can then define:
$NW_{Q,average} = (\Sigma_{i=1,n} NW_{Q,i})/n$.
The questionability rate of text i w.r.t. a precise query Q is then defined by:
$quest(Q,i) = NW_{Q,i}/NW_{Q,average}$.
This rate allows us to order texts w.r.t. to their questionability for a precise query Q, and then to select the most appropriate one, considering also CATEG, in particular for the quality of the response.

The last step consists in selecting the appropriate text fragment that responds the question. So far, our strategy is quite simple, and we have the following main situations:

- If the question matches with the title of the whole document, then the document is selected as a whole,
- If the question matches with the title or the goal of an instruction sequence, as defined in the grammar, then that whole sequence is selected. This is however a general rule which suffers some exceptions. In particular, for alternative sequences, it may be useful to select a larger fragment of the text.
- If the question matches with a goal within an instruction, then this instruction is returned to the user.

The adequacy of this rough strategy remains to be evaluated in depth. Since it is not easy to predict when a larger text fragment will be necessary, our strategy is to return a window that displays a priori the selected portion, however, the user can scroll it up or down to get a larger or a nearby text portion. Besides the response, in case of an indirect match (e.g. using more generic terms or synonyms), an explanation must be provided so that the user understand why he gets such as response. The explanation follows the template philosophy presented in WebCoop (Benamara et al., 04), outlining the terms that have been changed and why.

7 Perspectives

In this paper, we presented the general and rhetorical structure of procedural texts. We also investigated the structure of How-Questions, outlining those which really induce responses under the form of sets of instructions. We then showed how a procedural text can be characterized using relatively external and simple criteria. Finally, we briefly presented how to characterize the questionability of a text, and how the response retrieval mechanism can be constructed from this notion.

This work is still very experimental, it raises many questions. Our work needs to be deepened along many lines, including response accuracy (quality and scope), response generation, and the development of more elaborated evaluation methods, much more complex than e.g. the methods used in TREC for factoid questions. Procedural texts are also useful for answering Why questions (from goal and warning sections) that we intend to study.

References

[1] Adam, J.M., *Types de Textes ou genres de Discours ? Comment Classer les Textes qui Disent De et Comment Faire*, Langages, 141, pp. 10-27, 2001.

[2] Adam, J.M., *Types de Sequences Textuelles Elementaires*, Pratiques n56, Metz, 1987.

[3] Aouladomar, F., Saint-Dizier, P., *An Exploration of the Diversity of Natural Argumentation in Instructional Texts*, 5th International Workshop on Computational Models of Natural Argument, IJCAI, Edinburgh, 2005.

[4] Benamara, F., Saint-Dizier, P., *Advanced Relaxation for Cooperative Question Answering*, in: New Directions in Question Answering, in Mark T. Maybury, (ed), AAAI/MIT Press, 2004.

[5] Bieger, G.R., Glock, M.D., *The Information Content of Picture-text Instructions*, Journal of Experimental Education, 53, 68-76, 1984-85.

[6] Delin, J., Hartley, A., Paris, C., Scott, D., Vander Linden, K., *Expressing Procedural Relationships in Multilingual Instructions*, Proceedings of the Seventh International Workshop on Natural Language Generation, pp. 61-70, Maine, USA, 1994.

[7] Fellbaum, C., *WordNet An Electronic Lexical Database*, The MIT Press, 1998.

[8] Greimas, A., *La Soupe au Pistou ou la Conservation d'un Objet de Valeur*, in Du sens II, Seuil, Paris, 1983.

[9] Kosseim, L., Lapalme, G., *Choosing Rhetorical Structures to Plan Instructional Texts*, Computational Intelligence, Blackwell, Boston, 2000.

[10] Longacre, R., *Discourse Typology in Relation to Language Typology*, Sture Allen éd., Text Processing, Proceeding of Nobel Symposium 51, Stockholm, Almquist and Wiksell, 457-486, 1982.

[11] Luc, C., Mojahid, M., Virbel, J., Garcia-Debanc, C., Pery-Woodley, M-P., *A Linguistic Approach to Some Parameters of Layout: A study of enumerations*, In R. Power and D. Scott (Eds.), Using Layout for the Generation, Understanding or Retrieval of Documents, AAAI 1999 Fall Symposium, pp. 20-29, 1999.

[12] Luger, H.H., *Pressesprache*, Tubingen, Niemeyer, 1995.

[13] Mann, W., Thompson, S., *Rhetorical Structure Theory: Towards a Functional Theory of Text Organisation*, TEXT 8 (3) pp. 243-281, 1988.

[14] de Mattos Pimenta Parente, M-A., Steffen Holderbaum, C., Virbel J., Nespoulous, J-L., *Text Questionability as a predictor of story recall*, Thirteen Annual Meeting of the Society for Text Understanding, Madrid, 2003.

[15] Mortara Garavelli, B., *Tipologia dei Testi*, in G. Hodus et al.: lexicon der romanistischen Linguistik, vol. IV, Tubingen, Niemeyer, 1988.

[16] Muller,P., Tannier, X., *Annotating and Measuring Temporal Relations in Texts*, In Proceedings of Coling 2004, volume I, pages 50-56, Geneve, 2004.

[17] Qamar, H., *Quand Dire c'est: Ecrire-Comment-faire. Un Autre Type de Texte: le RECETTAL*, these soutenue l'Universite Lumiere, Lyon II, 1996.

[18] De Rijke, M., *Question Answering: What's Next?*, the Sixth International Workshop on Computational Semantics, Tilburg, 2005.

[19] Rosner, D., Stede, M., *Customizing RST for the Automatic Production of Technical Manuals*, in R. Dale, E. Hovy, D. Rosner and O. Stock eds., Aspects of Automated Natural Language Generation, Lecture Notes in Artificial Intelligence, pp. 199-214, Springler-Verlag, 1992.

[20] Saint-Dizier, P., *Verb Semantic Classes Based on 'Alternations' and WordNet-like criteria*, in : Predicative Forms in Natural language and lexical Knowledge Bases, Reds: Saint-Dizier,P., Eds: Kluwer Academic, Cambridge, USA, 1998.

[21] Saint-Dizier, P., *PrepNet: a Framework for Describing Prepositions: Preliminary Investigation Results*, the Sixth International Workshop on Computational Semantics, Tilburg, 2005.

[22] Schwitter, R., Rinaldi, F., Clematide, S., *The Importance Of How-Questions in Technical Domains*, TALN, Workshop Question-Reponse, Fes, Maroc, 2004.

[23] Takechi, M., Tokunaga, T., Matsumoto, Y., Tanaka, H., *Feature Selection in Categorizing Procedural Expressions*, The Sixth International Workshop on Information Retrieval with Asian Languages (IRAL2003), pp.49-56, 2003.

[24] Vander Linden, K., *Speaking of Actions Choosing Rhetorical Status and Grammatical Form in Instructional Text Generation* Thesis, University of Colorado, 1993.

[25] Werlich, E., *Typologie der Texte*, Heidelberg, Quelle and Meyer, 1975.

[26] Yin, L., *Topic Analysis and Answering Procedural Questions*, Information Technology Research Institute Technical Report Series, ITRI-04-14, University of Brighton, UK, 2004.

Towards a Framework for Collating Help-desk Responses from Multiple Documents

Yuval Marom and Ingrid Zukerman
School of Computer Science and Software Engineering,
Monash University
Clayton, VICTORIA 3800, AUSTRALIA
{yuvalm, ingrid}@csse.monash.edu.au

Abstract

Responses to help-desk email inquiries are often repetitive, sharing varying degrees of commonality. In addition, a significant proportion of the responses are generic, containing a very low level of technical content. In this paper, we present a corpus-based approach for identifying common elements in help-desk responses and using them to construct a new response. A help-desk domain is unique in that responses that contain even one incongruous sentence can alienate a user. It is therefore not always possible to automatically generate a complete response, because personalization is often better handled by human operators. Our system is designed to find and collate the generic portions of responses. We have adapted multi-document summarization techniques, and developed a measure that predicts the completeness of a planned response, thus indicating when a fully automated response is possible. Our evaluation shows that 14% of the responses in our corpus can be represented by complete generic responses.

1 Introduction

Email inquiries sent to help desks are often repetitive, and generally "revolve around a small set of common questions and issues".[1] This means that help-desk operators spend most of their time dealing with problems that have been previously addressed. Further, a significant proportion of help-desk responses contain a very low level of technical content, replying, for example, to inquiries about returning a product or questions addressed to the wrong group, or pointing out that the customer has provided insufficient detail about his/her problem. Organizations and clients would therefore benefit if an automated process was employed to deal with the easier problems, and the efforts of human operators were focused on difficult, atypical problems.

In this paper, we present an initial report of our corpus-based approach to achieving this objective with respect to email inquiries sent to Hewlett-Packard (HP). We are developing a system to automatically generate responses to users' requests on the basis of responses seen in a corpus of email dialogues. Since help-desk inquiries revolve around a small set of common problems, there is significant overlap in the content of the responses, but there are also differences arising from tailoring responses to particular user needs. For example, a response could consist of a generic description of how to install new printer drivers (the same for all printer models), preceded by a reference to a specific download location (different for each model). Also, a response in the corpus may refer to several, distinct problems raised by a user, where the answer to each problem appears repeatedly in the corpus, but the complete response does not.

In some cases, a new request might match a previous one very well, suggesting a traditional document retrieval approach, where a response document in the corpus can be re-used in its entirety. However, when a new request matches several previous requests whose responses have common but also different elements, document re-use is not appropriate. Instead, a response should be composed from parts of different responses.

This task is similar to query-relevant, multi-document summarization in the sense that different documents (i.e., candidate responses) must be combined to produce one response that is relevant to a user's request or interests. However, there is a significant difference between these tasks. Users of summarization systems will gloss over items of information that are not entirely relevant, whereas a help-desk response that contains even a single incongruous sentence will alienate the user. Therefore, the responses generated by our system must have very high relevance, even if this comes at the expense of completeness. If a complete response is not possible, it is more sensible to prompt a human operator to complete a partial response than to risk presenting incongruous information.

We postulate that there are two main types of information items that should be included in a response: generic, which are common to all (or most of) the responses that match a user's query, and specific, which address particular issues in the user's query. In the above example about drivers for printers, the general information about drivers would appear in most of the responses that match the given query, while the download information about a particular printer would appear only in specific replies. The approach proposed in this pa-

[1] `http://customercare.telephonyonline.com/ar/telecom_next_generation_customer/C.`

per identifies the former, i.e., information items that can be "safely" included in a reply. Further, we propose a measure to model our system's confidence in the completeness of a planned response composed of such safe elements. These are typically generic responses that have a low technical content.

The rest of this paper is organized as follows. In Section 2, we describe our corpus. Section 3 details our approach for generating the generic portions of help-desk responses and for assessing the completeness of planned responses. The evaluation of our approach is presented in Section 4. In Section 5, we discuss our first attempts at personalizing responses. Section 6 considers related work, followed by concluding remarks.

2 Corpus

Our corpus consists of 30000 email dialogues between users and help-desk operators at HP. These dialogues deal with a variety of user requests, which include requests for technical assistance, inquiries about products, and queries about how to return faulty products or parts. We have divided the corpus into topic-related datasets. For example, there is a "product replacement" (PRDRP) dataset with 1416 dialogues, and a "desktop" (DESKTOP) dataset with 590 dialogues. Further, we are focusing on 2-turn dialogues, as we are targeting user requests that can be dealt with using one response (about 80% of our corpus consists of 2-turn dialogues). Owing to time limitations, the procedures described in this paper were applied only to datasets comprising between 300 and 1500 (2-turn) dialogues, which corresponds to a total of 8000 dialogues.

Below are two responses from the PRDRP dataset.

R1: *I apologize for the delay in responding to your issue. Your request for a return airbill has been received and has been sent for processing. Your replacement airbill will be sent to you via email within 24 hours.*

R2: *I apologize for the delay in handling your issue. Your request for a return airbill has been received and has been sent for processing. Your replacement airbill will be sent to you via email within 24 hours.*

We can identify three functional "building blocks" for both responses: *apologize*, *confirm* and *inform*. Even though there is a minor difference between their opening sentences ("responding to" vs. "handling"), the responses are essentially identical. In contrast, response R3 below contains rather different building blocks, although it shares the first one with R1 and R2.

R3: *I apologize for the delay in responding to your issue. We are unable to send out replacement labels for return boxes. Please contact Technical Support at 1-800-OKCOMPAQ to have another box dispatched to you.*

In the next section we present our approach for representing and collating building blocks of responses.

3 Approach — Building blocks for responses

In order to find similarities between responses in our corpus, we believe it is necessary to represent building blocks at the sentence level. This is motivated by the characteristics of our task and domain: (1) the corpus contains repeated information at an intermediate level of abstraction (between entire responses and individual words), and (2) even a single incongruous sentence in a generated response could alienate a user. Further, the similarities between parts of responses should be abstracted from their exact wording, so that sentences that convey essentially the same meaning can be treated as the same building block. When doing so, care must be taken to ensure that we are able to confidently select representative sentences in the response generation stage.

We have developed a system that finds response building blocks by clustering sentences. We extract all the sentences from the responses in a particular dataset, and then cluster them into *Sentence Types (STs)*. This procedure should yield cohesive clusters for similar sentences, from which it is easy to select a representative sentence. In contrast, sentences that have differences would yield less cohesive clusters, thereby making it more difficult to select a representative sentence. This motivates a measure of cluster cohesion.

When generating a response to a new request, the system needs to know the building blocks for this response, i.e., the sentence types to use. In this paper we focus on the generation of the generic portion of responses. This generic portion may actually yield a complete response, such as R1 and R2, or a few sentences in a response, such as the first two sentences in R3. Our approach consists of three steps:

1. Finding *Response Types (RTs)*: groups of responses that share similar building blocks. Each group specifies the set of sentence types that the responses in the group agree on. We refer to this agreement as the *support* for the STs. In addition, we define a "semantic compactness" measure for the completeness of a response type, measured as the proportion of cohesive sentence types that are highly supported.

2. Producing a "model" response for each response type. This is achieved by selecting representative sentences from the cohesive sentence types that are supported by the group of responses.

3. Matching a new request with one or more response types. If only one response type is matched and it is considered complete by the semantic compactness measure, then its generated response can be sent directly to the user. Otherwise, a single response that is incomplete or several candidate responses (complete or otherwise) can be passed to an operator.

In this paper, we focus on the first two steps, i.e., constructing groups of responses, and generating a representative response for each group. In Section 5, we report on results of preliminary experiments involving the third step.

In the remainder of this section we give more detail on the different parts of the system: (1) identifying sentences types; (2) clustering responses according to the sentence types they contain; (3) calculating the "semantic compactness" of the response clusters; and (4) selecting sentences for inclusion in a response (Figure 1 illustrates Steps 1, 2 and 4).

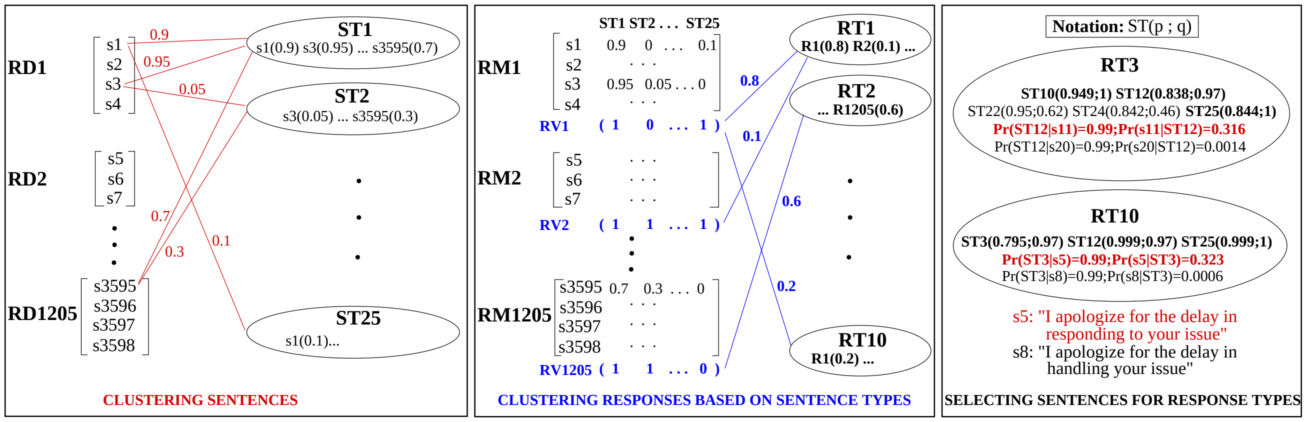

RD1
s1
s2
s3
s4
RD2
s5
s6
s7
RD1205
s3595
s3596
s3597
s3598
0.9
0.95
0.05
0.7
0.3
0.1
ST1
s1(0.9) s3(0.95) ... s3595(0.7)
ST2
s3(0.05) ... s3595(0.3)
ST25
s1(0.1)...
CLUSTERING SENTENCES
ST1 ST2 . . . ST25
RM1
s1 0.9 0 . . . 0.1
s2 . . .
s3 0.95 0.05 . . . 0
s4 . . .
RV1 (1 0 . . . 1)
RM2
s5 . . .
s6 . . .
s7 . . .
RV2 (1 1 . . . 1)
RM1205
s3595 0.7 0.3 . . . 0
s3596 . . .
s3597 . . .
s3598 . . .
RV1205 (1 1 . . . 0)
0.8
0.1
0.6
0.2
RT1
R1(0.8) R2(0.1) ...
RT2
... R1205(0.6)
RT10
R1(0.2) ...
CLUSTERING RESPONSES BASED ON SENTENCE TYPES
Notation: ST(p ; q)
RT3
ST10(0.949;1) ST12(0.838;0.97)
ST22(0.95;0.62) ST24(0.842;0.46) ST25(0.844;1)
Pr(ST12|s11)=0.99;Pr(s11|ST12)=0.316
Pr(ST12|s20)=0.99;Pr(s20|ST12)=0.0014
RT10
ST3(0.795;0.97) ST12(0.999;0.97) ST25(0.999;1)
Pr(ST3|s5)=0.99;Pr(s5|ST3)=0.323
Pr(ST3|s8)=0.99;Pr(s8|ST3)=0.0006
s5: "I apologize for the delay in responding to your issue"
s8: "I apologize for the delay in handling your issue"
SELECTING SENTENCES FOR RESPONSE TYPES

Figure 1: Sentence and response clustering, and response generation, for the PRDRP dataset.

3.1 Identifying sentence types

The representation used for clustering sentences is crucial, because it determines what constitutes similarity. In R1 and R2 the minor difference between 'handling' and 'responding' is inconsequential. However, if the last sentence of R1 were *"Your replacement airbill will be sent to you via email within **48** hours"*, the difference would be important (24 hours versus 48).

Our current implementation uses a bag-of-words approach with binary values. That is, each sentence is represented by means of a binary vector of size N (number of feature words in the dataset), where element j is 1 if (lemmatized) word w_j is present in the sentence, and 0 otherwise. Although this representation treats both of the above examples in the same way, we have found it a useful starting point. In the future, we plan to investigate context-dependent representations, such as that proposed by Lin [1998], and automatic tagging of certain types of words, such as numbers and case identifiers.

Once all the sentences have been extracted from the responses in a dataset, they are passed on to SNOB, a clustering program based on the Minimum Message Length principle [Wallace and Boulton, 1968]. SNOB yields m sentence types, where m varies for each dataset. For example, the PRDRP dataset has 25 sentence types, and the DESKTOP dataset has 40. Each sentence type ST_i, $i = 1, \dots, m$, is represented by means of a centroid *CST*$_i$ — an N-dimensional vector, such that *CST*$_i[j] = \Pr(w_j \in ST_i)$ is the probability that word w_j is used in ST_i. The left panel of Figure 1 illustrates the sentence-clustering process for the PRDRP dataset, which contains 1205 response documents (RD1, . . ., RD1205) comprising a total of 3598 sentences. As seen in this example, a sentence may probabilistically belong to more than one sentence type, e.g., $\Pr(ST1|s1) = 0.9$ and $\Pr(ST25|s1) = 0.1$ (these probabilities are provided by SNOB).

3.2 Clustering responses

We apply SNOB again to cluster responses into response types, but first we perform the following steps to represent responses by means of sentence types (these steps are illustrated in the middle panel of Figure 1 for the PRDRP dataset).

Representing sentences in terms of sentence types. We represent each sentence s_j by means of an m-dimensional vector, where m is the number of sentence types. Element i in the vector for sentence s_j contains $\Pr(ST_i|s_j)$ (the probability that s_j belongs to sentence type ST_i). We then combine the vector for each sentence in response R_k into a *Response Matrix* RM_k of size $n_k \times m$, where n_k is the number of sentences in R_k. For instance, as seen in the middle panel of Figure 1, RM1, the response matrix for response R1, comprises the vectors for sentences s1, s2, s3 and s4; the vector for s1 indicates that $\Pr(ST1|s1) = 0.9$, $\Pr(ST25|s1) = 0.1$ and $\Pr(ST_j|s1) = 0$ for $j = 2, \dots, 24$ (these probabilities sum to 1).

Representing responses in terms of sentence types. For each response matrix RM_k, we derive an m-dimensional *Response Vector* RV_k, such that for $i = 1, \dots, m$

$$RV_k[i] = \begin{cases} 1 & \text{if } \exists RM_k[j,i] \geq 0.1 \text{ for } j = 1, \dots, n_k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

That is, $RV_k[i] = 1$ indicates that sentence type ST_i has some presence in response R_k (with probability ≥ 0.1). We use a binary representation for the RVs because (1) it provides a reasonable first baseline for our system, and (2) the centroids of the resultant clusters have clear probabilistic semantics. In the future, we intend to investigate real-valued representations, e.g., set $RV_k[i]$ to be the maximum of the $RM_k[j,i]$ for all j.

Clustering. The response vectors are given to SNOB, which clusters them into response types. The number of response types varies for different datasets. For instance, PRDRP and DESKTOP have 10 and 9 response types respectively. Each response type RT_l is represented by means of a centroid *CRT*$_l$ — an m-dimensional vector, such that *CRT*$_l[i] = \Pr(ST_i \in RT_l)$ is the probability that sentence type ST_i is used in response type RT_l.

3.3 Calculating semantic compactness

The response clustering process yields a centroid for each response type. The centroid of a response type can be interpreted as a set of probabilities, $\mathbf{p} = \{p_1, p_2, \dots, p_m\}$, where

p_i represents the support in the response type for sentence type ST_i. In other words, p_i corresponds to the proportion of responses in the response type that use ST_i.

The *Semantic Compactness* (*SemCom*) measure predicts whether a complete, high-precision response can be generated for a response type. This measure is based on the support and cohesion of each sentence type. The former can be obtained from the support probabilities $\mathbf{p}$, while for the latter we define the following measure of cohesion $\mathbf{q} = \{q_1, q_2, \ldots, q_m\}$, where q_i for sentence type ST_i is defined as follows.

$$q_i = \frac{1}{N} \sum_{j=1}^{N} \delta(\mathit{CST}_i[j] \leq \alpha \vee \mathit{CST}_i[j] \geq 1 - \alpha) \qquad (2)$$

where N is the number of feature words in the data set, α is an empirically determined threshold, and δ is the boolean function

$$\delta(A) = \begin{cases} 1 & \text{if event A is true} \\ 0 & \text{otherwise} \end{cases}$$

By choosing a value for α close to zero, Equation 2 specifies that a sentence type is cohesive if there is a high proportion of words that are almost certainly present or almost certainly absent from this sentence type.[2] The rationale for this measure is that a cohesive group of sentences should agree strongly on the words they use and the words they omit. Hence, it is possible to obtain a sentence that adequately represents a cohesive sentence type, while this is not the case for a loose sentence type.

For example, the opening sentences in R1 and R2 belong to a sentence type which is a cluster consisting of 810 identical repetitions of the sentence from R1, and 15 identical repetitions of the sentence from R2. The cohesion of this sentence type is 0.97. An example of a non-cohesive sentence type is one which consists of sentences about part numbers and order numbers, such as *"Your part has been received"*, *"Please verify that this is the correct order number"*, *"No part return is required on this order"*, and *"The case number provided is not coming up in our system"*. These sentences share a few words (mainly 'part', 'order' and 'number'), but do not discuss anything specific about these words — we could not confidently select a representative sentence from this sentence type. The cohesion of this sentence type is 0.65.

The semantic compactness of a response type with support $\mathbf{p}$ and cohesion $\mathbf{q}$ is calculated as follows.

$$\mathit{SemCom}(\mathbf{p}, \mathbf{q}) = \frac{\sum_{i=1}^{m} \delta(p_i \geq \mathcal{T}_{\mathit{Hi}} \ \wedge \ q_i \geq \mathcal{T}_{\mathit{Coh}})}{\sum_{i=1}^{m} \delta(p_i \geq 0.1)} \qquad (3)$$

where $\mathcal{T}_{\mathit{Hi}}$ and $\mathcal{T}_{\mathit{Coh}}$ are empirically determined thresholds. *SemCom* measures the proportion of highly supported and cohesive sentence types among the sentence types that have some support (hence the low threshold of 0.1 in the denominator). If this proportion is high, the proposed response is deemed semantically compact, which means that it is a good candidate for automatic response generation. As the value of this proportion decreases, so does the confidence of automatically generating a complete response. During response generation, our system compares the semantic compactness of a proposed response with an empirically determined threshold, in order to determine whether a human operator should participate in the composition of a reply.

In Section 4.2 we evaluate the semantic compactness measure, and suggest a value for its threshold. Further, the results we report in that section were obtained with rather stringent thresholds ($\mathcal{T}_{\mathit{Hi}} = 0.75$, $\mathcal{T}_{\mathit{Coh}} = 0.9$ and $\alpha = 0.01$), in order to implement a cautious approach that avoids including potentially incongruous sentences in automatically generated responses. However, our sensitivity analysis shows that the quality of our responses is largely maintained even if we relax some of these thresholds [Marom and Zukerman, 2005].

3.4 Selecting sentences for inclusion in a response

Sentences can be (probabilistically) associated with multiple sentence types, so we need a method of selecting the most representative sentences to include in a response, while avoiding repetition. Filatova and Hatzivassiloglou [2004] address this problem, and we have implemented a modified version of their *adaptive greedy algorithm* for scoring sentences. Our system selects the most representative sentences from the most supported and cohesive sentence types.

The clustering program SNOB provides $\Pr(s_j|ST_i)$, the probability of a sentence s_j given a sentence type ST_i, which we can use as an indication of how representative this sentence is of the sentence type.[3] For example, when selecting a sentence from the sentence type corresponding to the opening sentences in R1 and R2, the sentence in R1 has a higher probability because it appears more frequently in that cluster.

The score of each sentence is calculated as follows.

$$\mathit{Score}(s_j) = \sum_{i=1}^{m} \Pr(s_j|ST_i) \times p_i \times \delta(q_i \geq \mathcal{T}_{\mathit{Coh}}) \qquad (4)$$

The last factor ensures that only cohesive sentence types contribute to the score of a sentence, thus safeguarding against potentially incongruous sentences.

Following the adaptive greedy algorithm, the system scores and sorts all the sentences, and then identifies which sentences represent sentence types that are already represented by higher-scoring sentences. The score of the lower-scoring sentences is then reduced by the contribution from the shared sentence type, thus reducing the chances of repetition in the response. We include in the response only sentences whose score is greater than zero. It is worth noting that our system does not currently address the order of sentence presentation, and therefore our evaluation is based purely on the content of a response.

The right panel of Figure 1 illustrates the generation of a response for two response types, RT3 and RT10 (for example, in RT3, the notation ST12(0.838;0.97) means that $p_{12} = \Pr(ST12 \in RT3) = 0.838$ and $q_{12} = 0.97$). RT3 contains five sentence types with a high value for

[2]This measure is a simplifi cation of entropy, in the sense that it yields non-zero values for extreme probabilities.

[3]Note that $\Pr(s_j|ST_i)$ is different from the posterior $\Pr(ST_i|s_j)$, which can be interpreted as the probability that s_j belongs to sentence type ST_i (shown in the left panel of Figure 1).

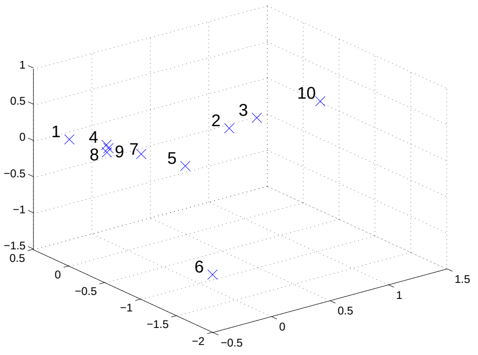


Figure 2: Response types for the PRDRP dataset.

$\text{Pr}(ST_i \in RT3)$, but only three of them have high cohesion (ST10, ST12 and ST25). Hence, we only include sentences from these sentence types in the response generated from RT3. In contrast, all the sentence types in RT10 have a high value for $\text{Pr}(ST_i \in RT10)$ and high cohesion — this response type has a semantic compactness of 1.0. Hence, a complete summary can be generated from these sentence types. As seen in Equation 4, since the ps and qs are high for all the sentence types in RT10, the selection of a representative sentence for these sentence types depends mainly on $\text{Pr}(s_j|ST_i)$. In this example, $s5$ is selected to represent ST3, as $\text{Pr}(s5|ST3)$ is much higher than $\text{Pr}(s8|ST3)$. The resultant generated response is identical to response R1.

4 Evaluation

In this section, we first illustrate the output produced by our system for two datasets (including the previously discussed PRDRP dataset). We then examine the predictive performance of our *SemCom* measure, and finally make some overall observations about the coverage and completeness of model responses.

4.1 Sample datasets

The PRDRP dataset

Figure 2 shows a 3D projection of the centroids of the response types discovered for the PRDRP dataset (the response types are represented by their numbers).[4] The figure shows (to some extent) how different the RTs are from each other. Response Type 10, discussed above, is the most significant one as it accounts for 862 responses (71%). According to Figure 2, the RTs most different from RT10 are RT1 and RT6. Both RTs have a perfect semantic compactness, they account for 72 and 49 responses respectively, and their generated responses are R4 and R5, respectively.

R4: *Your request for a return airbill has been forwarded to the proper group. You will receive your replacement airbill within 24 hours.*

[4]This figure is generated using Principal Component Analysis (PCA); it is a projection of the 25-dimensional centroid values onto the first three principal components discovered by PCA, which correspond to the axes in the figure. These components account for approximately 70% of the variability in the data (measured as the relative contribution of the first three eigenvalues of the covariance matrix).

R5: *I apologize for the delay in responding to your issue. Your request for a return airbill has been received. Additional information is needed to process your request. Please provide a case number or an order number so that we may send you a replacement label.*

These three response types together account for about 81.5% of the responses in the dataset. That is, the system is able to find three different kinds of generic responses, and confidently generate complete automatic responses that account for 81.5% of the actual responses.

The response-clustering procedure also finds groups of responses that cannot be replaced by model responses. For example, RT9 accounts for 55 responses, but has a semantic compactness of 0.0. This means that the responses in this cluster strongly disagree about which sentence types to use, or that the sentence types that they agree on are non-cohesive. Thus RT9 does not generate any sentences. An example of a response in this response type is R6, a fairly specific, personalized response.

R6: *It is usually required that the case number and order number are provided, however in your circumstances we will see what we can do to help. What other information can you provide? Do you have the serial number of the unit?*

The TAPEDRV dataset

Our findings show that different datasets have different proportions of responses that can be replaced by generic responses. The PRDRP dataset has the highest potential for the generation of a complete response, while only 35% of the responses in the TAPEDRV dataset (concerning tape drives) can be generated by our system in their entirety. However, our system can also generate partial responses from a response type. These responses contain sentences extracted from a subset of highly cohesive sentence types in the response type. An example of a partial response for the TAPEDRV dataset is R7.

R7: *Thank you for contacting HP's Customer Care Technical Center. We are only able to assist customers with in warranty products through our email services. At the present time, we have the following numbers to contact technical support for your out of warranty product.*

The response type that generated this response accounts for 19 responses, and has a semantic compactness of 0.25, which means that, on average, the generated response covers only a quarter of the actual responses. Examples of sentences that appear in the actual responses that make up this response type are: *"As your product is out of warranty, you can visit the link given below for complete details regarding the COLORADO 8GB TRAVAN tape drive: URL"*, and *"The 5 BG Internal travan drives should be recognized and configured automatically by Windows XP using the native QIC157 driver"*. The additional information in these sentences is very specific and quite unique, which accounts for the low semantic compactness of the response type. The part of the response generated by our system can be regarded as a generic opening segment, with the remainder of the response to be completed by an operator.

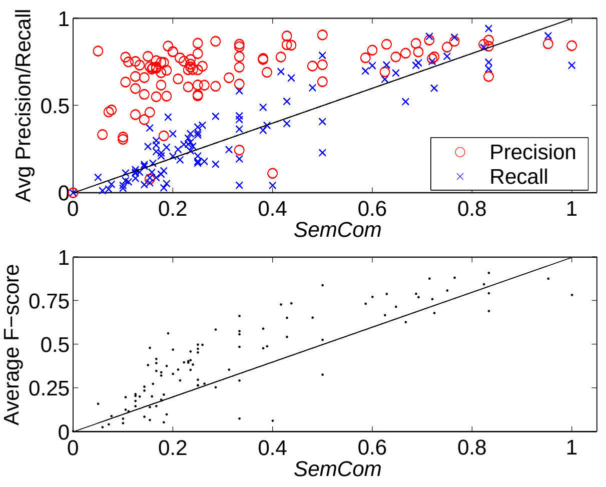


Figure 3: Relationship between semantic compactness and precision and recall (top plot), and F-score (bottom plot).

4.2 Predictive performance of *SemCom*

Our *SemCom* measure is designed to predict the completeness of an automatically-generated response composed of high-precision sentences. In order to determine the utility of this measure, we examine how well it correlates with the quality of the generated responses.

We assess the quality of a generated response r_g by comparing it with the actual responses in the response type from which r_g was sourced. To this effect, we use three well-known information retrieval measures: precision, recall and F-score [Salton and McGill, 1983]. Precision gives the proportion of words in r_g that match those in an actual response; recall gives the proportion of words in the actual response that are included in r_g; and F-score is the harmonic mean of precision and recall. Precision, recall and F-score are then averaged over the responses in r_g's response type to give an overall evaluation of r_g.[5] For example, RTs 1, 6 and 10 have respective average F-scores of 0.78, 0.82 and 0.80.

Figure 3 shows the relationship between semantic compactness and precision, recall and F-score for the 135 response types created for the different datasets. From the figure we see that precision is generally high, and is uncorrelated with *SemCom*. This is not surprising, as the sentence-selection process is designed to select high-precision sentences. Hence, so long as at least one sentence is selected, the text in the generated response r_g will agree with the text in the responses that are represented in r_g's response type. In contrast, recall is highly correlated with *SemCom*. A decrease in *SemCom* indicates that fewer sentences are included in the generated response, which therefore covers less of the information in the original responses. As expected from these results, the overall F-score is also highly correlated with semantic compactness. The linear and log correlations between the semantic compactness measure and F-score are 0.89 and 0.9 respectively, which demonstrate the high predictive power of the *SemCom* measure.

However, for the predictions made by *SemCom* to be useful, they must also agree with users' views. To test whether this is the case, we conducted a small, preliminary study as follows. We constructed four evaluation sets by selecting four response types with high semantic compactness (≥ 0.7), automatically generating a response from each response type, and selecting 15 actual responses from each response type for comparison.[6] Each evaluation set was given to two judges, who were asked to rate the precision and completeness of the generated response compared to each of the 15 responses in the set. Our judges gave all the automatically generated responses high precision ratings, and completeness ratings which were consistent with our semantic compactness measure.

4.3 Overall observations

Once we were confident that our semantic compactness measure can reliably predict the completeness of a generated response, we were interested to get an overall impression of the proportion of generic responses in our corpus. That is, we wanted to find out what proportion of responses could be represented by complete or partial model responses. For example, we saw in Section 4.1 that 81.5% of the responses in the PRDRP dataset can be represented by complete model responses.

Model responses are complete if their response types have a high semantic compactness. Figure 3 suggests that *SemCom* > 0.7 results in a high F-Score. If we consider this threshold to indicate a complete response, the response types that exceed it account for approximately 14% of the actual responses in the various datasets. If we consider a threshold of 0.4 to indicate a medium semantic compactness, then the additional response types that exceed it account for a further 6% of the responses — these response types would produce partial responses. The remaining 80% of the responses would have to be mostly written by an operator. However, this may be a pessimistic estimate, as some response types with a low *SemCom* yield reasonable partial responses, such as R7 whose response type has a *SemCom* of 0.25 (Section 4.1).

5 Tailoring responses to users' requests — preliminary trials

In order to enable the system to respond flexibly to specific user requests, user-driven summarization must be implemented. To this effect, we are considering two approaches: (1) *predicting response types* from request features in order to test whether model responses can be used to address some requests; and (2) *predicting sentence types* from request features in order to construct a response from parts of multiple responses in the corpus. Here we report on our preliminary trials for the first option, and present our ideas for the second option.

[5] In addition to these measures, which are calculated on a word-by-word basis, we experimented with the ROUGE evaluation procedure, which also takes into account word sequences [Lin and Hovy, 2003]. The simpler word-by-word evaluation correlated well with ROUGE, hence we only report on the former.

[6] Several of our automatically-generated responses match perfectly the operators' responses. Since these are obvious matches, they were not included in our study.

5.1 Predicting response types

We trained a Decision Graph [Oliver, 1993] (an extension of the decision trees described in [Wallace and Patrick, 1993]) to predict which RT is most appropriate for a given user request. We extracted various features as input to the Decision Graph, e.g., word unigrams with TF.IDF weights; word bigrams with binary weights; noun, verb, adjective and adverb frequencies; and email length (number of sentences). The features that turned out to be most significant were the word unigrams and bigrams.

For example, in one dataset the Decision Graph contained a split on the word 'xp', which differentiated two response types. These response types were very similar, both requesting more information from the user, and providing contact numbers for out-of-warranty products. The main difference between them was the sentence *"However, the software support for the Windows XP platform will be offered through a solution supplied by VERITAS. Please visit the link given below in order to get in touch with VERITAS"*. In a different dataset, the responses were so varied that for most of them the system could only generate the sentence *"Thank you, HP eServices"*. However, the Decision Graph predicted that if the words 'cp-2e' or 'cp-2w' (referring to specific router models) were present in the request, then a response type with very high semantic compactness could be used, resulting in response R8.

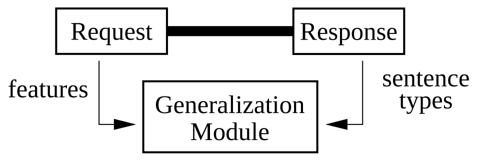
R8: *Based on the serial number or other information you've provided, your system is a consumer (home) product that is supported by a different group. The Consumer Product Support Group has been trained in the support of the Presario product line and they are best equipped to answer your questions. Please resubmit your question at URL.*

If the Decision Graph predicts a single response type with a high semantic compactness, then a fully automated response is possible. If, however, the response type is not complete, or more than one response type have been matched, then a human operator is presented with the response(s). This was generally the case in the trials we conducted. We built Decision Graphs for five datasets that had at least one response type with a high semantic compactness. The resultant graphs had between three and five leaves, most of which pointed to 2-3 response types with varying degrees of certainty.

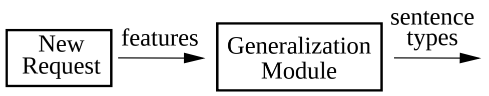
5.2 Predicting sentence types

Our results indicate that only a small proportion of requests can be addressed with complete model responses. We are currently investigating an alternative approach whereby we map features in users' requests directly to sentence types, and then compose a reply from sentence types. We believe that this level of granularity will enable our system to exhibit the functionality required for flexibly addressing user requests.

The general framework is shown in Figure 4, where a generalization module is used to learn mappings from features in users' requests to sentence types in the responses to these requests. The former would be similar to the features extracted for the Decision Graph approach, while an example of the latter is a binary pattern similar to the response vector RV defined in Equation 1 (also shown in the middle panel of Figure 1). The generalization module is essentially a supervised



(a) learning mappings between request features and sentence types, using all request-response pairs in the corpus



(b) predicting a set of sentence types for a new user request

Figure 4: Framework for a sentence-type based approach.

learning system that is trained on request-response pairs in the corpus (Figure 4(a)). The trained system is then used to predict which sentence types to use in a response to a new user request (Figure 4(b)). These predictions can be interpreted as the support for the sentence types given the request features. The procedure described in Section 3 can then be applied: the *SemCom* measure can be used to assess the completeness of a response comprising highly supported sentence types (Equation 3), and actual sentences can be scored and selected using Equation 4.

For example, we envisage that a feature corresponding to the time since the arrival of the user's request can predict the usage of the sentence type corresponding to the opening sentences in R1 and R2 (apologizing for the delay in the response). In another example, we envisage that the unigram feature 'xp' would predict the usage of two sentence types corresponding to the sentences shown in the example in Section 5.1 (providing a specific address for XP users).

6 Related Research

The idea of clustering text and then generating a summary from the clusters has been previously implemented in multi-document summarization systems [Radev *et al.*, 2000; Hatzivassiloglou *et al.*, 2001; Filatova and Hatzivassiloglou, 2004]. A key issue highlighted in such work is the choice of features used in the clustering. Radev *et al.* used low-level word-based features [Radev *et al.*, 2000], while Hatzivassiloglou and colleagues used higher-level, grammatical features obtained through part-of-speech tagging [Hatzivassiloglou *et al.*, 2001; Filatova and Hatzivassiloglou, 2004]. Our work differs from previous work on clustering and summarization in three respects. Firstly, the high-level features (sentence types) we use to cluster documents are learned from the corpus in an unsupervised manner, using as input only low-level, word-based features. Secondly, our reliance on sentence types enables us to identify response patterns beyond those identified by topic words, and hence allows us to generate multiple summaries within a single topic. Thirdly, our system avoids the inclusion of incongruous sentences, a restriction that is not traditionally addressed in multi-document summarization.

The completion of the user-driven implementation of our system will enable comparisons with other user-driven approaches, in particular those involving information retrieval (IR) techniques, such as [Berger and Mittal, 2000; Carmel *et al.*, 2000]. It will be interesting to test the effectiveness of our system in situations where a user's request retrieves multiple responses that do not overlap completely. In such situations, a traditional IR approach of simply finding the most suitable response document will not suffice, instead the response will need to be collated from multiple response documents.

7 Conclusion

The framework we are developing for collating help-desk responses from multiple documents is unique with respect to previous work on question answering and user-driven summarization. This is largely due to the nature of our domain, which is characterized by repetition and redundancy in request-response pairs, and by the strict demands placed by users on the relevance and coherence of a response. In addition, a help-desk corpus represents interactions with a typically large community of users, which introduces the scope for generalizing types of interactions, while at the same time affording the personalization of responses.

Uncertainty arises in the help-desk domain due to the presence of multiple responses that seem appropriate for a given query. To address this problem, our system extracts as much information for which there is support in the corpus, collating it into a planned response; measures the cohesion and completeness of this response; and produces complete or partial responses that have a high level of confidence.

The initial implementation of our system shows that a small fraction of the responses in our corpus can be represented by generic model responses. We have also presented a method for matching users' requests with model responses. However, in many cases a response to a user's request will require matching the user's request to individual sentence types, rather than complete responses, and we are currently investigating this issue.

The main application of our framework is the automation (or semi-automation) of help-desk responses. However, we believe that it can have other applications, such as automatic FAQ generation and extraction of answers from newsgroup discussions.

Acknowledgments

This research was supported in part by grant LP0347470 from the Australian Research Council and by an endowment from Hewlett Packard. The authors also thank HP for the extensive help-desk data, and Tony Tony for assistance with the sentence-segmentation software.

References

[Berger and Mittal, 2000] Adam Berger and Vibhu O. Mittal. Query-relevant summarization using FAQs. In *ACL2000 – Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 294–301, Hong Kong, 2000.

[Carmel *et al.*, 2000] David Carmel, Menachem Shtalhaim, and Aya Soffer. eResponder: Electronic question responder. In *CooplS '02: Proceedings of the 7th International Conference on Cooperative Information Systems*, pages 150–161, Eilat, Israel, September 2000.

[Filatova and Hatzivassiloglou, 2004] E. Filatova and V. Hatzivassiloglou. Event-based extractive summarization. In *Proceedings of ACL'04 Workshop on Summarization*, pages 104–111, Barcelona, Spain, 2004.

[Hatzivassiloglou *et al.*, 2001] V. Hatzivassiloglou, J.L. Klavans, M.L. Holcombe, R. Barzilay, M.Y. Kan, and K.R. McKeown. SimFinder: A flexible clustering tool for summarization. In *Proceedings of the NAACL Workshop on Automatic Summarization*, Pittsburgh, Pennsylvania, 2001.

[Lin and Hovy, 2003] C.Y. Lin and E.H. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Language Technology Conference (HLT-NAACL 2003)*, Edmonton, Canada, 2003.

[Lin, 1998] Dekang Lin. Automatic retrieval and clustering of similar words. In *COLING-ACL'98 – Proceedings of the International Conference on Computational Linguistics and the Annual Meeting of the Association for Computational Linguistics*, pages 768–774, Montreal, Canada, 1998.

[Marom and Zukerman, 2005] Yuval Marom and Ingrid Zukerman. Corpus-based generation of easy help-desk responses. Technical Report 2005/166, School of Computer Science and Software Engineering, Monash University, Clayton, Australia, 2005.

[Oliver, 1993] Jonathan J. Oliver. Decision graphs – an extension of decision trees. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pages 343–350, Fort Lauderdale, Florida, 1993.

[Radev *et al.*, 2000] D.R. Radev, H. Jing, and M. Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the ANLP/NAACL2000 Workshop on Summarization*, Seattle, Washington, 2000.

[Salton and McGill, 1983] G. Salton and M.J. McGill. *An Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[Wallace and Boulton, 1968] C.S. Wallace and D.M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968.

[Wallace and Patrick, 1993] C.S. Wallace and J.D. Patrick. Coding decision trees. *Machine Learning*, 11:7–22, 1993.

Breakthroughs and Challenges in Computational Semantics Implications for Question Answering

Johan Bos
School of Informatics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 9LW
Scotland, United Kingdom

Abstract

It's an exciting time for computational semanticists. Recent advances in robust parsing now give us the means to compute relatively fine-grained semantic representations for texts independent of the domain, using techniques from formal semantics such as lambda calculus and underspecification, and techniques from automated reasoning such as first-order theorem proving and model building. Finally we have reached a point where we can say goodbye to shallow processing and welcome more sophisticated techniques in semantic processing. Crossovers with previous efforts in named entity recognition, word sense disambiguation, and reference resolution are bound to happen (or are already happening) in the immediate future, providing a fleshed out framework for applications using computational semantics.

Given this rather optimistic view of the current state of the field, have we finally reached a point in time where we can announce that the problem of natural language understanding has been successfully solved? Or are there any clouds on the horizon?

I will address this issue by discussing two showcases for wide-coverage computational semantics and inference: open-domain question answering (in the context of the TREC and CLEF campaigns), and textual entailment determination (in the context of the PASCAL challenge). I will present a domain-independent framework for wide-coverage natural language processing based on Combinatorial Categorial Grammar (CCG) and Discourse Representation Theory (DRT) and discuss its strong and weak points in the light of these two applications, highlighting the use (and the lack) of background knowledge in semantic processing. This case study reveals (perhaps rather embarrassingly) the Achillis' heel of computational semantics: while we might have the right tools and the recipe for doing the job, there is still a vital ingredient missing.

References

K. Ahn, J. Bos, S. Clark, J.R. Curran, T. Dalmas, J.L. Leidner, M.B. Smillie & B. Webber (2004): Question Answering with QED and Wee at TREC-2004. In Voorhees and Buckland (eds.): The Thirteenth Text REtrieval Conference, TREC 2004.

P. Blackburn & J. Bos (2005): Representation and Inference for Natural Language. A First Course in Computational Semantics. CSLI Publications.

J. Bos (2005): Towards Wide-Coverage Semantic Interpretation. Proceedings of Sixth International Workshop on Computational Semantics IWCS-6. Pages 42–53.

J. Bos, S. Clark, M. Steedman, J.R. Curran & J. Hockenmaier (2004): Wide-Coverage Semantic Representations from a CCG Parser. Proceedings of the 20th International Conference on Computational Linguistics (COLING '04), Geneva, Switzerland.

J. Bos & K. Markert (2005): Combining Shallow and Deep NLP Methods for Recognizing Textual Entailment. In: Pascal, Proceedings of the First Challenge Workshop, Recognizing Textual Entailment. Pages 65–68.

H. Kamp & U. Reyle (1993): From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT. Kluwer.

M. Steedman (2001): The Syntactic Process. The MIT Press.

Toward Question Answering for Simulations

Mark G. Core, H. Chad Lane, Michael van Lent, Steve Solomon, Dave Gomboc, Paul Carpenter
The Institute for Creative Technologies, The University of Southern California
13274 Fiji Way, Marina del Rey, CA 90292 USA
core,lane,vanlent,solomon,gomboc,carpenter@ict.usc.edu

Abstract

The new research area of explainable artificial intelligence (XAI) allows users to question simulated entities whose motivations would otherwise be hidden. Here, we focus on the knowledge representation issues involved in building such systems.

1 Introduction

As artificial intelligence (AI) systems in military simulations become increasingly complex, it has been difficult for users to understand the activities of computer-controlled entities. Because military simulations are often used for their predictive power, the AI systems that drive them must be validated as behaving realistically and according to doctrine. Detailed specifications are drafted for these AI systems and the resulting behaviors are put under heavy scrutiny. In most cases, because the observer has no way to question AI-controlled entities, the observer's only recourse is watching numerous simulation runs looking for cases where faulty reasoning results in an incorrect action.

Military simulations are also used for training, replacing some or all of the soldiers in a live training exercise. Live training exercises often use after action reviews (AARs), typically led by a senior officer, to identity soldier and unit strengths and weaknesses. US Army Field Manual 25-101, "Battle Focused Training", states that "The OPFOR [opposing force] can provide valuable feedback on the training based on observations from their perspectives...the OPFOR can provide healthy insights on OPFOR doctrine and plans, the unit's action, OPFOR reactions to what the unit did." [Army, 1990][Appendix G] If the OPFOR are simulated entities, these entities must be able to answer questions to improve the student's understanding of their actions (human OPFORs in live exercises are available during AARs). The simulated friendly forces also need to participate in the AAR because otherwise human commanders may not see how their orders translate into the behavior of units and entities.

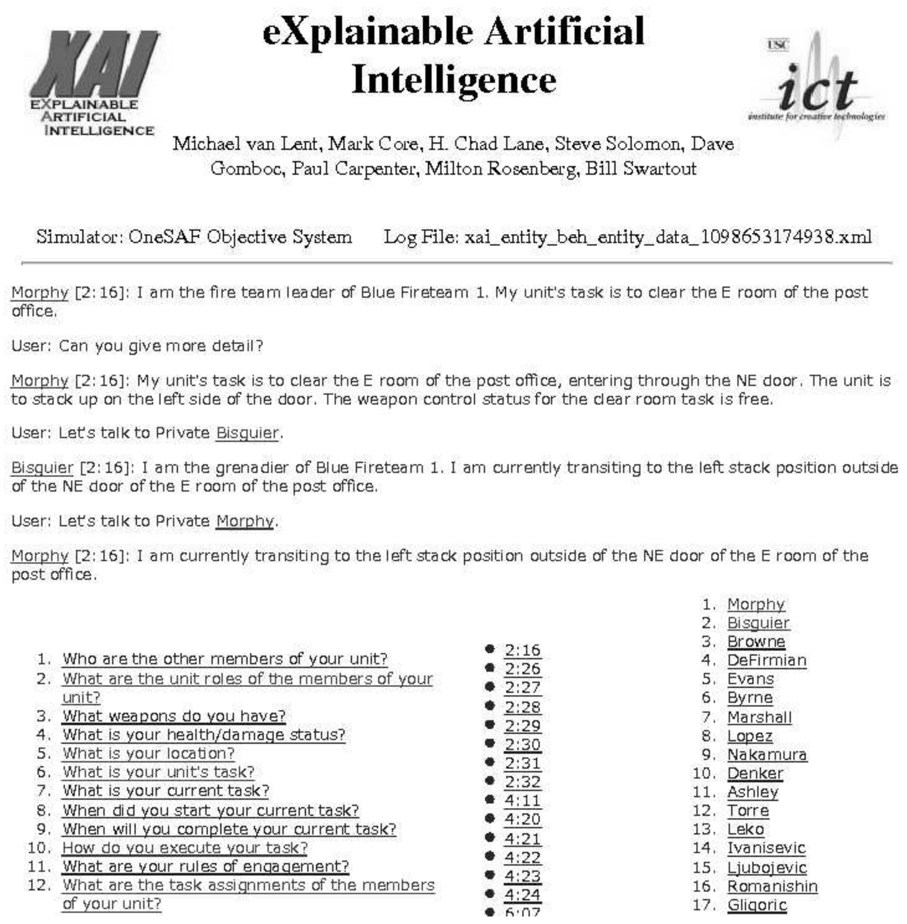
Figure 1 (a screenshot of our system's user interface) introduces the concept of an explanation system for simulated entities; following [van Lent *et al.*, 2004] we use the term, explainable artificial intelligence (XAI) system. Users select a time point to discuss, an entity to be questioned, and the question itself. Some of the questions are specific to the particular entity (e.g., what is your health?) while others concern a larger group (e.g., what is your unit's task?).

In this paper, we first discuss the challenges involved in building such a system and then present our new modular architecture for this task and our implementation of this architecture for the military simulation, the One Semi-Automated Forces Objective System (OOS) [Courtemanche and Wittman, 2002]. We highlight the central database of the architecture as it allows us to link abstract plans to actions recorded in the log files, meaning simulated entities can explain how they are attempting to achieve their goals.

2 XAI Challenge

Explanation systems for simulated entities have been built previously [Johnson, 1994; van Lent *et al.*, 2004] but were specific to the AI systems controlling the simulated entities in those applications, and not directly applicable to other AI systems. If those systems do not represent the information necessary for explanation then the user will be limited in the questions he can ask. Consider the example of sending a fire team to clear a room. Once the fire team is in position outside the room, the grenadier throws a grenade before the team enters the room. This could be encoded as a procedure (the step before entering the room is always throwing the grenade) in which case, the system cannot explain why the grenade was thrown.

This problem is not new; the literature review in [Swartout and Moore, 1993] points out that researchers studying explanation for expert systems fairly quickly agreed that the data structures of the expert system had to be designed with explanation in mind. Swartout and Moore advocated building a high-level knowledge base containing facts about the domain and problem-solving strategies, and using an automatic program writer to build an expert system from this specification. The problem is more complicated for XAI because the executable code must interface with an external simulation (a technical challenge) and be adopted by users of the resulting package (a social challenge). Military users must be convinced that the resulting AI controlled entities are as realistic as the AI systems built by more traditional approaches. Another issue is the extra effort needed to create such a knowledge base, but it may be the case that such effort is offset by



XAI EXPLAINABLE ARTIFICIAL INTELLIGENCE

eXplainable Artificial Intelligence

USC ict institute for creative technologies

Michael van Lent, Mark Core, H. Chad Lane, Steve Solomon, Dave Gomboc, Paul Carpenter, Milton Rosenberg, Bill Swartout

Simulator: OneSAF Objective System   Log File: xai_entity_beh_entity_data_1098653174938.xml

Morphy [2:16]: I am the fire team leader of Blue Fireteam 1. My unit's task is to clear the E room of the post office.

User: Can you give more detail?

Morphy [2:16]: My unit's task is to clear the E room of the post office, entering through the NE door. The unit is to stack up on the left side of the door. The weapon control status for the clear room task is free.

User: Let's talk to Private Bisquier.

Bisquier [2:16]: I am the grenadier of Blue Fireteam 1. I am currently transiting to the left stack position outside of the NE door of the E room of the post office.

User: Let's talk to Private Morphy.

Morphy [2:16]: I am currently transiting to the left stack position outside of the NE door of the E room of the post office.

1. Who are the other members of your unit?
2. What are the unit roles of the members of your unit?
3. What weapons do you have?
4. What is your health/damage status?
5. What is your location?
6. What is your unit's task?
7. What is your current task?
8. When did you start your current task?
9. When will you complete your current task?
10. How do you execute your task?
11. What are your rules of engagement?
12. What are the task assignments of the members of your unit?

- 2:16
- 2:26
- 2:27
- 2:28
- 2:29
- 2:30
- 2:31
- 2:32
- 4:11
- 4:20
- 4:21
- 4:22
- 4:23
- 4:24
- 6:07

1. Morphy
2. Bisquier
3. Browne
4. DeFirmian
5. Evans
6. Byrne
7. Marshall
8. Lopez
9. Nakamura
10. Denker
11. Ashley
12. Torre
13. Leko
14. Ivanisevic
15. Ljubojevic
16. Romanishin
17. Gligoric

Figure 1: Interface to XAI for OOS

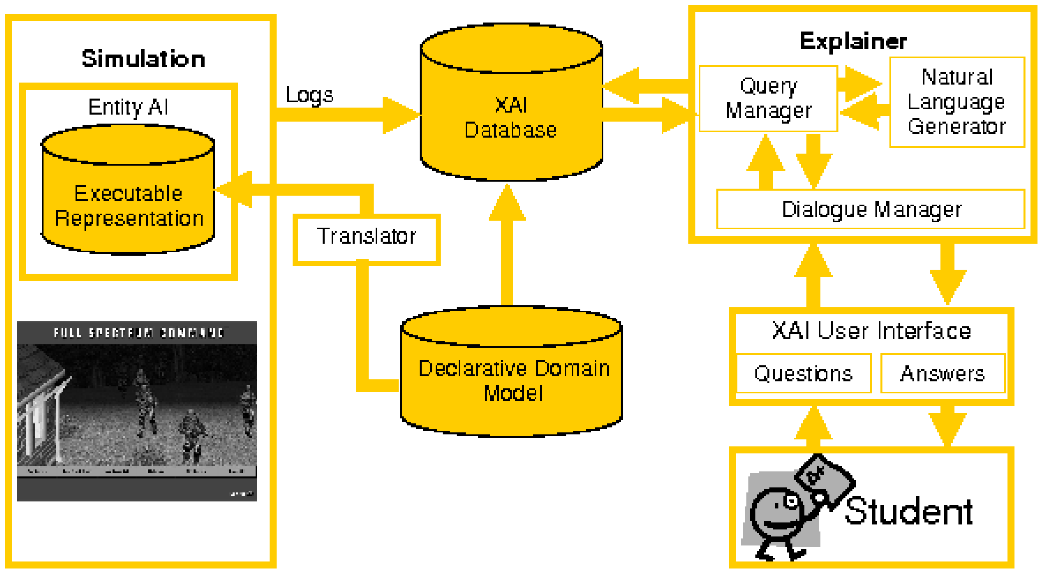


Figure 2: Domain Independent XAI Architecture

the increased usability of the system (e.g., the testing process may be quicker).

Swartout and Moore went on to discuss a plan-based approach to generating explanations, an architecture designed explicitly to facilitate:

- handling requests for clarification and elaboration
- customization based on a user model
- explanations at varying levels of detail
- natural explanations (e.g., human explanations often use pronouns and discourse markers)

We follow this advice in proposing a decoupled approach where the AI systems of a simulation are not changed but instead a declarative representation of the domain is translated into the knowledge representation used by the AI system; during explanation, the entity's actions can be reconnected to the original data structures. This domain representation would contain facts about the domain, definitions of terminology, and detailed representations of actions and problem solving strategies.

Consider the previous example of a procedure for clearing a room always having the "throw grenade" action before "enter room". The idea behind the procedure is that the grenade will suppress the enemy forces inside the room. System designers may not encode suppression as a precondition, but in the declarative domain model, we can define the concept suppression and include it as the goal of the throw grenade action in the context of clearing the room. The AI system will not use this information but it will be available during explanation to answer the question "Why was a grenade thrown into the room?"

Figure 2 shows how the declarative domain model augments the representations of an AI system. Each action type in the log file has a unique identifier linking it to its declarative representation and allowing the XAI database to reconstruct declarative representations of all the actions logged by the AI system. To answer user questions, the explainer selects information from the database and presents it via natural language and graphics displayed in the user interface.

3 XAI for OOS

Our first instantiation of the XAI architecture connects to the One Semi-Automated Forces Objective System (OOS), an entity based simulation system being developed by the U.S. Army [Courtemanche and Wittman, 2002]. XAI for OOS was a challenging project because of a short development time (2 months) and the fact that the target simulation was still a work-in-progress (the official release is in 2006). Interfacing early with a simulation or training aid is important because decisions by the developers of such programs can deeply impact the ability of an XAI system to explain entity actions and we plan to work with the OOS team to overcome the limitations discussed below.

The OOS team's approach to entity simulation is to encode declarative action representations in XML and use Java code to actually implement the described action. Our version of OOS only had the Java code (the XML descriptions were not ready) and we had to enter action descriptions by hand into the XAI database (a relational database).

An additional source of information is the simulation scenario which contains the static information associated with the simulated world, and the mission table. The static information defines all the entities including their rank, unit, and weapons carried. The mission table contains a series of unit tasks and their parameters (e.g., unit task = clear room(agent,room), agent = fireteam 1, room = east room of the warehouse). During the simulation, entities perform individual tasks in service of these unit tasks, and log files record the states of these entities (e.g., when they start and end tasks) and information about any weapon fire events. After the simulation is complete, XAI loads the log files into its relational database and the query generator searches for "interesting" time points. The time menu on XAI's interface is sensitive to the current entity (i.e., it displays the interesting time points for that entity). For each entity, the query generator identifies interesting times, defined as when the entity fired its weapon and the start, end, and mid points of entity tasks. This definition of interesting times is a placeholder; when deploying an XAI system, feedback from subject matter experts is vital for defining what questions the system should answer and what events are interesting.

After this initialization process is complete, the user can investigate the results of the simulation using the interface shown in figure 1. The screenshot shows 12 of the 16 questions supported by the system and includes questions about the entities' state (e.g., "What is your health/damage status"), questions about the scenario (e.g., "Who are the other members of your unit"), and questions about tasks. The user can ask for descriptions of the current individual (primitive) task, the current group task, parameters of the task, or how, in general, to perform the task.

Although this list of questions reflects our research into questions asked during military after action reviews, they are limited to information directly encoded in the log files. In our current architecture, the query manager executes queries to retrieve answers to questions. In future work, we will replace this component with a general purpose XAI reasoner capable of answering the more complicated questions necessary for military training purposes. Another influence on what questions can be answered is the completeness of the simulator's log files. In [Gomboc *et al.*, 2005] we discuss requirements for such log files and how the requirements impact what questions can be answered.

The question list is context sensitive, and all 16 questions are not always available. If the query generator determines that there was no weapon fire event at the current time, then weapon fire questions (13-15) will be removed from the list. The question list is also sensitive to the dialogue history as the request, "Can you give me more detail?", can only be made when the previous turn described the current unit task, or described how to perform a unit task; currently our system is only able to elaborate on these two answers.

The interface to XAI for OOS is controlled by a servlet engine that communicates via XML with the explainer. Users select an entity and a time point from the menus on the bottom of the screen, and then select a question from the same menu area. The middle frame of the page contains a dialogue history (all the questions asked by the user and the system responses).

Currently, the system starts by talking to the first entity of the first unit (in our test scenario, this is "Private Morphy") as shown in the screenshot. 2:16 is the first significant time for Morphy as he starts his task of clearing the east room of the post office. The first time a user talks to an entity, the dialogue manager creates an introduction (such as the one shown in the first line of the dialogue). Since the user has never spoken to Morphy, he introduces himself by giving his unit role, and since the user has not spoken to any other members of Blue Fireteam 1, Morphy describes the unit's task.

The process to create this introduction is the same process as answering the questions "what is your unit role?" and "what is your unit's task". The dialogue manager uses the query generator to retrieve the relevant information from the database and sends it to the natural language generation (NLG) module. Because of development time constraints, NLG was implemented with natural language templates (written in XSLT); slots in the templates are filled with information from the database. NLG formats names and times so that they appear as links in the user interface (clicking on them changes the current entity or time). We plan to use this facility in future work to encode formatting such as bulleted lists.

4 Discussion

Rather than simply writing an XAI module that only worked for its target AI system and simulator, we used our generic and modular architecture for XAI systems in building XAI for OOS. In a short time, with this new architecture, we have been able to exceed the capabilities of our predecessor system, XAI for Full Spectrum Command [van Lent *et al.*, 2004]. Because task information is linked to individual actions recorded in the log files, XAI for OOS can explain the generic method for achieving its current task as well as discussing the current parameters of this task.

The modularity of the system is important to allow interoperability with different AI systems and different simulators. As a portability test, we modified our system to accept log files from Full Spectrum Command (FSC). The OOS scenario that we have been using (light infantry) is very similar to the domain of FSC and we focused on supporting the questions that made sense in both domains. The major changes needed were changing the database schema to match the new format, updating the query manager so it could find the needed information in this new format, and changes to NLG to support new actions and objects.

The modularity of the system will make it easier to improve. For example, we plan to extend the dialogue manager to maintain a more extensive dialogue history. This code can be rewritten or replaced by a dialogue management toolkit without disturbing the other parts of the system. Even the user interface is simply a module of the system, and another interface supporting the explainer's XML communication format could replace our current interface with no changes to the rest of the system.

The modularity of our XAI architecture will also enable it to integrate with external software. Currently, with respect to training, XAI is best classified as a discovery system where users investigate the events of the simulation with no external direction (i.e., learning is left entirely up to the student). We are currently designing an intelligent tutoring module to provide the pedagogical presence necessary for effective training. The tutor could use the XAI system to illustrate an important lesson, or direct the student to use XAI to investigate an interesting time point, among other strategies.

A topic for future work is defining the general relationship between question answering systems and tutoring. One possibility is to teach the students about the QA system itself (how to use it). We are developing an investigation model that encodes the detective skills that an expert XAI user would possess. This model will be similar to troubleshooting guides for electronics (e.g., try to isolate the component causing the problem, then investigate the component in detail searching for faults). We can imagine developing investigation models for other QA tasks such as finding relevant research papers, or bargain shopping for airfares.

Acknowledgments

Thanks to Milton Rosenberg, William Swartout, and David Traum for their guidance and support. Note, the project described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the U.S. Government, and no official endorsement should be inferred.

References

[Army, 1990] FM 25-101: Battle Focused Training. Headquarters Department of the Army. Washington D.C., 1990.

[Courtemanche and Wittman, 2002] A. Courtemanche and R. Wittman. OneSAF: A product-line approach for a next-generation CGF. In *Proc. of the Eleventh SIW Conference on Computer-Generated Forces and Behavioral Representations*, pages 349–361, 2002.

[Gomboc *et al.*, 2005] D. Gomboc, S. Solomon, M. G. Core, H. C. Lane, and M. van Lent. Augmenting behavior models to support automated explanation and tutoring. In *Proc. of the Fourteenth Conference on Behavior Representation in Modeling and Simulation*, 2005.

[Johnson, 1994] W. L. Johnson. Agents that explain their own actions. In *Proc. of the Fourth Conference on Computer Generated Forces and Behavioral Representation, Orlando, FL*, 1994.

[Swartout and Moore, 1993] W. R. Swartout and J. D. Moore. Explanation in second generation expert systems. In J.M. David, J. P. Krivine, and R. Simmons, editors, *Second Generation Expert Systems*. Springer-Verlag, 1993.

[van Lent *et al.*, 2004] M. van Lent, W. Fisher, and M. Mancuso. An explainable artificial intelligence system for small-unit tactical behavior. In *Proc. of the Sixteenth Conference on Innovative Applications of Artificial Intelligence Conference*, Menlo Park, CA, 2004. AAAI Press.

A Question-Answering System for Portuguese

Carlos A. Prolo
PUCRS
Porto Alegre, Brazil
prolo@inf.pucrs.br

Paulo Quaresma
Irene Rodrigues
Pedro Salgueiro
Universidade de Évora
Évora, Portugal
{pq,ipr,pds}@di,uevora.pt

Renata Vieira
UNISINOS
São Leopoldo, Brazil
renata@exatas.unisinos.br

CONTENT AREAS:
Methodologies for Intelligently Answering Questions
Knowledge Representation and Integration
Language Processing

Abstract

In this paper we describe a Q&A system for Portuguese. The core of the system is a unification-based process that tries to fill in the gaps in the questions with the facts obtained from any given corpus with sentences of the language.

1 Introduction

This paper reports on an ongoing project to build a question-and-answer system for Portuguese. An initial version of this system was used in the Q&A task of the CLEF04 [Peters and Borri, 2004] with encouraging results, particularly the precision, as reported later in the paper.

The core of the approach is the unification of the open predicate expressed by the question asked and the closed predicates conveyed by the sentences in the corpus. This is done using Prolog. Consider the sentences in (1) and (2). The first line is the original version in Portuguese. The second line has the English equivalent, and the third line shows it with a *trace* (the ϵ) at the position the argument would normally be in the declarative form.[1]

(1) Quem matou Odete Roitman ?
Who killed Odete Roitman ?
Who$_i$ ϵ_i killed Odete Roitman ?

(2) O que Cristovão Colombo descobriu ?
What (did) Cristovão Colombo discover ?
What$_i$ (did) Cristovão Colombo discover ϵ_i ?

The question can then be written in a logical form as an open predicate – a predicate with unbound variables, – with the variable at the position corresponding to the trace, which is the missing argument being asked, as shown in (3) and (4), where W stands for the variable.

[1] See for instance [Haegeman, 1994] for the linguistic theories of *wh-movement*.

(3) **kill** (W , Odete Roitman)

(4) **discover** (Cristovão Colombo, W)

Now suppose we find the sentence in (5) in the corpus, whose associated closed predicate is in (6). We can unify the predicate with the one for sentence (1), obtaining $W = Laura$ which is the answer we are looking for.

(5) Laura killed Odete Roitman .

(6) **kill** (Laura , Odete Roitman)

2 System description

We describe in this section the main components used within the approach.

2.1 The parser

We use the Eckhard Bick's parser Palavras [Bick, 2000] to find the syntactic structure for both questions and corpus sentences. The parser provides rich morpho-syntactic information on both the lexical and constituent levels. For instance, upon finding the main verb of a clause the parser provides us with its lemma, which is used as the name of the predicate, allowing for independence of the particular verbal form used in each sentence. This is essential, especially in a morphologically rich language such as Portuguese.

The parser does not mark the position for empty arguments. However, it provides grammatical function of the constituents and assigns tags to the **wh** expressions, such as **SUBJ** for subject extraction, **ACC** for direct object extraction, and similar ones for prepositional object and adverbial adjunct. That allows for a good approximation of a unique predicate argument order for each verb. Of course, we have to rely on correct resolution of PP attachment (but see section 3), and there might be problems with verbs that take optional arguments.

2.2 Sentence Representation

A semantic representation (sr) for a text is given as a set of predicates with identification of each recognized entity. The format is as exemplified below in (8) for the sentence in (7). The example also shows how sub-constituents are treated.

(7) Um patologista defendeu que Jimi Hendrix morreu de asfixia após ter ingerido álcool e uma dose excessiva de barbitúricos.
(A pathologist defended that Jimmy Hendrix died of asphyxia after having ingested alcohol and an excessive dose of barbiturates.)

(8) **sr (entities:[**
A: indefinite, masc, sing,
B: definite, masc/fem, sing,
C: definite, fem, sing,
D: definite, masc, sing,
E: indefinite, fem, sing **]**,
predicates:[
pathologist (A),
defend (A,B),
name (B, Jimmy_Hendrix),
die (B),
rel (of, B, C),
asphyxia (C),
rel (after, C, D)
ingest (D)
alcohol (D)
dose (D)
excessive (D)
rel (of, D, E)
barbiturates (E) **])**

Each entity in the text is introduced as a variable over which we can predicate. The attributes regarding definiteness, gender, number, etc., obtained from the parser will help, for instance, to restrict the possible matchings (e.g. masculine vs. feminine). This is a very strong feature of Portuguese as a morphologically rich language.

Notice that prepositional phrases are conveyed as **rel** predicates since at this level we do not know exactly what semantic relation they play. We will come back to this issue ahead when we talk about the ontology.

For questions a similar representation is built, in which the wh-pronoun – or similar word that stands for the type of the answer – will be specially marked as the initially unbound variable whose value we will be interested after unification with the semantic representation of any given text. (9) and (10) provide an example for a very simple question.

(9) Quem morreu ?
(Who died ?)

(10) **sr (entities:[**
var F: interrog(who), masc/fem, sing/plur **]**,
predicates:[
die (F) **])**

When the question is unified with (8), F unifies with B, and we know that the answer is Jimmy Hendrix.

2.3 The Ontology

So far we have presented examples for which the answers can be derived immediately from the semantic representation that resembles the constituent structure of the sentences. Often this is not so straightforward. An ontology was integrated into the system using the language OWL and ISCO [Abreu, 2001], containing general rules relating concepts, which when applied to the initial formulas allows for it to be turned into a more general and informative form.

The ontology was created using a semi-automatic approach. First, a top level ontology was defined, with generic concepts, such as, actions and entities. Then, a simple low-level ontology of domain concepts is automatically extracted from the documents and merged with the top-level ontology. The automatic extraction of concepts is based on the tree-like syntactic structure and it detects and extracts triples of verb-subject-object. These triples are then related using semantic information supplied by the parser Palavras and merged with the top-level ontology. A more detailed description of this process can be found at [Saias and Quaresma, 2005]. The ontology was useful to derive some logical rules to deal with the generic relation *rel*. If we have classes such as *person* and *thing*, which are related in some other class, such as *own*, we derive a rule that says that if a person is in a relation *of* with a *thing*, this relation may be of type *own*, as illustrated below in Figure 1. This corresponds to the semantic/pragmatic module of the system.

```
class person.
    name N;
    ...

class thing.
    ...

class own:
    person A;
    thing B.

own(A,B) <- rel(of, B, A),
            person(A),
            thing(B).
```

Figure 1: Example of Logical Rules of the Ontology

A typical case is the handling of the predicate *rel* that arises from the prepositional phrases. From the noun phrase *"O gato do Manuel"* (*"the cat of Manuel"* $\rightarrow$ *"Manuel's cat"*), the system first generates a predicate *rel (of, A, B)* which is then converted into *owns (B, A)* by use of the ontology rules that know about the use of the preposition of to express possession of animals by persons.

As an additional example, consider the question in (11). The system was able to correctly answer it, unifying it with (8), only because the interrogative pronoun *como* (*how*) was turned into an *"of"* relation modifying the verb, as shown in (12) (cf. "**Of what** *did Jimmy Hendrix die ?*").

(11) Como morreu Jimi Hendrix ?
How did-die Jimmy Hendrix ?

(12) **sr (entities:[**
F: definite, masc/fem, sing
var G: interrog(what), masc, sing **]**,
predicates:[
die (F)

name (F, Jimmy_Hendrix)
rel (of, F, G) **])**

This approach results in an overgeneration of rules and thus many alternative interpretations for *rel* relations. This is not however a problem for the final goal of getting the correct answer for questions. Indeed it is part of the solution. The key point here is that the same piece of information can be expressed in many different ways and so can the questions. By providing logical predicates one for each interpretation, we improve our chances of having one of them unified and hence obtaining an answer.

As simple as it may be right now, the ontology has proven very useful for answering the questions. However, it is still a major issue since it is not powerful enough in some cases and should be substantially improved. A major point is nominalization. The nominal *"A morte do Pedro"* (*"Peter's death"*) carries the information that Peter died. Relative clauses also indirectly convey information about the NP head. The noun phrase *"The apartment bought by John"* presupposes that John bought the apartment referred to by the NP. Appositives, pervasive in newspaper articles, carry an equative relation between nominals. Named entity recognition – distinguishing persons, locations, etc. – is important, but also the understanding of which common nouns in a question trigger the search for which entities (e.g *"Which city"* means we are looking for a place). The improvement of the ontology is currently a major issue in the project.

3 The tricks of the trade

In this section we cover a few selected issues related to complex syntactic or semantic aspects of language which are hard to solve in a principled approach, but which are seen as very relevant to improve the effectiveness of the system. They are "tricks", because they do not handle the issues in a principled way. However, they are intended to improve the quality of the answers approximately as if they were handled properly in a principled way.

3.1 Reciprocal Verbs and Alternations

Many verbs can have their arguments switched over, roughly preserving the meaning, such as in (13) [Levin, 1993, p. 36]. If we just proceed as we have shown before, when these verbs appear in a question, our chance of having an appropriate sentence to match it immediately drops to 50%, since the order of the arguments in both the question and the corpus sentence is random. The trick here is to replicate the predicate in the question, with the arguments switched in the second occurrence, in an *OR* construction.

(13) Maria casou com Pedro.
(Mary married Peter.)
Pedro casou com Maria.
(Peter married Mary.)

Among the many verbs that can take this alternation is the verb *"ser"* in its equative usage – the equative *be*,[2] which appears extremely often in questions.

[2]Portuguese already has two different verbs for *be* – *"ser"* and *"estar"*, but still *"ser"* can be used in either equative or predicative constructions.

Currently we do not distinguish verbs that can take the alternation from those that cannot. Neither we distinguish the equative from the predicative construction. We replicate the predicates for the main verb on all questions. Whenever this is inappropriate it is still unlikely to cause a problem since the alternate predication will not succeed to unify. Try to find an answer for *"Who died of Jimmy Hendrix?"*

A principled account of this kind of phenomena would imply to have a good verb ontology since there are hundreds of other kinds of verb alternations(e.g., see [Levin, 1993] for English), perhaps none of them, individually, nearly as relevant to our task as the one we just covered, but that as a group would make a difference. Another issue is that even reciprocality has other manifestations, depending on particular verb groups (cf. *"Mary and John married."*)

3.2 Passives and Verb Features

We have not conduced a thorough study of the relevance of particular verb features to the quality of the answers. We know about the importance of agreement features – number, person and gender, – to help determining the features of arguments when their morphology is ambiguous. But this is already accounted for by Bick's parser.

An important feature is the passive voice, which inverts the order of the arguments and should be considered when generating the predicates in the semantic representation. It is pervasive in language, but in particular, it tends to appear in questions when one does not want to mention the agent of some action, as in *"When was Kennedy assassinated?"* We could use the information from the parser to do the argument inversion. However, currently this is already taken into account by the overgeneration of the trick for reciprocality.

At first glance, other syntactic features of the verb, such as tense, mood, or aspect, do not appear to be so relevant for the effectiveness of the system, but this deserves more careful evaluation.

3.3 Prepositional Attachment

Current parsers are notoriously bad at correctly attaching prepositional phrases to their heads. Bick's parser tends to attach the PPs to the lower possible candidate – which is statistically the most likely place. The trick here is to make available a duplicate, alternative **rel** predicate for which the attachment is higher. Consider the sentence in (14). Because the parser wrongly attaches the PP to the lower noun *leukemia* instead of to the verb *die* the sentence does not help to answer *"When did John die?"* By providing an alternative attachment to the verb the answer comes straightforwardly.

(14) Paulo morreu de leucemia in 1956.
(Paul died of leukemia in 1956.)

The question remains of how much can we be hurt if our hypothesis of higher attachment is plainly wrong. Although we have not made a precise statistical analysis, empirical observation has shown that this is very unlikely to cause trouble due to the usually very distinct nature of the attachment candidate heads. That is, the wrong attachment alternative will never succeed unifying.

3.4 Parts-of-Speech Mistakes

We realized that the parser makes occasional mistakes in the parts-of-speech. When this happens for a sentence of the corpus even one that carries the answer for the question, this can often be overcome, since the answer may also be contained in other sentences of the corpus. However when the mistake happens at the question, then it its highly destructive. In order to overcome this problem we may use the fact that the initial morphological analysis provides us with all the possible parts of the speech of each word. Here the trick is to allow for the creation of multiple alternatives for each of the possible combination of parts of speech. The *"combinatorial explosion"* is not a problem due to the usually small size of the questions.

4 Evaluation

This system has been used to compete in the question-answering track of the CLEF 2004 [Peters and Borri, 2004] and got the best score among the three Portuguese runs and the 7th global best score (among 57 runs) [Quaresma *et al.*, 2004].

Of the 199 questions[3] it found an answer for 72, of which 47 (or 65%) were correct, 18 (or 25%) were inexact and only 7 were wrong. However for the remaining 127 for which it gave no answer, only 9 did not really have an answer in the corpus provided in the track. That is, the system was, in a sense, extremely conservative in giving an answer. But when it did, it generally did well.

Two main reasons contributed to the high number of missing answers. One is that the inference process is based in Prolog and, due to the computational complexity, it was not possible to make inferences over the complete knowledge base built from the analysis of the documents. So the documents had to be pre-selected from the full collection using an extension of the SINO engine [Greenleaf *et al.*, 1997] that did not perform to content, giving very low recall values. So for many questions the documents that contained the answer were not even taken into account in the inference process.

At present, changes are being made to the Prolog interpreter allowing inferences over the full knowledge base.

The second main reason for the low number of answers given can be generally attributed to the ontology. The inference process relies heavily on the ontology. The tricks in the previous section are nothing but practical solutions to tasks that belong to the ontology. (Only the PP attachment trick was used in the CLEF contest.)

5 Conclusions

We presented a question answering system, that integrates: parsing; semantic representation and inference using ontology. The reasonably high precision is a strong characteristic of the system, and progress has been made to drastically improve the recall without neglecting precision. In particular we have to improve the following aspects. Efficiency in parsing is seen as a must , together with a good indexing system, so that we can have timely access to the documents to be processed. Improvement of the ontology is being pursued, integrated with practical solutions, to allow for the successful recover of the answer.

[3]1 of the 200 questions was discarded by the judges

Acknowledgments

This work was partially supported by the funding agencies of Portugal and Brazil, GRICES, ICCTI, CAPES and CNPq.

References

[Abreu, 2001] Salvador Abreu. Isco: A practical language for heterogeneous information system construction. In *Proceedings of INAP'01*, Tokyo, Japan, 2001.

[Bick, 2000] E. Bick. *The Parsing System "Palavras". Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework*. Aarhus University Press, 2000.

[Greenleaf *et al.*, 1997] G. Greenleaf, A. Mowbray, and G. King. Law on the net via austlii - 14 m hypertext links can't be right? In *In Information Online and On Disk'97 Conference, Sydney*, 1997.

[Haegeman, 1994] Liliane Haegeman. *Introduction to Government and Binding Theory*. Blackwell, Cambridge, USA, 1994.

[Levin, 1993] Beth Levin. *English verb classes and alternations*. University of Chicago Press, Chicago, 1993.

[Peters and Borri, 2004] C. Peters and F. Borri, editors. *Working Notes for the CLEF 2004 Workshop*. Bath, UK, 2004.

[Quaresma *et al.*, 2004] Paulo Quaresma, Luis Quintano, Irene Rodrigues, and Pedro Salgueiro. The university of Évora approach to qa@clef-2004. In Carol Peters and Francesca Borri, editors, *Question-Answering Track of the Cross Language Evaluation Forum*, pages 403–412, Bath, UK, 2004. Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Richerche, ISTI-CNR, Italy.

[Saias and Quaresma, 2005] José Saias and Paulo Quaresma. A methodology to create legal ontologies in a logic programming information retrieval system. *Law and the Semantic Web*, 2005. *To appear*.

Semantic Knowledge in Question Answering Systems

Vincent Barbier, Brigitte Grau, Anne-Laure Ligozat, Isabelle Robba and Anne Vilnat
LIMSI-CNRS, BP 133
91403 Orsay cedex
France
e-mail : FirstName.Name@limsi.fr

Abstract

QA systems need semantic knowledge to find in documents variations of the question terms. They benefit from the use of knowledge resources such as synonym dictionaries or ontologies like WordNet. Our goal here is to study to which extent variations are needed and to determine what kinds of variations are useful or necessary for these systems. This study is based on different corpora in which we analyze semantic term variations, based on reference sets of possible variations.

1 Introduction

Most QA systems are composed of three main components. First, question analysis extracts terms from the question and finds the expected type of the answer. Then, a search engine searches the collection for documents. To this end, one or more successive requests are built with the question terms, and possibly with term variations. Finally, answers are selected following relevance criteria taking into account syntax and semantics. To find relevant documents, QA systems have to identify variations of question terms in these documents.

Our goal in this paper is to study to which extent variations are needed and to determine what kinds of variations are useful or necessary for these systems.

To this end, we present on the one hand an evaluation of our own strategy. Our QA system, working both on French and English languages, takes into account semantic variations of simple or composed terms of the question in order to cut down the set of documents retrieved by the search engine; this paper presents an evaluation of how relevant this strategy is, focusing on the French system.

On the other hand, we show to which point a QA system is able to find answers without requiring any semantic resource and to which extent results would be enhanced by such resources. This study is based on different corpora, in which we study semantic term variations. Two corpora come from the evaluation on French: one is made of all the correct answers given by the participants, the other is our set of answers. The last corpus is an automatically built corpus of correct and incorrect passages from TREC-11 questions.

After a state of the art on QA systems, we present our system FRASQUES, then we describe the studies we made on different corpora. We also describe the dedicated corpus we constituted to prevent us from the bias introduced by the use of the participants' results; finally we detail the variations present in this corpus, thanks to Wordnet ontology.

2 Semantic knowledge for selecting documents in QA systems

In order to improve document selection in QA systems, several strategies can be conceived. They consist in using semantic knowledge, present in thesauri or lexicons, at different stages of the system: i) for elaborating the query given to the search engine; ii) on the results of the search engine, for selecting the best documents or extracting small passages. QA systems generally make use of thesauri by selecting words close to question words according to semantic or lexical relations, such as synonymy, hyperonymy and hyponymy.

In the first strategy, namely query elaboration, a first problem consists in choosing the right keywords in the question; then a second problem is raised by the search of related words. Keyword selection is often based on the morpho-syntactic category of question words. They can also be weighted or considered differently in the query according to pre-established rules or to their weights in a reference corpus.

In addition to keyword selection, it can be interesting to consider their linguistic variations in order to take into account some lexical distances between questions and answer-sentences. [Moldovan *et al.*, 2003] generate morphologic, lexical and semantic variations of question keywords from WordNet ([Fellbaum, 1998]), and introduce them progressively in the queries when their system do not return enough answers. [Yang and Chua, 2002]'s system merges two kinds of knowledge sources, the Web and WordNet, for extending queries: after questioning the Web, they keep those words that are the most correlated to question words and consider them as query terms since they seem relevant in the question context. They also add some related words found in WordNet.

[Ittycheriah *et al.*, 2001] have tested different document retrieval techniques, with and without query expansion. Applying expansion mechanisms as filtering criteria for selecting answers in retrieved documents gives better results than applying them for expanding requests. We also chose this solution in our systems, FRASQUES (for French language) and QALC (for English language).

3 FRASQUES System

FRASQUES roughly follows the same principles than QALC, our English QA system, even if they slightly differ in their realisation. Both are made of four main modules, colored with gray in Figure 1.

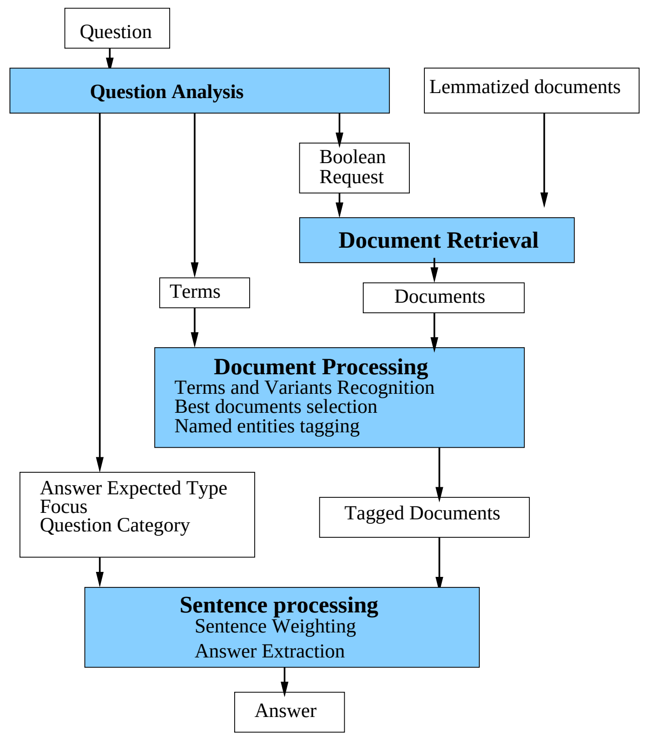


Figure 1: FRASQUES system

Question analysis proceeds in two steps. First, some information is determined such as the expected type of the answer, when this type belongs to our named entity list. Second, the lists of synonyms for non empty words of the question are built (this point is detailed in section 4.1).

The search engine, Lucene[1], is a boolean engine. To interrogate the French collection, Lucene is given a set of requests built from the non empty words of the questions. If no documents are retrieved (or a number smaller than a given threshold), the collection is searched again with fewer terms (see details section 4.3).

The documents retrieved by Lucene are re-indexed by Fastr ([Jacquemin, 1999]) in order to recognize morphological, syntactic or semantic variants of simple or composed terms of the question. These terms are weighted, and thus documents are weighted in turn. Documents are then re-ordered and a sub-set of them is considered. The named entities tagging module is then applied to these documents. The final module is in charge of extracting the answer from weighted sentences. The process differs depending on the fact that the question expects an answer which is or is not a named entity.

[1]http://jakarta.apache.org/lucene/docs/index.html

4 Analysis of the FRASQUES system

The corpora we analyze in this section come from the EQueR evaluation campaign. They are composed of the correct passages (250 characters maximum) returned by the participants, plus the results of FRASQUES.

4.1 The questions

For each question, FRASQUES question analysis module determines several kinds of information, among which three sets are more thoroughly studied in this article: i) the set of non-empty words of the question, ii) the set of their synonyms extracted from Fastr and iii) the set of their synonyms extracted from EuroWordNet.

In EQueR, the main task consisted of 500 questions. Among these, 33 did not have any Fastr synonym, and 73 did not have any EuroWordNet synonym. The average number of words per question was 5.6 while the average of Fastr synonyms was 12.8, which is relatively high, especially since among the 500 questions, there were 592 proper names, which rarely accept synonyms. The average of EuroWordNet synonyms per question was 7.1. Thus, there are nearly twice as many Fastr synonyms as EuroWordNet synonyms, which can be explained by the low coverage of EuroWordNet.

4.2 Quantitative analysis of participants correct passages

The correct passages given by the participants constitute an interesting corpus to analyze. To gauge the benefits brought by knowledge sources such as synonyms, we calculated the presence rate of synonyms in correct passages.

The corpus is composed of 2213 passages, that is an average of 4.7 passages per question (only 30 questions have not been answered). Among these passages, 82% do not contain any Fastr synonym, and 88% do not contain any EuroWordNet synonym. Only the words of the question obtain an significant rate as shown Table 1, containing the average rate of question words or synonyms per question.

Question words	60.4
Question words as Fastr synonyms	3.6
Question words as EuroWordNet synonyms	2.7

Table 1: Question words and synonyms in correct passages

Several reasons explain those rather low rates of synonyms in the corpus. First, the synonym bases are not the ones the other participants use, moreover few of them take into account such knowledge. Second, in EQueR, a lot of correct answers could be found with the words of the question. It seems (it is also true in TREC campaigns) that there is often at least one formulation close to the question, which is probably due to the large amount of documents (1.5 gigabytes).

4.3 FRASQUES answers

The set of documents returned by FRASQUES is also interesting to exploit. This corpus can be divided into two parts: the documents returned by the search engine Lucene, and the documents selected and ordered after indexation by Fastr. On the basis of these two corpora, we investigated the influence of our use of semantic knowledge on the passage selection at the different stages of the question answering process.

When querying the search engine, we favour documents containing all words contained in the question. If no such document exists, or if there are too few of them, the constraints on the query are relaxed by omitting some of the words in the question. First, a query composed of all the non-empty words of the question if formed. If a threshold number of documents (fixed at 200), is not reached, a new query is constructed which contains the focus of the question, its main verb and its proper names. Then relaxation consists in suppressing the verb, and constructing different queries for each proper name. When we take off words, their variants may still be found in the returned documents.

For the EQueR campaign, we ran the system twice, in order to test different document selection strategies. For the first run, all proper names were used without considering the threshold of 200 documents; for the second run, we checked the number of documents after each query.

To evaluate our strategy, we listed all the various short answers, in order to have a set of admissible template answers as large as possible. We then evaluated our corpus of documents and the passages returned with respect to these answers. For each run of our system, we counted the number of occurrences of template answers in our corpus, after the first two steps of our question answering process, namely document selection by the search engine, and selection by Fastr.

The search engine returns documents containing a template answer for only 73 to 76 % of the questions. This can be explained by several factors : imprecision in the choice of the keywords of the question, which are selected only due to their morpho-syntactic tagging, errors of lemmatization, problems of anaphor and so on.

The selection of 50 documents after indexing by Fastr does not entail a decrease in the number of correct documents. The first run makes use of synonyms only for multiterms, while the second run also searches synonyms of monoterms. The second run could be expected to have a better recall, but this is not the case. This can be explained by the high degree of similarity between the questions and some of their correct sentences, and also by the noise introduced by searching "incorrect" synonyms.

Multiterm semantic variants have been found by Fastr in 40 questions (9% of the questions). These variations enabled the system to link for example the phrases "transfert d'animal" and "transport des animaux", or "avocat de M." and "défenseurs de M.". The synonyms used in these cases seem more relevant than those used for recognition of monoterm variation, which can be explained by the fact that monoterms lack a context which could enable to choose between all the possible synonyms. Multiterm variants here prove their interest, and it could be useful to favour them in other steps of the question answering process, in order to reiterate document retrieval with found synonyms.

We also carried out evaluation experiments to determine how relevant it is to use the words of the question when looking for the answer. This is summarized in Table 2. For each sentence returned by our system, we counted the number of words of the question present, as well as the number of Fastr and EuroWordNet synonyms. Then we made similar experiments where we took only in consideration those sentences which were judged correct: those have higher scores in terms of occurrences of words of the question and of synonyms, which justifies our current approach of passage selection.

	Correct passages	All passages
Words of the question	69.7%	57.3%
Fastr synonyms	4.9%	4.3%
EuroWordNet synonyms	4.0%	3.2%

Table 2: Question words and their synonyms in the extracts returned by FRASQUES

The rates of synonyms in the sentences we returned correspond to those found in the corpus of answers of all participants. Like the corpus of participants, our corpus contains a high number of sentences containing no synonym : between 78 and 80%, depending on the origin of the synonyms and of the sentences. This very low occurence rate of synonyms is probably due to the lexical proximity between the questions and the answering sentences, as it was foreseeable.

5 Extension to other variations

In order to study the reliability of more extended variations, such as those given by WordNet relations, we have built semi-automatically a corpus of answers. Questions are taken from TREC11 QA evaluation and answers are extracted from the TREC11 Aquaint collection. The corpus is composed of 123 questions and 1066 pertinent answer passages (corresponding to a paragraph or 3 sentences), which makes a mean of 8.7 passages per question.

For each question, we collected a set of pertinent and non-pertinent paragraphs. At first, passage pertinence is automatically evaluated thanks to an answer pattern, which is a regular expression. But this method is quite noisy: among the passages considered as relevant, only one of three passages is really pertinent. This method reduces dramatically the human work, but a manual validation is still required. Now, we detail the method used to constitute the corpus.

5.1 Request and Corpus Filtering

In most system, query variation is limited to synonym or morphological variations which makes difficult to study more complex variations. Our aim is to study the various possible variations of each term of a question with as little bias as possible. In this purpose we decided to collect the answers by building one specific query for each studied question term.

For each query, we omit one term of the question, which will be the studied term. This term is not represented, neither by its actual form nor by a variation. Thus a variation of this term should not be favoured over another one.

The query is a conjunction of disjunction of terms. Each disjunction of terms represents a term of the question and is a set of variations of this term obtained thanks to WordNet.

The allowed variations are synonyms, plus the most frequent word of all synsets at a distance of two WordNet relations or less. The distance of the variations is reduced in order to prevent the generated query to bring too noise, which is already important. Last, the less significant terms of the question are not used in the queries.

Named Entities are considered to be better filters than common nouns and other grammatical categories. For the other terms, the term significantness is estimated by human judgment. This human ranking permits the system to automatically build more judicious queries. For example, in order to study the variations of the term "destroy", the query will be :

Pompeii & expansion(volcano) & expansion(ancient)
with: *expansion(volcano)=(mountain|mount|crater|volcano)*

Once the documents have been fetched, pertinent and non-pertinent documents are separated according to the answer pattern. Note that thanks to this method, the pertinent and non-pertinent documents are fetched with the same query, which is once more aimed at limiting bias possibilities.The last step is the manual validation of the corpus.

5.2 Study of the Variations

We searched the kinds of variations between the terms of the question and words in the retrieved passages. A term variant exists if a path of WordNet relations links the synset containing the word of the question to a synset containing the word of the passage. The links taken into account are any combination of WorNet relations, except for glosses and morphological derivations. The passages are about 180 words long.

As to measure the frequency of kinds of variants, we aggregated them into a small number of classes. We chose to classify them according to the WordNet path length. Classes are synonyms (lemmas are different but words belong to a same synset) and words when they are distant from n relations (with n from 1 to 4).

We counted how many links each passage, either pertinent or not, contains. We only considered the less distant variation of each term of the question, if exists. Figure2 shows the average frequency of classes of links in both pertinent and non-pertinent passages, and the ratio between those two numbers.

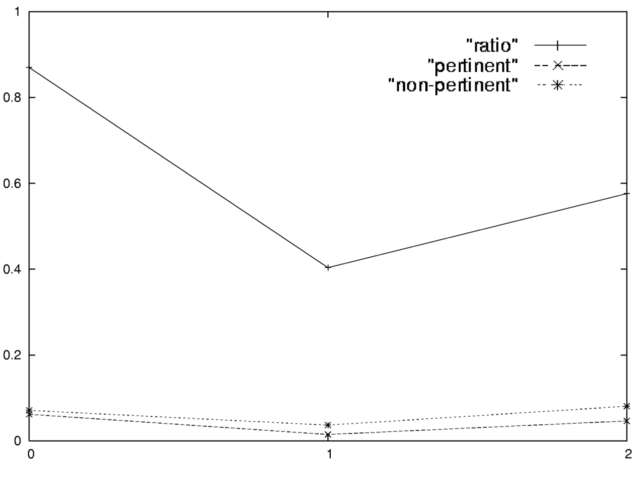


Figure 2: Precision of extended variations

We notice that the frequency of synonyms (point 0 on the x-axis in Figure2) is similar to the results we obtained in our preceding study. It appears that links compound of two relations are more frequent than 1-relation links. Moreover, they achieve a better precision. These 2-relation links often consist in hypernymy relations followed by hyponymy relations. For instance, *town*, a correct variant for *city* is given by the following path: *city* (hypernym) *municipality* (hyponym) *town*.

A question containing the word *wife* can be answered thanks to the word *husband*, because they are related through: *wife* (hypernym) *spouse* (hyponym) *husband*.

These observations show that term expansion can benefit of the use of compound relations and that variations should not be limited to synonymy or to one link hyponymy relations.

6 Conclusion

In TREC as in EQueR campaigns, systems that obtain the best results make use of semantic knowledge sources. Intuitively, one can assume that such information is necessary in open domain question answering. Nevertheless, few of these robust systems have evaluated how their results are enhanced by using this kind of information, because doing this evaluation is sometimes complex or even impossible because of the system architecture. In this paper we propose a solution which consists in exploring result corpora aiming to find what kind of knowledge was really used.

This evaluation allowed us to check that uncontrolled use of synonyms gives very few improvements in the system results; on the contrary, multiterms provide a context that makes possible the discrimination of synonyms and the selection of relevant variations.

Most importantly, this study allows us to evaluate to about 85% the rate of correct answers that may be found without quite any semantic knowledge. This rate is rather high, but not sufficient, and obtaining best results necessarily means a better use of semantic knowledge.

References

[Fellbaum, 1998] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[Ittycheriah *et al.*, 2001] A. Ittycheriah, M. Franz, and S. Roukos. Ibm's statistical question answering system - trec-10. In *Proceedings of the Tenth Text retrieval conference*, Gaithersburg, MD, 2001. NIST.

[Jacquemin, 1999] C. Jacquemin. Syntagmatic and paradigmatic representations of term variation. In *Proceedings, ACL'99*, pages 341–348, University of Maryland, 1999.

[Moldovan *et al.*, 2003] D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154, 2003.

[Yang and Chua, 2002] H. Yang and T.-S. Chua. The integration of lexical knowledge and external resources for question answering. *Proceedings od The Eleventh Text Retrieval Conference*, 2002.

A Question Typology and Feature Set for QA

Lili Aunimo
University of Helsinki
Department of Computer Science
FIN-00014 University of Helsinki, Finland
lili.aunimo@helsinki.fi

Abstract

This paper presents a typology and a feature set to be used in question classification for question answering. The typology and feature set are evaluated using a set of general domain questions.

1 Introduction

A general domain question answering (QA) system typically takes as its input a variety of semantically different natural language questions and their reformulations. In order to extract the answer to these questions from text, the answer type and the extraction pattern related to that type is needed. A question typology is used to map between the questions and the answer types. Each answer type then has a set of extraction patterns related to it. This paper proposes a question typology for general domain factual QA, and a set of relevant features that can be extracted from the questions and used to classify them according to the typology.

Almost all general domain QA systems rely on some kind of question typology. An exhaustive typology containing 140 different question types has been created by Hovy et al.[Hovy *et al.*, 2002]. This typology is provided with surface text patterns that allow the system to find answers of the desired type from text. However, the accuracy of a classifier classifying questions according to the typology presented is not reported.

Li et al. have experimented with the use of different feature combinations in question classification [Li *et al.*, 2004]. They report an accuracy of 92.5 % when classifying questions into 6 coarse classes and an accuracy of 89.3 % when classifying questions into 50 fine classes. Their classifiers are based on the Winnow algorithm and the number of active features used is counted in hundreds. Question classification results reported by Suzuki et al. [Suzuki *et al.*, 2003] are in line with the experiments of Li et.al, and they also found out that using both named entities and semantic information is necessary to perform high-performance question classification.

2 The Proposed Typology

The proposed typology is developed bearing in mind the following three main requirements that we determined for a good question typology that is to be used in a QA system:

1. The types of the typology are the answer types (also called targets) for the questions. These types should be automatically identifiable from natural language text. For example: The type of the question *Who is the inventor of television?* is *PERSON*. The answer to he question is a person name, and person names can be automatically identified from natural language text with a reasonable accuracy.

2. The typology should be such that the questions can be automatically classified according to it. This means that there has to be a feature set based on which the classification is possible. Further, the features must be such that they can be inferred programmatically from the questions.

3. The typology should be generalizable, which means that it should apply to different domains and languages.

The typology is developed for general domain question answering systems such as those evaluated in the TREC[1] and CLEF[2] QA evaluation campaigns. As a starting point, we used the Multieight-04 Corpus [Magnini *et al.*, 2005]. Multieight-04 corpus classifies questions into eight classes. Based on the two-layered question taxonomy of Li and Roth [Li and Roth, 2002] that is created for the questions of TREC 10, we modified the original question typology. As can be seen from the Table 1, the Multieight-04 Corpus contains 8 classes and our typology 18 classes. The figure also lists the class names and their frequencies in the first 314 questions of the 700 questions of the Multieight-04 Corpus. As can be seen from the Table 1, the class other has been split into 11 finer classes. The 7 other classes from the multieight-04 corpus have been taken as such. In addition, the original classification of individual questions into classes has been revised and thus the frequencies of classes are different even though the dataset is the same. For example, the questions *What is Judge Borsellino's first name?* and *What office does Ariel Sharon hold?* are classified as *OTHER* in the multieight-04 corpus and as *PERSON* in the new classification.

[1]http://trec.nist.gov/
[2]http://clef-qa.itc.it/

Fine Typology			MultiEight-04	
Class	**#**		**Class**	**#**
PERSON	69		PERSON	61
LOCATION	50		LOCATION	48
MEASURE	45		MEASURE	44
ORGANIZATION	44		ORGANIZATION	37
TIME	38		TIME	39
MANNER	20		MANNER	21
SUBSTANCE	8		OTHER	49
OBJECT	7		OBJECT	15
ABSTRACT	7			
ORGANIC	6			
UNIT	4			
ACTION	4			
EVENT	4			
NATIONALITY	3			
MEDICAL	2			
COLOR	1			
LANGUAGE	1			
AWARD	1			

Table 1: The distribution of the first 314 questions from the Multieight-04 Corpus over the fine typology proposed and the coarse typology originally in the corpus.

3 The Features

The second requirement for a good typology that is listed in Section 2 is that in order to be able to automatically classify the questions according to a typology, there has to be a suitable feature set. Further, this feature set must be such that the features for a given question can be inferred programmatically.

3.1 Transformation of Questions into Features

The construction of a typology and of a feature set that can be used to automatically classify questions according to the typology would be a relatively easy task if the questions could be manually transformed into lists of features. However, in a QA system, the incoming natural language questions must be automatically transformed into lists of features. Thus, the features have to be such that they can be programmatically inferred from the question. According to our initial studies, the set of seven features that we propose is such. The features consist of lemmatized words, part-of-speech (POS) tags, punctuation marks, semantic tags and target tags. The semantic tags used are the following: *country, capital, language, color, profession, person, award, organization, unit, nationality, event, measure, time, named entity, location*. The target tagset consists of the following tags: *location, nationality, person, measure, organization, time, medical, manner, award, event, color, language*. Before transforming the question into features, it is lemmatized and marked with POS tags, semantic tags and target tags. The order of application of the above phases is important as the subsequent phases use information from previous ones.

In detail, the transformation of a question into a list of features involves the following steps:

1. Lemmatize and tag the question with POS tags using a parser from Connexor[3].
2. Tag the question with semantic tags based on word lists, WordNet[Fellbaum, 1998] and regular expressions containing words, POS information and semantic tags.
3. Tag the question with target tags using regular expressions consisting of question words, semantic tags and POS information.
4. Tokenize the question. A token can be a multiword entity tagged by a semantic or target tag or a single word.
5. Form the features. The features First, Second, Third, Fourth, Fifth and Last token describe the corresponding tokens of the question. Feature extraction from questions is done in three different ways:

Features First token, Second token and Last token: If the token is a semantic or target tag, the feature value is this tag. Otherwise, the token is a single word, and its lemmatized form is the feature value.

Features Third token, Fourth token and Fifth token: If the token is a semantic or target tag, the feature value is this tag. If the token is an open class part-of-speech word, i.e. an adjective, an adverb, a substantive or a verb, the POS tag is the feature value. Otherwise it is the word. If the token is empty, the feature value is *NIL*.

Feature Target Tag: If the question contains a target tag, it is the value of this feature. Otherwise the value is *NIL*.

3.2 Experiments with the Proposed Feature Set

We applied the above described transformation of natural language questions into lists of features to a dataset consisting of the 314 first questions of the Multieight-04 Corpus. For each of the seven features, the ten most frequent values as well as the number of different values in the dataset are given in table 2. The features *First token*, *Second token* and *Last token* are open class which means that they don't have a predefined value set. The rest of the features are closed class and thus they have a predefined value set.

The first six features describe the tokens found at different positions of the sentence: the first, second, third, fourth, fifth and last tokens. The last feature can refer to any token of the sentence and it is *NIL* in 63 % of the questions. Tokens can be single words, word sequences or punctuation. For example, the question

```
What <TT:time>year</TT:time> was
<ST:person>Thomas_Mann</ST:person>
awarded <ST:award>the_Nobel_Prize</ST:award>?}
```

has the 6 tokens *what*, *year*, *Thomas Mann*, *awarded*, *the Nobel Prize*. (TT:time is a target tag and ST:person and ST:award are semantic tags.) The values for the features describing the third, fourth and fifth tokens may also be *NIL*. This is the case if the question is segmented only into 3, 4 or 5 tokens, i.e. it has less than 3, 4 or 5 words or tokens. For example, the question *Who is Shimon Peres?* is represented as

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

Name	Open	Values and frequencies	#
First token	yes	what(120), who(59), how(57), where(23), when(16), name(13), in(12), on(5), tell(2), give(2)	14
Second token	yes	be(117), do(32), many(30), what(14), tim(12), a(11), org(11), mea(9), loc(8), can(6)	53
Third token	no	NIL(43), the(68), V(57), N(30), person(16), ne(13), A(11), loc(10), tim(9), a(5)	37
Fourth token	no	NIL(97), V(60), N(33), the(15) profession(13), person(13), A(11), ne(7), NUM(6), of(6)	35
Fifth token	no	NIL(138), of(26), N(23), v(23) the(12), EN(12), A(8), ADV(8), person(7), ne(6)	35
Last token	yes	ne(28), organization(24), person(22), county(21), time(17), profession(11), location(11), locate(10), world(7), org(6)	122
Target tag	no	nil(198), org(26), loc(26), tim(23), mea(11), per(11) nat(3), med(3), uni(1), man(1)	19

Table 2: The proposed set of seven features. For each feature, the table lists its name, openness, ten most common values and the total number of possible values.

Features	18 Classes	8 Classes	7 Classes
Baseline 6 features	71,3 %	71,3 %	78%
Semantic 7 features	80,3 %	83,1 %	85,3 %

Table 3: Classification accuracy using three different typologies and feature sets.

who, be, NIL, NIL, NIL, person, NIL. The question has only three tokens and no target tag.

4 Evaluation

In order to evaluate the typology and the feature set constructed, we performed six experiments using three different typologies and two different feature sets. The classification was performed using the C4.5 [4] decision tree [Quinlan, 1993] induction and classification algorithm. The accuracy figures reported in Table 3 are obtained by 10-fold cross-validation [Duda *et al.*, 2001].

The typology containing 8 classes is the original typology of the Multieight-04 Corpus. The typology of 7 classes is the Multieight-04 Corpus typology where the classes *OTHER* and *OBJECT* have been merged. The finer typology is the one described in Section 2. The baseline feature set consists of six features: the first, second, third, fourth, fifth and last words of the question. The features first, second and last word are the lemmatized forms of the corresponding words. The features third, fourth and fifth word are either the POS tags or the lemmatized forms of the corresponding words. If the word belongs to an open word class (i.e. verb, noun, adjective or adverb) or if it is a numeral, the feature is its POS tag. Otherwise it is the lemmatized form of the word. The semantically enhanced feature set consisting of seven features is the one described in Section 3.2. The results are shown in table 3.

The semantic features clearly enhance the classification performance. The fact that the classification using also the semantic features into 18 classes is less accurate than classification into 8 or 7 classes might be due to the small number of training examples: 284 or 283 training examples versus 30 or 31 testing examples. For some classes, the whole dataset of 314 contains only one example. Detailed classification figures are shown in Tables 4 and 5. In general, the common classes are classified very accurately, on the expense of the rare classes. Experiments using a similar feature set and typology as the baseline feature set and the typology of 8 classes described above, have been performed on Finnish questions[Aunimo and Kuuskoski, 2005]. The accuracy of the classifier has been about 71% which is very close to the 71,3% accurasy reported here.

Real class	**# Classified as**								
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
a PERSON	64			3				2	
b LOCATION	1	43		5					
c MEASURE			42	2					
d ORGANIZATION	4			40					
e TIME	2				36				
f MANNER	1		1	1		17			
g SUBSTANCE	1			3			4		
h OBJECT	1			4				1	
i ABSTRACT		1		2				2	1
j ORGANIC	2			2			1		
k UNIT				2		1	1		
l ACTION				1			2		
m EVENT		1	1	1				1	
n NATIONALITY									
o MEDICAL	2								
p COLOR				1					
q LANGUAGE				1					
r AWARD				1					

Real class	**# Classified as**								
	j	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**
a PERSON						1			
b LOCATION									
c MEASURE				1					
d ORGANIZATION									
e TIME									
f MANNER									
g SUBSTANCE									
h OBJECT			1						
i ABSTRACT			1						
j ORGANIC	1								
k UNIT									
l ACTION		1							
m EVENT									
n NATIONALITY					2	1			
o MEDICAL									
p COLOR									
q LANGUAGE									
r AWARD									

Table 4: Classification results using the semantically enhanced features in detail.

5 Future Work and Conclusions

The work reported here shows that the use of semantic features enhances the performance of a question classifier for QA. In addition, the question typology has an important effect on classifier performance: few classes that are evenly distributed among the questions gives better results than a fine-

[4]The software is available at http://www.rulequest.com/Personal/c4.5r8.tar.gz

Class	Accuracy %
PERSON	94,1,
LOCATION	87,8
MEASURE	93,3
ORGANIZATION	90.9
TIME	94,7
MANNER	89,5
SUBSTANCE	50,0
OBJECT	14,3
ABSTRACT	14,3
ORGANIC	16,7
UNIT	0
ACTION	0
EVENT	0
NATIONALITY	33,3
MEDICAL	0
COLOR	0
LANGUAGE	0
AWARD	0

Table 5: Classification accuracy by class using the semantically enhanced feature set.

grained typology with a skewed distribution. However, natural language questions that are posed to a QA system tend to need a finegrained classification where classes are unevenly populated. The typology and the feature set presented in this paper were developed to address just these problems. However, a large body of work remains to be done. First of all, the experiments described above should be done using a bigger dataset because only such a dataset will give more reliable results on the finegrained classification task. Experimenting with a larger dataset will also show how well our semantic tagging and semantic target tagging methods are applicable to unseen data.

In addition to experimenting with a larger dataset, the typology and feature set should be ported to other languages in order to see how generalizable they are. The final evaluation forum for the typology and feature set presented here is a full QA system. A QA system will show how the typology matches the answers to be extracted form unstructured text.

References

[Aunimo and Kuuskoski, 2005] Lili Aunimo and Reeta Kuuskoski. Reformulations of finnish questions for question answering. In *Proceedings of the 15th Nordic Conference of Computational Linguistics*, Joensuu, Finland, May 2005. To appear.

[Duda *et al.*, 2001] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

[Fellbaum, 1998] Christiane Fellbaum, editor. *WordNet : an electronic lexical database*. MIT Press, 1998.

[Hovy *et al.*, 2002] Eduard Hovy, Ulf Hemjacob, and Deep ak Ravichandran. Question/answer typology with surface text patterns. In *Proceedings of the Human Language Technology Conference*, San Diego, USA, 2002.

[Li and Roth, 2002] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

[Li *et al.*, 2004] Xin Li, Dan Roth, and Kevin Small. The role of semantic information in learning question classifiers. In *Proceedings of the 1st International Joint Conference on Natural Language Processing*, Hainan Island, China, 2004.

[Magnini *et al.*, 2005] B. Magnini, A. Vallin, C. Ayache, G. Erbach, A. Peñas, M. de Rijke, P. Rocha, K. Simov, and R. Sutcliffe. Overview of the CLEF 2004 Multilingual Question Answering Track. In C. Peters, P. D. Clough, G. J. F. Jones, J. Gonzalo, M. Kluck, and B. Magnini, editors, *Multilingual Information Access for Text, Speech and Images: Results of the Fifth CLEF Evaluation Campaign*, volume 3491 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[Suzuki *et al.*, 2003] Jun Suzuki, Hirotoshi Taira, Yutaka Sasaki, and Eisaku Maeda. Question classification using HDAG kernel. In *Proceedings of the Workshop on Multilingual Summarization and Question Answering, held in conjunction with the 4 1th Annual Meeting of the Association for Computational Linguistics.*, 2003.

Recognition of alternation paraphrases: a robust and exhaustive symbolic approach

Marilisa Amoia
Dept of Computational Linguistics
University of the Saarland
Saarbrücken Germany
amoia@coli.uni-sb.de

Claire Gardent
CNRS/Loria
Campus Scientifique BP 239
54506 Vandoeuvre-les-Nancy, France
claire.gardent@loria.fr

Abstract

In this paper we show how by incorporating linguistic knowledge such as that encoded in Verbnet and WordNet in a shallow parser like XIP, it is possible to build a robust semantic parser which can cope with paraphrastic constructions involving alternations and/or lexical synonymy. In the future, we want to extend the parser taking into consideration more complex cases of paraphrastic constructions.

Topics: use of language resources for reasoning in question-answering.

1 Introduction

A salient feature of natural language is that it allows paraphrases that is, it allows different verbalisations of the same content. Thus although the various verbalisations in (1) may have different pragmatic or communicative values, they all share a core semantic content, the one approximated by a traditional montagovian compositional semantics.

(1) a. This key opens the safe.
The safe opens with this key.
b. The laboratory merges with the firm.
The laboratory and the firm merge.
c. Jean hit the wall with a stick.
Jean hit the stick on the wall.
d. I give books to John.
I give John books.

Linguists have long noticed the pervasiveness of paraphrases in natural language and attempted to caracterise it. Thus for instance Chomsky's "transformations" capture the relation between one core meaning (a deep structure in Chomsky's terms) and several surface realisations while [Mel'čuk, 1988] presents sixty paraphrastic rules designed to account for paraphrastic relations between sentences.

More recently, work in information extraction (IE) and question answering (QA) has triggered a renewed research interest in paraphrases as IE and QA systems typically need to be able to recognise various verbalisations of the content. Because of the large, open domain corpora these systems deal with, coverage and robustness are key issues and much on the work on paraphrases in that domain is based on automatic learning techniques (see for example [Barzilay and Lee, 2003] and [Shinyanma *et al.*, 2002]) .

In this paper, we investigate an alternative research direction and present a symbolic treatment of paraphrases which (for the moment) is restricted to alternations and/or lexical synonymy. For this type of paraphrases, we present a *robust* and *wide coverage* system which, by integrating the detailed knowledge of alternations and of lexical synonymy encoded in VerbNet and WordNet in a cascaded finite state parser (Xerox Incremental Parser henceforth, XIP), assigns two such paraphrases one and the same semantic representation.

The paper is structured as follows. Section 2 presents the linguistic resources used and the type of semantic representations produced by our approach. Section 3 shows how we extend XIP to integrate VerbNet and WordNet information so as to assign paraphrases the same semantic representation. Section 4 presents an evaluation of the system based on a set of annotated examples extracted from VerbNet. Section 5 concludes with pointers for future work.

2 Lexical resources: VerbNet and WordNet

VerbNet is a broad-coverage domain-independent verb-lexicon [Kipper *et al.*, 2000] which encodes syntactic and semantic information for about 4000 english verbs. The verbs are organised in classes which refine Levin's classes [Levin, 1993] and capture generalisations about the regular association between syntactic and semantic verb properties.
More specifically, a VerbNet class has the following components: (i) the set of english verbs belonging to that class each verb being annotated with the WordNet meaning(s) relevant to that class, (ii) the set of theta roles which can be mapped to the arguments of these verbs, (iii) selectional restrictions on the arguments and (iv) a set of frames consisting of an indentifier, an example, a syntactic description and a decompositional semantics common to all verbs in that class. Figure 1 pictures the VerbNet frame for the class meander-47.7.

For the treatment of paraphrases, the information contained in VerbNet is useful for several reasons. First, VerbNet documents the alternations of each verb such as those given in 1 and those illustrated by Figure 1 e.g., :

(2) a. The river meanders through the valley

Members: *cascade(1), climb(4), crawl(), cut(), drop(), go(7), meander(1), plunge(), run(3), straggle(2), stretch(1), sweep(5), tumble(), turn(), twist(), wander(4), weave(4), wind(1 2)*
Thematic Roles and Selectional restrictions: Location[+concrete] Theme[+elongated]
Frames: Intransitive (+ path PP) "The river runs through the valley" Theme V Prep[+path] Location Prep(during(E),Theme,Location) exist(during(E),Theme)
Locative Inversion "Through the valley meanders the river" Prep[+path] Location V Theme Prep(during(E),Theme,Location) exist(during(E),Theme)
There-insertion "There meanders through the valley a river" there V Prep[+path] Location Theme Prep(during(E),Theme,Location) exist(during(E),Theme)
There-insertion "There meanders a river through the valley" there V Theme Prep[+path] Location Prep(during(E),Theme,Location) exist(during(E),Theme)

Figure 1: VerbNet representation of the *meander-47.7* class

b. Through the valley meanders the river
c. There meanders through the valley a river
d. There meanders a river through the valley

By constructing the group of arity preserving alternations a verb participates in, it is then possible to identify all meaning preserving alternations a verb can occur in. Further since VerbNet associates with each verb class a thematic grid and a decompositional semantics, it becomes possible to develop a parser which, based on this knowledge can build identical semantic representations for alternations paraphrases. Thus for instance, using the thematic role information associated in VerbNet with the *meander-47.7* class, all sentences in 2 can be assigned the same basic semantic representation:

River(R) & Valley(V) & Meander(E) & Location(E,V) & Theme(E,R)

Another feature of VerbNet which makes it attractive for the treatment of paraphrases is its linking with WordNet. As indicated above, the verbs of a VerbNet class are annotated with the WordNet meaning(s) relevant to that class. Now because WordNet records the synonyms of a given word usage, this linking between VerbNet and WordNet also gives us access to synonymic paraphrases. In the case at hand for instance, the set of synonyms retrieved from WordNet for meaning 1 of *meander* is *weave, wind, thread, meander, wander*. By integrating this information in a parser lexicon and combining it with the knowledge of alternations given by VerbNet, we can thus obtain a parser wich assign one and the same semantic representation to the following sentences.

(3) a. The river meanders through the valley
b. The river weaves through the valley
c. Through the valley weaves the river
d. There weaves through the valley a river
e. There weaves a river through the valley

3 Extending XIP to recognise VerbNet alternation paraphrases

In what follows, we show how the knowledge encoded in VerbNet can automatically be integrated in a robust parser thereby supporting the recognition of the set of alternation and/or lexically synonymic paraphrases covered by VerbNet.

3.1 XIP

XIP ([Ait-Mokhtar *et al.*, 2002]) is a rule-based, shallow parser based on finite state techniques that garantees robustness by adopting incrementality: the input sequence is processed by a layered grammar, each grammar layer being applied sequentially. As the input is processed, it is either enriched or left unchanged – the output is the input sequence as annotated by the sequential application of the rules from the different layers. By ordering the grammar rules appropriately, data which is either infrequent or incorrect can therefore be handled.

We use XIP version 3.10 (2000-2001) as developed at the Xerox Research Europe. This version includes the NTM tokenizer and morphological analyser based on finite states technology, the HMM statistical POS tagger and a grammar for English. The parser is also reasonably efficient running at a speed of 1 300 words per second on a Pentium II 50.

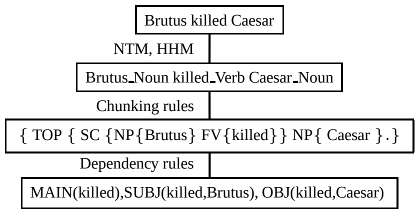

Brutus killed Caesar
NTM, HHM
Brutus_Noun killed_Verb Caesar_Noun
Chunking rules
{ TOP { SC {NP{Brutus} FV{killed}} NP{ Caesar }.}
Dependency rules
MAIN(killed),SUBJ(killed,Brutus), OBJ(killed,Caesar)

Figure 2: XIP representation of the sentence *Brutus killed Caesar.*

The grammar for English includes two types of subgrammars namely, chunking and dependency grammars. The **chunking grammar** describes constituency structure and consists of layered groups of chunking rules which apply either to partially ordered bags of nodes or to ordered subsequences. The following rule, for example, builds an NP chunk if the list of nodes contained in the current XIP stack contains a determiner which is followed (also not immediately ?*) by a noun

15 > NP=Det, ?*, Noun.

The **dependency grammar** supports the specification of (functional, thematic, semantic, anaphoric, etc.) relations between words or chunks and is based on the (layered) specification of groups of dependency rules of the form:

| ⟨subtree_pattern⟩ |
if ⟨ conditions ⟩
⟨dependency_term ⟩

where `subtree_pattern` is a tree matching expression describing structural porperties of part of the input tree, `conditions` is any Boolean expression built up from dependency terms and `dependency_term` is a term of the form `name<flist>(a1, ..., aN)` with `name` the name of a dependency relation, `flist` a list of features associated with that dependency relation and `a1, ..., aN` the relation arguments.

The following dependency rule for example,
| SC(?*, FV[trans:+]{?*,#1[last:+]}}, NP[time:]{?*, #2[last:+]}|
$VCOMP[dir:+](\#1,\#2)$

recognises a VCOMP dependency between a transitive verb (#1), head of a finite verb chunk FV within a clause chunk (SC), and the head (#2) of an NP chunk with negative time feature.

3.2 Incorporating Verbnet into XIP

To support a robust and large scale treatement of alternation paraphrases, we integrated VerbNet information into a XIP lexicon and we specified dependency rules which use this information together with the VerbNet set of lexico-syntactic patterns in order to assign a given input sequence the thematic grid assigned to that sequence by VerbNet.

3.3 The verb lexicon

The XIP verb lexicon associates each verb with its VerbNet class and with the WordNet Synset identifier corresponding to the relevant usage of the verb in that VerbNet class. For instance, the verb *meander* is assigned the following lexical entry: *meander*: `verb`+=[meander-47.7, pred=c01828635].
The *VerbNet class* will be used both to guide syntactic parsing and to support semantic construction. Further, the VerbNet semantic class is used in the rule to specify the syntax/semantic interface of the verb.

The *WordNet synset information* on the other hand serves to group together synonyms. That is, all verbs in a VerbNet class which belongs to the same WordNet synset will be assigned the same semantic representation.

The VerbNet class and synset assignment was made automatically on the basis of both VerbNet and WordNet information. At present, the lexicon contains 4225 verbs corresponding to 2779 WordNet synsets and 352 VerbNet verb classes. However since word sense disambiguation is not integrated in XIP, we only consider the most frequent meaning of a verb. In future, we intend to bypass this limitation by tagging the input verbs with meaning identifiers.

3.4 Semantic construction

To assign identical semantic representation to alternation paraphrases, we augment the XIP grammar with a set of *thematic grid dependency rules*. These rules assume as input the output of the existing XIP parser that is, a representation of the input including both constituency and grammatical functions (subject, object, etc.) information. Based on this information, a thematic grid (dependency) rule identifies a given VerbNet pattern and specifies a mapping between syntactic and thematic argument.

Suppose the sentence to be parsed is:

(4) The river meanders through the valley

As indicated in section 2, VerbNet associates to that usage of the verb *meander* the grid of theta roles Theme V Prep[+path] Location, where the canonical subject of the verb is assigned a Theme role and a prepositional object introduced by a path denoting preposition a Location role.

In the XIP framework, such a specification is captured by the following (simplified) dependency rule:

```
if( (  MAIN(#1[coil9_6])
    || MAIN(#1[coil9_61])
    ...
    || MAIN(#1[meander47_7])
    ...
    || MAIN(#1[waltz51_5])
    )
     & VDOMAIN[passive:~](#1,#11)
     & SUBJ-SEM(#1,#2) & ~OBJ(#1,?)
     & VMOD[post](#1,#4) & PREPD(#3,#4)
     & (#3[vnpath])
  )
EVENT(#1),Theme(#1,#2),Location(#1,#4).
```

In words: if the main verb #1 (which should not be in passive mode) is associated via lexical lookup with one of the listed VerbNet classes (and in particular with the *meander47_7* class) and it has no object but has a subject #2 and a postposed verb modifier #4 introduced by a path denoting preposition #3, then the semantic representation produced is EVENT(#1), Theme(#1,#2),Location(#1,#4).

As this rules applies to the input sequence (4), the following representation is output where indeed the correct thematic representation has been produced:

EVENT(running_C01870464_MEANDER47_7:+)
LOCATION(running_C01870464_MEANDER47_7:+,valley)
THEME(running_C01870464_MEANDER47_7:+,river).

The extended XIP grammar has 425 semantic rule ordered by specificity, the most specific rules occurring first and the least specific last. Since within one grammar layer only the first applicable rule is used, this ensures that the syntactic configuration captured by the rule that is executed is indeed the most appropriate.

This rule ordering also allows an appropriate treatment of the difference between adjuncts and subcategorised PPs. Being more specific, the rules describing verbs taking prepositional arguments will be tested before the general rules describing the combination of verbs with adjuncts and so will be preferred in case they can apply. Consider the two sentences in 5.

(5) a. Sharon shivered from fear.
EVENT(C01834682),CAUSE(C01834682,fear),
EXPERIENCER(C01834682,Ann)

b. Sharon breakfasted in the garden.
EVENT(C01149559), AGENT(C01149559,Ann)

In the first sentence, the PP is described in VerbNet as an element of the subcategorisation frame of the verb *shiver* which is mapped to the CAUSE role. In contrast, in the second sentence the PP is treated as an adjunct and is not assigned a thematic role.

To define the specificity ordering over the thematic rules, we first generalised the syntactic frames to 68 more general templates by ignoring prepositional and selectional information. The resulting set of templates was then organised in a

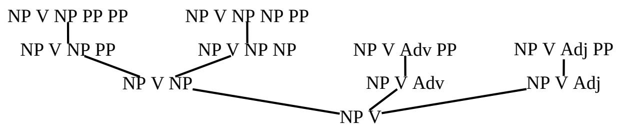


Figure 3: Hierarchy of syntactic patterns in Xip

hierarchy (cf. Figure 3) which was then used to automatically order the XIP thematic rules.

To cover unknown input and more specifically verbs whose VerbNet class is not given in the lexicon, we introduce an additional postprocessing step which performs a default thematic grid assignment on the basis of the 68 abstract rule templates used to specify rule ordering. For instance, suppose that the input sentence is 6 and that the VerbNet class for "stand" is not given in the lexicon. In such a case, the general rule specifying a syntactic configuration of the form `PP[loc] V NP`, will assign the locative PP an `arg2` role and the NP an `arg1` role thereby producing the given semantic representation.

(6) a. 'On the pedestal stood a statue
EVENT(stood), ARG1(stood,statue), ARG2(stood,pedestal)

More generally, the postprocessing step will assign default underspecified thematic roles to maximal projection phrases occurring in the input on the basis of surface syntax information. Note that the use of underspecified thematic roles renders the obtained semantic representations similar to those assumed by PropBank [Kingsbury *et al.*, 2002].

4 Evaluation

To evaluate the extended XIP parser, we extracted from Verbnet the 1012 example sentences it contains together with their thematic role annotation. We then applied the parser to the resulting set of sentences and automatically compared the thematic grid output by the extended XIP parser with the thematic grid described by the VerbNet annotation. We obtained the following results:

- 71% of the sentences were assigned the correct representation (i.e. the same roles assignment as in VerbNet),
- 15% of the sentences were assigned the correct syntactic pattern but the wrong theta roles because the parser has no ontological restrictions on roles
- 4% of the sentences were assigned a default pattern because either the tagger was not able to recognize the class appartenance of the verb, or the verb class assignment in the lexicon did not allow the syntactic pattern illustrated by the given sentence,
- 10% of the sentences could not be mapped onto a VerbNet pattern because the constituency information delivered by XIP was insufficient.

In sum, the evaluation shows that the robust parser developed deals appropriately with 71% of the VerbNet data and that there is reasons to hope that it can be further improved by incorporating selectional restrictions on roles and by improving the basic constituency grammar.

5 Conclusion

In this paper, we have shown how to integrate the linguistic knowledge of alternations encoded in VerbNet and the verbal lexical synonymy information encoded in WordNet into a robust parser which assigns alternation paraphrases one and the same semantic representation. A first evaluation shows that the parser has an accuracy of 71%.

Current and future work concentrates on (i) improving the current results by improving the grammar and integrating selectional restrictions, (ii) extending the paraphrastic coverage by considering additional paraphrasing mechanisms such as morphoderivational variants, cross categorial synonyms and antonyms and (iii) evaluating the system on real text corpora.

An additional line of research concerns the usability of the existing parser for automatically tagging real text either with VerbNet thematic grid information using the detailed thematic grid dependency rules or with less specific PropBank thematic grid information by resorting to the less detailed rule templates used in the postprocessing step.

Acknowledgments

We would like to thank the Xerox company for making XIP available to us. We would also like to thank the Contrat Plan Etat Région: Ingénierie des Langues, du Document et de l'Information Scientifique, Technique et Culturelle for partially funding the research presented in this paper.

References

[Ait-Mokhtar *et al.*, 2002] S. Ait-Mokhtar, J P. Chanod, and C. Roux. Robustness beyond shallowness: incremental deep parsing. *Natural Language Engineering*, 8(2/3):121–144, 2002.

[Barzilay and Lee, 2003] Regina Barzilay and L. Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of NAACL-HLT*, 2003.

[Kingsbury *et al.*, 2002] Paul Kingsbury, Martha Palmer, and Mitch Marcus. Adding semantic annotations to the penn treebank. In *Proceedings of the Human Language Technology Conference*, San Diego, California, 2002.

[Kipper *et al.*, 2000] Karin Kipper, Hoa Trang Dang, and Martha Palmer. Class-based construction of a verb lexicon. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, July-August 2000. Morgan Kaufmann.

[Levin, 1993] Beth Levin. *English Verb Classes and Alternation, A preliminary Investigation*. The University of Chicago Press, Chicago, 1993.

[Mel'čuk, 1988] I. Mel'čuk. Paraphrase et lexique dans la theorie linguistique sens-texte. *Lexique*, 6:13–54, 1988.

[Shinyanma *et al.*, 2002] Y. Shinyanma, S. Sekine, K. Sudo, and R. Grishman. Automatic paraphrase acquisition from news articles. In *Proceedings of HLT*, 2002.

On the Effective Use of Cyc in a Question Answering System

Jon Curtis, Gavin Matthews, David Baxter
Cycorp, Inc. 3721 Executive Center Drive, Suite 100, Austin, TX 78731.
{jonc,gmatthew,baxter}@cyc.com

Topics: flexibility supported by discourse/user model, new types of questions, reasoning with incomplete knowledge, response generation.

Abstract

We describe a commercial question-answering system that uses AI – specifically, the Cyc system – to support passage retrieval and perform deductive QA, to produce results superior to what each question-answering technique could produce alone.

1 Introduction

This paper describes a working prototype of a commercial question-answering system that uses artificial intelligence in conjunction with NLP-driven passage retrieval in a way that integrates the two markedly different approaches to question-answering and produces better results than either approach could yield alone.[1] MySentient[2] Answers 1.0 draws upon the Cyc system, a large knowledge-base, commonsense reasoning system. We describe three areas in which Cyc's knowledge contributes to the system's overall question-answering ability, both directly in deductive question-answering, and indirectly, by supporting passage-retrieval. In particular, we focus on the use of Cyc for:

1. augmentation of NLP-based passage retrieval by generating NL expansions of key concepts mentioned in a question;
2. answering question types that pose a challenge to passage retrieval methods, such as procedural ("How do I …?") and cost/benefit ("Why should I …?"); and
3. paraphrasing the results of deductive question answering as NL strings for display to an end user.

We close with a discussion of the current limitations of the integrated system and a description of anticipated extensions to the use of Cyc in future versions.

2 Cyc

Cyc is a state-of-the-art artificial intelligence program that has been in development since 1984. Drawing upon the world's largest general-purpose knowledge base of over 164,000 concepts and 3,300,000 facts (rules and ground assertions) relating them[3], Cyc is the only AI program in existence today that can reasonably claim to have some degree of common sense. Cyc's knowledge is represented in CycL, a higher-order logical language based on predicate calculus. Every assertion in Cyc is represented in a context, or *microtheory*, which allows the representation of competing theories. Like ordinary concepts, microtheories are explicitly represented as first-class objects in Cyc, giving Cyc a reflective capability to reason about its own representations. Microtheories form a hierarchy that facilitates knowledge re-use (assertions stored in the most general contexts are always available), and inferential focus (given a query posed in a specific microtheory, other knowledge from sibling or more specific microtheories will not come into play). Cyc's inference engine combines general theorem proving (*e.g.* rule chaining) with specialized reasoning (*e.g.* subsumption and transitivity).

Cyc has been used in commercial web-search systems (*e.g.* HotBot) and in question-answering systems, most recently in a purely deductive system for answering AP chemistry questions, developed in collaboration with Vulcan, Inc. [Friedland *et al*., 2004]. Cyc's rôle in the MySentient system heralds its first appearance in a commercial question-answering system. MySentient makes use of Cyc pervasively, as a means to augment NLP-based QA, as the basis for a deductive QA module, and in other capacities, such as clarification and profiling, that will be touched upon here.

3 MySentient Answers 1.0

MySentient Answers 1.0 is a working question-answering system, designed by MySentient Ireland (R&D) Ltd. of Dublin, Ireland, and implemented in collaboration with Cycorp, Inc. of Austin, Texas, and the Center for Natural Language Processing in Syracuse, New York[4]. MySentient Answers has been the subject of extensive demonstration to interested commercial parties and is expected to be available for public access in the near future.

[1] The work described in this paper was made possible by the financial backing of MySentient.

[2] "MySentient" is a registered trademark of MySentient Ireland (I.P.) Limited.

[3] MySentient uses a carefully-chosen subset of the full Cyc knowledge base with 137,000 concepts and 1,700,000 facts.

[4] See http://www.mysentient.com/, http://www.cyc.com/, and http://www.cnlp.org/

3.1 Architecture

MySentient Answers is based on the *S-Core* architecture, which integrates disparate components into a uniform XML-based interface. Components each receive a *storybook* giving the history of the interaction, and their output is appended to the appropriate element. This design gives some of the flexibility of a blackboard architecture, yet allows some powerful simplifying assumptions.

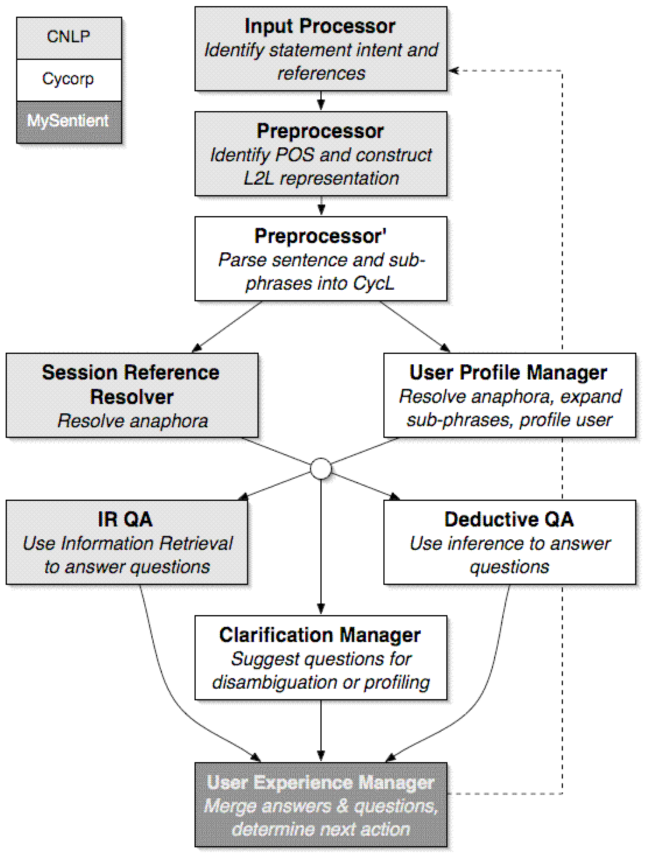

CNLP
Cycorp
MySentient
Input Processor
Identify statement intent and references
Preprocessor
Identify POS and construct L2L representation
Preprocessor'
Parse sentence and sub-phrases into CycL
Session Reference Resolver
Resolve anaphora
User Profile Manager
Resolve anaphora, expand sub-phrases, profile user
IR QA
Use Information Retrieval to answer questions
Deductive QA
Use inference to answer questions
Clarification Manager
Suggest questions for disambiguation or profiling
User Experience Manager
Merge answers & questions, determine next action

Figure 1: MySentient Answers system architecture, showing selected components.

3.2 Question Answering Modules

The S-Core architecture allows any task to be attempted in parallel (or in sequence) by multiple competing modules. In MySentient Answers, there are several question-answering modules: various NLP passage-retrieval modules developed by CNLP, and a deductive question-answering module based on Cyc. All QA modules return NL answers.

CNLP's question-answering capabilities are grounded in a quasi-logical representation – Language-to-Logic, or "L2L" that has proven successful in recent TREC question answering tracks [Diekema, *et al*., 2000].

3.3 Methodology

The Cyc Knowledge Base is essentially open-domain, but deployments of MySentient Answers will be focused on specific domains that reflect the needs and interests of the customer. The final goal of the project is that the customer's domain experts will perform much of this specialization, using a suite of MySentient Authoring Tools. At the current stage of development, these tools are in a rudimentary state; therefore, much of the authoring done in support of the results described in this paper has been simulated by blending prototype tools with the intervention of skilled ontologists. This simulation not only provides a proof-of-concept of the run-time question-answering system, but also permits an informed comparison to be made between the representation capabilities required and the feasible capabilities of the planned authoring tools (see Section 8, "Current Limitations and Future Directions," below). The corpus used for this simulation was provided by the Motley Fool UK, based on its website.[5]

While coverage of a corpus must start with the corpus itself, it is also necessary to concentrate on test queries; Cyc's coverage of the Motley Fool domain therefore had two foci.

The corpus was subjected to automated analysis for noun phrase identification and interpretation, plus extraction of glossary entries. A manual pass of review and correction to ensure broad coverage of the domain followed this.

Known and blind question sets were prepared (by both Cycorp and MySentient) based on the corpus. The known question sets were analyzed by question type (see Question Types of Interest), and strategies for broad coverage of each question type were devised and implemented.

Independently, the quality-assurance team performed daily tests based on the question sets. Results were evaluated on a primitive basis by automatic comparison with a growing set of input/output pairs. Each input/output pair was classified as correct, incorrect, or correct but badly paraphrased. Incorrect results were reviewed with a focus on identifying and resolving the broad class of defect (such as missing or erroneous knowledge) rather than fixing problems specific to particular questions.

The intensive ontological engineering effort for the Motley Fool UK domain was performed over a four-week period, and took 691.25 person-hours. The source corpus was equivalent to about 200 pages of text and a total of 286 test questions were prepared for that domain. The NLP-based components also underwent a training process against the Motley Fool corpus; however, this training was done independently of the simulated authoring/ontological engineering effort done for Cyc. As a limited test of how MySentient Answers benefits from integrating both NLP and deductive approaches to question-answering, MySentient prepared 132 questions that were posed to the system, and for each question, each QA system was scored on whether it produced a satisfactory answer. In borderline cases, a half-point was awarded.

Overall, the multiple CNLP QA modules scored 63% and the Cyc DQA module scored 34%. This asymmetry is to be expected because of the relative maturity of NLP systems in the QA domain. The federation of QA modules (taking the

[5] "The Motley Fool" is a registered trademark of Motley Fool, Inc. See http://www.fool.co.uk/ for the corpus website.

high score for each question) scored 79%, a significant improvement over the individual QA modules. In several cases, both modules gave usefully different answers that, taken together, form a rounded answer to the user's question. Two interesting examples are:

In response to "How do I protect myself from credit card fraud?" the CNLP module returned a passage that described online fraud guarantee and internet delivery protection, whereas the Cyc module returned sentences advocating PIN secrecy, comparing receipts, and reporting credit card loss.

In response to "Should we get married or live together?" the CNLP module returned a passage about the legal rights accorded to married couples, whereas the Cyc module returned sentences describing the economic benefits of cohabitation and the tax benefits of marriage.

It is important to note that this test does not isolate all federation factors. In particular, the Cyc-based DQA uses on upstream CNLP modules for the identification of noun and verb phrases, while the CNLP QA modules make use of Cyc-derived expansions. Nevertheless, the limited test described supports the view that the use of deductive question answering in tandem with a NLP QA system can significantly boost system effectiveness.

4 Discourse Modeling

Cyc's contributions to MySentient Answers are grounded in a discourse model, generated on the fly. This model stores information from a user's session, in CycL, so that Cyc can reason over it. Each discourse model is associated with a microtheory structure that is defined for each user and can be extended across sessions to preserve useful information about the user and his or her interactions with the system.

The most general microtheory in the structure is the user profile, which contains data intended to persist between sessions, and so is available for inference any time the user logs in. Though not currently a mature feature of the system, the user profile gathers information that can be used to improve the quality of subsequent interactions. For example, if the user asks for recommendations for Mazda truck accessories during one session, and later asks for directions to service stations for "my vehicle," the system should be able to use the knowledge, gained during the previous session, that the user has a Japanese vehicle. These features are still prototype technologies, and are therefore described in the "Current Limitations and Future Directions" section of this paper. The ability to profile the user is an exciting distinguishing feature of the MySentient system.

For NL parsing and generation, the user-session model includes a lexical microtheory sub-structure that was originally developed for and used in the DARPA-funded, Cyc-based KRAKEN knowledge-acquisition system [Panton, *et al.*, 2002]. The most general microtheory of this sub-structure is a user-specific lexicon, from which the contents of the appropriate general Cyc lexicon are visible. Though British English is the assumed default language for the test domain, the system is can determine the appropriate language on a per-session basis. Once that language is determined, an assertion is added to the Cyc KB linking the user-specific lexicon to the general lexicon for that language. So for British English, Cyc will generate a sub-context link from the user-specific lexicon to #$BritishEnglishMt, making its data about British spellings, common words, etc., available. For more information on the representation and use of lexicons in Cyc, see [Burns and Davis, 1999].

Directly below the user-specific lexicon are two more lexical microtheories, which are used for inference by Cyc's parser and NL generation. By design, the parsing and generation microtheories are siblings; user-specific lexical information, such as information about how the user referred to a concept, is stored in the user-specific lexicon, where it is available for both parsing and generation.

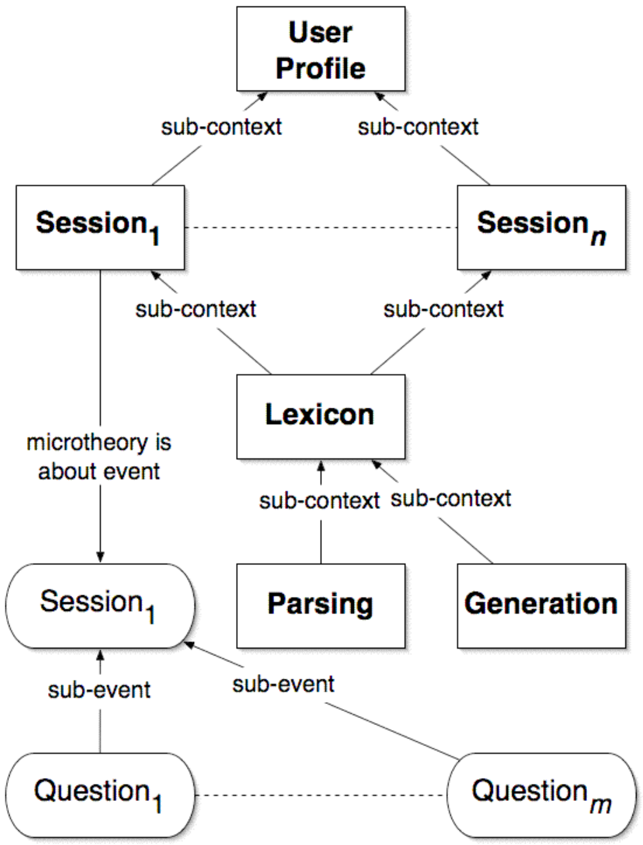
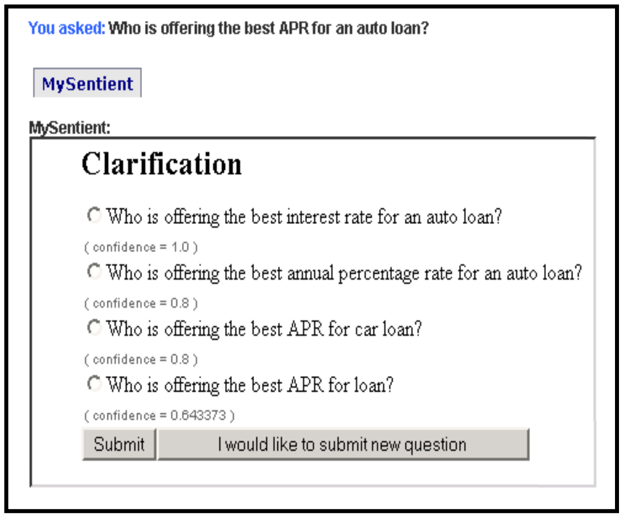


Figure 2: Part of discourse model. Boxes show microtheories; ovals show events. Not shown are user-independent super-contexts and other ontology.

Between the user profile and the user-specific lexicon are one or more session microtheories, containing the vast majority of user-specific assertions. Sessions themselves are explicitly represented as events, allowing their internal, temporal structure to be modeled. Each session revolves around the user asking one or more questions; these question-asking events are also explicitly represented as proper sub-events of the session to which they belong. Information about the user's question, such as the grammatical features of the constituent phrases (e.g., "market indicators" is a bare plural expression, "the stock market" is definite singular), the CycL semantics for the question (when available), and hypotheses about what the question is about, are all related to the user's question-asking event, using special discourse modeling vocabulary that can be leveraged by the Cyc inference engine to support query expansion, deductive question-answering, and natural language generation. Those processes are described in the following sections.

5 Query Expansion

Query expansion is the process of altering an input question, or a (quasi-)formal representation thereof, typically by adding or replacing terms. The modifications a query undergoes during the expansion process is determined by analysis of the query terms. *E.g.* "AIM" might have the *expansion* "Alternative Investment Market." Expansions can be used to focus a query, often by contributing to a set of query words, or the categorization of its answer-type. The most common approach to query expansion seen in the literature is to leverage a dictionary-based program, such as WordNet, to produce syn-sets, hypernyms and hyponyms [Hovy, *et al.*, 2000], or to find appropriate, hard-to-predict part-of-speech variations for noun compounds ("attorneys general," and not "attorney generals" as an expansion of "attorney general") [Bilotti, 2004], or to find stemming information for query-words [Bacchin and Melucci, 2004].


You asked: Who is offering the best APR for an auto loan?

MySentient

MySentient:

Clarification

Who is offering the best interest rate for an auto loan?
(confidence = 1.0)
Who is offering the best annual percentage rate for an auto loan?
(confidence = 0.8)
Who is offering the best APR for car loan?
(confidence = 0.8)
Who is offering the best APR for loan?
(confidence = 0.643373)

Submit | I would like to submit new question

Figure 3: MySentient's Clarification module uses expansions to suggest re-phrasings of the user's question.

A limiting feature of these approaches is a near-complete reliance on lexical methods: Only the relationships between *terms* are considered; the *semantic* relationships between *concepts* are not.[6] MySentient's expansion-generation is a departure in this regard. Key phrases in the user's question are identified, translated into CycL, and placed in the discourse model. The User Profile Manager then reasons over this formal representation of the *meanings* of query-words to identify concepts that form the semantic basis for expansions. Other modules, such as the Deductive Question Answering Module and the Clarification Manager can also use these phrase-level translations.

Several expansion strategies are explicitly represented in the Cyc Knowledge Base, each defining criteria that a concept must meet to be used as the semantic basis for expansions. When a strategy is executed, inference seeks concepts that meet the relevant criteria. The resultant bindings are then sent to a Natural Language generation function that generates NL for those bindings. The generated strings, along with the strategy used and a confidence, are outputted by the User Profile Manager as proposed expansions for the original input term.

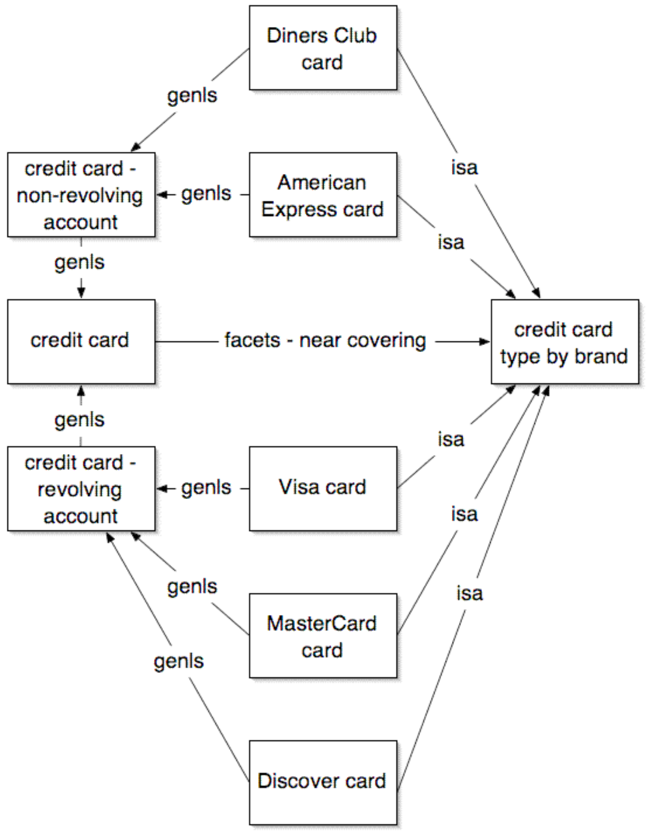
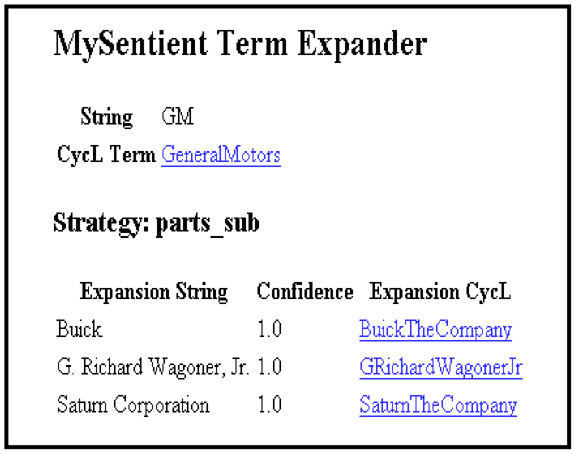


Figure 4: Part of Cyc's credit card ontology with faceting. The rightmost node is a second-order collection; all other nodes are first-order collections.

Cyc is agnostic as to how the expansions should be used; for example an NLP question-answering module might treat them as conjuncts in a Boolean rewrite, or they might be used in answer-type classification. Empirical evaluation of various strategies — based on CNLP's determination of their usefulness for passage-retrieval by their technology, and MySentient's exacting wall-clock performance criteria — led to the decision to include two strategies in MySentient Answers 1.0: A synonym-generation strategy that calls

[6] At least not directly. Some techniques (*e.g.* LSA) approximate semantic "closeness" by measuring co-occurrence in a corpus. Semantic closeness, however, is not a first-order semantic relationship; the authors contend that the semantic relationships that explain semantic closeness are more valuable for expansion.

upon the Cyc Lexicon to simulate traditional NLP-based query expansion[7], and a "classification" strategy[8] that uses definitional, or "type" information to generate expansions useful for categorization. Given an input string, "APR," the User Profile Manager uses the synonym-generation strategy to return the unabbreviated "annual percentage rate," and the classification strategy to return the more general "interest rate." Although the use of expansions by the NLP question answering module is not visible to the end user, the results of expansions are nevertheless discernable in MySentient Answers 1.0: MySentient has implemented a simple Clarification module prototype that substitutes high-


MySentient Term Expander

String GM
CycL Term GeneralMotors

Strategy: parts_sub

Expansion String	Confidence	Expansion CycL
Buick	1.0	BuickTheCompany
G. Richard Wagoner, Jr.	1.0	GRichardWagonerJr
Saturn Corporation	1.0	SaturnTheCompany

Figure 5: For "GM", we get the CycL term #$GeneralMotors. The "parts" of this term include two sub-divisions and the CEO.

confidence expansions into the original query, and presents them to the user as proposed re-phrasings. For example, if a user asks, "Who is offering the best APR for an auto loan?" the Clarification manager will offer as re-phrasings, "Who is offering the best interest rate for an auto loan?", "Who is offering the best annual percentage rate for an auto loan?" and "Who is offering the best APR for car loan?"[9]

As noted above, other Cyc-based expansion strategies are available, but are turned off by default. Nevertheless, these are worth describing as examples of how the space of possible expansions is extended through a semantic approach. Among these strategies is a "conceptually related" algorithm that finds closely associated concepts, using significant semantic relationships. For example, asked to expand "asthma," the conceptually related strategy will return "lung" because asthma is known to be a specialization of *lung disease*, and lung diseases are ailments that affect the lungs. This same strategy will also return "medical insurance," because medical insurance provides coverage for medical problems, and asthma is a kind of medical problem.

Another strategy is the "specializations" strategy, that, when given a term that maps to a collection, will return salient specializations of that collection. For example, given the input string "credit card," this strategy will return the names of the various brands of credit card, such as "VISA," "MasterCard," "American Express," and "Discover." Cyc draws on the knowledge that the collections representing these cards are all instances of a higher order collection, #$CreditCardTypeByBrand, that facets #$CreditCard by the various brands. By restricting the search to collections that are part of a faceting hierarchy, the strategy is able to avoid returning less helpful specializations of #$CreditCard (e.g., "stolen credit card," "credit card printed at a factory in London,") that the system might know about, but are more or less arbitrary sub-collections, and not part of a more intuitive conceptual hierarchy.

Finally, Cycorp has implemented two parts-based strategies, *parts_super* and *parts_sub*, that use Cyc's knowledge of the structure of types and particular individuals to return expansions that, in the *parts_super* case, reflect that concepts placement in a structure, and in the *parts_sub* case, reflect that concept's internal structure. For example, given the input string "GM" and using the *parts_sub* strategy, Cyc returns "Buick" and "Saturn Corporation," two subdivisions of General Motors, as well as "G. Richard Wagoner, Jr.", the current CEO of GM.

6 Deductive Question Answering

Like the User Profile Manager, the Cyc-based Deductive Question Answering module (DQA) uses explicitly represented strategies. The highest-confidence strategy queries the knowledge base with a CycL representation of the user's question. As such, this strategy depends on the total success of the Natural Language Preprocessor module (based on the parsing technology described in [Panton, *et al.*, 2002]), in mapping English to CycL. In cases where syntactic or semantic ambiguity in the user's question results in competing CycL interpretations, simple heuristics (such as preferring the least complex CycL expression) are used to rank the interpretations. The top-ranked interpretation is identified in the discourse model as the default interpretation of the user's question, while the other candidates are recorded as possible interpretations, available for later clarification.

The DQA module retrieves that CycL interpretation and uses it to query the Cyc Knowledge Base. (If there are no sentential interpretations, DQA moves on to the next strategy.) In Cyc, a query consists a CycL formula and several query-properties, including: a *microtheory*, or context, from which to ask the query; the *temporal index and granularity* (do we want bindings that satisfy the formula now, ever, all

[7] The synonym strategy is an exception to the rule that strategies first identify relevant CycL terms and then paraphrase each.; instead, all synonyms are generated from a single CycL term that is the best interpretation of the user's phrase.

[8] The classification strategy differs from other strategies in that it uses a semantic closeness metric to assign confidences to its outputs. Thus as an expansion of "auto loan," "loan," being closer in Cyc's generalization hierarchy, gets a higher confidence than "obligation," a more distant (and abstract) generalization.

[9] The agreement error reflects the simplicity of the Clarification module's current substitution algorithm. That the input phrase "an auto" has an indefinite article is recorded in Cyc's discourse model; the module can be "smartened" to use this information.

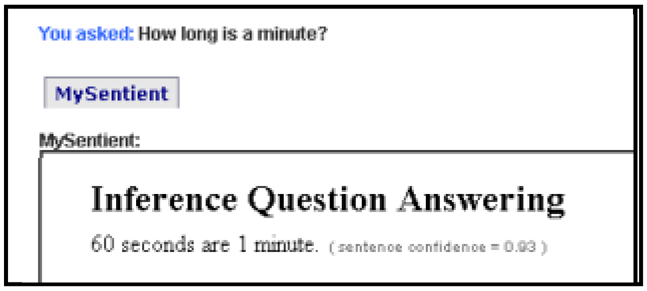
the time, *etc.*?); a limit on the number of *transformations*, or inference steps that chain rules; and a *time* limit.

For DQA queries: the microtheory is the user's session; the temporal index and granularity are "any time", allowing the system to find temporally-qualified answers; the number of transformations and the like are determined by the nature of the question (primarily the main predicate); and the time is distributed from a (configurable) 30 second budget.

6.1 Question-types of Interest

The problem of parsing arbitrary English to a formal, logical representation is only partly solved. Thus a deductive question answering system that accepts arbitrary NL input will necessarily be limited both in the expressiveness of the formal language (the vocabulary), and in inherently difficult problems in resolving quantifier scope, negation, implicature, and context-sensitivity. At the same time, IR and passage-retrieval systems are limited by their lack of understanding: Even systems with the ability to classify a question as "about" a type, or as falling into a certain, common class, are fragile in some areas. Such systems are unable to handle questions that require comprehension of the relevant document corpus; though such systems can often return passages that contain an answer to the user's question, many questions do not have answers encapsulated by a particular passage in the text, but are nevertheless answerable from the content contained within the entire document set.

Given these restrictions, the Cyc-based Deductive QA module was optimized for questions that, given the corpus and test queries, appear representative of prevalent question


You asked: How long is a minute?

MySentient

MySentient:

Inference Question Answering

60 seconds are 1 minute. (sentence confidence = 0.93)

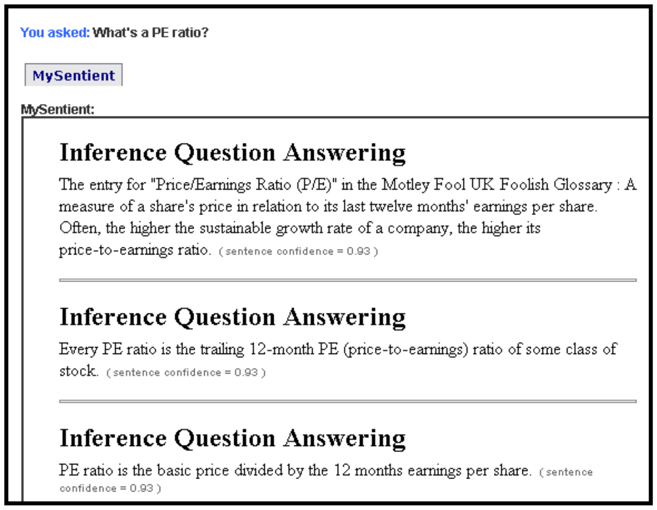
Figure 6: DQA answering a commonsense question not covered in the document corpus.

types. Significantly, the cost of targeting DQA to handle such question types is amortized by its reusability. This is in distinction to NLP-based QA systems, which generally need to be re-trained from scratch against a new corpus.

Commonsense, Off-topic Questions

In general, IR and passage-retrieval systems are limited by the corpora they draw upon in answering questions. While answering (sometimes difficult or technical) questions relevant for the domain defined by the corpora, they can appear quite intelligent or insightful. However, such systems are, by their very nature, easily "gamed" by users who wish to disabuse an otherwise sympathetic audience of the notion that the system is genuinely smart, or capable of understanding what's being asked.

Classic examples include questions that are easily answered by any intelligent agent that can understand the meanings of the words involved, such as "What colour is a blue car?" or simple general knowledge like "What is an amoeba?" and "How long is a minute?" – questions that Cyc has the knowledge to answer. Though Cyc cannot answer every conceivable commonsense question, its ability to apply common sense to both in-domain and out-of-domain problems is expected to give MySentient Answers the general look and feel of intelligence.


You asked: What's a PE ratio?

MySentient

MySentient:

Inference Question Answering

The entry for "Price/Earnings Ratio (P/E)" in the Motley Fool UK Foolish Glossary : A measure of a share's price in relation to its last twelve months' earnings per share. Often, the higher the sustainable growth rate of a company, the higher its price-to-earnings ratio. (sentence confidence = 0.93)

Inference Question Answering

Every PE ratio is the trailing 12-month PE (price-to-earnings) ratio of some class of stock. (sentence confidence = 0.93)

Inference Question Answering

PE ratio is the basic price divided by the 12 months earnings per share. (sentence confidence = 0.93)

Figure 7: Cyc's DQA module gives answers for a definitional question both from a glossary, and from a formal representation of the concept.

Definitional Questions

Questions in this category are of the familiar, "What is …?", "Who is … ?" "Define …" "What can you tell me about …?" variety. Though many corpora, including the Motley Fool UK corpus used to develop MySentient Answers, include glossaries of important concepts, in general a passage-retrieval system will only succeed in reliably returning glossary entries if one of the two following conditions hold: 1) the glossary entries are formatted so as to contain "tip-off" key-words or phrases (e.g., "APR *is defined as* …") or distinctive formatting (e.g., a bolded entry followed by a colon) that the system is trained to recognize; or 2) the question-answering system is sufficiently permissive in what it will return, that any passage (including the glossary entry) that contains the relevant query-word will be picked up.

Cyc's question-answering strategy for definitional questions is to infer good answers using the predicates #$definitionalDisplaySentence and #$interestingSentence that relate statements to some of the concepts that they are centrally about. The Cyc Knowledge Base contains a handful of general rules that allow Cyc to return sentences constructed from definitional predicates, such as #$isa and #$genls, as well as others. If Cyc has the glossary entry for a term, it will use these rules to construct an "interesting sentence" for that term that includes the glossary entry.

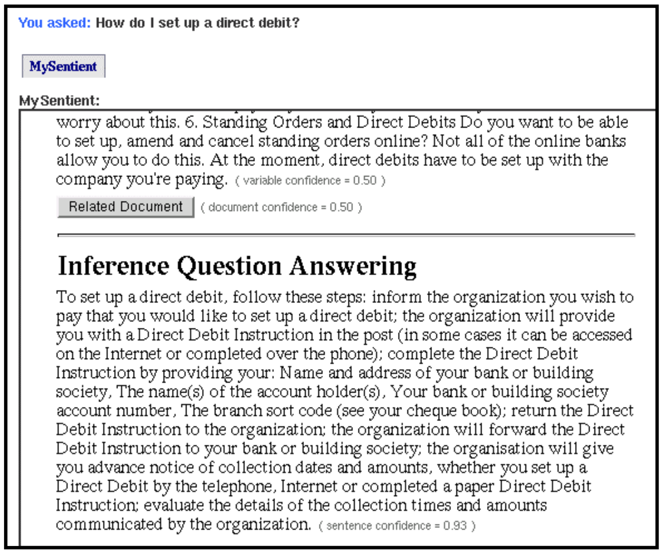
Taxonomic Questions
Taxonomic questions are those that ask the system to identify sub-types, or sub-classes of a focal concept. For example, "what are the different types of bank account?" is answered by producing a list or hierarchy of bank account-types. Such questions are relatively easy for any ontology-based deductive system, yet somewhat difficult for a system that relies solely on IR or passage-retrieval techniques.

Procedural Questions
In many document corpora, "recipes" for achieving a goal are not condensed into a single passage or even article. Often, process knowledge is spread across a corpus, or portions of it are not made explicit, left to the reader as an exercise in small-step inference. Under such circumstances, procedural questions can be difficult for a passage-retrieval system to answer.

The DQA module succeeds in these circumstances, drawing upon Cyc's hierarchy of event types and vocabulary for describing the structure of events. The CycL predicate #$properSubEvents identifies the top-level sub-events of an event, and temporal relations such as #$startsAfterEndingOf and #$startsNoEarlierThanStartingOf describe the order of sub-events. Each sub-event can have its own internal structure in similar fashion, allowing for a recursively constructed, arbitrarily deep representation of an event.

In Cyc, processes and their stages are represented as collections of events, so that process knowledge is represented with rules concluding to #$properSubEvents and the temporal ordering predicates described above. Specialized "rule macro" vocabulary allows for a compact representation of these often complex rules, making efficient reasoning about processes possible.

When asked a procedural "How do I …?" question, the NL-to-CycL parser identifies the collection of events (the process) referred to. The DQA module then asks the Cyc KB for all top-level stages in temporal order. The result is a


You asked: How do I set up a direct debit?

MySentient

MySentient:

worry about this. 6. Standing Orders and Direct Debits Do you want to be able to set up, amend and cancel standing orders online? Not all of the online banks allow you to do this. At the moment, direct debits have to be set up with the company you're paying. (variable confidence = 0.50)

Related Document (document confidence = 0.50)

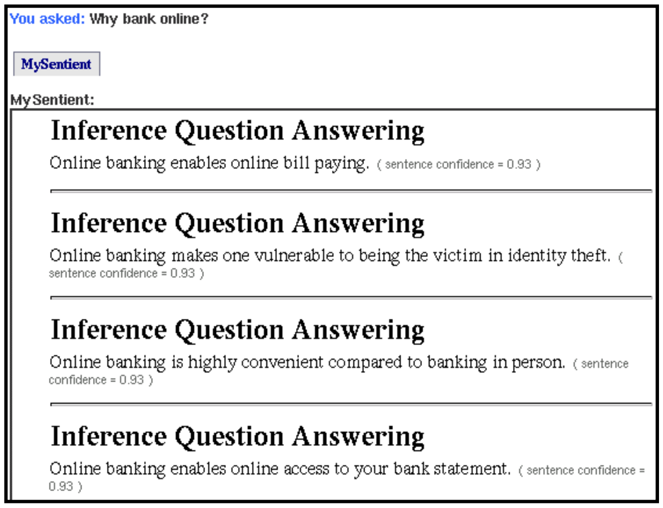
Inference Question Answering

To set up a direct debit, follow these steps: inform the organization you wish to pay that you would like to set up a direct debit; the organization will provide you with a Direct Debit Instruction in the post (in some cases it can be accessed on the Internet or completed over the phone); complete the Direct Debit Instruction by providing your: Name and address of your bank or building society, The name(s) of the account holder(s), Your bank or building society account number, The branch sort code (see your cheque book); return the Direct Debit Instruction to the organization; the organization will forward the Direct Debit Instruction to your bank or building society; the organisation will give you advance notice of collection dates and amounts, whether you set up a Direct Debit by the telephone, Internet or completed a paper Direct Debit Instruction; evaluate the details of the collection times and amounts communicated by the organization. (sentence confidence = 0.93)

Figure 8: DQA answers a "How do I ...?" question with a step-by-step procedure.

fully bound CycL sentence that relates a process to this list, which is then paraphrased appropriately in NL. The output is a step-by-step description of the process.

Cost/Benefit Questions


You asked: Why bank online?

MySentient

MySentient:

Inference Question Answering

Online banking enables online bill paying. (sentence confidence = 0.93)

Inference Question Answering

Online banking makes one vulnerable to being the victim in identity theft. (sentence confidence = 0.93)

Inference Question Answering

Online banking is highly convenient compared to banking in person. (sentence confidence = 0.93)

Inference Question Answering

Online banking enables online access to your bank statement. (sentence confidence = 0.93)

Figure 9: In answer to a "Why" question, the DQA module lists possible costs and benefits.

Questions in this category often take forms such as, "Why should I …?" or "What's the best reason to …?". Again, a passage retrieval system will do well on such questions insofar as the corpus contains explicit FAQ pages or articles with helpful headings or titles such as "Why should I …?" Even then, either the question-answering system must have received just the right input (e.g., a "Should I X or Y?" input to get back a passage entitled "Should we X or Y?") or else contain an internal table of equivalent phrasings (e.g., a question of the form, "Should I X or Y" is answerable by any passage that contains, "Why would I X over Y"). In either case, unless a passage contains some sort of explicit header or clue as to its relevance to this area, it will be passed over by a passage retrieval system.

In the DQA module, such questions are handled by asking Cyc for CycL sentences that are salient for consideration in a cost/benefit analysis of a given action type. Using the higher-order features of CycL, these sentences are inferred from assertions that identify certain *predicates* as relevant for cost/benefit analysis. For example, the assertion:

(#$costBenefitPredForSitType #$typePromotesRisk
#$Event #$doneBy 1)

tells the inference engine that the relation #$typePromotesRisk is relevant in a cost/benefit analysis of "doing" any type of event. (The "1" is the argument position of the event-type in the #$typePromotesRisk sentence.) Thus, if the user asks, "Should I bank online?", Cyc tries to prove a sentence of the form:

(#$typePromotesRisk #$OnlineBanking …)

For example:

(#$typePromotesRisk #$OnlineBanking
#$performedBy #$IdentityTheft #$victim)

which means that "performing" an online banking event increases ones vulnerability to being the victim of identity theft. Upon proving such a sentence, it is returned as a binding for the original query, and paraphrased into English.

6.2 Alternative DQA Strategies

As noted above, the general problem of parsing arbitrary English into inference-friendly CycL has not been fully solved. As such, the discourse model will not always contain a CycL translation of the user's question; indeed, for unfamiliar or complicated question-types, this will frequently be the case. Also, even when a CycL interpretation is available, there is no guarantee that the knowledge needed to answer the question is in the Cyc Knowledge Base. Thus, in order to be as effective as possible, the DQA module has been designed to reason from incomplete knowledge: it can invoke a number of lower-confidence question-answering strategies that do not depend on the total success of NL-to-CycL parsing. Because these strategies operate from a state of less information than the primary question-answering strategy, they are necessarily more brittle – specifically, more prone to returning inappropriate (but factually correct) answers. Nevertheless, these strategies provide some level of robustness against parse-failure or unanticipated discourse modeling problems, and have indeed resulted in a general increase in coverage over the target question-set. These strategies are described in order of the level of discourse model information required for them to apply, from most to least:

Topic-based Responsiveness

This strategy uses partial parse information, based on the translation into CycL of identified key phrases from the user's query. Where two or more phrases have been successfully assigned Cyc semantics, this strategy searches for interesting or informative links between them. For example, if asked a question from which only "LSE" and "AIM" are understood, Cyc would return a sentence about their relationship, such as "The Alternative Investment Market is a junior market to the London Stock Exchange."

Interesting Sentences about Terms

Like the Topic-based Responsiveness strategy, this strategy also works by reasoning over the CycL semantics of phrases from the query. This strategy, however, is more broadly applicable (and so of lower confidence), using individual query phrases and #$interestingSentence reasoning to return summary or definitional information for each phrase.

Glossary-driven Sub-string Matching

Unlike the other backup strategies, Glossary-driven Sub-string Matching does not require that any part of the question be parsed. If a query matches the title of a slurped glossary entry, then that glossary entry is returned as the answer. This strategy thus guards against knowledge gaps in the Cyc KB (*e.g.*, "Free Float" is not represented in the Knowledge Base, but the Motley Fool UK glossary entry for that term is), as well as unanticipated parser failure for definitional questions that would otherwise return glossary entries using standard question-answering methods.

7 Natural Language Generation

For a system that uses a formalized representation of the input question and performs deduction against a knowledge base whose content is also represented formally, the problem of presenting the results of a query to the end user in a readable and useful way is especially difficult.

Many systems that perform deduction will typically reduce the problem to that of generating natural-looking NL from the bindings that the inference engine returns in response to an open query. Others will attempt to augment this process by providing additional "context" – terse passages of relevant text, links to web-pages relevant to some of the entities returned as bindings, or general information in the knowledge base about those entities [Vargas-Vera and Motta, 2004; Breck *et al.*, 1999].

The generation of English answer-text from the results of deductive question answering is handled in MySentient Answers in a very different way. The answer-text generator has access not only to the variable bindings, but also to the proof tree produced by the Inference Engine, and hence to the supporting assertions in the KB. This allows for more informative and nuanced presentation of inference answers.

For instance, given the query "Who are the officers of Martha Stewart Omnimedia?" the Inference Engine finds two bindings: Susan Lyne and Martha Stewart.

Rather than somewhat misleadingly presenting these two bindings without qualification, the answer-text generator inspects the inference datastructures to find that the "Susan Lyne" answer is supported by the following line of reasoning:

1. Susan Lyne is asserted to hold the position of Chief Executive Officer in Martha Stewart Omnimedia. This assertion is in a microtheory whose contents are temporally qualified to hold during the time period from November 11, 2004 through the present.
2. *CEO* is a specialization of *Officer in Organization*.
3. If someone holds a specialized version of some position in an organization, the person may be concluded to hold the more general position.

Of these supports, (2) and (3) do not mention the binding "Susan Lyne," so Cyc chooses to present (1), and passes it to the CycL-to-NL paraphrase module. This module is independent of the explanation-generation module, and is simply tasked with rendering a CycL sentence into English. It uses the Cyc Lexicon (part of the Cyc KB), which contains mappings from atomic concepts onto names and lexical entries, and phrase-generation templates for functors. These templates are recipes for the compositional construction of natural language phrases (not strings) that have syntactic and semantic information. This information permits grammatical manipulation, such as tense, agreement or sentential force (*e.g.* question or statement.). Once the phrase is built, a string is generated from it, and returned.

For the Susan Lyne assertion, this module uses the temporal qualification on the assertion's microtheory to generate the adverbial phrase "since November 11, 2004" and to assign present perfect tense to the head verb, producing this sentence:

Since November 11, 2004, Susan Lyne has held the position of chief executive officer in Martha Stewart Living Omnimedia.

Using a similar approach, the following answer text for the "Martha Stewart" binding is produced:

From 1998 to March 15, 2004, Martha Stewart held the position of corporate president in Martha Stewart Living Omnimedia.

Thus the answer text includes not only the bindings found, but also the temporal qualification for each and the specific position held, while omitting more general, less pertinent facts and rules used to reach the conclusions. Furthermore, it does so using general principles for determining the best support to show, and existing KB assertions and paraphrase functionality.

8 Current Limitations and Future Directions

MySentient Answers 1.0 is a fully functional question-answering system, but certain areas require further development before full integration. These include clarification, anaphora resolution, external knowledge sources, and authoring tools.

8.1 Clarification

As described above, the MySentient Answers system includes a module to generate clarification questions that suggest replacement questions from the expansions generated by the User Profile Manager. Use of the Cyc KB and inference engine will allow the system to not only generate more sophisticated questions for the user, but also solicit useful information about the user. As presently envisioned, this falls into the following types:

Term-Level Disambiguation: This type of question seeks to disambiguate a term (typically a Noun Phrase) from within a question.

What did you mean by "IRA"? ...

Sentence-Level Disambiguation: This type of question seeks to resolve ambiguity at the sentence level by suggesting replacements for the entire question.

What did you mean by 'Can I get a mortgage and rent the house out?'? ...

Precisification: This sounds very similar to Sentence-Level Disambiguation but is subtly different in both implementation and effect. This attempts to take a (possibly answerable) question, and suggest more precise forms for it. The new questions can not only help QA systems find answers, but will allow them to filter out irrelevant ones.

How big is Afghanistan? →
Which of the following questions did you mean to ask?
What is Afghanistan's population?
What is Afghanistan's gross domestic product?
What is Afghanistan's land area?

Topic Redirects: This suggests potentially relevant information sources. It is intended that the author can suggest key topics, and relevant resources (with associated URLs).

Are you interested in sellers of mortgages?

Interview Questions: These use Cyc's knowledge base to determine what sorts of information about discourse entities is commonly available and important to know in order to induce relevant questions:

What breed is your dog?

Such interview questions are intended not only to make it easier for QA modules to answer the user's question, but also to gather profile information about the user. This technology is based on the Salient Descriptor first developed for the KRAKEN system as part of DARPA's Rapid Knowledge Formation (RKF) programme [Witbrock, *et al*., 2003].

8.2 Anaphora Resolution

Cyc's discourse modeling enables the system to make significant headway into the problem of resolving anaphoric pronouns and noun phrases, which has been recognized as a difficult and important problem in question answering [Vicedo and Ferrandez, 2000].

The anaphora resolution implemented for MySentient Answers makes the simplifying assumptions that 1) definite NPs can be the antecedents of anaphoric NPs and pronouns, and 2) such NPs refer to instances of the relevant type (so "the shark" is interpreted as referring to a particular fish, though there are contexts where it does not, *e.g.* "The shark is a ferocious predator."). Cyc's anaphora resolution proceeds by searching backward through the discourse model for the most recent possible antecedent, eliminating candidates by applying both linguistic and semantic knowledge. On the linguistic side, for example, "he," being singular, would not resolve to "us," which is plural. On the semantic side, the referent of "he," presumably a male animal, cannot be identical to the referent of "my mother," who is represented in the discourse model as a woman.

8.3 External Knowledge Sources

Cyc's inference engine has the capability of drawing information, not only from its knowledge base, but also from external knowledge sources such as databases, and structured websites [Masters and Güngördü, 2003]. Information from multiple external sources (and the KB) can be combined in one inference.

There are two main difficulties associated with use of this technology in a system such as MySentient Answers: the task of authoring the formal semantics of an external knowledge source is still time-consuming and requires extensive training; and the use of external sources, especially websites, generally makes it difficult to ensure that the system is fast enough to be responsive to the user.

8.4 Authoring Tools

As described above, the prototype system was specialized for the Motley Fool UK domain by a combination of prototype authoring tools and manual ontological engineering. It is anticipated that, eventually, the customer will perform almost all authoring. A number of authoring tools are planned to support both NLP and Cyc-based modules. Those that most directly support Cyc's rôle are:

Concept Extractor: This component processes domain-relevant documents to identify concepts (primarily noun phrases), relate them to existing Cyc terms, and conjecture type information for novel terms. This uses Cycorp's Noun Learner, developed under Phase I of the AQUAINT project.

Coverage Checker: This component uses Cyc's Knowledge Base to ensure that the terms identified by the Concept Extractor are adequately represented, and identify knowledge gaps in the form of questions. Like the Interview clarification strategy described in section 8.1, the coverage checker is based on the Salient Descriptor.

Term Lexifier: This component, allows an author to relate Cyc terms (both denotational and sentential) to their natural language representations. These mappings can be used for both parsing and paraphrase generation. This is based on emerging Cycorp technology, and early RKF experiments.

Ontology Editor: This component projects a stratified digraph onto a relevant subset of the Cyc ontology, permitting a GUI to display the graph to enable browsing of and modification to the ontology. This component ties together the foregoing components, by visualizing of the results of the Concept Extractor, allowing the user to answer the Concept Extractor's questions, and providing access the Term Lexifier. This is novel technology developed for this project.

Acknowledgments

The authors would like to thank MySentient, especially Mike Mendoza, John Kranz, Dave Wade-Stein, Scott Gosling, Sean O'Connor, Mike Krell, and Rob Halsted. Also, the authors thank CNLP, in particular, Elizabeth Liddy, Eileen Allen, Ozgur Yilmazel, Nancy McCracken, and Niranjan Balasubramanian. Finally, the authors acknowledge the other contributors to the MySentient project at Cycorp: Robert C. Kahlert, Karen Pittman, Dave Schneider, Ben Gottesman, Peter Wagner, Linda Aramil, Jennifer Sullivan, Larry Lefkowitz, Michael Witbrock, Steve Reed, Matt Watson, Jim Zaiss, Blake Shepard, Chris Deaton, Casey McGinnis, Brett Summers, Kevin Knight, Pace Reagan, Keith Goolsbey, Kathy Panton, Chester John, and Amanda Vizedom.

References

[Friedland *et al.*, 2004] Noah S. Friedland, Paul G. Allen, Gavin Matthews, Michael Witbrock, David Baxter, Jon Curtis, Blake Shepard, Pierluigi Miraglia, Jurgen Angele, Steffen Staab, Eddie Moench, Henrik Oppermann, Dirk Wenke, David Israel, Vinay Chaudhri, Bruce Porter, Ken Barker, James Fan, Shaw Yi Chaw, Peter Yeh, Dan Tecuci, Peter Clark. Project Halo: Towards a Digital Aristotle. *AI Magazine*, 25(4): 29-48, Winter 2004.

[Diekema, *et al.*, 2000] Diekema, A. Liu, X., Chen, J., Wang, H., McCracken, N., Yilmazel, O., and Liddy,E.D. Question Answering: CNLP at the TREC-9 Question Answering Track. In *Proceedings of the 9th Text REtrieval Conference*, pages 501–510, Gaithersburg, MD, USA, November 2000.

[Panton, *et al.*, 2002] Kathy Panton, Pierluigi Miraglia, Nancy Salay, Robert C. Kahlert, David Baxter, Roland Reagan. Knowledge Formation and Dialogue Using the KRAKEN Toolset. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligence*, pages 900–905, Edmonton, Canada, July 28–August 1, 2002.

[Burns and Davis, 1999] K.J. Burns and A.R. Davis. Building and Maintaining a Semantically Adequate Lexicon Using Cyc. In *Breadth and Depth of Semantic Lexicons*, 2(3):397–425, June 1992.

[Hovy, *et al.*, 2000] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question Answering in Webclopedia. In *Proceedings of the 9th Text REtrieval Conference*, pages 655–664, Gaithersburg, MD, USA, November 2000.

[Bilotti, 2004] Matthew W. Bilotti, Query Expansion Techniques for Question Answering. Masters Thesis, Massachusetts Institute of Technology, 2004.

[Bacchin and Melucci, 2004] Michela Bacchin and Massimo Melucci, Expanding Queries using Stems and Symbols. In *Proceedings of the 13th Text REtrieval Conference*, Gaithersburg, MD, USA, November 2004.

[Vargas-Vera and Motta, 2004] Maria Vargas-Vera and Enrico Motta. AQUA – Ontology-based Question Answering System. In *Proceedings of the Third Mexican International Conference on Artificial Intelligence*, pages 468–477, Mexico City, Mexico, April 2004.

[Breck *et al.*, 1999] Eric Breck, John Burger, Lisa Ferro, David House, Marc Light, Inderjeet Mandi. A Sys called Qanda. In *Proceedings of the Eighth Text REtrieval Conference*, pages 499–506, Gaithersburg, MD, USA, November 1999.

[Witbrock *et al.*, 2003] Michael Witbrock, David Baxter, Jon Curtis, Dave Schneider, Robert Kahlert, Pierluigi Miraglia, Peter Wagner, Kathy Panton, Gavin Matthews, Amanda Vizedom. An Interactive Dialogue System for Knowledge Acquisition in Cyc. In *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems*, pages 138–145, Acapulco, Mexico, August 2003.

[Vicedo and Ferrandez, 2000] Jose L. Vicedo and Antonio Ferrandez. Importance of Pronominal Anaphora Resolution in Question Answering Systems. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 555–562, Hong Kong, China, October 2000.

[Masters and Güngördü, 2003] Chip Masters and Zelal Güngördü, Structured Knowledge Source Integration: A Progress Report. In *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS 03)*. Pages 562-566, Piscataway, New Jersey, 2003.

Knowledge Representation for Semantic Entailment and Question-Answering

Rodrigo de Salvo Braz **Roxana Girju** **Vasin Punyakanok** **Dan Roth** **Mark Sammons**
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL, 61801, USA
{braz, girju, punyakan, danr, mssammon}@cs.uiuc.edu

Abstract

Semantic entailment is the problem of determining if the meaning of a given sentence entails that of another. *Question-answering* can be reduced to this problem by rephrasing the question as a statement that is entailed by correct answers. In [Braz *et al.*,] we present a principled approach to semantic entailment that builds on inducing re-representations of text snippets into a hierarchical knowledge representation along with an optimization-based inferential mechanism that makes use of it to prove semantic entailment.

This paper provides details and analysis of the knowledge representation and knowledge resources issues in the above approach. We analyze our system's behavior on a collection of question-answer pairs and use it to motivate and explain some of the design decisions in our hierarchical knowledge representation, that is centered around a predicate-argument type abstract representation of text.

1 Introduction

Semantic entailment is the task of determining, for example, that the sentence: "*WalMart defended itself in court today against claims that its female employees were kept out of jobs in management because they are women*" entails that "*WalMart was sued for sexual discrimination*".

Determining whether the meaning of a given text snippet *entails* that of another or whether they have the same meaning is a fundamental problem in natural language understanding that requires the ability to abstract over the inherent syntactic and semantic variability in natural language [Dagan and Glickman, 2004]. This challenge is at the heart of many high level natural language processing tasks including Question Answering, Information Retrieval and Extraction, Machine Translation, and others that attempt to reason about and capture the meaning of linguistic expressions.

Research in natural language processing in the last few years has concentrated on developing resources that provide multiple levels of syntactic and semantic analysis, resolve context sensitive ambiguities, and identify relational structures and abstractions (from syntactic categories like POS tags to semantic categories such as named entities).

However, beyond these resources, in order to support fundamental tasks such as inferring semantic entailment between two text snippets, there needs to be a unified knowledge representation of the text that **(1)** provides a hierarchical encoding of the structural, relational and semantic properties of the given text, **(2)** is integrated with learning mechanisms that can be used to induce such information from raw text, and **(3)** is equipped with an inferential mechanism that can be used to support inferences over such representations.

Relying on general purpose knowledge representations — FOL, probabilistic or hybrids — along with their corresponding general purpose inference algorithms does not resolve the key issues of *what to represent* and *how to derive a sufficiently abstract representation* and, in addition, may lead to brittleness and complexity problems. On the other hand, relying only on somewhat immediate correspondences between question and candidate answers, such as shared words or shared named entities, has strong limitations. We avoid some of these problems by *inducing* an abstract representation of the text which does not attempt to represent the full meaning of text, but provides what could be seen as a *shallow semantic* representation; yet, it is significantly more expressive than extraction of straightforward phrase-level characteristics. We induce this into a description-logic based language that is more restricted than FOL yet is expressive enough to allow both easy incorporation of language and domain knowledge resources and strong inference mechanisms.

Unlike traditional approaches to inference in natural language [Schubert, 1986; Moore, 1986; Hobbs *et al.*, 1988] our approach (1) makes use of *machine learning* based resources in order to induce an abstract representation of the input data, as well as to support multiple inference stages and (2) models inference as an *optimization* process that provides robustness against inherent variability in natural language, inevitable noise in inducing the abstract representation, and missing information.

In this paper, we focus on the hierarchical knowledge representation used by our system. We also present a brief explanation of the inference algorithm — described in detail in a companion paper [Braz *et al.*,] — and a detailed experimental analysis of our system that highlights the advantages of the hierarchical approach. Along with the formal definition and justification developed here for our computational approach to semantic entailment, our knowledge representation and algorithmic approach provide a novel solution that addresses some of the key issues the natural language research community needs to resolve in order to move forward towards higher level tasks of this sort. Namely, we provide ways to represent knowledge, either external or induced, at multiple levels of abstractions and granularity, and reason with it at the appropriate level. The preliminary evaluation of our approach is very encouraging and illustrates the significance of some of its key contributions.

1.1 General Description of Our Approach

We reduce the problem of question answering to that of textual semantic entailment. Whether a candidate text actually answers a question can be inferred by deciding if the question, converted to a statement with a placeholder, is entailed by the candidate answer. For example, "John bought the book yesterday downtown" is a correct answer to the question "Who bought the book?" because it entails "XXX bought the book" (the placeholder can be thought of as an existentially quantified variable).

Specifically, given two text snippets S (source) and T (target) where typically, but not necessarily, S consists of a short paragraph and T, a sentence, textual semantic entailment is the problem of determining if $S \models T$, which we read as "S *entails* T". This informally means that *most people would agree that the meaning of S implies that of T*. More formally, we say that S *entails* T when some representation of T can be "matched" (modulo some meaning-preserving transformations to be defined below) with some (or part of a) representation of S, at some level of granularity and abstraction. The approach consists of the following components:

KR: A Description Logic based hierarchical knowledge representation, EFDL (Extended Feature Description Logic), [Cumby and Roth, 2002], into which we re-represent the surface level text, augmented with induced syntactic and semantic parses and word and phrase level abstractions.

KB: A knowledge base consisting of syntactic and semantic rewrite rules, written in EFDL.

Subsumption: An extended subsumption algorithm which determines subsumption between EFDL expressions (representing text snippets or rewrite rules). "Extended" here means that the basic unification operator is extended to support several word level and phrase level abstractions.

First, a set of machine learning based resources are used to induce the representation for S and T. The entailment algorithm then proceeds in two phases: (1) it incrementally generates re-representations of the original representation of the source text S by augmenting it with heads of subsumed rewrite rules, and (2) it makes use of an optimization based (extended) subsumption algorithm to check whether any of the alternative representations of the source entails the representation of the target T. The extended subsumption algorithm is used both in checking final entailment and in determining when and how to generate a re-representation in slightly different ways.

Figure 1 provides a graphical example of the representation of two text snippets, along with a sketch of the extended subsumption approach to decide the entailment.

2 Hierarchical Knowledge Representation

Our abstract representation of text passages is based on a predicate-argument structure at both semantic and syntactic levels. The semantic representation captures predicate-argument relations following the PropBank [Kingsbury *et al.*, 2002] representation. PropBank is a semantically annotated version of the Wall Street Journal portion of Penn Treebank and provides consistent semantic role labels across different syntactic realizations of the same verb as shown the in following example:

[*Mary*]/ARG0 *left* [*the room*]/ARG1.

Here, ARG0 represents the *leaver* and ARG1 the *thing left* as arguments of the verb *leave*.

Currently, we focus only on verb-argument relations, but our representation can be easily extended to other types of predicates, such as nouns, adjectives, and adverbs. This representation is induced by a machine learning based Semantic Role Labeler ([Punyakanok *et al.*, 2004]) that identifies the verb's arguments and semantically annotates them with corresponding PropBank labels in context.

At the syntactic level, the representation captures full parse information for the text considered. For this, we rely on Collin's parser [Collins, 1999]. For example,

[*Mary*]/NP *left* [*the room*]/NP/ [*in a hurry*]/PP,

where the two noun phrases and the prepositional phrase are attached to the verb *left*.

Besides these, we also consider a word-level representation. Each word is annotated with various information such as part-of-speech and lemma.

The abstract representation is also enriched with other types of knowledge, such as named entities (eg, "John Smith" is a PERSON), qualifier labels (eg, "some people", "no children"), negation (eg, "didn't succeed"), modality (eg, "could", "might"), temporal information (eg, "after an event", "before an action"), and coreference (both pronoun and name coreference).

The most significant aspect of our knowledge representation is its **hierarchy**. It captures the semantic, syntactic, and lexical levels of abstraction thus described and is used by the inference algorithm to exploit these inherent properties of the language. The hierarchical representation provides flexibility as the source and target snippets can be matched at the corresponding level. Most importantly, it provides a way to abstract over variability in natural language by supporting inference at a higher than word level, and thus also supports the inference process in recovering from inaccuracies in inducing the representation. Consider, for example, the following pair of sentences, in which processing at the semantic parse level exhibits identical structure, despite significant lexical level differences.

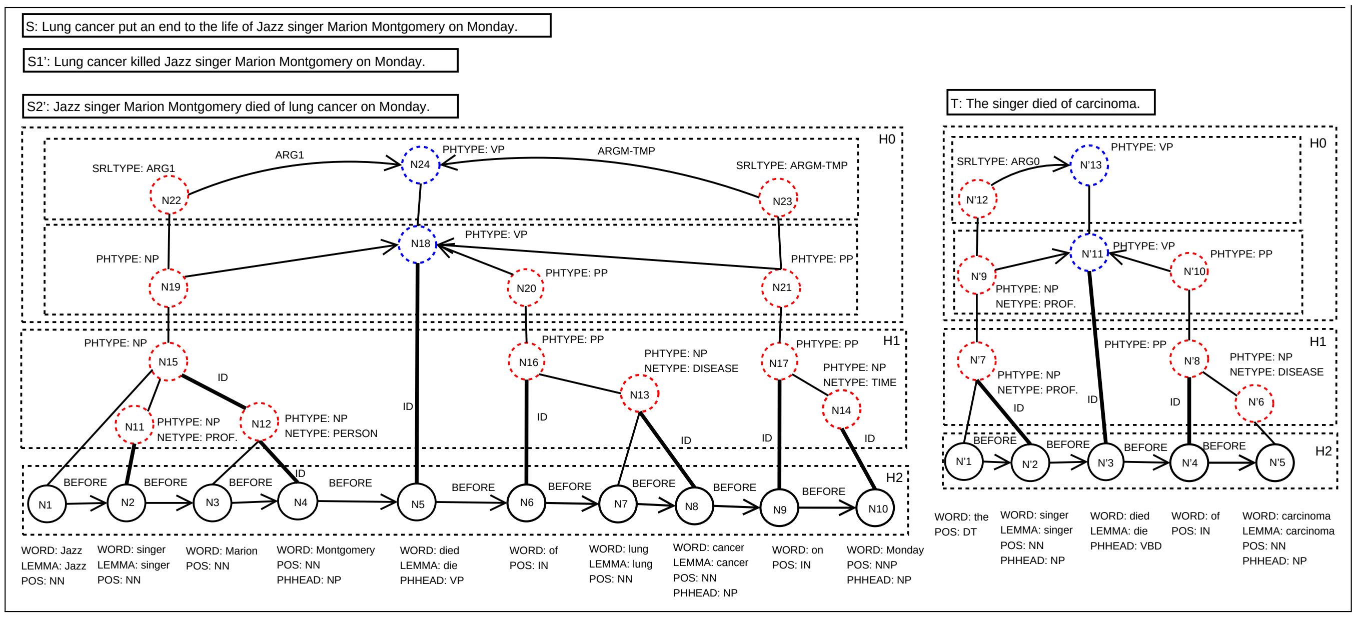

S: Lung cancer put an end to the life of Jazz singer Marion Montgomery on Monday.
S1': Lung cancer killed Jazz singer Marion Montgomery on Monday.
S2': Jazz singer Marion Montgomery died of lung cancer on Monday.
T: The singer died of carcinoma.
H0
H1
H2
WORD: Jazz LEMMA: Jazz POS: NN
WORD: singer LEMMA: singer POS: NN
WORD: Marion POS: NN
WORD: Montgomery POS: NN PHHEAD: NP
WORD: died LEMMA: die PHHEAD: VP
WORD: of POS: IN
WORD: lung LEMMA: lung POS: NN
WORD: cancer LEMMA: cancer POS: NN PHHEAD: NP
WORD: on POS: IN
WORD: Monday POS: NNP PHHEAD: NP
WORD: the POS: DT
WORD: singer LEMMA: singer POS: NN PHHEAD: NP
WORD: died LEMMA: die PHHEAD: VBD
WORD: of POS: IN
WORD: carcinoma LEMMA: carcinoma POS: NN PHHEAD: NP

Figure 1: Example of *Re-represented Source & Target* pairs as concept graphs. The original source sentence S generated several alternatives including S'_1 and the sentence in the figure (S'_2). Our algorithm was not able to determine entailment of the first alternative (as it fails to match in the extended subsumption phase), but it succeeded for S'_2. The dotted nodes represent phrase level abstractions. S'_2 is generated in the first phase by applying the following chain of inference rules: #1 (genitives): "Z's W → W of Z"; #2: "X put end to Y's life → Y die of X". In the extended subsumption, the system makes use of WordNet hypernymy relation ("*lung cancer*" IS-A "*carcinoma*") and NP-subsumption rule ("*Jazz singer Marion Montgomery*"' IS-A "*singer*"). The rectangles encode the hierarchical levels (H_0, H_1, H_2) at which we applied the extended subsumption. Also note that in the current experiments we don't consider noun plurals and verb tenses in the extended subsumption, although our system has this capability.

S: "[*The bombers*]r/ARG0 *managed* [*to enter* [*the embassy building*]/ARG1]/ARG1."[1]

T: "[*The terrorists*]/ARG0 *entered* [*the edifice*]/ARG1."

On the other hand, had the phrase *failed to enter* been used instead of *managed to enter* , a negation attribute associated with the main verb would prevent this inference. Note that failure of the semantic parser to identify the semantic arguments ARG0 and ARG1 will not result in a complete failure of the inference, as described in the inference section: it will result in a lower score at this level that the optimization process can compensate for (in the case that lower level inference occurs).

In the following subsection we present a formal description of the knowledge representation.

2.1 Formal Description of The Knowledge Representation

The hierarchical representation of natural language sentences, defined formally over a domain $\mathcal{D} = \langle \mathcal{V}, \mathcal{A}, \mathcal{E} \rangle$ which consists of a set $\mathcal{V}$ of typed elements, a set $\mathcal{A}$ of attributes of elements, and a set $\mathcal{E}$ of relations among elements. We use a Description-Logic inspired language, *Extended Feature Description Logic (EFDL)*, an extension of FDL [Cumby and Roth, 2002] . As described there, expressions in the language have an equivalent representation as *concept graphs*, and we refer to the latter representation here for comprehensibility.

Nodes in the concept graph represent elements — words or (multiple levels of) phrases. *Attributes* of nodes represent properties of elements. Examples of attributes (they are explained in more detail later) include {LEMMA, WORD, POS, PREDICATE_VALUE, PHTYPE, PHHEAD, NETYPE, ARGTYPE, NEGATION}. The first three are word level, the next three are phrase level, NETYPE is the named entity of a phrase, ARGTYPE is the set of semantic arguments as defined in PropBank [Kingsbury *et al.*, 2002] and NEGATION is a negation attribute. Only attributes with non-null values need to be specified.

Relations (roles) between two elements are represented by labeled edges between the corresponding nodes. Examples of roles (again, explained in more detail later) include: {BEFORE, ARG0, . . . ARG5}; BEFORE indicates the order between two individuals, and ARG0 represents the relations between a predicate (verb) and its argument..

Figure 1 shows a visual representation of a pair of sentences rerepresented as concept graphs.

Concept graphs are used both to describe instances (sentence representations) and rewrite rules. Details are omitted here; we just mention that the expressivity of these differ - the body and head of rules are simple chain graphs, for inference complexity reasons. Restricted expressivity is an impor-

[1]The verbs "*manage*" and "*enter*" share the semantic argument "[*the bombers*]/ARG0".

tant concept in Description Logics [Baader *et al.*, 2003], from which we borrow several ideas and nomenclature.

Concept graph representations are induced via state of the art machine learning based resources that a part-of-speech tagger [Even-Zohar and Roth, 2001], a syntactic parser [Collins, 1999], a semantic parser [Punyakanok *et al.*, 2004; 2005], a named entity recognizer [2], and a name coreference system [Li *et al.*, 2004] with the additional tokenizer and lemmatizer derived from WordNet [Fellbaum, 1998]. Rewrite rules were filtered from a large collection of paraphrase rules developed in [Lin and Pantel, 2001] and compiled into our language; a number of non-lexical rewrite rules were generated manually. Currently, our knowledge base consists of approximately 300 inference rules.

Rule representation

A rule is a pair (lhs, rhs) of concept graphs (lhs is the rule's *body*, while rhs is its *head*). These concept graphs are restricted in that they must be *paths*. This restricts the complexity of the inference algorithm while keeping them useful enough for our purposes. As a shorthand, more than one path can be described in lhs, but in this case the rule is implicitly treated as more than one rule, each with one path and the same rhs.

lhs describes a structure to match in the sentence concept graph, while rhs describes a new predicate (and related attributes and edges) to be added to the sentence concept graph in case there is a match. rhs can also describe attributes to add to one or more existing nodes without adding a new predicate, provided no new edges are introduced. These restrictions ensure that the data representation always remains, from the subsumption algorithm's perspective, a set of overlapping trees.

Variables can be used in lhs so that we can specify which entities have edges/attributes added by rhs. Rules thus allow rewrite of (part of) original sentence; e.g. we can encode DIRT [Lin and Pantel, 2001] rules as predicate/argument structures and use them to allow (parts of) the original sentence to be re-represented via paraphrase, by linking existing arguments with new predicates.

3 Algorithmic Semantic Entailment

This section follows the presentation in [Braz *et al.*,] and formally defines and justifies our algorithmic approach to semantic entailment.

Let $\mathcal{R}$ be a knowledge representation language with a well defined syntax and semantics over a domain $\mathcal{D}$. Specifically, we think of elements in $\mathcal{R}$ as expressions in the language or, equivalently, as the set of interpretations that satisfy it [Lloyd, 1987]. Let r be a mapping from a set of text snippets $\mathcal{T}$ to a set of expressions in $\mathcal{R}$. Denote the images of two text snippets S, T, under this mapping by r_S, r_T, respectively. Given the set of interpretations over $\mathcal{D}$, let M be a mapping from an expression in $\mathcal{R}$ to the corresponding set of interpretations it satisfies. For expressions r_S, r_T, the images of S, T under $\mathcal{R}$, their model theoretic representations thus defined are denoted $M(r_s)$, $M(r_t)$.

Conceptually, as in the traditional view of semantic entailment, this leads to a well defined notion of entailment, formally defined via the model theoretic view; traditionally, the algorithmic details are left to a *theorem prover* that uses the syntax of the representation language, and may also incorporate additional knowledge in its inference. We follow this view, and use a notion of *subsumption* between elements in $\mathcal{R}$, denoted $u \sqsubseteq v$, for $u, v \in \mathcal{R}$, that is formally defined via the model theoretic view – when $M(u) \subseteq M(v)$. Subsumption between representations provides an implicit way to represent entailment, where additional knowledge is conjoined with the source to "prove" the target.

However, the proof theoretic approach corresponding to this traditional view is unrealistic for natural language. Subsumption is based on *unification* and requires, in order to prove entailment, that the representation of T is entirely embedded in the representation of S. Natural languages allow for words to be replaced by synonyms, for modifier phrases to be dropped, etc., without affecting meaning. An extended notion of subsumption is therefore needed which captures sentence, phrase, and word-level abstractions.

Our algorithmic approach is thus designed to alleviate these difficulties in a proof theory that is too weak for natural language. Conceptually, a weak proof theory is overcome by entertaining multiple representations that are equivalent in meaning. We provide theoretical justification below, followed by the algorithmic implications.

We say that a representation $r \in \mathcal{R}$ is *faithful* to S if r and r_S have the same model theoretic representation, i.e., $M(r) = M(r_s)$. Informally, this means that r is the image under $\mathcal{R}$ of a text snippet with the same meaning as S.

Definition 1 *Let S, T be two text snippets with representations r_S, r_T in $\mathcal{R}$. We say that $S \models T$ (read: S* semantically *entails T) if there is a representation $r \in R$ that is faithful to S and that is subsumed by r_T.*

Clearly, there is no practical way to exhaust the set of all those representations that are faithful to S. Instead, our approach searches a space of faithful representations, generated via a set of rewrite rules in our KB.

A *rewrite rule* is a pair (lhs, rhs) of expressions in $\mathcal{R}$, such that $lhs \sqsubseteq rhs$. Given a representation r_S of S and a rule (lhs, rhs) such that $r_S \sqsubseteq lhs$, the augmentation of r_S via (lhs, rhs) is the representation $r'_S = r_S \wedge rhs$.

Claim 1 *The representation r'_S generated above is faithful to S.*

To see this, note that as expressions in $\mathcal{R}$, $r'_S = r_S \wedge rhs$, therefore $M(r'_S) = M(r_S) \cap M(rhs)$. However, since $r_S \sqsubseteq lhs$, and $lhs \sqsubseteq rhs$, then $r_S \sqsubseteq rhs$ which implies that $M(r_S) \subseteq M(rhs)$. Consequently, $M(r'_S) = M(r_S)$ and the new representation is faithful to S.

The claim gives rise to an algorithm, which suggests incrementally *augmenting* the original representation of S via the rewrite rules, and computing subsumption using the "weak" proof theory between the augmented representation and r_T.

[2]Named entity recognizer from Cognitive Computation Group, http://l2r.cs.uiuc.edu/∼cogcomp

Informally, this claim means that while, in general, augmenting the representation of S with an expression rhs may restrict the number of interpretations the resulting expression has, in this case, since we only augment the representation when the left hand side lhs subsumes r_S, we end up with a re-representation that is in fact equivalent to r_S. Therefore, given a collection of rules $\{(lhs, rhs)\}$ we can chain them, and incrementally generate faithful representations of S. Consequently, this algorithm is a sound algorithm[3] for semantic entailment according to Def. 1, but it is not complete. Its success depends on the size and quality of the rule set[4] applied in the search.

Two important notes are in order. First, since rewrite rules typically "modify" a small part of a sentence representation (see Fig. 1), the augmented representation provides also a compact way to encode a large number of possible representations. Second, note that while the rule augmentation mechanism provides a justification for an algorithmic process, in practice applying rewrite rules is somewhat more complicated. The key reason is that many rules have a large fan-out; that is, a large number of heads are possible for a given rule body. Examples include synonym rules, equivalent ways to represent names of people (e.g., John F. Kennedy and JFK), etc. We therefore implement the mechanism in two ways; one process which supports chaining well, in which we explicitly augment the representation with low fan-out rules (e.g., Passive-Active rules); and a second, appropriate to the large fan-out rules. In the latter, we abstain from augmenting the representation with the many possible heads but take those rules into account when comparing the augmented source with the target. For example, if a representation includes the expression "JFK/PER", we do not augment it with all the many expressions equivalent to "JFK" but, when comparing it to a candidate in the target, such as "President Kennedy", these equivalencies are taken into account. Semantically, this is equivalent to augmenting the representation. Instead of an explicit list of rules, the large fan-out rules are represented as a functional black box that can, in principle, contain any procedure for deciding comparisons. For this reason, this mechanism is called *functional subsumption*.

The resulting algorithmic approach is therefore:

(1) Once an EFDL representation for S and T is induced, the algorithm incrementally searches the EFDL rewrite rules in KB to find a rule with a body that subsumes the representation of S. In this case, the head of the rule is used to *augment* the EFDL representation of S and generate a new (equivalent) representation of S. KB consists of syntactic and semantic EFDL rewrite rules expressed at the word, syntactic and semantic categories, and phrase levels; applying them results in new representations S'_i that capture alternative ways of expressing the surface level text.

(2) Re-representation S'_is are processed via the extended subsumption algorithm against the representation of T. The notion of extended subsumption captures, just like the rewrite rules, several sentence, phrase, and word-level abstractions. The extended subsumption process is also used when determining whether a rewrite rule applies.

Rewrite rules and extended subsumption decisions take into account relational and structural information encoded in the hierarchical representation, which is discussed below. In both cases, decisions are quantified as input to an optimization algorithm that attempts to generate a "proof" that S entails T.

4 Inference Model and Algorithm

As natural languages allow the expression of various concepts in different ways without affecting meaning, an exact subsumption approach that requires the representation of T be entirely embedded in the representation of S'_i is unrealistic.

Extended subsumption is designed to take advantage of the hierarchical representation at various levels of abstraction at the sentence, phrase, and word-level. Thus, nodes in a concept graph are grouped into different hierarchical sets denoted by $H = \{H_0, \ldots, H_j\}$ where a lower value of j indicates higher hierarchical level (more important nodes).

The inference procedure recursively matches the corresponding H_j nodes in T and S'_i until it finds a pair whose constituents do not match. In this situation, a *Phrase-level Subsumption* algorithm is applied.

Figure 1 exemplifies the matching order between S'_i and T based on constraints imposed by the hierarchy.

We solve the subsumption problem by formulating an equivalent Integer Linear Programming (ILP) problem[5]. Details about the extended subsumption and the inference algorithm can be found in a companion paper [Braz *et al.*,].

5 Experimental Evaluation and Discussion

We tested our approach on a collection[6] of question-answer pairs develop by Xerox PARC for a pilot evaluation of Knowledge-Oriented Approaches to Question Answering under the ARDA-AQUAINT program. The PARC corpus consists of 76 Question-Answer pairs annotated as "true", "false" or "unknown" (and an indication of the type of reasoning required to deduce the label). The question/answer pairs provided by PARC are designed to test different cases of linguistic entailment. The corpus concentrates on examples of strict and plausible linguistic (lexical and constructional) inferences and indicates whether it involves some degree of background world knowledge. The focus is on inferences that can be made purely on the basis of the meaning of words and phrases. The questions are straightforward and therefore easily rewritten (by hand) into statement form. One sentence pair involving qualifiers was reordered to test qualifier subsumption.

[3]Soundness depends on a "correct" induction of the representation of the text; we do not address this theoretically here.

[4]The power of this search procedure is in the rules. lhs and rhs might be very different at the surface level, yet, by satisfying model theoretic subsumption they provide expressivity to the re-representation in a way that facilitates the overall subsumption.

[5]Despite the fact that this optimization problem is NP hard, commercial packages have very good performance on sparse problems such as this one [Xpress-MP,].

[6]The data is available by following the data link from http://l2r.cs.uiuc.edu/~cogcomp.

For evaluation reasons, we used only two labels in our experiments , "true" and "false", corresponding to "S entails T" and "S does not entail T". The "unknown" instances were classified as "false".

Of these 76 sentence pairs, 64 were perfectly tagged by our Semantic Role Labeler (SRL), and were used as a noise-free test set to evaluate our system.

This section describes the development of our system along two dimensions: one adds more structure and annotation of words and phrases, while the other adds semantic analysis components. We first outline the progressive development of our system from lexical level matching to the full system. Then, for each version of the system and for each semantic analysis component we give one or more examples of sentence pairs affected by the new version/component. Finally, we present a summary of the performance of each version of the system on the noise-free and noisy data sets.

As baseline we use lexical-level matching based on a bag-of-words representation with lemmatization and normalization (LLM). We use this as the starting point for our system. We then add Semantic Role Labeling, which gives us simple verb-level predicate-argument structure; this version of the system is labeled "SRL + LLM". Finally, we add full parse and shallow parse structure, which allows the parsing of the SRL arguments into hierarchical structures with key entities as the roots and modifiers as the leaves. This version of the system, labeled "SRL + deep structure", also uses Named Entity annotation from our Named Entity Recognizer (NER).

The system presently supports three semantic analysis modules: verb phrase compression, discourse analysis, and qualifier analysis. The different semantic analysis modules depend on different levels of structure: verb phrase compression requires word order and part of speech; discourse analysis requires full parse information and part of speech, and qualifier analysis requires full and shallow parse information and part of speech. These components are added in the above order for each version of the system that supports them.

Finally, there is a Knowledge Base module comprising rewrite rules that encode paraphrase and inference information. Most rules require SRL information, though some use only word order and the words themselves. In evaluating the LLM system with the KB enabled, only word-based rules can fire.

A. LLM

We use the LLM as the starting point for our full entailment system. The LLM system ignores a large set of stopwords, which for certain positive sentence pairs allows entailment when the more sophisticated systems require a rewrite rule to map from the predicate in S to the predicate in T. For example: since the list of stopwords includes forms of "be", the following sentence pair will be classified "true" by LLM, while the more sophisticated systems require a KB rule to link "visit" to "be (in)":

S: *[The diplomat]/ARG1 visited [Iraq]/ARG1 [in September]/AM_TMP*
T: *[The diplomat]/ARG1 was in [Iraq]/ARG2*

The SRL+LLM system will extract verb frames for "visit" in S and "was" in T, and will fail the subsumption check at the verb level. For LLM, the only words of T that register are "diplomat" and "Iraq", and as these are present in S, LLM will return "true".

Of course, LLM is insensitive to small changes in wording. For the following sentence pair, LLM returns "true", which is clearly incorrect:

S: *Legally, John could drive.*
T: *John drove.*

B. SRL + LLM

The next version of the system first tries to match SRL annotation for S and T, and if this matches, it uses LLM to determine argument subsumption.

The advantage of SRL+LLM over LLM is evident when, for example, arguments are interchanged between two verbs. This case is not represented in the PARC dataset, but the following sentence pair gives such an example:

S: *[The president]/ARG0 said [[the diplomat]/ARG0 left [Iraq]/ARG1]/ARG1*
T: *[The diplomat]/ARG0 said [[the president]/ARG0 left [Iraq]/ARG1]/ARG1*

In this case, the non-stopwords in S and T are identical, so LLM will label this pair "true". However, SRL will attach different ARG0s to "said" and "left", and will therefore correctly label this sentence pair "false".

The disadvantage of using SRL is that it generates predicate frames for some verbs that are ignored as stopwords by LLM, such as "went" in the following example:

S: *[The president]/ARG0 visited [Iraq]/ARG1 [in September]/AM_TMP*
T: *[The president]/ARG0 went to [Iraq]/ARG1.*

Where LLM ignores "went", SRL generates a predicate node, causing subsumption to fail at the predicate level and generating an incorrect "false" label.

In this data set, there are more instances like the second case above than like the first; the result is a drop in performance. However, the rest of this section will show that the SRL forms a crucial backbone that supports a more successful approach.

The Verb Processing module

The Verb Processing (VP) module rewrites certain verb phrases as a single verb with additional attributes. It uses word order and Part of Speech information to identify candidate patterns and, when the verbs in the construction in the sentence match a pattern in the VP module, the verb phrase is replaced by a single predicate node with additional attributes representing modality ("CONFIDENCE") and tense ("TENSE").

The VP module presently recognizes modal constructions, tense constructions, and simple verb compounds of the form "VERB to VERB" (such as "manage to enter"). In each case, the first verb is compared to a list that maps verb lemmas to tenses and qualifiers; for example, "has" is recognized as a tense auxiliary and results in the attribute "TENSE: past" being added to the second verb's node; the "has" node is then eliminated and the graph structure corrected.

In the example below, the VP recognizes the modal construction and adds the qualifying attribute "CONFIDENCE: potential" to the main verb node, "drive":

S: *Legally, John could drive.*
T: *John drove.*

Subsumption in the SRL+LLM system then fails at the verb level, returning the correct value "false" for this sentence pair.

This module also acts as an enabler for other resources (such as the Knowledge Base). This may result in a decrease in performance when those modules are not present, as it corrects T sentences that may have failed subsumption in the SRL+LLM system because the auxiliary verb was not present in the corresponding S sentence:

S: *Bush said that Khan sold centrifuges to North Korea.*
T: *Centrifuges were sold to North Korea.*

The SRL+LLM system returns the correct answer, "false", for this sentence pair, but for the wrong reason: SRL generates a separate predicate frame for "were" and for "sold" in T, and there is no matching verb for "were" in S.

When the VP module is added, the auxiliary construction in T is rewritten as a single verb with tense and modality attributes attached; the absence of the auxiliary verb means that SRL generates only a single predicate frame for "sold". This matches its counterpart in S, and subsumption succeeds, as the qualifying effect of the verb "said" in S cannot be recognized without the deeper parse structure and the Discourse Analysis module.

On the PARC corpus, the net result of applying the VP module when the KB is not enabled is either no improvement or a decrease in performance, due to the specific mix of sentences. However, the importance of such a module to correctly identify positive examples becomes evident when the knowledge base is enabled, as the performance jumps significantly.

C. SRL + Deep Structure

The next version of the system uses full and shallow parse, Named Entity and Part of Speech information to identify substructure in SRL predicate arguments. Specifically, the system identifies the key entity in each argument and modifiers such as adjectives, titles, and quantities. By default, we assume that T must be less specific than S (this is not always correct, but requires a new module to determine when a more general argument entails a more specific one, and when not).

The new system correctly labels the following example, which is incorrectly labeled by the LLM and SRL+LLM systems:

S: *No US congressman visited Iraq until the war.*
T: *Some US congressmen visited Iraq before the war.*

The new system includes the determiners "no" and "some" as modifiers of their respective entities; subsumption fails at the argument level because these modifiers don't match.

However, the new system also makes new mistakes:

S: *The room was full of women.*
T: *The room was full of intelligent women.*

The LLM and SRL+LLM systems find no match for "intelligent" in S, and so return the correct answer, "false". However, the SRL+deep structure system allows unbalanced T adjective modifiers, assuming that S must be more general than T, and returns "true".

Verb Phrase Module

Adding the Verb Phrase module results in performance changes similar to those when it is enabled with the SRL+LLM system, for the same reasons.

Discourse Analysis Module

The Discourse Analysis (DA) module detects the effects of an embedding predicate on the embedded predicate. It uses the full parse tree to identify likely candidate structures, then compares the embedding verb to a list mapping verbs to modality (CONFIDENCE). The main distinction that is presently supported is between "FACTUAL" and a set of values that distinguish various types of uncertainty. This allows different assumptions to be supported; for example, if we wish to assume that when something is said, it is taken as truth, we can treat the CONFIDENCE value "REPORTED" as entailing "FACTUAL" and vice versa. The module attaches the appropriate CONFIDENCE attribute value to the embedded verb node; if this attribute is not matched during subsumption, subsumption fails.

The following example highlights the importance of the way an embedded predicate is affected by the embedding predicate. In this example, the predicate "Hanssen sold secrets to the Russians" is embedded in the predicate "The New York Times reported...".

S: "*The New York Times reported that Hanssen sold FBI secrets to the Russians and could face the death penalty.*"

T: "*Hanssen sold FBI secrets to the Russians.*"

Our system identifies the following verb frames in S and T:

S-A: "*[The New York Times]/A0 reported [that Hanssen sold FBI secrets to the Russians...]/A1*"

S-B: "*[Hanssen]/A0 sold [FBI secrets]/A1 to [the Russians]/A3*"

T-A: "*[Hanssen]/A0 sold [FBI secrets]/A1 to [the Russians]/A3*"

During preprocessing, our system detects the pattern "[VERB] that [VERB]", and classifies the first verb as affecting the confidence of its embedded verb. The system marks the verb (predicate) "sold" in S with attribute and value "CONFIDENCE: REPORTED". Thus the subsumption check determines that entailment fails at the verb level, because by default, verbs are given the attribute and value "CONFIDENCE: FACTUAL", and the CONFIDENCE values of the "sold" nodes in S and T do not match. This is in contrast to LLM and SRL+LLM, both of which return the answer "true".

The next example demonstrates that the implementation of this embedding detection is robust enough to handle a subtly different sentence pair: in this case, the sentence structure "Hanssen, who sold..." indicates that the reader should understand that it is already proven (elsewhere) that Hanssen has sold secrets.

S: “*The New York Times reported that Hanssen, who sold FBI secrets to the Russians, could face the death penalty.*”

T: “*Hanssen sold FBI secrets to the Russians.*”

Our system identifies the following verb frames in S and T (using the full parse data provided by Collins’ parser to connect “who” to “Hanssen”):

S-A: “*[The New York Times]/A0 reported [that Hanssen, who sold FBI secrets to the Russians...]/A1*”

S-B: “*[Hanssen]/A0 sold [FBI secrets]/A1 to [the Russians]/A3*”

T-A: “*[Hanssen]/A0 sold [FBI secrets]/A1 to [the Russians]/A3*”

During preprocessing, the system does not detect an embedding of “sold” in “reported”, and so does not attach the attribute and value “CONFIDENCE: REPORTED” to the verb “sold” in S. During the subsumption check, the “sold” verbs now match, as both are considered factual.

Qualifier Module

The Qualifier module allows comparison of qualifiers such as “all”, “some”, “any”, “no”, etc. In the experimental results summarized below, adding the Qualifier module does not improve performance, because in all the PARC examples involving qualifiers, a different qualifier in S and T corresponds to a negative label.

However, the qualifier module will correctly analyze the following sentence pair, which is a reordered pair from the PARC corpus:

S: *All soldiers were killed in the ambush.*
T: *Many soldiers were killed in the ambush.*

The default rule – non-identical argument modifiers cause subsumption to fail – is incorrect here, as S entails T. The Qualifier module correctly identifies the entailment of “many” by “all”, and subsumption will succeed.

5.1 Experimental Results

We present the evaluation results across two dimensions. On the horizontal axis we represent the baseline, system version#1 (SRL+LLM), and system version#2 (SRL + Deep structure). On the vertical axis we represent various knowledge modules used to enrich the basic representation. The results are summarized below in two tables. Table 1 represents the evaluation of the system with and without the KB inference rules and when SRL is perfect (i.e., considering only examples on which our SRL tool gives correct annotation). Table 2 shows the performance when the SRL system is used with the entire dataset, including those examples on which the SRL tool makes mistakes.

The results obtained with perfect semantic argument structure (perfect SRL) are provided here to illustrate the advantages of the hierarchical approach, as noise introduced by SRL errors can obscure the effects of the different levels of the hierarchical representation/subsumption.

The improvement across the vertical dimension, which represents additional analysis resources that use the structural information supported by the given system configuration, is monotonic, indicating the benefits of semantic analysis of the structural information.

The improvement across the horizontal dimension, which represents successively finer structural representation, is clearly best for the system with full parse information. To summarize, the system behaves consistently, showing improvement as additional hierarchical structure and additional semantic analysis resources are added. These results show that the hierarchical approach is valid.

Work In Progress

Presently, the system works well when the SRL annotation it depends on is mostly or completely correct. We are working on the following items to further improve the system’s performance:

- **Back-off Strategies to Handle Missing SRL Information.** We are working on a secondary step in each of the first two levels of the subsumption algorithm, in which we will use the Full Parse/dependency information to seek candidates to substitute for “missing” arguments and even verbs.
- **Verb Phrase Matching.** While the present SRL annotation allows us to handle simple verb phrases by decomposition, this is potentially error-prone.

  We are developing a module to address the problem of verb phrases (as highlighted in example 2 above) by preprocessing sentences to collapse simple verb phrases into a single node. This requires identifying the main verb to associate with the replacement node, and adding attributes to represent any qualification associated with the supporting verb (in this case, “manage” does not require new attributes at our current level of operation, but phrases like “failed to enter” and “tried to enter” would require negation and intention attributes respectively, both of which should result in subsumption failing unless similar qualifications are present in the matching verb/verb phrase).
- **More sophisticated quantity matching.** Clearly there is a need to use numbers as a critical matching element in the phrase-level subsumption phase. However, matching numbers is not necessarily straightforward, as they can be qualified in a number of ways – “at least $1 million”, “over $1 million”, “more than $1 million”, “one million dollars”, etc. We can identify number boundaries in many cases with our Named Entity Tagger, and use this to identify number qualifiers of key entities. However, determining whether numbers match requires further processing to relate qualifiers, different notations, etc.

6 Previous Work

Knowledge representation and reasoning techniques have been studied in NLP for a long time [Schubert, 1986; Moore, 1986; Hobbs *et al.*, 1988]. Most approaches relied on First Order Logic representations with a general prover and without using acquired rich knowledge sources.

	without KB			with KB		
	LLM	**SRL+LLM**	**SRL + Deep structure**	**LLM**	**SRL+LLM**	**SRL + Deep structure**
Base	59.38	56.25	62.50	62.50	60.94	68.75
VP	N/A	57.81	62.50	N/A	67.19	75.00
DA	N/A	N/A	71.88	N/A	N/A	82.81
Qual	N/A	N/A	71.88	N/A	N/A	82.81

Table 1: System's performance obtained for the PARC question-answer pairs without with perfect SRL. This corresponds to 64 question-answering pairs. The empty buckets (N/A) indicate that the module in the left hand column could not be used with that column's system configuration.

	without KB			with KB		
	LLM	**SRL+LLM**	**SRL + Deep structure**	**LLM**	**SRL+LLM**	**SRL + Deep structure**
Base	61.84	55.26	61.84	64.47	59.21	67.11
VP	N/A	55.26	60.52	N/A	63.16	69.74
DA	N/A	N/A	68.42	N/A	N/A	77.63
Qual	N/A	N/A	68.42	N/A	N/A	77.63

Table 2: System's performance obtained for the PARC question-answer pairs on the full data set. The empty buckets (N/A) show that no knowledge information could be used.

Significant development in NLP, specifically the ability to acquire knowledge and induce some level of abstract representation could, in principle, support more sophisticated and robust approaches. Nevertheless, most modern approaches developed so far are based on shallow representations of the text that capture lexico-syntactic relations based on dependency structures and are mostly built from grammatical functions in an extension to keyword-base matching [Durme *et al.*, 2003]. Some systems make use of some semantic information, such as WordNet lexical chains [Moldovan *et al.*, 2003], to slightly enrich the representation. Other have tried to learn various logic representations [Thompson *et al.*, 1997]. However, none of these approaches makes global use of a large number of resources as we do, or attempts to develop a flexible, hierarchical representation and an inference algorithm for it, as we present here.

7 Conclusions

This paper presents a principled, integrated approach to *semantic entailment*. We developed an expressive knowledge representation that provides a hierarchical encoding of structural, relational and semantic properties of the text and populated it using a variety of machine learning based tools. An inferential mechanism over a knowledge representation that supports both abstractions and several levels of representations allows us to begin to address important issues in abstracting over the variability in natural language. Our preliminary evaluation is very encouraging, yet leaves a lot to hope for. Improving our resources and developing ways to augment the KB are some of the important steps we need to take. Beyond that, we intend to tune the inference algorithm by incorporating a better mechanism for choosing the appropriate level at which to require subsumption. Given the fact that we optimize a linear function, it is straightforward to learn the cost function. Moreover, this can be done in such a way that the decision list structure is maintained.

8 Acknowledgments

We grateful to Ron Kaplan and Xerox PARC for sharing the collection of question-answer pairs with us. This research is supported by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program, a DOI grant under the Reflex program, NSF grants ITR-IIS-0085836, ITR-IIS-0085980, and an ONR MURI Award.

References

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *Description Logic Handbook*. Cambridge, 2003.

[Braz *et al.*,] R. Braz, R. Girju, V. Punyakanok, D. Roth, and M. Sammons. An inference model for semantic entailment in natural language. In *Proceedings of the National Conference on Artificial Intelligence*.

[Collins, 1999] M. Collins. *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, Computer Science Department, University of Pennsylvenia, Philadelphia, 1999.

[Cumby and Roth, 2002] C. M. Cumby and D. Roth. Learning with feature description logics. In S. Matwin and C. Sammut, editors, *The 12th International Conference on Inductive Logic Programming (ILP-02)*, pages 32–47. Springer, 2002. LNAI 2583.

[Dagan and Glickman, 2004] I. Dagan and O. Glickman. Probabilistic textual entailment: Generic applied modeling of language variability. In *Learning Methods for Text Understanding and Mining*, Grenoble, France, 2004.

[Durme *et al.*, 2003] B. Van Durme, Y. Huang, A. Kupsc, and E. Nyberg. Towards light semantic processing for question answering. HLT Workshop on Text Meaning, 2003.

[Even-Zohar and Roth, 2001] Y. Even-Zohar and D. Roth. A sequential model for multi class classification. pages 10–19, 2001.

[Fellbaum, 1998] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[Hobbs *et al.*, 1988] J. R. Hobbs, M. Stickel, P. Martin, and D. Edwards. Interpretation as abduction. In *Proc. of the 26th ACL*, pages 95–103, 1988.

[Kingsbury *et al.*, 2002] P. Kingsbury, M. Palmer, and M. Marcus. Adding semantic annotation to the Penn treebank. In *Proceedings of the Human Language Technology conference (HLT).*, San Diego, CA, 2002.

[Li *et al.*, 2004] X. Li, P. Morie, and D. Roth. Identification and tracing of ambiguous names: Discriminative and generative approaches. In *Proceedings of the National Conference on Artificial Intelligence*, 2004.

[Lin and Pantel, 2001] D. Lin and P. Pantel. DIRT: discovery of inference rules from text. In *KDD '01*, pages 323–328, 2001.

[Lloyd, 1987] J. W. Lloyd. *Foundations of Logic Progamming*. Springer, 1987.

[Moldovan *et al.*, 2003] D. Moldovan, C. Clark, S. Harabagiu, and S. Maiorano. Cogex: A logic prover for question answering. In *HLT-NAACL*, 2003.

[Moore, 1986] R. C. Moore. Problems in logical form. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*. Kaufmann, Los Altos, CA, 1986.

[Punyakanok *et al.*, 2004] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proc. of the 20th International Conference on Computational Linguistics (COLING)*, Geneva, Switzerland, August 2004.

[Punyakanok *et al.*, 2005] V. Punyakanok, D. Roth, and W. Yih. The necessity of syntactic parsing for semantic role labeling. In *Proc. of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[Schubert, 1986] L. K. Schubert. From english to logic: Contex-free computation of 'conventional' logical translations. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*. Kaufmann, Los Altos, CA, 1986.

[Thompson *et al.*, 1997] C. Thompson, R. Mooney, and L. Tang. Learning to parse NL database queries into logical form. In *Workshop on Automata Induction, Grammatical Inference and Language Acquisition*, 1997.

[Xpress-MP,] Xpress-MP. Dash Optimization. Xpress-MP. http://www.dashoptimization.com/products.html.

Using Information Fusion for Open Domain Question Answering

Tiphaine Dalmas and **Bonnie Webber**
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 bonnie@inf.ed.ac.uk

Content Areas:
web question answering, information fusion, shallow inference

Abstract

In open domain Question Answering, answer candidates are ranked according to individual features such as matching the answer type expected by the question. We report on a technique based on the fusion of candidate answers and their context into answer neighbourhoods to provide better features for ranking and allow shallow reasoning.

1 Introduction

Research efforts in automated Question Answering (QA) have focused on understanding questions to retrieve correct answers. This includes deep parsing, lookups in ontologies, question typing and machine learning of answer patterns appropriate to question forms. In such context, answer candidates are seen as competitors and ranked according to individual features such as match with the expected answer type and number of question words in context. This ignores potentially relevant relations between answer candidates. In recent work [Clarke *et al.*, 2001; Brill *et al.*, 2002], frequency count of answer candidates, i.e. equivalence by string matching, has proved to be useful to identify correct answers. We propose to investigate further relations and focus on the analysis of answer candidates and their context, in the belief that their relationships can be exploited as well as individual features.

Illustrating this intuition is the infamous *Where is the Taj Mahal?* example [Burger *et al.*, 2002]. There are several Taj Mahal in the world, the most famous one being in *Agra, India*. Recognising that the two distinct strings, *Agra* and *India*, apply to the same world referent and building an answer cluster {*Agra, India*} increases the likelihood of either candidate being an answer through the cumulative frequency counts of both occurrences. Considering *Agra* and *India* as competitors reduces that likelihood. The same goes for other correct answer candidates such as *Atlantic City*, *New Jersey*. The first intuition is thus to try to automatically discover such relations between answer candidates. After a first experiment using WordNet to relate candidates, we noticed that answer candidates that would be judged to be incorrect answers were actually of interest. In a second experiment, we performed fusion not only on answer candidates but also on phrases from the context, e.g. *architecture*, *Trump*, *casino*. We introduced a new linking based on word co-occurrences and built more topic-oriented clusters. The *Taj Mahal* question can then be provided with distinct clusters such as {*architecture, Agra, India*}, {*casino, Trump, Atlantic City, NJ*}.

In this paper, we first motivate research on multiple answers, showing that they are not a rare case and thus one could benefit of such multiplicity. We review work in information fusion and its applications in QA. We then describe two experiments based on answer comparison and information fusion to identify answer neighbourhoods. We show that such modeling, even based on shallow techniques, improves the overall accuracy and robustness of a baseline QA system by 20%.

2 Background

2.1 Multiple Answers

The background to this work is the quantity of questions with multiple answers. To quantify this, we investigated cases where different extractions were considered acceptable answers to questions in TREC QA [Voorhees, 2002] and in a corpus of reading comprehension tests produced by the MITRE corporation based on texts from CBC4Kids [Light *et al.*, 2001].

For TREC QA, we calculated the percentage of multiple answers using the patterns provided by NIST judges to evaluate systems. These patterns are regular expressions, one for each similar answer (in terms of pattern-matching). We counted each separate line of patterns as a separate answer.

The figures are given below[1]. These figures are actually an *under-count* because a single line can stand for several answers, e.g. the single regular expression `(18|19|20) million` corresponds to three different answers. But even with this *under-count*, the proportion of multiple extractions is significant.

[1]The proportion of multiple answers in TREC 11 is significantly less because systems were required to give only one answer per question. Still, several questions had more than one distinct answer pattern.

TREC	8	9	10	11	CBC
# questions	200	693	500	500	481
No answer	2	11	67	56	0
Single answ.	129	304	211	378	173
Multi. answ.	69	378	222	66	308
% of multiple	34.5	54.5	44.4	13.2	64

There are several reasons for multiple answers, including ambiguity or complexity in the question, variability or how the same information is presented, and possible contradictory information. (See [Buchholz and Daelemans, 2001] for a case study of complex answers.) Nonetheless, the number of answers there are to a question is *not* proportional to the size of the corpus (around 3GB for TREC questions, 500 words for CBC questions) but rather to how *informative* the corpus is with respect to a question.

Our original intuition concerned *types* of questions which could be answered in more than one way. Classification of questions in the TREC 8 through TREC 10 test sets by their WH-word leads to the following six classes (representing around half of the initial corpus) along with the frequency with which they have more than one answer.

class	# questions	Multiple answ.
who/famous-for	215	40.5%
when	93	44.1%
how-adj/adv	107	46.7%
where	118	64.5%
why/cause-effect	16	75%
definition	120	80.8%

While definition and cause/effect questions very often have more than one acceptable answers, the proportion is high for all six question types. This goes beyond the usual distinction in TREC QA between factoid questions (expecting only one answer) and definition/list question (for which multiple answers are allowed).

As the designer of a QA system, there are two things one can do with multiple answers: simply choose among them, or try to use them – for example in a complex answer. If answers are extractions (strings), the only way of using them is to count how often each occurs and choose the one with the highest frequency. (This is essentially what is done in [Brill *et al.*, 2002]). If we move beyond strings though, we can exploit multiple answers in additional ways that allow greater movement towards fluent and user-oriented answers. We believe fusion, as a mean of merging information and structuring data, will allow that step.

2.2 Information Fusion

Information fusion is a term that refers to the merging of information that originates from different sources. Fusion is not always required, even for complex answers. For example, among answer candidates for the question *What were Christopher Columbus' three ships?* are the extractions *the Nina, the Pinta, and the Santa Maria*; *The Santa Maria, The Nina, and The Pinta*; *Pinta ship*. The answer patterns that TREC QA has for assessing answers to this question are: *Nina.*Pinta.*Santa Maria*, *Santa Maria.*Nina.*Pinta*, and all the required variants to match the correct coordination. However, complex answers have not always been anticipated in a single text. Parts of the answer may occur independently, in which case the answer has to be *reconstructed* by analyzing the *relationships* occurring between nuggets of information found in different places.

In multi-document summarization, which has similar problems of data redundancy and heterogeneity in data as QA over large corpora, [Mani and Bloedorn, 1999] have proposed a graph-based technique to identify similarities and differences among documents in order to *construct* a summary, while [Barzilay *et al.*, 1999] report a technique based on information fusion to *generate* new sentences summarizing information dispersed in several documents.

In QA, [Girju, 2001] demonstrated the benefit of *answer fusion* to retrieve list of correct answers. Her approach is top-down, directed by question type (cause, effect and definition) for which relational patterns, such as *X caused by Y*, were pre-computed to unfold a dynamic ontology from the extractions found in the search corpus.

In this paper, we demonstrate the value of a bottom-up approach to fusion without pre-computing relational patterns and applicable to any question type. The next sections describe QAAM, our software for QA Answer Modeling, and report results for two evaluations: (1) one experiment on location questions and (2) another one based on TREC QA open domain questions using the web as search corpus.

3 QA Answer Modeling

Our answer models are directed graphs in which nodes correspond to entities projected from the question and candidate extractions and edges convey relationships between them. The graph represents the fusion of information contained in the set of extractions. To generate such a model, two steps are required: (1) normalizing the extractions to be projected as nodes into the model and (2) discovering relationships between them. Once a model is built, answer candidates can be ranked using both individual features and graph features (i.e. fusion-based features). The system we implemented to automatically generate models is called QAAM.

3.1 Projection

We use *extractions* as a broad term to refer to answer candidates. Extractions can actually be of several kinds: a passage, a sentence, a phrase or simply a keyword. In our first experiment, we used answers provided by other TREC QA systems. Such answers were usually nominal phrases and we projected these directly, otherwise the extraction was split into nominal phrases. In our second experiment, QAAM's input was a list of web snippets split into sentences, and nominal phrases were projected into nodes.

To standardize such phrases, we represent each as a list of attribute-value pairs (features). On the work reported here, we use three features: (1) the list of tokens in the phrase, (2) the list of lemmas (first experiment) or stems (second experiment) and (3) the extraction and position at which the phrase has been found. Normalized extractions form the nodes of the graph. We apply the same process to the question as to answer candidates in order to discover relations not only between potential answers but also between the question and the extractions. We distinguish three types of nodes:

- *question nodes* projected from the question
- answer nodes that directly match an answer type expected by the question type. (A distance question expects a number followed by a distance unit.) We currently have 20 question types and 84 patterns. We call these directly matching nodes *nuclear* nodes, i.e. nodes that contain a core information.[2]
- remaining nodes are called *satellites*.

For answers that are only weakly typed, nodes that are not question nodes are all considered as nuclear. (For instance, we do not have a specific pattern for question expecting a definition.)

The next step consists in discovering what the relationships between nodes are.

3.2 Relation Discovery

There are infinitely many relations that could hold (1) between a question and its answer candidates, and (2) among answer candidates.

Relations between question and answers have been studied in the context of question typing, and several strategies have been proposed. Question ontologies are helpful to anticipate the type of the answer, especially Named Entities (e.g. location questions or questions asking for a person name). [Prager *et al.*, 2001] describe the use of WordNet hypernyms to answer definition questions. [Girju, 2001] focused on cause-effect relationships, as well as hypernym relations for definition questions.

There are fewer studies on relationships that could hold between answers. This research falls into two categories. Several groups, including [Clark *et al.*, 2001; Brill *et al.*, 2001], use answer redundancy and frequency, i.e. answer equivalence, to help answer selection. Other groups, such as [Buchholz and Daelemans, 2001] and [Webber *et al.*, 2002], have proposed different formalizations of answer relations to handle multiple and/or complex answers. Table 1 proposes a comparison of the two.

Both approaches assume that the system has found correct answers and the considered relations are between correct answers only. Our approach considers all answer candidates (correct and incorrect) as well as question words. For that reason, we also prefer the term *information fusion* rather than *answer fusion* (Girju), as our modeling involves fusing extractions that are not always correct answers. This is a realistic claim for two reasons: (1) QA is still far from providing 100% accurate answer extractions[3] and (2) an answer can be composed of elements that, considered independently, are not answers (e.g. *Pinta ship* alone is not a correct answer to the question *What were Christopher Columbus' three ships?*) and related candidates, although not fitting the answer type (e.g. *Morse/1844* for *When was the telegraph invented?*), can also help answering.

[2]These patterns should not be confused with the relational patterns that Girju uses for top-down fusion. We use patterns make a distinction between different kinds of nodes, not to identify their relations.

[3]The best score on factoid questions at TREC QA 13 was 77%, the median score being 17%, which shows that the average QA system performance is still low.

Table 1: Relationships between answers

[Webber *et al.*, 2002]	[Buchholz and Daelemans, 2001]
answers determined to be equivalent (mutually entailing)	*different measures*, *different designations*, *time dependency*
answers that differ in specificity (one-way entailing)	*granularity*
answers that are mutually consistent but not entailing can be replaced by their conjunction (aggregation)	*collective answers*
answers that are inconsistent, or *alternative answers*	*many answers*, *ambiguity in the question*, *different beliefs*

Instead of defining templates in the lines of research in Information Extraction (see for instance [Yangarber, 2000]), in which, for a time question about an event, one searches for placenames or actors related to a given date, we propose to look at two relationships: equivalence and inclusion. (We use $\leftrightarrow$ to denote an equivalence and $\rightarrow$ an inclusion.) In our second experiment, we introduce contextual comparison (word co-occurrences) to our modeling.

Relations are infered in a unsupervised way. Models are automatically generated using the techniques described below. Both experiments evaluate models on a task-based basis, so we do not have figures for the precision and quality of the models. The reason for choosing a task-based evaluation is because models generate many relationships. Given N candidates, the size of required matrix to encode relations is $(N * (N-1))/2$, e.g. for a question with 50 nodes (question words plus answer candidates), 1225 relations would have to be evaluated against a previously annotated gold standard, and this for each question. Since we are yet experimenting with the type of relations to be discovered, we stick to unsupervised inference and use a task-based evaluation.

We now review the techniques we use for inference. Equivalence is infered using either WordNet lookups (synonyms) or shallow string comparison (lemma/stem comparison, abbreviation recognition and Edit-Distance similarity). Table 2 provides examples for each technique.

Table 2: Equivalence Inference.

Techniques	**Examples**
Lemmatization/stemming	illnesses $\leftrightarrow$ illness
Abbreviation	US $\leftrightarrow$ United States
Edit-Distance	Hindenburg $\leftrightarrow$ Hindenberg
WordNet synonym	treaty $\leftrightarrow$ pact

To infer inclusion, we use four WordNet pointers: hyponym, meronym, partonym and membership, and an approximation of inclusion based on overlap computation and context analysis (see Table 3). Contextual inclusion is computed by checking the overlap between the context feature (see Section 3.1) of two nodes. For instance if the context of *bipolar disorder* is a subset of the context of *mental illness*, the system makes the assumption that *bipolar disorder* entails *mental illness* because each time *bipolar disorder* is mentioned somewhere, it co-occurs with *mental illness* (but not the reverse because *mental illness* also occurs independently elsewhere). This form of inclusion/entailment was inspired by work on light-weight inference [Monz and de Rijke, 2001].

Table 3: Inclusion inference.

Techniques	**Examples**
WordNet hyponym	symptom → vertigo
WordNet meronym	water → hydrogen
WordNet partonym	Scotland → Edinburgh
WordNet membership	European Union → France
Lexical head	expert → sunspot expert
Bag subset	sun → sun core
Context	mental illness → bipolar disorder

3.3 Identifying Answer Neighborhoods

Once a model is built, the topology of the graph provides clues for ranking nodes and identifying answer neighborhoods, i.e. highly connected group of nodes. Figure 1 shows a model generated in our first experiment on location questions for the question *Where is Glasgow?*. Each node corresponds to a candidate answer (*London* appeared twice, and *Glasgow* appeared among candidates). This graph has two partitions: A is the largest one and contains the correct answer nodes *Britain* and *Scotland*, linked by inclusion, and both pointing to *Glasgow*. Partition B (*Munich*) is an isolated node. Such neighborhood properties give clues to which node could be a correct answer and which parts of the graph refer to a same answer or to potential alternatives. For example *Glasgow*, *Manchester* and *London* are distinct siblings, while *Britain*, *Scotland* and *Glasgow* are on the same branch.

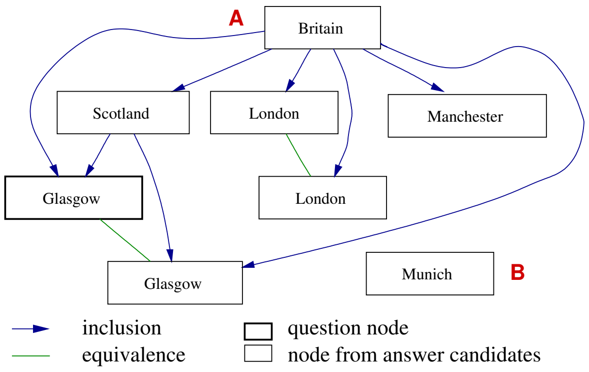


Figure 1: Answer model for *Where is Glasgow?*

Figure 2 shows that, even if no relation had been inferred between a question node and the other nodes, fusion still helps identifying which cluster of nodes is more likely to contain a correct answer given the question *Where is the Valley of the Kings?* The cluster {*Egypt*, *Luxor*} would be considered more likely than the singleton *Ohio*.

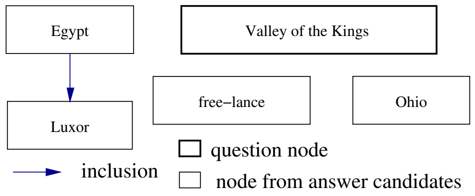


Figure 2: Answer model for *Where is the Valley of the Kings?*

Notice that in this pilot experiment, we assumed all nodes to be nuclear (apart from question nodes). Thus *free-lance* was an answer candidate but, being isolated, it had a low ranking.

To rank candidate answers based on the model, QAAM evaluates different properties of each node to assess the likelihood of a node's being an answer. Such properties are individual (node type) as well as graph-based computations, e.g. number of children, partition size, relations with a question node. (See next section for the selection algorithm.)

The selection can vary in specificity and amount of information provided. For instance, a general answer to *Where is Glasgow?* corresponds to the root node *Britain*. A detailed answer would output the full path from Glasgow, i.e. *Scotland, Britain*. Models provide inference mechanisms for further processing of the answers, for instance more elaborated answers. However, there is not yet any evaluation material to assess the correctness of such answer clusters. The evaluations in this paper consist in displaying a rank-ordered list of nodes and evaluating it against answer patterns. Such evaluation assesses the quantitative value of fusion in terms of answer correctness but does not evaluate the quality of the clustering process.

4 Pilot Experiment on Location Questions

The object of this first experiment was to check the feasibility of answer modeling on real data, given that QA systems still produce many incorrect answers. [Greenwood, 2004] notes that *the average performance by participants in TREC 11 was approximately 22%*. Only the best TREC 11 system [Moldovan *et al.*, 2002] correctly answered 415 question over 500, for a score of 85.6%. Given this average performance, the experiment was meant to determine whether it is worth generating answer models to derive better answers under such conditions.

To check the impact of information fusion, we compared the approach described in Section 3 against a technique that compares each extraction with the question but ignores relations among the extractions.

We selected 85 questions from TREC 8 to TREC 11 that request locations and for which there is an obvious inclusion or entailment relationship between a word of the question and the expected answer (*Where is X?* X is part of <*answer*> or

<*answer*> 'includes' X). Candidate extractions were taken from the list of judgements produced by TREC systems. For each question, we assigned it 5 incorrect answer candidates and 1 to 5 correct ones, depending on the number of correct judgements available. 44.7% of the questions had one correct answer, 17.6% had two, 12.9% had three, only 3.5% had four and 21.3%, five.

The task was defined as a ranking problem. As described in Section 3, QAAM takes as input a question and a list of extractions, generates a graph and outputs an ordered list of strings, each corresponding to a different node. We inferred equivalence and inclusion using WordNet, abbreviation recognition and lexical head comparison. Contextual comparison was not included, as TREC judgements correspond to exact answers and thus a limited context.

Ranking was based on comparing the following node properties:

(a) Does the node derive from the question or is it equivalent to a question node?
(b) How many question nodes is the node directly related to?
(c) How many question nodes are present in its partition of the graph?
(d) How large is its partition?
(e) How many children does it have by transitive inclusion?
(f) How many nodes is it equivalent to?

(a) excludes nodes that paraphrase a question node such as *Glasgow* in Figure 1[4]. (d) measures the size of the partition of the node being considered. (f) measures redundancy, while (b) and (c) check the relation of the node to the question. (e) gives a measure of specificity. The fewer are its children by inclusion, the more specific we take a node to be. For instance, *Britain* has more children by inclusion than *Scotland* and is thus considered less specific. Notice that (d), (e) and (f) count both question and answer nodes.

The two following approaches were then compared:

- `-fusion`, which sorts nodes based only on features that relate the question and a single answer – i.e. (a) and (b).
- `+fusion`, which makes use of all the features in the order: (a), (b), (c), (d), (e) and (f), and reflects relations between (i.e. fusion of) multiple nodes. The order defines a preference: For instance, a specific node (e) that occurs only once would be preferred to a less specific node that is more redundant (f).

Notice that features (c), (d), (e) and (f) make use of relations between any kind of nodes and thus cannot be used by `-fusion`. For instance, according to feature (c), *London* in Figure 1 is related to one question node, *Glasgow*, by following paths between answer nodes. This cannot be used for `-fusion`. If the path does not actually contain any answer node, it is then equivalent to feature (b), which is a `-fusion` feature. Finally, the node with the longest string was selected for tie-breaking.

[4]Only for questions such as *What do you call a newborn kangaroo?* should the answer be a paraphrase of (part of) the question.

The final list of rank-ordered nodes was then evaluated against the TREC answer patterns. Table 4 shows that QAAM, when it made use of relationships between answer nodes (`+fusion`), identified more correct answers than it did by simply looking up relations between question nodes and answer nodes (`-fusion`). The *inclusion* relationship was especially useful to detect clusters of related answers and was the main relation inferred (2/3 of the relations). What was surprising was the proportion of relations involving incorrect answers (2/3 of the total number of relations). To check the role of incorrect answer nodes, we assumed an oracle could identify correct answer nodes so that only relations involving correct answer nodes and/or question nodes are considered.

Table 4: FRS stands for first rank score, i.e. the percentage of questions for which each system ranked a correct answer first. MRR (Mean Reciprocal Rank) measures the overall reranking strategy.

	standalone		with an oracle	
	FRS	MRR	FRS	MRR
`-fusion`	49%	0.63	65%	0.71
`+fusion`	72%	0.82	78%	0.85

While an oracle significantly improved the performance of the `-fusion` system, the improvement was significantly less when using fusion. The models contained two kinds of incorrect answer: (1) out of topic and (2) wrong but related. The latter kind helped building larger partitions more likely to contain a correct answer. Overall, a strategy that carries out information fusion among answers candidates (even incorrect ones) is not only better but more robust and resistant to incorrect answers than a strategy that considers only relations between questions and answers alone.

5 Web Answer Model Generation

Our second experiment addresses two limitations of the first experiment, considering additional question types and a greater diversity of answers.

In order to get a greater diversity of answers, we used the web as a search corpus rather than ACQUAINT (TREC QA corpus). This highlighted the fact that a more data-mining oriented interpretation of the QA task would view it as determining what the answers are for that question *with respect to a specific corpus*. For instance, given the question *What is vertigo?*, TREC 10 systems found the following answers in the AQUAINT corpus: *dizziness*, *disorientation*, *sensation motion*, *tinnitus*, *skewed balance*. The same question posed to the Web using Google finds, in addition, that it is a company, a Photoshop plug-in, a paragliding and hang-gliding competition, a comics series and an Alfred Hitchcock movie. (The AQUAINT corpus contains mentions of the film but it was not among the accepted answers because a TREC QA requirement is to provide the most expected answer). TREC QA systems make extensive use of external knowledge sources, such as WordNet, in which the only interpretation for vertigo is the medical symptom. Answers are thus biased towards the resource used, instead of being representative of the search corpus (hence our bottom up approach).

In this second experiment, we used *Wee*, our own shallow Google-based QA system, to provide us with web sentences from which we could build models and extract candidate answers. For each question, Wee posed a query to Google consisting of keywords from the question and took back the top 100 snippets that Google returned. Snippets were split into sentences (about 350 sentences per question), tokenized, and given as input to QAAM. Node features were reduced to a normalisation of answer strings by stemming. To infer the equivalence relationship, we used simple string matching and the Edit-Distance algorithm to deal with misspellings, which are frequent in web data. WordNet lookups used for relation inference were replaced by overlap computation and a shallow context comparison.

We developed this version of QAAM on TREC 10 and evaluated it on TREC 11. For development, we used an extended set of answer patterns that included correct web answers that were not present in the TREC set. Results for TREC 11 were evaluated with TREC answer patterns only. We compared three strategies against a sentence level baseline (Wee output). The models were based on the fusion of the top 100 sentences (taken from the approximately 350 sentences associated with the 100 Google snippets) of the baseline. Each strategy generated a cluster of nodes as answer. The lists differed in how answers were partitioned into answer neighbourhoods. Examples for each strategy are given below (*What is vertigo?*):

- **twins** partitions the answer model into clusters of equivalent nodes: {*dizziness, DIZZINESS*}, {*Hitchcock*}, {*Alfred Hitchcock*}, {*software*}, {*3D plug-ins for Adobe*}, {*sensation of spinning*}.
- **family** identifies family clusters, i.e. nodes considered equivalent or related by an inclusion relationship based on overlap, e.g. *Hitchcock* → *Alfred Hitchcock*): {*dizziness, Dizziness*}, {*Hitchcock, Alfred Hitchcock*}, {*software*}, {*3D plug-ins for Adobe*}, {*sensation of spinning*}.
- **extended family** generates clusters of nodes of the same family or related by their context: {*sensation of spinning, dizziness, Dizziness*}, {*Hitchcock, Alfred Hitchcock*}, {*software, 3D plug-ins for Adobe*}.

During this experiment, we noticed question nodes aggregated many answers nodes because they are ambiguous. They generated large clusters that were not comparable with sentence level answers. For instance, *vertigo* would cluster nodes referring to individual symptoms of the disease as well as to features of the film and software. To disambiguate such clusters and reduce the size of partitions, we considered question nodes as barriers to partitioning.

Clusters were then ranked by size (sum of the frequency of each member) and evaluated against answer patterns to assess their correctness. Sentences provided by the baseline (Wee output) were assessed in the same way and we compared the recall for each strategy. To *What is vertigo?*, the first sentence provided by the baseline was:

National Institute on Deafness and Other Communication Disorders The primary NIH organization for research on Dizziness and Vertigo is the National Institute.

Notice the length of an answer cluster is similar in length to a sentence-based answer, however the keyword density is higher. (Results are not comparable with TREC results: A TREC answer is a key phrase whereas our answers are either a sentence (baseline) or a list of key phrases (fusion approaches).)

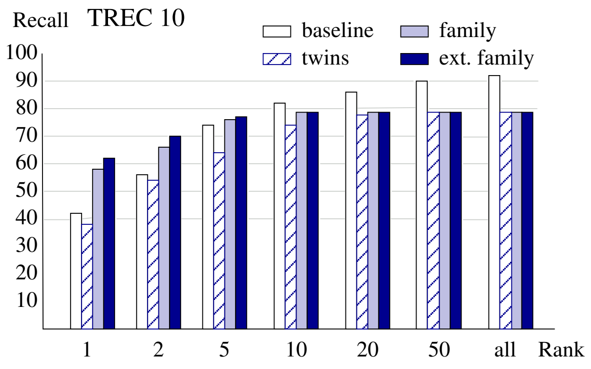


Figure 3: TREC 10 results

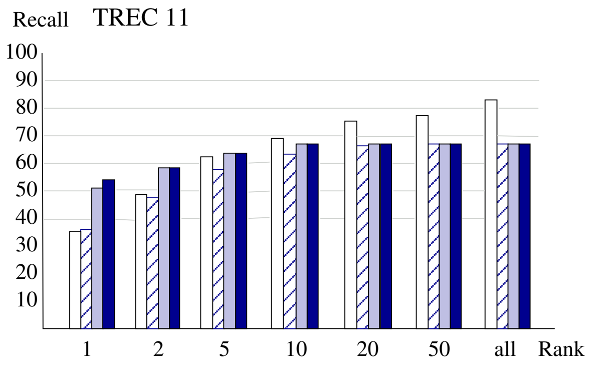


Figure 4: TREC 11 results

Figures 3 and 4 show average *recall at rank* for TREC 10 and TREC 11 questions.

The baseline results are low on first-ranked answers. As is often the case, the more answers accepted, the better the recall (up 92% for TREC 10 and 83% for TREC 11) but the lower the precision. On TREC 10, the best recall is achieved when providing on average 200 candidate extractions. This means that more than 80% of TREC questions are answerable using the web. However, at rank 1, which is the level of accuracy required for QA, sentence-based reranking provides only a third of the questions with a correct answer.

Both data sets show that the main improvement was obtained by using fusion and especially the *inclusion* relationship (*family* and *extended family*), with the best results achieved using *extended family*. In both TREC 10 and 11, baseline recall was improved by 20% for first ranked answers. On average, QAAM proposed 12 distinct partitions as answers per question, which resulted from the fusion of the top 100 snippets. The baseline uses the full set of snippets (on average 350 per questions), which explains why the baseline performs better on the long run. With fusion, recall at rank thresholds around rank 10 on both figures and indicates how much more precise the fusion strategy is compared to the baseline.

Relation discovery, and consequently fusion, performed at different levels depending on the question type. Questions asking for a definition (*What is e-coli?*, *What does CPR stand for?*) or a biographic information (*Who was Galileo?*) and factoid questions asking for a term, a location or a date were best answered using fusion. Numeric answers (e.g. duration, weight, distance) were more difficult to compare because unit variations and number approximations led to smaller partitions with similar answer likelihood. Answers describing an action (*What is an eclipse? moon blocks sunlight*) could not be fused properly since only nominal phrases were projected.

However, this large-scale experiment (TREC 10 and 11 represent 1000 questions) shows that information fusion can serve as a strong basis for reranking for any question type, even when based on shallow techniques. Although clusters are not exact answers, they are better nests for correct answers than plain sentences, and their structure can serve as the basis for shallow reasoning over answer relationships.

6 Conclusion and Future Work

In this paper, we demonstrated, through two experiments we carried out, the value of information fusion and answer comparison in terms of answer correctness.

We have shown that incorrect but related answers can help to build a supportive network around a correct answer and to interpret and eventually classify results. Such nodes, which we call *satellites*, appear to be of two kinds: (1) words denoting or relating to a question type (e.g. the word *"location"* for spatial questions) and (2) words relating to the world referent of a correct answer or an event involved in it (*architecture, mausoleum* for the Indian Taj Mahal, or the birth date for a question asking for a birth place). Questions for which the expected answer has a very specific linguistic form (e.g. questions asking for a year, i.e. a four-digit answer) are easier to answer from QAAM clusters because the distinction between satellite and nuclear nodes can be done using pattern matching. For questions with a less well defined answer type, such as definition questions, the distinction is not as straightforward. For instance *What is pastrami made of?* is well answered with *beef*. *Food*, although conceptually true, is incorrect because too general. In future work, we would like to investigate further QAAM graphs to identify ways of automatically classify satellite versus nuclear nodes. In the pastrami example, the fact that *food* is too general of a concept to be given as answer is evident from QAAM graph because it includes many other nodes (*meat, beef, turkey...*) and includes *pastrami* itself. Besides the advantages of clustering, introducing collective features, it seems worth investigating what further inference mechanisms, based on relationships, can be used from the models to improve answer selection and eventually answer generation.

Another research area we would like to explore is the value of clustering for further answer processing. In current evaluations, *Agra* and *India* are considered as two distinct correct answers. Using fusion, it is possible to identify them as referring to the same answer. As mentioned above, there is not yet any evaluation material to assess the correctness of such answer clusters. An automated way of evaluating without involving heavy annotation work could be to compare the results of the two strategies, with and without fusion, in a rendering task such as picture retrieval or summarization. Since QAAM clusters are structured and contain background information, we believe that a rendering based on fusion would better acknowledge answer multiplicity, considering an answer as a structured entity as opposed to a string.

Acknowledgment Tiphaine Dalmas's work is funded by the School of Informatics at the University of Edinburgh.

References

[Barzilay *et al.*, 1999] R. Barzilay, K. R. McKeown, and M. Elhadad. Information Fusion in the Context of Multi-Document Summarization. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 550–557, 1999.

[Brill *et al.*, 2001] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data Intensive Question Answering. In *10th Text Retrieval Conference*, pages 393–400, 2001.

[Brill *et al.*, 2002] E. Brill, S. Dumais, and M. Banko. Analysis of the AskMSR Question-Answering System. In *Empirical Methods in Natural Language Processing Conference*, 2002.

[Buchholz and Daelemans, 2001] S. Buchholz and W. Daelemans. Complex answers: A case study using a WWW question answering system. *Natural Language Engineering 1 (1)*, 2001.

[Burger *et al.*, 2002] J. Burger, C. Cardie, V. Chaudhri, R. Gaizauskas, S. Harabagiu, D. Israel, C. Jacquemin, C. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weishedel. Issues, Tasks and Program Structures to Roadmap Research in Question and Answering. *NIST*, 2002.

[Clark *et al.*, 2001] C. L. A. Clark, G. V. Cormack, and T. R. Lynam. Exploiting Redundancy in Question Answering. In *24th ACM-SIGIR International Conference on Research and Development in Information Retrieval*, pages 358–365, 2001.

[Clarke *et al.*, 2001] C. L. A. Clarke, G. V. Cormack, and Th. R. Lynam. Exploiting Redundancy in Question Answering. In *24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 358–365, 2001.

[Girju, 2001] R. Girju. Answer Fusion with On-Line Ontology Development. In *North American Chapter of the Association for Computational Linguistics - Student Research Workshop*, 2001.

[Greenwood, 2004] Mark Greenwood. Answer Finder: Question Answering from Your Desktop. In *Computational Linguistics UK*, pages 75–80, 2004.

[Light *et al.*, 2001] M. Light, G. Mann, L. Hirschmann, E. Riloff, and E. Breck. Analyses for Elucidating Current Question Answering technology. *Natural Language Engineering*, 7(4):325–342, 2001.

[Mani and Bloedorn, 1999] I. Mani and E. Bloedorn. Summarizing Similarities and Differences Among Related Documents. *Information Retrieval*, 1:35–67, 1999.

[Moldovan *et al.*, 2002] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatasu, A. Novishi, A. Badulescu, and O. Bolohan. LCC Tools for Question Answering. In *11th Text Retrieval Conference*, 2002.

[Monz and de Rijke, 2001] C. Monz and M. de Rijke. Light-Weight Inference for Computational Semantics. In *Inference in Computational Semantics*, pages 59–72, 2001.

[Prager *et al.*, 2001] Prager, Chu-Caroll, and Czuba. Use of WordNet Hypernyms for Answering What-Is Questions. In *10th Text Retrieval Conference*. NIST, 2001.

[Voorhees, 2002] E. M. Voorhees. Overview of the TREC 2002 Question Answering Track. In *11th Text Retrieval Conference*, page 1. NIST, 2002.

[Webber *et al.*, 2002] B. Webber, C. Gardent, and J. Bos. Position statement: Inference in Question Answering. In *LREC Workshop on Question Answering: Strategy and Resources*, pages 19–26, 2002.

[Yangarber, 2000] R. Yangarber. *Scenario Customization for Information Extraction*. PhD thesis, New York University, 2000.

Supervised Machine Learning Techniques for Question Answering

Ingrid Zukerman and Pawel Kowalczyk and Michael Niemann
School of Computer Science and Software Engineering
Monash University
Clayton, VICTORIA 3800, AUSTRALIA
{ingrid,pawel,niemann}@csse.monash.edu.au

Bhavani Raskutti
Telstra Research Laboratories
770 Blackburn Road
Clayton, VICTORIA 3168, AUSTRALIA
Bhavani.Raskutti@team.telstra.com

Abstract

In this paper, we discuss three components of our question answering system, focusing on our application of machine learning techniques. In particular, we describe our use of Support Vector Machines (SVMs) for the selection of sentences that contain answers to queries. Our SVMs are trained on attributes that reflect two types of information: (1) semantic query features, and (2) the relationship between these features and candidate answer sentences. Our evaluation yields encouraging results, pointing the way to further investigation of machine learning techniques.

1 Introduction

The growth in popularity of the Internet highlights the importance of developing systems that generate responses to queries targeted at large unstructured corpora. The development of systems for question answering has been investigated for some time, and has been allocated significant human resources. This human investment has prompted the investigation of machine learning techniques for different aspects of question answering, e.g., [Suzuki *et al.*, 2002; Zhang and Lee, 2003]. The research discussed in this paper is part of this trend. We report on the application of machine learning techniques for the following tasks.

- Query analysis – where we automatically classify queries according to their type (Section 2.2). This classification is used in the sentence selection process.
- Document retrieval – where we learn features of query words that affect retrieval performance (Section 3).
- Sentence selection – where we learn features of queries and of their relationship with candidate sentences that help us find sentences that are most likely to contain the answer to a query (Section 4.3). We are currently focusing on factoid answers. However, our techniques may be adapted to extracting other types of answers from candidate sentences.

In Sections 2, 3 and 4, we describe our use of machine learning techniques and the obtained results for query classification, document retrieval and sentence selection respectively. In Section 5 we discuss related research, followed by concluding remarks.

2 Query Analysis

Query analysis consists of two main sub-tasks: feature extraction and query classification. Feature extraction obtains features that are useful for classification as well as for later stages of the question answering process. Query classification divides queries into different question types, which are indicative of the kind of answer that is expected. The idea is that this distinction should assist in the sentence selection process (and later in answer extraction). Hence, the Support Vector Machines (SVMs) used for sentence selection are trained separately for each question type (Section 4.3).

2.1 Query feature extraction

We extract 11 query features that are used to train the SVM that performs query classification (Section 2.2) and the SVM that performs sentence selection (Section 4.3). Our SVMs are trained using SVM^{light} (`http://svmlight.joachims.org/`). The query features are obtained by first parsing the query using Charniak's probabilistic parser (`ftp://ftp.cs.brown.edu/pub/nlparser/`),[1] and then performing rule-based extraction. The application of the extraction rules depends on the syntactic structure of the query as described below. Figure 1 illustrates the parse tree and the seven non-null features for the query "What two European countries are connected by the St. Gotthard Tunnel under the Alps?".

- **Topic** – what the query is about. The topic is chosen in the following order: (1) the first named entity in the query, or (2) the head noun of a what/which question, or (3) the grammatical object, or else (4) the first NP in the sentence.
- **Answer type** – the type of the expected answer. If the query starts with a `[wh NP]`, the answer type is the NP. Otherwise, if the question has the form `[wh be OBJ]`, then the answer type is the NP of the object, e.g., who was *the US president* in 1929?
- **Action** – the main verb in the query if it isn't "be", or the main verb from the relative phrase (`RelP`) if the query has the form `[wh be NP RelP]` or `[wh NP be NP RelP]`.
- **Named entities** – sets of proper nouns that appear in the same noun group.

[1]Charniak's parser was trained on the Penn Treebank corpus, which contains mainly declarative sentences. In the future, we plan to retrain the parser on a corpus that includes more questions.

Query:
What two European countries are connected by the St. Gotthard Tunnel under the Alps?

Parse tree:

```
(S1
  (SBARQ
     (WHNP (WP What))
     (SQ (NP (CD two) (NNP European) (NNS countries))
         (VP (AUX are)
     (VP (VBN connected)
                (PP (IN by)
                    (NP (NP (DT the) (NNP St.)
                            (NNP Gotthard) (NN Tunnel))
                        (PP (IN under) (NP (DT the)
                            (NNP Alps))))))))
      (. ?)))
```

Features:

TOPIC:	[European]
ANSWER TYPE:	[two European countries]
ACTION:	[connected]
NAMED ENTITIES:	[European|St. Gotthard|Alps]
SUBJECT:	[two European countries]
CONSTRAINTS:	prepP [by the St. Gotthard Tunnel| under the Alps]
REMAINING LEMMAS:	[What are ?]

Figure 1: Parse tree and features for a sample query.

- **Subject** – the grammatical subject of the query. In general, the subject is the first NP in a query. For queries that have a relative phrase, the subject is the NP to which the relative phrase is attached; and if the query starts with a PP, then the subject is the NP of the PP.
- **Object** – the grammatical object of the query. If a query has a relative phrase, this is the object of the relative phrase. Otherwise, it is the object of the main verb.
- **Adverbial, adjectival, verb-based and prepositional constraints** – these contain phrases found anywhere in the query, except for phrases inside the NPs listed in other features (but they include any other phrases that are attached to these NPs).
- **Remaining lemmas** – query lemmas that are not in any of the other features.

These syntactic features were selected because (1) they carry semantic content which has the potential to lead to the correct answer, and (2) they contain text segments that would normally be found together in the vicinity of the correct answer. The idea is that these distinctions will enable the SVM to assign higher weights to features that are important for identifying an answer (Section 4.3).

2.2 Query classification

We have identified the following 11 query types, which are indicative of the kind of answer that is expected: *location, number, time, attribute, person, process, term, organization, howDoYouSay, object* and *other*. The first five query types, which are detailed below, comprise 85% of the queries in TREC11 and TREC12. The results presented in Section 4.5 are for these query types.

- **location**, e.g., "In what *country* did the game of croquet originate?".
- **number**, e.g., "*How many* chromosomes does a human zygote have?".
- **time**, e.g., "What *year* was Alaska purchased?".
- **attribute** – an attribute of the query's topic, e.g., "What is Australia's *national blossom*?".
- **person**, e.g., "*Who* is Tom Cruise married to?".

It is worth noting that there is nothing intrinsically important about these particular query types. The main factor is their potential to improve question-answering performance.

We used an SVM to classify queries into these query types [Kowalczyk *et al.*, 2004]. The SVM was trained on the features described in Section 2.1, plus two WordNet features [Miller *et al.*, 1990]: the top four WordNet senses for (1) the Action and (2) the Answer Type of the query. The SVM obtained an average recall of 92% and average precision of 93% (averaged over 20 runs) for the 911 queries from TREC11 and TREC12.

3 Document Retrieval

Document retrieval is done using the vector space model moderated by boolean constraints and two adjustments to the TF.IDF score. As seen in Table 1, these modifications significantly boost retrieval performance.

The following constraints were applied to filter documents returned by the vector space model.

C1: At least one of the proper nouns in a query must appear in a candidate document.

C2: If a query has two or more content lemmas (i.e., not a stop word), then the candidate document must have at least two content lemmas.[2]

The following adjustments were applied to the TF.IDF score of a document.

- Adjustment suggested by decision graphs – we used decision graphs [Oliver, 1993] (an extension of the decision trees described in [Wallace and Patrick, 1993]) to analyze the influence of frequency-based query features on retrieval performance. *DGraf*, the decision graph program, identified a region of high retrieval performance for queries whose lemmas have a frequency below a threshold $\tau_F = 1600$ [Zukerman *et al.*, 2003]. We treat this observation as a suggestion for increasing the TF.IDF score of a lemma l as follows.

$$\text{TF.IDF}_\alpha(l) = \begin{cases} \alpha\text{TF.IDF}(l) & \text{if Freq}(l) < \tau_F \\ \text{TF.IDF}(l) & \text{otherwise} \end{cases}$$

  where α (> 1) is an empirically determined factor.
- Empirically obtained adjustment – the TF.IDF score of a document is given extra weight based on the number of query lemmas found in the document, as follows.

$$\text{TF.IDF}(D) = \text{\#ofLemmasFound}^\beta \sum_{i=1}^{n} \text{TF.IDF}_\alpha(l_i)$$

  where n is the number of content lemmas in a query, β is an empirically determined exponent, and l_i is the ith lemma in the query.

[2]Document retrieval is lemma based, i.e., document words are lemmatized prior to indexing, and query words prior to retrieval.

Retrieval method	Answ'ble queries
200 docs:	
Basic TF.IDF	52.7%
TF.IDF+ constraints	69.6%
TF.IDF+ constraints + LemmasFound	84.5%
TF.IDF+ constraints + LemmasFound + DGraf	87.5%
100 docs: TF.IDF+ const + LemFound + DGraf	84.1%
50 docs: TF.IDF+ const + LemFound + DGraf	79.0%

Table 1: Performance of four retrieval methods for 200 documents. Performance of best method for 100 and 50 documents.

The retrieval performance of our system was assessed using the *number of answerable queries,* which returns the number of queries for which the system has retrieved at least one document that contains the answer to a query [Zukerman *et al.*, 2003]. This measure gives an upper bound for the performance of a retrieval or question-answering system. Nonetheless, we find it more suitable for the question answering task than the standard precision measure [Salton and McGill, 1983]. For example, consider a situation where 10 correct documents are retrieved for each of 2 queries and 0 correct documents for each of 3 queries, compared to a situation where 2 correct documents are retrieved for each of 5 queries. Average precision would yield a better score for the first situation, failing to address the question of interest for the question-answering task, namely how many queries have a chance of being answered, which is 2 in the first case and 5 in the second case. This is the number represented in our *number of answerable queries* measure.

Table 1 shows the results obtained for the different retrieval methods when the top 200 documents are retrieved.[3] These results were obtained with $\alpha = 3$ and $\beta = 4$ (these values yielded the best performance among the tried values $1 \leq \alpha \leq 4$ and $1 \leq \beta \leq 6$). It is worth noting that if we accept correct answers from other retrieved documents (in addition to the documents identified by TREC), answerable queries goes up to 89.2% for 200 retrieved documents. However, at present, we are using only the TREC-identified documents for the sentence selection stage. Table 1 also shows the retrieval performance obtained for the top 100 and top 50 documents, which are considered in our sentence selection experiments (Section 4).

4 Sentence Selection

Our sentence selection process receives as input queries for which our document retrieval process found at least one correct document among the top M retrieved documents. We have experimented with lower values of M than those used for document retrieval ($M = 100$ and $M = 50$ instead of 200) owing to time and computer memory limitations. Columns 4 and 5 in Table 2 show the breakdown of the input queries with answers in the top 100 and top 50 documents according to the five main query categories.

[3]Our retrieval performance is calculated for the 842 TREC11 and TREC12 queries that have answers according to the TREC judgment files (Table 2).

Query type	Number of queries			
	total	w answers in TREC	w answers in 100 docs	w answers in 50 docs
Location	207	198	180	175
Number	187	163	134	122
Time	145	139	121	115
Attribute	121	112	88	82
Person	118	106	93	86
Sub-total	778	718	616	580
Other	133	124	92	85
Total	911	842	708	665

Table 2: Five main query types at different stages of the question answering process.

In order to isolate the different stages of the question answering process, we train the sentence-selection SVM using the query classes that were employed to train the query-classification SVM, rather than the classes returned by this SVM (Section 2.2).

The sentence selection process progressively narrows down the space where the answer to a query may be found, starting from a set of documents, and ending with a set of ranked sentences. In the next stage of our project, we will consider the extraction of candidate answers from these sentences.

The initial reduction steps are performed algorithmically, and the ranking of sentences is done by means of SVMs – one SVM for each type of query. The training (and testing) is performed separately for each of the five main query types, since each of these query types have different distinguishing characteristics. Prior to generating candidate sentences for SVM training and testing, we perform the following actions on each of the top M retrieved documents: (1) document reduction, and (2) sentence filtering.

4.1 Document Reduction

In this step, we estimate the portion of a retrieved document that is likely to contain the answer to a query, and retain only this portion for further processing. First, we find the *document center* – the shortest span in the document that contains the query content lemmas that were found in the document. For example, consider a query that contains content lemmas $l_1 l_2 l_3$. If only l_1 and l_3 appear in a document, and the shortest span between these lemmas corresponds to positions p_1 and p_3, then the document center comprises the lemmas between these two positions. In addition, due to buffer-size limitations, we remove documents whose center exceeds L_C sentences ($L_C = 40$).

We then take L_S sentences before and after the document center (bounded by the beginning/end of the document). These sentences plus the center constitute the area of the document where the rest of the processing is done. Figure 2 shows the ratio of positive to negative SVM vectors (corresponding to sentences with and without correct answers respectively) as a function of L_S for $L_S = \{2, 5, 10, 15, 20, 40\}$. As can be seen from this Figure, for Time and Location queries, the number of negative vectors is

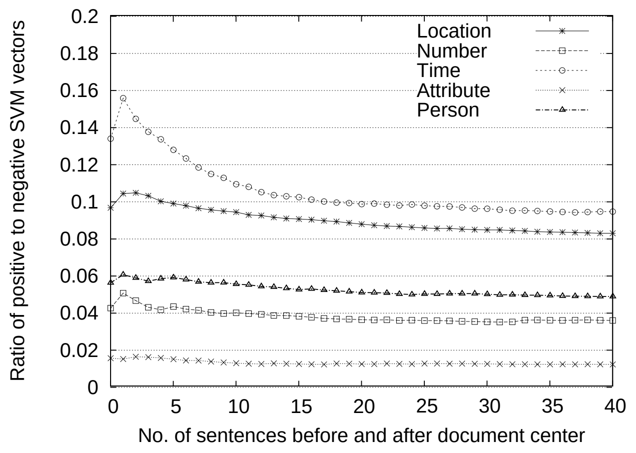


Figure 2: Ratio of positive to negative SVM vectors as a function of number of sentences before and after the document center (top 50 documents).

about 10 times larger than the number of positive vectors, for Person and Number queries, it is about 20 times larger, and for Attribute queries about 50 times. In addition, for the first four types of queries, the ratio of positive to negative vectors improves slightly when two sentences around the center are considered, while for Attribute queries, the ratio remains fairly constant. In Section 4.5, we examine the relationship between these ratios and question-answering performance.

4.2 Sentence Filtering

In this step, we eliminate sentences that are unlikely to contain the answer to a query. To this effect, we (1) identify noun groups (NGs) that are likely to meet the criteria of the query type under consideration,[4] and (2) remove sentences that are unlikely to contain suitable answers.

Identifying NGs that meet the query criteria

First, we use Charniak's probabilistic parser to parse all the sentences obtained in the previous step, and extract all the NGs from these parse trees. The identification of NGs that meet the query criteria depends on the query type. We perform two types of NG identification: *positive* and *negative*.

- Positive identification consists of recognizing NGs that meet the query criteria. This allows us to confidently remove the remaining candidate NGs. Numbers, and dates and times have been automatically tagged in a copy of the corpus, therefore positive identification is done for Number and Time queries. In addition, locations and people are usually identified by means of proper nouns, hence we retain only NGs composed mainly of proper nouns (possibly interleaved with prepositions) for Location and Person queries. In contrast, for Attribute queries we retain NGs composed of common nouns.
- Negative identification consists of removing NGs that clearly do not meet the query criteria. This is done using manually constructed *filter files*, which contain designations for honorifics, locations (e.g., city, airport, school), organizations (e.g., company, society), currency, etc, and common people names. The choice of filter file(s) depends on the query type. For instance, location designators are used to remove locations from Person queries (e.g., "John Wayne Airport").

[4]The answers to queries of the types considered in this paper are NGs. Hence, in the rest of this paper, we will mention only NGs, but our techniques are also applicable to answers that are verb groups.

In the future, we intend to investigate the use of a name-entity recognizer as a filter in addition to or instead of these processes.

Removing unpromising sentences

First, we remove the sentences that do not contain any of the candidate NGs which remain after the previous step. We then remove sentences that fail the following requirements: (1) the sentence must contain at least one content lemma from the SUBJECT or OBJECT of the query, and (2) the sentence must contain at least one noun from one of the named entities in the query. This filtering process eliminates about 80% of the sentences that do not contain the answer to the queries, but at the same time it eliminates about 60% of the sentences that contain the answer. We are currently investigating filtering processes that retain more sentences with correct answers without increasing excessively the number of sentences without correct answers.

4.3 Training the SVMs

The eventual objective of the SVMs is to identify NGs that answer the given queries. However, our current focus is to find sentences that contain the answer. Our SVMs are trained and tested on the sentences in the document center plus L_S sentences before and after the document center. As indicated in Section 4.1, we considered values for L_S between 2 and 40. For $L_S = 2$, this process generates about 34,000 sentences per query type on average (averaged over the five query types), of which only a small fraction are positive (i.e., contain the answer).

We train a separate SVM for each query type, but the same SVM components are used for all the query types. Each sentence yields one SVM vector, which comprises 12 main components: (1) whether the sentence answers the query correctly; (2-11) one component for each of the first 10 query features described in Section 2 (at present we do not use the Remaining Lemmas feature); and (12) an additional "summary" component. That is, the general form of an SVM vector is:

1	2	3	...	11	12
IsAnswer	Topic	AnswerType	...	PrepConstraint	Summary

where positive vectors have a value of 1 for the IsAnswer component, and negative vectors have a value of 0.

The 2nd to the 11th SVM component (for the first 10 query features presented in Section 2.1) are described below, followed by a description of the summary component. The elements of these components are illustrated with respect to the sample query and candidate answer sentence in Figure 3.

Component for query feature *f* (10 components)

The SVM component for query feature f_i $(i = 1, \ldots, 10)$ represents different aspects of this feature and of its relationship with a candidate sentence. This component comprises

Query:
What is the southwestern-most tip of England?

Parse tree:
```
(S1 (SBARQ (WHNP (WP What))
       (SQ (VP (AUX is)
            (NP (NP (DT the)
                    (X (JJ southwestern) (: -))
                    (JJS most) (NN tip))
                (PP (IN of) (NP (NNP England))))))
     (. ?)))
```

Features:

TOPIC:	[England]
ANSWER TYPE:	[the southwestern most tip of England]
NAMED ENTITIES:	[England]
OBJECT:	[the southwestern most tip of England]
REMAINING LEMMAS:	[What is ?]

Sentence:
MARAZION, England_ The very southwestern tip of England, the county of Cornwall is a place set apart.

Answer: Cornwall

Parse tree:
```
(S1
 (S (NP (NNP MARAZION))
    (PRN (, ,)
      (S (NP (NNP England))
         (VP (VBZ _)
             (NP (NP (DT The) (ADJP (RB very)
                                    (JJ southwestern))
                     (NN tip))
                 (PP (IN of) (NP (NNP England))))))
             (, ,))
    (NP (NP (DT the) (NN county))
        (PP (IN of) (NP (NNP Cornwall))))
    (VP (AUX is)
        (NP (NP (DT a) (NN place))
            (VP (VBN set) (ADVP (RB apart)))))
    (. .)))
```

Figure 3: Sample query and candidate sentence from a document.

three segments: *query only*, *similarity* and *similarity with neighbouring sentences*.

Query only provides information about the query itself. This segment includes the following elements.

- *Count* – the number of content lemmas in the feature.
- *IsQuote* – whether the feature is a quote.
- $\textit{ReltvAvg}\text{IDF} = \frac{\text{avg IDF of feature lemmas}}{\max_i\{\text{avg IDF of feature } i \text{ lemmas}\}}$

  This element represents the relative "importance" of the query feature, i.e., features which contain lemmas that are infrequent in the corpus are deemed more important than features that contain frequent lemmas.

Similarity reflects the goodness of the match between the content lemmas in the feature and the candidate sentence. This segment will enable the SVM to identify which features should be well represented in a successful sentence and which are not so crucial. Similarity includes the following elements (in the following equations, "lemmas" refers to content lemmas).

- $\textit{Recall} = \frac{\text{\# feature lemmas in sentence}}{\text{\# feature lemmas}}$
- $\textit{Precision} = \frac{\text{\# feature lemmas in sentence}}{\text{\# lemmas in sentence}}$
- $\textit{FuzzyMatch} = \frac{\text{\# feature lemmas in span}}{\max\{\text{\# feature lemmas}, \text{\# lemmas in span}\}}$

  where *span* is the shortest contiguous string (in the sentence) that contains the feature lemmas. The inclusion of feature lemmas in the denominator is necessary to avoid anomalous results when only a few feature lemmas appear in the sentence, and they are next to each other. This yields a span (denominator) that is equal to the numerator, and hence a value of 1 for FuzzyMatch, which would wrongly indicate a good outcome. Note that this measure replicates Recall if the span contains fewer lemmas than the feature.
- $\text{TF.IDF}\textit{Recall} = \frac{\sum \text{TF.IDF of feature lemmas in sentence}}{\sum \text{TF.IDF of feature lemmas}}$

  This element is like Recall, but uses the TF.IDF of each feature lemma instead of 1.

For example, Recall is 0.75 for the OBJECT feature in the example in Figure 3, as three of its content lemmas ("southwestern", "tip" and "England") appear in the sentence; Precision is 3/10; FuzzyMatch is 0.75, as the span "southwestern tip of England" contains three of the content lemmas in the feature; and TF.IDFRecall is 0.71.

Similarity with neighbouring sentences is like the above Similarity segment, but calculated with respect to the sentence before and the sentence after the current sentence (four elements each for the sentence before and the sentence after). The idea is that even if the candidate sentence does not contain query features, it could still contain the answer if the features appear in neighbouring sentences and paragraphs.

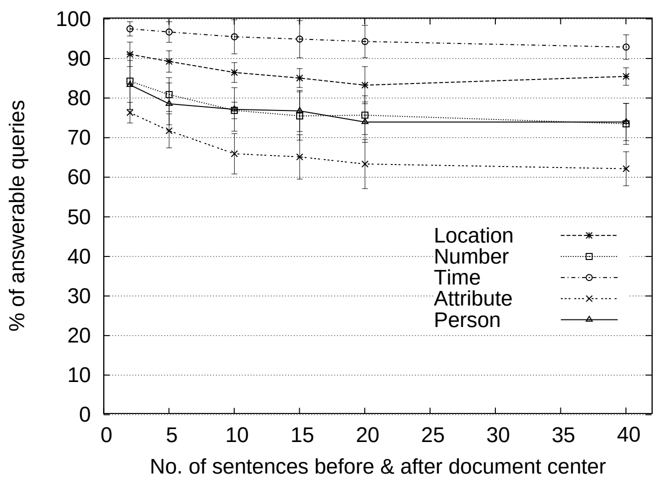
This segment has two additional elements that indicate whether neighbouring sentences or paragraphs contain information relevant to the query.

- *Sentence distance* (in number of sentences) to the closest sentence containing a lemma from the feature.
- *Paragraph distance* (in number of paragraphs) to the closest paragraph containing a lemma from the feature.

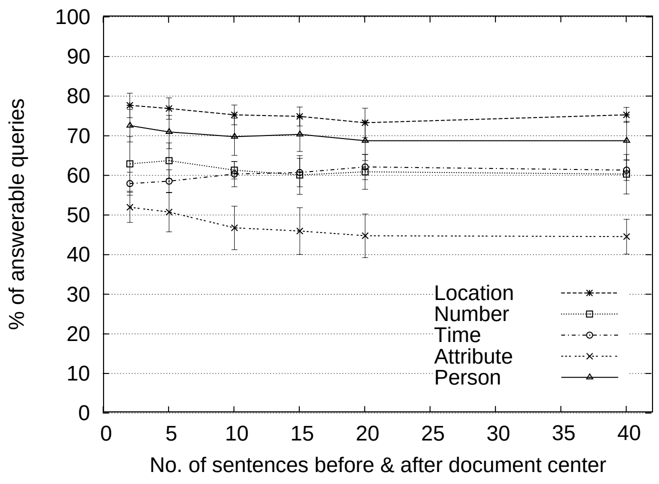
Summary component

This component reflects the overall quality of a candidate sentence. It contains the following elements:

- *AveragePrecision*, *AverageRecall*, *AverageFuzzyMatch* and *Average*TF.IDF*Recall*, averaged over the non-empty features of the query.
- *RatioNon-emptyFeatures* =
  $$\frac{\text{\# query features with} \geq 1 \text{ lemma in sentence}}{\text{\# non-empty query features}}$$
  This element represents the proportion of non-empty query features for which the sentence contains at least one content lemma. For instance, RatioNon-emptyFeatures for the example in Figure 3 is 1 (all the non-empty query features, without counting REMAINING LEMMAS, appear in the sentence).



(a) % answerable queries from queries with positive vectors.



(b) % answerable queries from queries with answers in retrieved documents.

Figure 4: Question answering performance for Location, Number, Time, Attribute and Person queries as a function of the number of sentences before and after the document center; results for 50 retrieved documents and top 20 sentences returned by the SVMs.

4.4 Selecting a Sentence

Since documents are sometimes replicated nearly verbatim in different TREC news sources, a particular sentence may appear more than once in the list of candidate sentences. However, some values in its SVM vector (and its SVM score) may differ if this sentence is preceded or followed by different sentences. Thus, prior to sentence selection, we remove such duplicate sentences, retaining only the highest scoring one. The candidate sentences are then sorted in descending order of their score, and the top N sentences are returned. Our results for $N = 1, \ldots, 20$ are discussed in Section 4.5.

4.5 Evaluation

As stated above, the current focus of our evaluation is on the identification of the sentence that contains the correct answer to a query. Our evaluation was performed on the five main query categories: location, number, time, attribute and person. Owing to memory limitations, our SVM was trained only on 20% of the queries for each of these categories, and tested on 80% with five-fold cross validation.

First, we determined whether the positive-to-negative vector ratio depicted in Figure 2 is indicative of the question answering performance of the system. To this effect, we trained and tested the SVMs using different numbers of sentences before and after the document center ($L_S = \{2, 5, 10, 15, 20, 40\}$). These trials were conducted for 50 and 100 retrieved documents. Interestingly, the number of retrieved documents had a negligible effect on performance (differences were below one standard deviation). Still, the number of answerable queries for 50 retrieved documents was slightly higher than for 100 documents for Attribute and Person queries, and about the same for the other types of queries. We postulate that this is because the additional retrieved documents yield a small increase in documents with answers and a large increase in documents without answers, resulting in the deterioration of the ratio of positive to negative SVM vectors (according to Table 2, going from 50 to 100 retrieved documents yields only 36 additional correct documents in total, while requiring the inspection of 50 additional documents for each query).

Figures 4(a) and 4(b) depict the average percentage of answerable queries in the top 20 sentences returned by each of the five SVMs as a function of the value of L_S for 50 retrieved documents. Figure 4(a) shows the precision of the results returned by the SVMs, and Figure 4(b) depicts the question-answering performance as a percentage of the answerable queries from the retrieved documents. According to both plots, the value of L_S has a marginal effect on question-answering performance, with a slightly better performance obtained for $L_S = 2$ for four query types (the best performance for Time queries is obtained for $L_S = 20$, but this result is not statistically significant).

The distinction between queries with and without positive vectors allows us to differentiate between two causes for failing to answer a query: failure due to shortcomings in our machine learning approach, i.e., features used in the SVMs, and failure due to the removal of correct sentences by our filtering process (Section 4.2). Our system exhibits a creditable performance for Location queries according to both plots, with Person queries yielding a somewhat lower performance. In contrast, Time queries go from being the best performers in the positive-vectors plot to being third or fourth in the documents-with-answers plot, with the performance of Number queries also dropping between both plots, although less dramatically. This means that our filtering process is quite discriminating for Time queries, and somewhat less discriminating for Number queries, improving precision at the expense of coverage. Finally, Attribute queries exhibit the worst

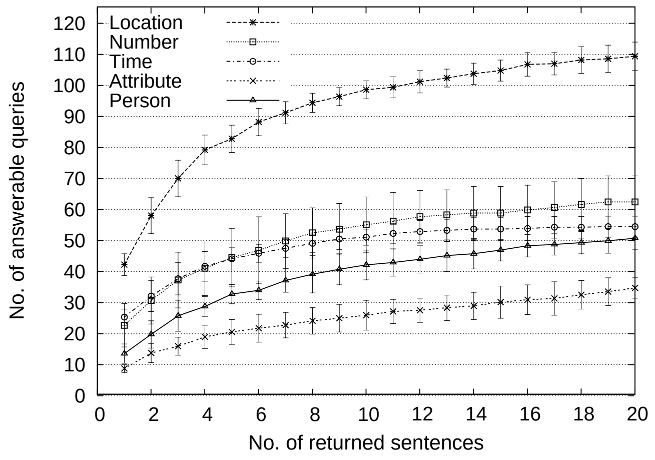


Figure 5: Number of answerable queries for $L_S = 2$ and 50 retrieved documents, for Location, Number, Time, Attribute and Person queries, as a function of the number of sentences returned by the SVMs ($N = 1, 2, \ldots, 20$).

performance according to both plots. This is because an attribute may be any noun group, while the answers to the other four query types are more easily identifiable: number, and time and date are tagged in the corpus, and person and location are proper noun groups. Clearly, our sentence selection performance can be improved by employing a better sentence filtering process (Section 4.2). However, this process is more challenging for Attribute queries than for the other types of queries, due to the many kinds of potential answers for Attribute queries. This indicates that a finer classification of Attribute queries may be useful.

Overall, the best performance is obtained when 50 documents are retrieved, and only two sentences before and after the document center are considered. We now examine the results obtained with this setting in further detail. Figure 5 depicts the average number of answerable queries for the five query types as a function of the number of sentences returned by the SVMs ($N = 1, \ldots, 20$); the error bars show one standard deviation. Recall that these results were obtained for 80% testing. For instance, the 109 answerable Location queries for 20 returned sentences constitute 78% of the 140 Location queries in the test set. As expected, the percentage of answerable queries goes up with the number of sentences. In fact, the gradient of the curves is still substantial at 20 sentences, indicating that additional sentences should be considered.

Our results indicate that our approach shows promise and merits further investigation. Specifically, we are currently pursuing two main avenues of research: (1) the identification of additional SVM attributes; and (2) the application of more sensitive, query-type relevant filtering rules.

5 Related Research

Machine learning techniques have been used mainly for query classification and answer selection.

Zhang and Lee [2003] and Hacioglu and Ward [2003] used SVMs for automatic query classification. Zhang and Lee considered two grains of classifications, coarse (6 classes) and fine (50 classes), while Hacioglu and Ward considered a fine-grained classification only. Zhang and Lee experimented with five machine learning methods and with bag-of-words and syntactic attributes, while Hacioglu and Ward used word-based primitive attributes, which were then composed into complex attributes. We considered an intermediate grain classification of 11 classes. Although we used only a modified bag-of-words approach, we obtained significantly better results for our classification than those obtained by Zhang and Lee [2003] for a coarse classification. However, as seen in Section 4.5, our sentence selection results show that our attribute class may be too broad, and could benefit from a finer classification.

Several researchers applied machine learning techniques to learn answer selection patterns, e.g., [Radev *et al.*, 2000; Pasca and Harabagiu, 2001; Suzuki *et al.*, 2002; Echihabi and Marcu, 2003]. Radev *et al.* [2000] annotated spans in the corpus with labels that indicate the type of the information in the spans, and trained a regression algorithm to learn the weights of attributes extracted from candidate answer spans. These weights were then used to rank the spans. Pasca and Harabagiu [2001] employed a perceptron to learn a comparison function used to rank candidate answers. Both Radev *et al.* and Pasca and Harabagiu used attributes that reflect the extent of the match between the query terms and the terms in a text snippet or span, e.g., number of matched query terms, distance between query terms and a candidate answer of the expected type, and order of query terms in the sentence. Such attributes were also used by Suzuki *et al.* [2002]. However, they considered more attributes than Radev *et al.* and Pasca and Harabagiu, and distinguished between different types of attributes, e.g., keyword, question focus, and semantic category (examples of keyword attributes are: average number of stems, inflections and parts-of-speech in the query that match a candidate answer). Suzuki *et al.* compared the performance of several supervised learning systems, showing that SVMs outperform decision trees and maximum entropy. Echihabi and Marcu [2003] adopted a different approach, where they inferred answer patterns for queries. Echihabi and Marcu derived answer patterns from sentences that contain the answer to a query, and used these patterns to train a noisy channel model to identify the maximum probability answer sentence that "yields" a given query.

Our work combines sentence patterns with attribute extraction, as our system trains its SVMs on attributes that reflect the relationship between semantic query features and candidate answers. Czuba *et al.* [2002] used attributes similar to some of ours to train classifiers to predict whether a list of retrieved sentences contains the answer to a query. This indicates that our attributes show promise for the answer-selection stage of our project.

6 Conclusion

We have described three components of a question answering system – query analysis, document retrieval and sentence selection – focusing on their use of machine learning techniques. In particular, we have examined the use of SVMs for sentence selection, and described how we train SVMs using attributes of semantic query features and attributes of the relationship between these features and candidate sentences. Our results illustrate the promise of machine learning techniques for different aspects of the question answering problem, and indicate that significant progress can still be made before the limits of this methodology are reached.

Acknowledgments

This research was supported in part by the ARC Centre for Perceptive and Intelligent Machines in Complex Environments. The authors thank Eugene Charniak for his modifications to his probabilistic parser.

References

[Czuba *et al.*, 2002] Krzysztof Czuba, John Prager, and Jennifer Chu-Carroll. A machine-learning approach to introspection in a question answering system. In *ACL2002 – Proceedings of the ACL Workshop on Empirical methods in Natural Language Processing*, pages 265–272, Philadelphia, Pennsylvania, 2002.

[Echihabi and Marcu, 2003] Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *ACL2003 – Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Sapporo, Japan, 2003.

[Hacioglu and Ward, 2003] Kadri Hacioglu and Wayne Ward. Question classification with Support Vector Machines and error correcting codes. In *Companion Volume of the Proceedings of HLT-NAACL 2003 – Short Papers*, 2003.

[Kowalczyk *et al.*, 2004] Pawel Kowalczyk, Ingrid Zukerman, and Michael Niemann. Analyzing the effect of query class on document retrieval performance. In *AI'04 – Proceedings of the 17th Australian Joint Conference on Artificial Intelligence*, pages 550–561, Cairns, Australia, 2004.

[Miller *et al.*, 1990] George Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to WordNet: An on-line lexical database. *Journal of Lexicography*, 3(4):235–244, 1990.

[Oliver, 1993] Jonathan J. Oliver. Decision graphs – an extension of decision trees. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pages 343–350, Fort Lauderdale, Florida, 1993.

[Pasca and Harabagiu, 2001] Marius Pasca and Sanda Harabagiu. High performance question/answering. In *SIGIR'01 – Proceedings of the 24th ACM International Conference on Research and Development in Information Retrieval*, pages 366–374, New Orleans, Louisiana, 2001.

[Radev *et al.*, 2000] Dragomir Radev, John Prager, and Valerie Samn. Ranking suspected answers to natural language questions using predictive annotation. In *Proceedings of the Sixth Applied Natural Language Processing Conference*, pages 150–157, Seattle, Washington, 2000.

[Salton and McGill, 1983] G. Salton and M.J. McGill. *An Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[Suzuki *et al.*, 2002] Jun Suzuki, Yutaka Sasaki, and Eisaku Maeda. SVM answer selection for open-domain question answering. In *COLING'02 – Proceedings of the International Conference on Computational Linguistics*, pages 974–980, Taipei, Taiwan, 2002.

[Wallace and Patrick, 1993] C.S. Wallace and J.D. Patrick. Coding decision trees. *Machine Learning*, 11:7–22, 1993.

[Zhang and Lee, 2003] Dell Zhang and Wee Sun Lee. Question classification using Support Vector Machines. In *SIGIR'03 – Proceedings of the 26th ACM International Conference on Research and Development in Information Retrieval*, pages 26–32, Toronto, Canada, 2003.

[Zukerman *et al.*, 2003] Ingrid Zukerman, Bhavani Raskutti, and Yingying Wen. Query expansion and query reduction in document retrieval. In *ICTAI2003 – Proceedings of the 15th International Conference on Tools with Artificial Intelligence*, pages 552–559, Sacramento, California, 2003.

Information Synthesis: A Glance at the Future

Marie-Francine Moens
Katholieke Universiteit Leuven
Tiensestraat 41 B-3000 Leuven, Belgium
marie-france.moens@law.kuleuven.be

Abstract

Because of the current information overload, information synthesis by the machine becomes increasingly important. Information synthesis from text regards the composition or combination of diverse content parts or elements so as to form a coherent whole. We define information synthesis from a cognitive and linguistic viewpoint. We clarify its relation to question answering, summarization, and linking of information. An overview of the many applications in professional and Web contexts is given.

The second part of the talk focuses on the technologies. A short overview of current technologies for information synthesis from text introduces the most promising research avenues. Especially the role of information extraction technologies is stressed as they offer the necessary basis for representing content and for reasoning with it. Information extraction has the additional advantage that it can be applied – though often relying on separate methods - on different media (e.g., text, images, video) allowing information synthesis across media.

A final part discusses the bottlenecks when developing synthesis systems for realistic text repositories in open and closed domains.

Where are the 'Killer Applications' of Restricted Domain Question Answering?

Michael Minock
Umeå University
Department of Computing Science
C445 MIT-huset Umeå, Sweden 90187
mjm@cs.umu.se

Abstract

From a language technologist's point of view, the penetration of natural language interfaces onto today's web is somewhat disappointing; it seems that information retrieval, forms based, metaphor-based and hyper-link interfaces dominate all points of the design space. While open domain question answering promises to rival or extend information retrieval systems, restricted domain question answering systems likewise represent a rival to forms-based interfaces. The purpose of this position paper is to discuss the properties of potential web-based 'killer applications' of restricted domain question answering. The paper entertains a set of candidate domains, proposes a general methodology for building restricted domain interfaces and highlights some near term challenges that must be confronted.

1 Introduction

Although open domain question answering is a very promising area, the position in this paper is that an effort must likewise be undertaken for a set of promising restricted domains. Though one may view restricted domain question answering as a special case of open domain question answering, employing an isolated set of domain documents and perhaps a tailored lexicon and grammar, the view here is more expansive; it is assumed that domain knowledge and data are explicitly represented to one degree or another. This enables the introduction of domain concepts and facts in addition to domain 'documents'. It also points toward a revival of question answering over databases [1; 3]. In any case a resulting restricted domain question answering system, with access to domain knowledge, is presumably more able to use reasoning in support of question answering. The answer itself is expected to be direct, consisting of some combination of natural language, tabular data, graphs, video clips, documents, etc.

This paper shall refrain from making explicit assumptions about the nature of a representation backing a restricted domain system, however it is assumed that it will consist of a conceptual model (or ontology) which covers some significant number of entities (instances or assertions). We assume that this domain information is built-up and maintained in some coherent, quality state through some combination of fact extraction from a set of documents and hand crafted knowledge representations or databases. Furthermore, and perhaps optimistically, we assume that a suitable natural language interface may be built over a given domain that: 1.) lets the user pose questions in a full natural language; 2.) offers paraphrases when user questions are unclear or ambiguous; 3.) accurately describes answers to avoid misunderstandings. Given these assumptions, the thought experiment in this paper is to ask, *assuming that problems of representation and natural language processing can be handled, what domains are ripe for restricted domain question answering on the web?* Since most web users are already comfortable with key-word search, forms and hyper-links, the application of natural language question answering to a restricted domain must somehow eclipse these techniques, individually or in combination.

Section 2 of this position paper outlines some desiderata for promising restricted domains. Section 3 proposes a set of restricted domains and discusses these proposals in relation to the desiderata. Section 4 proposes a general development methodology for building restricted domain question answering systems. Section 5 discusses some of the high priority challenges that need to be addressed to open the way for the deployment of such systems. Section 6 concludes this position paper.

2 Desiderata for Restricted Domains

To enjoy success on the web, a restricted domain must be:

D1: Circumscribed

- D1.1: topic is focussed
- D1.2: level of detail is evident
- D1.2: knowledge may be represented and data is factual

D2: Complex

- D2.1: spans more than several concepts

- D2.2: entities have complex properties and are involved in complex relationships
- D2.3: numerous entities

D3: Practical

- D3.1: answers to single sentence questions are useful to an identified group of users
- D3.2: data and knowledge acquisition and maintenance is feasible
- D3.3: query volume is high

2.1 Circumscribed

If D1.1 is adhered to a user may ascribe a bounded understanding to the system and thus revisit the system when they wish to know something about the domain. Normally this is not be too difficult to achieve. If for instance we pick the domain of 'Aquarium Fish: Species, Habits and Care' a user would have a reasonable idea of what the domain encompasses, though they would perhaps need to experiment a bit to get an idea. If the domain is too general, say 'Current events', then the user might have a bit more difficulty ascertaining and remembering what the system really 'knows' about.

Though D1.1 bounds the topic in conceptual space, D1.2 bounds the level of detail that is to be captured. For example assume that we have the topic 'World War II: leaders, footage and battles'. It may be that we could ask questions such as "show some footage of the battle of Britain", but of questions such as "which was the first suburb of Minsk to fall to the Nazi armies?" are not likely to be represented.

Another consideration is whether the information of the domain is simple enough to be adequately captured with contemporary knowledge and data representation. For example the aquarium domain is probably somewhat simpler ontologically that the World War II domain. The judgement of the knowledge requirements is qualitative, but, unlike our computers we do have common sense and can rate certain applications as requiring significantly more knowledge representation requirements than others.

2.2 Complex

The key issue here is to insure that the domain has sufficient structure to warrant a question answering interface. For example if D2.1 is not adhered to and the domain is spread over only a few concepts (tables), then a forms based interface will probably suffice. For example if we consider flight information of a particular airline (e.g. `www.sas.se`), we see that a forms based interface suffices. At one point however, when the schema spans more than a few concepts, standard forms based approaches break down. It is at this point that other techniques become necessary such as query construction tools, visual query languages, hierarchies of forms, natural language menus, etc. It stands to reason that such a point is also where natural language question answering may find a break through as well.

If, in violation of D2.2, the objects of user interest really don't have complex properties associated with them other than their own names or textual content, then perhaps a key-word based search techniques is most appropriate. Consider for example a set of documents that have only very coarse descriptors and sets of associated keyword. Since there are only very simple predicates that objects may satisfy, an 'advanced search' option that mixes keyword and simple predication (e.g. date, domain name, etc.) will suffice.

Naturally, as noted in D2.3, if the number of entities in the domain is limited then the user would probably be better off just reading a list or navigating through a hypertext document.

2.3 Practical

Since we expect single sentence questions, naturally a user should be able to get something of value as an answer. Thus if a domain fails D3.1 and the content is either too well known or of very marginal interest, then who will be interested in querying it? For example if one has a geography database of the capitals of various countries, who, other than those interested in testing the system, would really use such an interface. A better option is to click through the CIA world fact book. The same is true for a database over relatively uninteresting data. If for instance, I put my contacts and calendar into a database, who other than myself would really be interested in querying it. Again we have to use our common sense to gauge the how well a domain meets D3.1.

The consideration in D3.2 is how feasible is it to build a representation of the domain. In general the cost of constructing the database and its interface's linguistic configuration must be justified by the quantity of queries that are posed over the lifetime of the interface (D3.3). Factors such as the timeliness of the data and the speed at which users wish to know such data enter into judging a candidate domain on these measures.

2.4 Correlations among the Desiderata

Naturally there are correlations among the desiderata above. For example if a domain qualifies on D2.1 then it is likely to qualify on D2.2 and D2.3 as well. In fact all the sub-desiderata of the three major desiderata seem to positively correlate. Moreover D1 and D2 on a whole seem to be inversely correlated.

Of course these desiderata are simply a set of characteristics arrived at through introspection and are merely subjective judgments, overlapping and probably incomplete. Still they provide a convenient starting point to organize the search for the illusive 'killer application' of restricted domain questions answering.

3 Some Candidate Domains

We now evaluate a set of application domains. These domains are introduced with a simple description followed by an opinion of how well they rate on the above desiderata.

A1: **NLI to a photo album:** Pictures are classified based on when and where they are taken, by who, of what, of whom, etc. This application scores well on D1, but poorly on D2.1 and D2.2. A hybrid approach based on key word search of picture captions and forms based predications on picture date, size, location is deemed to be a superior approach.

A2: **Simple geography facts:** Users may ask questions about cities, countries, languages, religions and ethnic groups. This application scores relatively well on D1 and D2, though it could be argued that D1.2 is a little weak. The problem is mostly D3.1. Still there is some hope that with more complete data and integration of maps, etc. that some type of useful interface will emerge. An initial example of such a domain may be accessed through the interface [6] at `www.cs.umu.se/~mjm/step` which is backed by the Mondial data set[5].

A3: **Bus schedules:** An interface allowing residents to query for times and destination of busses within the local community. This application scores well on D1, but perhaps less well on D2, especially D2.1 and D2.3. For D3 it seems like it could generate a fair number of queries especially if the interface was readily accessible via mobile devices. An example interface of this type may be accessed at `www.idi.ntnu.no/~tagore/bustuc/`

A4: **City events information:** An interface to city events, hotels, restaurants and other attractions. This type of application does a fair job on D1 and a good job on D2 if the city is large. Some issues surround D1.2. If the city is a popular tourist destination it could do well on D3.1 and D3.3 as well. D3.2 is something of a weak spot, but it could be offset by D3.3. The work [7] discusses such an interface for the city of Cottbus.

A5: **Natural language assistant to a GIS:** Allow for natural language querying of a wide variety of locations on a map. For example "show the houses that are worth between 100k and 200k dollars that are less than 10km from a lake." Such an application scores OK on D1, though there are some questions surrounding spatial representation in D1.3. Provided that there are numerous types of objects represented on the map with complex properties, it is likely that the application will score well on D2 as well. The site could generate enough interest to justify the considerable cost associate with D3.2. One specific idea is to build an interface to the UNESCO world heritage sites database.

A6: **Nobel prize database:** A database that contains information on the winners of the Nobel prizes including nationality, pictures, age, academic affiliation, etc. This idea rates well on D1, and receives a perhaps just passing score on D2. The database itself does not seem to be difficult to build and does not need to be updated often. Finally the domain might be of interest to the public.

A7: **World Cup scores and highlights:** Allows for users to query for games, player and team statistics for the world cup tournament. This domain rates well on D1 and D2. Although the cost of building a database may be considerable, if the interface worked well, the expected number of queries may also expected to be high.

A8: **Software catalog:** Allows for users to query for software with certain properties, running on different platforms, etc. This type of domain is a bit weak on D1. There are simply so many different concepts that might be involved. The system is certainly complex enough (D2) and for D3 there seems to be a great need for this type of service, though it must be said that building and maintaining the database is likely to be a very difficult task.

A9: **Continental European travel information:** Allows users to pose queries over train schedules, attractions, hotels, etc. This domain has difficulties with D1 but does well on D2. It may have difficulties with D3.2, but this could be offset with D3.3. A fragment of this domain has been treated in [2].

4 A Proposed Development Methodology

The following is a proposed methodology to roll out a restricted domain question answering system:

S1: Identify a domain with the above desiderata

S2: Collect a large number of candidate questions over the domain.

S3: Build a conceptual model that covers the bulk of questions from S2.

S4: Define a representational model that corresponds to the conceptual model of S3.

S5: Populate the representational model with 'documents' and facts

S6: Configure the natural language interface over the representational model

S7: Offer the system to a user community

Note that S2 may be carried out through a brain storming session or through hidden operator ("wizard of Oz') experiments with a sample user group. Note that steps S3, S4 and most importantly S5 are carried out after the selection of the domain; in general we view the prospects of using legacy databases without significant restructuring as overly optimistic. Step S6 is the configuration of the natural language interface over the restricted domain. Note that this step will probably involve significant work even if general linguistic grammars are used; mapping domain independent grammars to restricted domain models is very challenging and can only be partly automated. Finally in S7 the system is presented to a user community. At this point S7 becomes a source of queries and the development iterates through steps S2 through S7 ad infinitum.

5 Some Challenges

Of course each domain of section 3 has different ontological requirements, and thus have separate challenges, not addressed here, that must be confronted. The challenges sited in this section apply to all the candidate domains.

5.1 Natural Language Processing

Needless to say, the natural language interfaces to restricted domains must be sophisticated. One key issue is that interfaces must support both generation and understanding; a system must be able to paraphrase user questions either during answer presentation or in the case of a low confidence parses. Another issue is that the systems will need to be able to support some special types of questions. In particular it seems that superlatives are a difficult type of question which will need to be parsed (e.g. "Give the person who has won the most Nobel prizes."). A variant of this type of question is to request the top ranking set of answers (e.g. "Give the 10 most populated countries in Asia."). Mapping such requests to the logic that expressed them is non-trivial. Finally it should be noted that the natural language interfaces must be linguistically complete enough to offer to parse the vast majority of questions over the domain. This includes some method to cope with non syntactic inputs as well as a tolerance for simple spelling errors.

5.2 Cooperative Query Answering

It seems that most if not all domains will need to have at least some support for cooperative query answering [4]. At its core this means support for automated or semi automated query generalization and specialization. These capabilities enable relaxation of queries which obtain no answers and query refinement when user queries are too broad. A capability of identifying false presuppositions within user queries is also important (e.g. "List cities in Grance with more than 1 million people" should be answered with "There is no country or region named 'Grance'.").

Somewhat outside the traditional topics of cooperative querying answering, systems must in one way or another handle conceptual and data incompleteness. Conceptual incompleteness may perhaps be handled by extending the domain model with dummy concepts near to the domain topic. Meta-level responses would be generated when user queries touch upon these out of range concepts. Additionally logical meta-data may be used in cases that there is data incompleteness (e.g. "Give the cities in Sweden" could be answered with "the cities in Sweden are ..., but the database only contains cities in Sweden with more than 20,000 people.")

5.3 Open Evaluation

To facilitate better comparison of systems, restricted domain query answer prototypes should be available for anonymous querying on the web. The page at `www.cs.umu.se/~mjm/step` lists all the systems of which I am aware (including my own system STEP [6]). Additionally, for those who do offer web demonstrations, it would be helpful to publish the system's complete configuration. Naturally it would also be helpful if all domain data was available as well. If more groups were to do this, perhaps more systematic comparisons of various techniques could be carried out.

6 Conclusions

This position paper has proposed that concerted efforts be made to promote restricted domain question answering on the web. Web interfaces make no special assumptions about the user group (visually impaired, talking on a phone, etc.) and because of this, we must confront the fact that other interface techniques, such as forms, hyper-links, menus, key-word searches will compete directly with natural language interfaces. Still, it is argued here that there is a set of circumscribed, complex and practical domains that are best served by natural language question answer interface. Such applications constitute a break through point for natural language access to structured data/knowledge. If a group can deploy such a solution that always beats forms based and keyword based rivals, this will go a long way toward vitalizing work in natural language interfaces and ontologies for that matter.

References

[1] I. Androutsopoulos and G.D. Ritchie. Database interfaces. In R. Dale, H. Moisl, and H. Somers, editors, *Handbook of Natural Language Processing*, pages 209–240. Marcel Dekker Inc., 2000.

[2] F. Benamara. Cooperative question answering in restricted domain : the WEBCOOP experiment. In *ACL04 workshop on Question Answering in Restricted Domains*, 2004.

[3] A. Copestake and K. Sparck Jones. Natural language interfaces to databases. *The Natural Language Review*, 5(4):225–249, 1990.

[4] T. Gaasterland, P. Godfrey, and J. Minker. An overview of cooperative answering. *Intelligent Information Systems*, 1(2):127–157, 1992.

[5] W. May. Information extraction and integration with FLORID: The MONDIAL case study. Technical Report 131, Universität Freiburg, Institut für Informatik, 1999.

[6] M. Minock. A phrasal approach to natural language access over relational databases. In *Proc. of NLDB*, Alicante, Spain, 2005.

[7] B. Thalheim and T. Kobienia. Generating DB queries for web NL requests using schema information and DB content. In *Proc. of NLDB*, pages 205–209, 2001.