A SYNTACTIC CANDIDATE RANKING METHOD FOR ANSWERING NON-COPULATIVE QUESTIONS

Abolfazl Keighobadi Lamjiri

A thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

April 2007

Your file Votre référence
ISBN: 978-0-494-30132-6
Our file Notre référence
ISBN: 978-0-494-30132-6

Abstract

A Syntactic Candidate Ranking Method for Answering Non-copulative Questions

Abolfazl Keighobadi Lamjiri, Ph.D.

Concordia University, 2007

Question answering (QA) is the act of retrieving answers to questions posed in natural language. It is regarded as requiring more complex natural language processing (NLP) techniques than other types of information retrieval such as document retrieval. QA is sometimes regarded as the next step beyond search engines that ranks the retrieved candidates. Given a set of candidate sentences which contain keywords in common with the question, deciding which one actually answers the question is a challenge in question answering. In this thesis we propose a linguistic method for measuring the syntactic similarity of each candidate sentence to the question. This candidate scoring method uses the question head as an anchor to narrow down the search to a subtree in the parse tree of a candidate sentence (the target subtree). Semantic similarity of the action in the target subtree to the action asked in the question is then measured using WordNet::Similarity on their main verbs. In order to verify the syntactic similarity of this subtree to the question parse tree, syntactic restrictions as well as lexical measures compute the unifiability of critical syntactic participants in them. Finally, the noun phrase that is of the expected answer type in the target subtree is extracted and returned from the best candidate sentence when answering a factoid open domain question.

In this thesis, we address both closed and open domain question answering problems. Initially, we propose our syntactic scoring method as a solution for questions in the Telecommunications domain. For our experiments in a closed domain, we build a set of customer service question/answer pairs from Bell Canada's Web pages. We show that the performance of this ranking method depends on the syntactic and lexical similarities in a question/answer pair. We observed that these closed domain questions ask for specific properties, procedures, or conditions about a technical topic. They are sometimes open-ended as well. As a result, detailed understanding of the question and the corpus text is required for answering them.

As opposed to closed domain question, however, open domain questions have no restriction on the topic they can ask. The standard test bed for open domain question answering is the question/answer sets provided each year by the NIST organization through the TREC QA conferences. These are factoid questions that ask about a person, date, time, location, etc.

Since our method relies on the semantic similarity of the main verbs as well as the syntactic overlap of counterpart subtrees from the question and the target subtrees, it performs well on questions with a main content verb and conventional subject-verb-object syntactic structure. The distribution of this type of questions versus questions having a 'to be' main verb is significantly different in closed versus open domain: around 70% of closed domain questions have a main content verb while more than 67% of open domain questions have a 'to be' main verb. This verb is very flexibility in connecting sentence entities. Therefore, recognizing equivallent syntactic structures between two copula parse trees is very hard. As a result, to better analyze the accuracy of this method, we create a new question categorization based on the question's main verb type: *copulative* questions ask about a state using a 'to be' verb, while *non-copulative* questions contain a main non-copula verb indicating an action or event.

Our candidate answer ranking method achieves a precision of 47.0% in our closed domain, and 48% in answering the TREC 2003 to 2006 non-copulative questions. For answering open domain factoid questions, we feed the output of Aranea, a competitive question answering system in TREC 2002, to our linguistic method in order to provide it with Web redundancy statistics. This level of performance confirms our hypothesis of the potential usefulness of syntactic mapping for answering questions with a main content verb.

Acknowledgments

First and foremost, I would like to thank my research advisor, Dr. Leila Kosseim for her extraordinary support in all aspects of my graduate life. Her pleasant personality, her superb guidance, encouragement and enthusiasm, and the care and respect she shows for her students have made working with her a great pleasure. I would like to thank Professor Radhakrishnan for giving me meticulous guidelines in crucial stages all along my Ph.D. Working with him, I learned to have the big picture of a problem, a list of phases to achieve the goal, and always, a short list of milestones for the next couple of weeks.

I would also like to thank Dr. Bergler and Professor Suen, and Professor Ramachandran for serving on my Dissertation Committee and Professor Nie for serving on my Qualifying Examination Committee. Their expertise, questions and suggestions have been very useful on improving my Ph.D. work.

I also thank Osama El-Demerdash at CLaC lab for innumerable hours of discussion and joint work on word sense disambiguation and feature extraction for questions. I am very grateful to have been surrounded by an amazing team of graduate students and researchers that have made my graduate life very pleasant. I would like to thank all my colleagues in the CLaC laboratory, especially Julien Dubuc, Majid Razmara, and Alex Beaudoin.

Last but not least, I would like to thank my family for their endless love, support, and care during all these years, especially my father, Heidar, to whom this thesis is lovingly dedicated.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the amount of textual information grows everyday, accessing the right information for a specific purpose is becoming more and more difficult. In this situation, information retrieval tools find a large number of documents, but leave the browsing and refining the query for retrieving the right information to the user. These post-processing tasks are structured together in the form of a Question Answering (QA) system.

1.1 Question Answering

Question Answering is a type of information retrieval. Given a collection of documents (such as the Web or a local collection), the system should be able to retrieve answers to questions posed in natural language.

The optimal question answering technique would look up the answer in a comprehensive world knowledge base. Building such a knowledge is an endeavor that a few have started a long time ago, in projects such as CYC [Len95] (1986). However, subjectivity, inexactness, and incompleteness of human knowledge about entities and their relationships have prevented the common acceptance of these tools. Even with such a knowledge base, un-informative conclusions can be inferred if no human *common sense* is involved in the reasoning process. For example, consider the question *Q- "Why did John roller-skate to McDonalds last night?"* taken from the 2001 QA roadmap [BCC+01]. Interestingly, if someone produces the answer *"Because he was hungry."*, the questioner might not

be satisfied, as chances are that the questioner really wanted to know *Q- "Why did John roller-skate instead of walk or drive or use some other reasonable means of transportation?"*. In this case it is clear that the question asks about *the act of roller-skating*, not the purpose. Therefore, the classification of questions based on their focus cannot be performed, unless world knowledge and common sense reasoning capabilities are added to a QA system [BCC+01]. John Burger et al. in the QA roadmap [BCC+01] suggest the use of a taxonomy of question types as a simple substitute for understanding a question. However, for full understanding, issues such as implicatures and ambiguities should be handled.

A group of methods used in QA are based on keyword techniques to locate interesting passages and sentences from the retrieved documents, and filter them based on the presence of the desired answer type within that candidate text. Ranking is then done based on syntactic features such as word order, location, or lexical similarity to the query. When using massive collections with good data redundancy, some systems use templates to find the final answer in the hope that the answer is just a reformulation of the question. If you posed the question "What is a dog?", the system would detect the pattern "What is a X" and look for documents containing "X is a Y". This often works well on simple factoid questions seeking factual tidbits of information such as names, dates, locations, and quantities.

However, in the cases where simple question reformulation or keyword techniques will not suffice, more sophisticated syntactic, semantic and contextual processing must be applied to extract or construct the answer. These techniques include named-entity recognition, syntactic relation detection, coreference resolution, syntactic alternations, word sense disambiguation, logic form transformation, logical inferences (abduction), common sense reasoning, temporal or spatial reasoning, etc. These systems will also very often utilize world knowledge that can be found in ontologies such as WordNet [Fel98], to augment the available reasoning resources through semantic connections and entity definitions.

More difficult queries such as 'Why' or 'How' questions, hypothetical postulations, spatially or temporally constrained questions, dialog queries, badly-worded or ambiguous questions will all need deeper understanding of the question. Complex or ambiguous document passages likewise need deeper NLP (Natural Language Processing) techniques applied to be understood.

QA systems generally run the following processes in turn (see Figure 1.1):

1. They first process the question in order to select important keywords as the clue to search for the documents that might include the answer. This step includes tokenization, part-of-speech tagging, stop word filtering, and stemming. In addition to the keywords, the type of the expected answer is extracted using the question word or other salient features; for example, the question word 'who' is associated with a person and the word 'year' after the question word 'what' hints that the answer is expected to be a date.

2. Selected question keywords are sent to an information retrieval engine to get relevant documents. For a better recall, the query can be expanded by including synonyms of keywords beforehand.

3. In order to narrow down the search to the passage and sentence that provide the answer, retrieved documents are chunked into paragraphs and sentences. The sentences (or paragraphs) that seem to be related to the question are selected for further analysis. Sometimes, a window is moved on the text in order to choose a text buffer of a predefined size that includes all the keywords.

4. Finally, candidate sentences (or paragraphs) are ranked and the answer phrase is extracted from the best candidate by applying techniques such as mapping a pattern and selecting the word that fills in the answer field or simply, by finding the word that mostly co-occurs with the question keywords. Usually in this phase, a few candidates are extracted first before choosing one of them as the final answer.

Current state of the art in the field of QA includes using redundancy from the Web, transforming the text into logical predicates and abductive proving to reason about these predicates, knowledge modeling in closed domain, interactive and contextual question answering, etc., that we will review in Chapter 2.

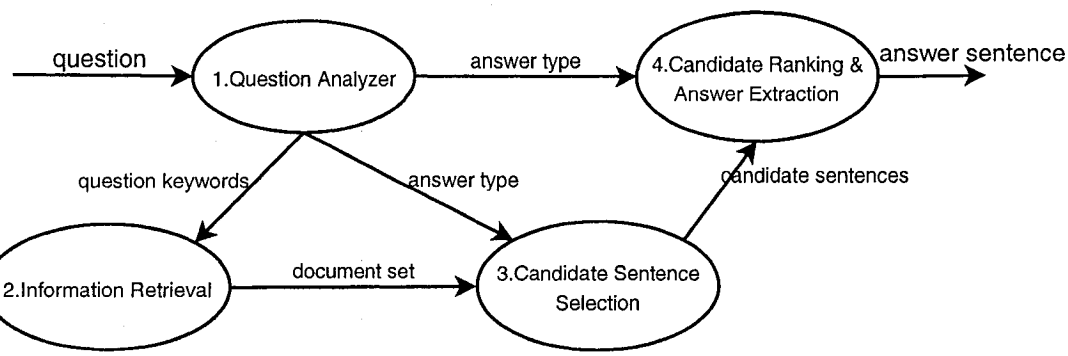

Figure 1.1: Architecture of a typical QA system.

1.2 Categories of Questions

Researchers in the field of question answering have typically classified questions based on their semantic type, arranged hierarchically in taxonomies (e.g. comparison, definition, spatial or temporal, procedural, etc.). Popular text complexity criteria are based on surface text features such as the average length of sentences in the text or the average number of syllables in words [Gre04]. In general, the shorter a sentence, the easier it is to understand. Writing style and use of metaphors or colloquial language, for example, are factors that affect the text complexity that cannot be automatically recognized. We, on the other hand, are interested in the syntactic complexity of questions and the corpus text.

We categorized our closed domain technical question/answer pairs based on their lexical and syntactic similarity. We manually built three complexity categories: simple, average, and challenging; *simple* questions have a simple and conventional grammatical structure with most of the question keywords appearing in the answer sentence without any lexical differences; *average* questions are longer and have a rather complex parse tree; synonyms of question keywords might appear in the answer sentence; and answering *challenging* questions is not possible unless we use advanced NLP techniques such as co-reference resolution and pragmatic knowledge.

Questions are commonly categorized structurally into factoid questions that ask for names, dates, locations, quantities, etc., versus complex questions, that ask for a list of entities, a procedure, an explanation, etc. [BCC+01]. In our closed domain corpus, we realized that the distribution of copulative (having a 'to be' verb) versus non-copulative (having a 'content' verb) questions was significantly different than in open domain; in the TREC question sets (explained in Chapter 3),

figures show that only around one third of the questions are non-copulative while this category comprises more than 70% of the questions in closed domain; most factoid open domain questions ask about a state, date, name, etc. in a copulative syntactic form: *Q66.2-"Who was the on-board commander of the submarine?"*[1]. More examples will be provided when describing our document collections and the question/answer sets in Chapter 3. Although there are other copula verbs in English, such as 'look', 'feel', 'taste', 'smell', 'sound', etc., that can be used to connect the subject to an adjective, we only make a distinction between 'to be' versus non-'to be' verbs, because this verb has so much flexibility in connecting sentence entities that equivallent syntactic structures are very hard to recognize for this verb. This phenomenon led us to create a new question categorization based on the question's main verb type, into copulative versus non-copulative. Most non-copulative questions have a conventional syntactic structure: a content verb, a subject, one or more objects and modifiers; one of these syntactic constituents is replaced by the question word. Analyzing the performance of the state of the art QA systems shows that they perform slightly better on the copulative category, practically showing that non-copulative questions are more difficult to answer. To the best of our knowledge, work on copulative versus non-copulative questions has not been investigated in the past. Results in open domain show that our ranking method outperforms the average accuracy of current QA systems for non-copulative questions (Chapter 7).

In the following, we review a few common question classification criteria used in the QA field.

1.2.1 Factoid versus Complex

A factoid question is a fact-based, short answer question such as *Q121.5-"Her book caused what pesticide to be banned?"* (regarding 'Rachel Carson') or *Q123.2-"Where was Vicente Fox educated?"*. The answer to a factoid question is a noun phrase referring to the name of a person, location, entity, date, etc. The accuracy of a question answering system is reported as the Mean Reciprocal Ranking (MRR) score in the field. It is equal to the inverse of the position of the first correct answer in the list [VT99]:

$$Score(answer\ list) = \frac{1}{Rank(first\ correct\ answer)}$$

More complex questions ask for an explanation, such as *Q85.5-"Why did the Grand Cayman turn*

[1]By numbering the questions from now on we refer to the standard factoid questions composed by the NIST organization (`http://trec.nist.gov`) in TREC QA conferences.

away a NCL ship?". A question might ask for more than one entity in the form of a *list* question; list questions ask for different instances of a particular kind of information to be returned, such as *"List the names of chewing gums"*. Answering such questions requires a system to assemble an answer from information located in multiple documents. Some issues that might arise during answering a list question are:

- Detection of overlapping and contradictory information,
- Credibility of information sources,
- Event tracking, comprising first story detection as well as story continuation.

Another form of a complex question is the *definition* questions. A definition question asks for interesting information about a particular person or thing such as *Q117.1-"What kind of plant is kudzu?"*. Definition questions also require locating information in multiple documents, removing nonessential information.

1.2.2 Contextual and Interactive Questions

In a contextual and interactive scenario, QA is seen as part of a dialogue interaction where a question can refer to previous questions or answers. To answer such sequences of questions, systems should keep track of features of the context, while receiving novel information. Resolution of references within questions, in the text, and the reference to previous answers are major issues in this area. For example, the following question sequence was used in the question answering track of TREC 2001:

1. Which museum in Florence was damaged by a major bomb explosion in 1993?
2. On what day did this happen?
3. Which galleries were involved?
4. How many people were killed?
5. Where were those people located?
6. How much explosive was used?

1.2.3 Open versus Closed Domain Questions

Some question answering systems deal with questions on any topic, while others are expert in only a specific domain (for example, touristic information [Ben04], medicine or the construction sector [ZSD+04]).

Open domain question answering deals with questions about every topic, and can only rely on general ontologies and world knowledge. On the other hand, these systems usually have much more data available from which they extract the answer.

Open Domain Question Answering

The focus of open domain QA is on systems that can function in unrestricted topics. Systems are basically provided with a corpus of documents (typically raw journal articles) to use for finding answers to questions. Ever since the TREC QA track started in 1999, the focus of most of the participating systems has been mainly on factoid questions; i.e. questions which start with 'When', 'Who', 'How much', etc., as opposed to 'Why' and 'What' question words. The task in the TREC 2003 QA track was a combined task that contained list and definition questions in addition to factoid questions. Each category was evaluated separately.

Most open domain QA systems use documents from the Web as the major source of information. There are two benefits for using the web:

- Questions in a wide variety of topics can be answered;
- Up-to-date information is always available somewhere on the Web;
- Information redundancy on the Web helps in detecting and filtering incorrect information and keeping the piece of information that appears most often as the correct information.

However, inconsistent information, reliability of a source, and network latency in accessing the Web are issues that arise when dealing with the information on the Web. While we focus on finding the information by processing text sources, some types of information can be extracted from pockets of structured and semi structured knowledge sources on the Web too. For example, the CIA World Fact book provides political, geographic, and economic information about every country in the world; 50states.com contains numerous properties related to US states from the state bird to land area;

Biography.com has collected profiles of over twenty five thousand famous people; the Internet Movie Database stores entries for hundreds of thousands of movies, including information about their cast, production staff, and dozens of other properties.

One approach to improve the accuracy of a question answering system is to restrict the domain it covers. By restricting the question domain, the size of the knowledge base to build becomes smaller [CSH+04]. This opens up new issues under closed domain question answering.

Closed Domain Question Answering

The term closed domain refers to answering questions on a restricted set of topics (for example, medicine or tourism). Questions asked in a specific domain usually are not factual questions: they tend to be longer, more complex, open-ended and ask for specific properties, procedures, or conditions, and their answers are longer and more complex [DNK06]. As a result, a system expert in that domain is expected to perform a detailed understanding of the question and the text to be able to extract the correct answer. For the question *"What materials are best suited for houses in the Montreal area?"* [ZSD+04] about construction, general QA systems cannot determine an appropriate answer type for this question (or use general domain resources such as WordNet). To answer more specific questions than those on general concepts, one has to use more relative technical knowledge. However, world knowledge is infinite; even the largest knowledge base can only store a part of all concepts and technical terms for all domains.

Yet another feature of QA in closed domain is that, instead of being a single noun phrase, usually an answer should be a larger chunk (e.g. an entire sentence) in order to satisfy what is asked by the end user. Because of these characteristics, closed domain QA has recently been a popular topic of research in QA [Me05, (ed04].

In a document collection relevant to a specific topic, candidate redundancy is less present: the content of documents cover various issues instead of including repeated information [DNK06]. Additionally, low candidate redundancy makes it very unlikely to find an answer with a straightforward grammatical and lexical similarity to the question. Finally, precision is very important in closed-domain question answering because of their practical applications, such as customer service and teaching, that require high reliability: "*no answer* is preferred to a wrong answer" [CSH+04].

Closed domain QA has the disadvantage of being less general than open domain systems. On the other hand, it can be seen as an easier task because NLP systems can exploit domain-specific knowledge frequently formalized in ontologies.

1.3 Challenges in the Field

Although much work has been done in the field, many open research questions still remain:

- *User-centered question answering* includes areas such as keeping a profile information about the user, dialog with the user, and keeping track of a local context information about the current situation [KKM+06]. This area is also referred to as cooperative question answering in the sense that the answer is adapted depending on the user profile, or the context. This field emerged as early as mid 70s in which quite advanced reasoning models were developed (mostly 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. For example, to answer a question that asks about the proportion of students who passed course X in Fall 2006, returning *"course X was not offered in that semester"* instead of *"No one passed course X"* is quite informative and helpful for the user.

- *Cross-lingual question answering*, in which documents or questions are in different languages (see the CLEF Competitions web site at `http://clef.iei.pi.cnr.it/`).

  In a multilingual QA task two main variables need to be considered: the source language, i.e. the language in which the questions are formulated, and the target language, i.e. the language of the document collection. A cross-language QA system should enable users to search documents that are written in a language they do not know, which is a promising application in a multilingual society. Answer-strings, which are usually retrieved from the corpus without any changes, could be translated into the source language.

- *Natural language generation* is required when the information for answering a question comes from different sources (e.g. for answering list questions). Usually, a modularization of the generation process is assumed which roughly distinguishes between a strategic (deciding what to say) and a tactical (deciding how to say it) part. This strategy-tactics distinction is partly

mirrored by a distinction between text planning and sentence generation. Text planning is concerned with working out the large-scale structure of the text (answer) to be produced and may also comprise content selection [Ae96].

- *Reasoning and text understanding* would help in answering complex questions, as well as interpreting complex passages in the corpus. Resolving the causality in the text for example is needed for answering 'Why' questions. Another perspective of reasoning 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 and the arguments that justify the response. Reasoning under uncertainty or with incomplete knowledge is another challenge in this area.

Most question answering systems use plain text on the Web as their source of information. Because of the very large complexity of natural text, there is still a long way before being able to produce accurate, cooperative, and robust QA systems. Lately, several communities have started to work together on common grounds to achieve these goals.

1.4 Statement of the Problem

Given a set of candidate sentences each containing a fair number of question keywords, deciding which one actually answers the question is a challenging decision that a question answering system must solve. Some statistical systems use lexical patterns built by reformulating the question words to filter sentences that seem to be irrelevant to the question and boost the ones that seem to be more similar to the answer sentence. Others return the most frequent phrase in the set that is of the expected answer type as the answer. Linguistic systems however, take into consideration the syntactic relation of key words to make a more informed decision on the relevance of a candidate sentence to the question.

In our work, we address the problem of finding the best candidate sentence (the $4th$ process in Figure 1.1) by measuring the syntactic and semantic similarity of candidate-sentences to the question. We combine three features for this purpose: the semantic similarity of the main verbs, lexical word overlap, and the syntactic overlap of counterpart subtrees. We will show that this linguistic method

does not require any domain knowledge and is portable to other domains. Specially, by providing it with the redundancy statistics from the Web, we will show that this ranking method is applicable to open domain.

1.5 Outline of the Thesis

In this thesis, we start off by reviewing different techniques used in state of the art question answering systems. We will look at different techniques as well as domains of application in the literature review chapter (Chapter 2). For grounding the statement of the problem and providing a test bed to evaluate our work, we used two document collections. We present the characteristics of each collection and the question sets in Chapter 3.

We describe our syntactic method for ranking the candidate sentences in Chapter 4. Chapter 5 explains the processes involved in extracting candidate sentences to be ranked by this method. Chapters 6 and 7 give evaluation results in closed and open domains respectively; the error introduced while extracting candidate answers is reported separately to focus on the accuracy of our candidate ranking method. Finally, we will conclude and give future directions for our work in Chapter 8.

Chapter 2

Literature Review

Most current question answering systems exploit web searchers and the redundancy of candidates: they return the most frequent noun phrase in proximity of the question keywords, which is of the expected answer type. The most redundant noun phrase, is assumed to be most probably the answer.

QA systems that incorporate linguistic analysis however, rely on precise input, such as a correct parse tree, which results in answers of high precision while recall stays relatively low. Syntactic and semantic analysis of sentences is also relatively costly and not feasible to apply on all the candidate sentences.

In this chapter we review statistical and linguistic QA systems designed to answer open domain or closed domain questions. We categorize the systems that use high-level linguistic tools, such as linguistic analysis of parse trees as linguistic, while the ones that work with abundant samples for learning or optimization as statistical.

2.1 Open Domain Linguistic Approaches

The techniques presented here are potentially applicable for texts with complex content (such as ambiguity, co-reference structures, multiple interaction of entities, etc.). Linguistic techniques generally have a high precision while their recall is low.

Knowledge-rich approaches are costly with respect to their processing time: converting available unstructured text to some recognizable and meaningful structure (such as a parse tree) is costly and

error-prone. So they are not applicable to the whole corpus; hence, such techniques are usually run on small passages retrieved by fast knowledge-poor techniques.

2.1.1 Using the Semantics of Arguments

In this section, we review a number of semantic-based methods that understand the question and candidate answers to some degree before selecting the answer.

The National University of Singapore QA System

With one of the best performing QA systems at the TREC 2004 track, the National University of Singapore QA [SJT+04] uses the Jaccard coefficient to test pair wise similarity of the predicate arguments marked by the ASSERT parser [PWH+04]. ASSERT is a shallow parser trained on the PropBank [PGK05] corpus and uses support vector machine classifiers. It tags the arguments of a verb with labels from ARG0 up to ARG5, as suggested in PropBank. The Jaccord coefficient ignores stop-words and uses a bag-of-words feature for scoring. To measure the semantic similarity of the question verb v_1, from the verb in the answer frame, v_2, they look for v_2 starting from the first sense of v_1, using hyponymy, entailment, and gloss links from WordNet. The final similarity score will be $\sqcap_1^L W_i$, in which $\sqcap$ is the multiplication symbol, L is the length of the relation path, and W_i is the weight of relation R, the same as Moldovan et al. used in [MN02]. Their semantic expansion is limited to a depth of two, since further expansion does not improve the results.

The similarity of two frames is computed as

$$Sim(F_1, F_2) = \alpha \times Sim_v(v_1, v_2) + (1 - \alpha) \times Sim_A(A_1, A_2)$$

combining the similarity of the verbs, $Sim_v(v_1, v_2)$, and their arguments $Sim_A(A_1, A_2)$.

The similarity between the two sets of arguments is measured at the lexical level by a fuzzy matching method; they consider all arguments in a frame together as a bag of independent terms. They argue that since semantically related verbs are considered, the semantic roles of the arguments might be different in different frames. Compared to our work, the use of Minipar [Lin93] gives us a label for each argument, and we let arguments of the same type to unify. In addition to bag-of-words however, we keep stop words and find the common syntactic links (e.g. *"his red car"* versus *"the car*

at the door of his house"). We gave this syntactic feature twice as much contribution.

To compute $Sim_A(A_1, A_2)$, where A_1 and A_2 are two argument sets [SJT+04],

$$A_1 = \{(l_{1,1}, T_{1,1}), (l_{1,2}, T_{1,2}), \ldots, (l_{1,m}, T_{1,m})\}$$

$$A_2 = \{(l_{2,1}, T_{2,1}), (l_{2,2}, T_{2,2}), \ldots, (l_{2,n}, T_{2,n})\}$$

where $l_{i,j}$ is the argument label of the jth argument of A_i and $T_{i,j}$ is the set of terms in the jth argument of A_i. The similarity is computed as:

$$Sim_A(A_1, A_2) = \frac{|T_1 \cap T_2|}{|T_1 \cup T_2|}$$

where, T_1 and T_2 are computed as below (stop words are removed from the text):

$$T_1 = \bigcap_{i=1}^{m} T_{1,i} \ \ and \ \ T_2 = \bigcap_{j=1}^{n} T_{2,j}$$

If the question and the answer passages contain more than one semantic frame, they compute pair wise frame similarity scores between a question frame and an answer frame, and pick the maximum score as representing the semantic similarity between the question and that specific passage.

For factoid TREC 2005 questions, they obtain an accuracy of 0.666, while the maximum result achieved was 0.713 (see Section 7.1 for a discussion on evaluation).

The QED System

The QED system [ABC+01] developed by the university of Edinburgh uses an A^* algorithm to unify the question-DRS (Discourse Representation Structure) with the passage-DRSs and picks the one with the highest score. In the scoring process, the highest score is given to the passage that perfectly matches the question: i.e. the same verb, arguments and argument order, as in the question. Low scores are considered for less perfect matches (obtained by relaxed unification: synonym verbs and nouns, unifying a verb with a more general or specific one). Again, here the level of detail this system considers is higher, specially with their Prolog logic prover. In this configuration, explicit predicates embed world knowledge.

In a similar work, Kaisser et al. [KSW06] introduce a similar semantic structures built from the entries in the FrameNet [BFL98], VerbNet [Sch05] and PropBank [PGK05] semantic lexicons. They generate potential answer sentences to a given question using these lexical resources. They use the sample sentences for each frame provided in the lexicons for this purpose. They show that using this method, they can indeed search for answers that are indirectly related to the question[1].

In facing a question such as *"Who bought YouTube?"*, they parse it using MiniPar [Lin93] and simplify the resulting parse tree into a semantic structure:

```
head: purchased(V)
subj: Who
whn:  Who
obj:  YouTube
```

The main verb (head) is then sent to the lexical resources and the semantics of the answer (who) as well as patterns that predict occurrence of the answer are produced:

```
Buyer[Subj,NP] VERB Goods[Obj,NP] Seller[Dep,PP-from]
Buyer[Subj,NP] VERB Goods[Obj,NP] Money[Dep,PP-for]
Buyer[Subj,NP] VERB Goods[Obj,NP] Recipient[Dep,PP-for]
```

Using the *Commerce-goods-transfer* semantic relation between the frame 'buy.v' from FrameNet and the frames 'purchase.v', 'retail.v', 'retailer.n', 'vend.v', 'vendor.n', 'sale.n' and 'sell.v', lexical answer patterns such as the following are generated[2]:

```
ANSWER[NP] purchased YouTube
ANSWER[NP] (has|have|had) purchased YouTube
YouTube (has|have) been purchased by ANSWER[NP]
ANSWER[NP] bought YouTube
ANSWER[NP] (has|have) bought _YouTube
YouTube (has|have) been bought by ANSWER[NP]
YouTube was sold to ANSWER[NP]
YouTube had been sold to ANSWER[NP]
...
```

Since none of the three lexical resources they use contain an entry for the verb 'be' (copulative questions as we categorized them), this method is only applied on questions with a content main

[1] Lexical match of the verbs is considered to be a direct relation between the question and the answer sentence.
[2] Syntactic variations in active/passive, past/present, etc. senses are applied here as well.

verb. For 254 such factoid questions of TREC 2002, the baseline still perform better (34%) than the combined result they achieve (26%); the baseline simply uses a simulated lexicon entry with the following three senses for every non-copulative verb:

```
sense 1 (intransitive): [ARG0] VERB
sense 2 (transitive): [ARG0] VERB [ARG1]
sense 3 (ditransitive): [ARG0] VERB [ARG1] [ARG2]
```

For factoid 2006 TREC questions however, they achieve a precision of 32.3%, placing their results in the top 5 best performing methods.

The Sapere Prototype QA System

Katz and Lin [KL03] have a ternary subject-verb-object scheme and use predicate logic; applying a constraint satisfaction method to find an answer that satisfies the syntactic/semantic constraints on the answer. Our method provides flexibility in matching the arguments and allows them to have a slight syntactic difference. They focus on the following two problems and adapt a first order predicate logic formalism to address them:

1. 'Semantic symmetry' such as *"What do frogs eat?"* versus *"What eats frogs?"*. These two sentences are similar only at the word level.

2. 'Ambiguous modification' for example in *"the largest volcano in the Solar System?"* versus *"the largest planet in the Solar system"* or *"Even the largest volcanoes found... backyard, the Solar System"*. In each sentence, the adjective *'largest'* modifies a different entity.

The applicability of this comprehensive state-of-the-art method is shown successfully on sixteen questions, with a precision of 0.84 ± 0.11, while basic keyword matching results in a precision of 0.29 ± 0.11. In larger scale open-domain questions though, breaking the text into small grain predicate logic form is less feasible.

2.1.2 Logic and Proof

LCC's PowerAnswer 3 QA System

In Language Computer Corporation's PowerAnswer 3 system [MBT06], the best performing TREC 2006 QA system, a target and its related questions are transformed into fine grained logic entities for the COGEX [MCH03] logical prover to extract the correct answer. Different strategies are developed to address various classes of questions. The question processing module emphasizes on the temporal constraints on the target and each question. If a date that unifies with the temporal requirements of the input question is detected, a temporal boost is applied to the candidate answer (scaled according to the syntactic proximity of the detected date to the candidate answer). The eXtendedWordNet [MM01] lexicon provides the world knowledge information for reasoning steps in the natural language logic prover, COGe.g. COGEX uses a three layered logical representation which captures the syntactic, semantic and temporal propositions encoded in a text fragment; negations, quantifiers and conditional statements are detected and resolved using automatically learned transformation rules. Keyword alternation information is provided by the Question Processing module as well.

After analyzing the logic form and the parse trees of each text fragment, several NLP axioms are used to break down complex nominals and coordinating conjunctions into their constituents. Lexical chain axioms from WordNet help identify the semantic entailment between the question and a candidate answer. For example, given the eXtended WordNet lexical chain

```
(v-call#4, send_for#1) -DERIVATION-> (n-caller#5  -HYPERNYM->
(n-announcer#1) -DERIVATION-> (v-announce#3)
```

between call.v and announce.v, the system constructs three axioms:

```
call_VB(e1,x1,x2) -> caller_NN(x1),
caller_NN(x1) -> announcer_NN(x1) and
announcer_NN(x1) -> announce_VB(e1,x1,x2).
```

Heuristics ensure the high quality of these relations; for example, the length of a relevance chain is limited to three and chains with an IS-A relation followed by a HYPONYMY link are discarded.

Allowing the later relation, for instance, would infer *"John lives in Detroit"* from *"John lives in Chicago"*:

```
Chicago -IS-A-> city -HYPONYMY-> Detroit
```

Discovering and implementing insufficient knowledge or inference rules needs deep analysis of QA reasoning steps done by linguists/experts. This system obtained an accuracy of 57.8%, the highest scored TREC 2006.

The JAVELIN QA System

In another work, Nyberg [DHKN03a] introduces a light-weight fuzzy unification as an extension to an earlier work, JAVELIN [NMC+03]. Table 2.1 shows how the question *"When was Wendy's founded?"* and part of the passage (answer in this example) are converted to predicate logic. In the question for example, x_6 is an object of x_2, the founding event, occurring in the past ($TENSE(x_2, |past|)$), and so on. Table 2.2 explains the meaning of selected predicates in the logical sentences used. This representation is built from grammatical functions, and does not utilize semantic roles. Grammatical functions are extracted using *KANTOO* [DHKN03b] for converting questions and the *Link Grammar Parser* [GLS95] for analyzing answer passages. KANTOO is a parser which produces a deep, detailed syntactic analysis of the question and a functional structure (f-structure) which specifies the grammatical functions of its components, such as subject, object, adjunct, etc. The output of KANTOO for the question *"When was Wendy's founded?"* is shown in Figure 2.1.

The following is the output of the Link Parser for the sentence *'David Thomas founded Wendy's in 1969'*:

```
  +------------------------Xp-------------------------+
  |                         +-------MVp--------+      |
  +---------Wd----------+   +------O-------+   |      |
  |         +-G-+---G--+--S--+        +-YS--+   +-IN+   |
  |         |   |      |     |        |     |   |   |   |
LEFT-WALL R. David Thomas founded.v Wendy s.p in 1969 .
Constituent tree:
(S (NP R. David Thomas)
   (VP founded
       (NP (NP Wendy s))
       (PP in
  (NP 1969)))
.)
```

```
(  (Brill-pos VBN)
   (adjunct (
     (object (
       (Brill-pos WRB)
       (atype temporal)
       (cat n)
       (ortho ``When")
       (q-focus +)
       (q-token +)
       (root when)
       (tokens 1)))
     (time +)))
   (cat v)
   (finite +)
   (form finite)
   (modified +)
   (ortho founded)
   (passive +)
   (punctuation (
     (Brill-pos ``.")
     (cat punct)
     (ortho ?)
     (root ?)
     (tokens 6)))
   (qa (
     (gap (
       (atype temporal)
       (path (*MULT* adjunct
object))))
     (qtype entity)))
   (root found)
   (subject (
     (BBN-name person)
     (Brill-pos NNP)
     (cat n)
     (definite +)
     (gen-pn +)
     (human +)
     (number sg)
     (ortho ``Wendys")
     (person third)
     (proper-noun +)
     (root wendy)
     (tokens 3)))
   (tense past)
   (tokens 5))
```

Figure 2.1: Output of the KANTOO parser for the question *"When was Wendy's founded?"*

When was Wendys founded?	R. David Thomas founded Wendys in 1969.
ROOT(x6, —Wendys—)	ROOT(x6, —Wendys—)
ROOT(x2, —found—)	ROOT(x2, —found—)
ADJUNCT(x2,x1)	ADJUNCT(x2,x1)
OBJECT(x2,x6)	OBJECT(x2,x6)
SUBJECT(x2,x7)	SUBJECT(x2,x7)
	ROOT(x7, —R. David Thomas—)
TYPE(x2, —event—)	TYPE(x2, —event—)
TENSE(x2, —past—)	
ROOT(x1,x9)	ROOT(x1, —1969—)
TYPE(x1, —time—)	TYPE(x1, —time—)
ANS(x9)	

Table 2.1: Conversion of the questions and retrieved passages to logical forms [NMC+03].

predicate	example	comments
ROOT	ROOT(x13, —John—)	the root form of entity/event x13
OBJECT	OBJECT(x2,x3)	x3 is the object of verb or preposition x2
SUBJECT	SUBJECT(x2,x3)	x3 is the subject of verb x2
DET	DET(x2,x1)	x1 is a determiner/quantifier of x2
TYPE	TYPE(x3, —event—)	x3 is of the type *event*
TENSE	TENSE(x1, —present—)	x1 is a verb in *present* tense
EQUIV	EQUIV(x1,x3)	semantic equivalence: *apposition:* "John, a student of CMU" *equality operator in copular sentences:* "John is a student of CMU"
ATTRIBUTE	ATTRIBUTE(x1,x3)	x3 is an adjective modifier of x1: adjective-noun: "stupid John" copular constructions: "John is stupid"
PREDICATE	PREDICATE(x2,x3)	copular constructions: "Y is x3" ROOT(x2, —be—) SUBJECT(x2,Y) PREDICATE(x2,x3)
POSSESSOR	POSSESSOR(x2,x4)	x4 is the possessor of x2 "x4s x2" or "x2 of x4"
AND	AND(x3,x1) AND(x3,x2)	"John and Mary laughed." ROOT(x1, —John—) ROOT(x2, —Mary—) ROOT(x4, j laugh j) TYPE(x4, —event—) AND(x3,x1) AND(x3,x2) SUBJECT(x4,x3)
ANS	ANS(x1)	only for questions: x1 indicates the answer

Table 2.2: Meaning of selected predicates used in JAVELIN [NMC+03].

Then, a unification algorithm is used to match question representations with the representations of the extracted passages (which might contain an answer). JAVELIN then uses a fuzzy unification, since a strict boolean unification based on lexical items is not as flexible as is needed:

> For instance, knowing that *Benjamin killed Jefferson* would not answer the question *Who murdered Jefferson?*, using a strict unification strategy [NMC+03].

To implement the *'fuzzy unification'* they treat the relations between question terms as a set of weighted constraints; the more constraints the passage satisfies, the higher the score that candidate passage gets. A number of similarity measures are also used between relevant terms ($sim(|Kill|, |Murder|) = 0.8$ for example).

However, the similarity function domain of words is limited in this approach. They use WordNet, and similarity is limited to the words in one synset. A complete approach to word similarity also

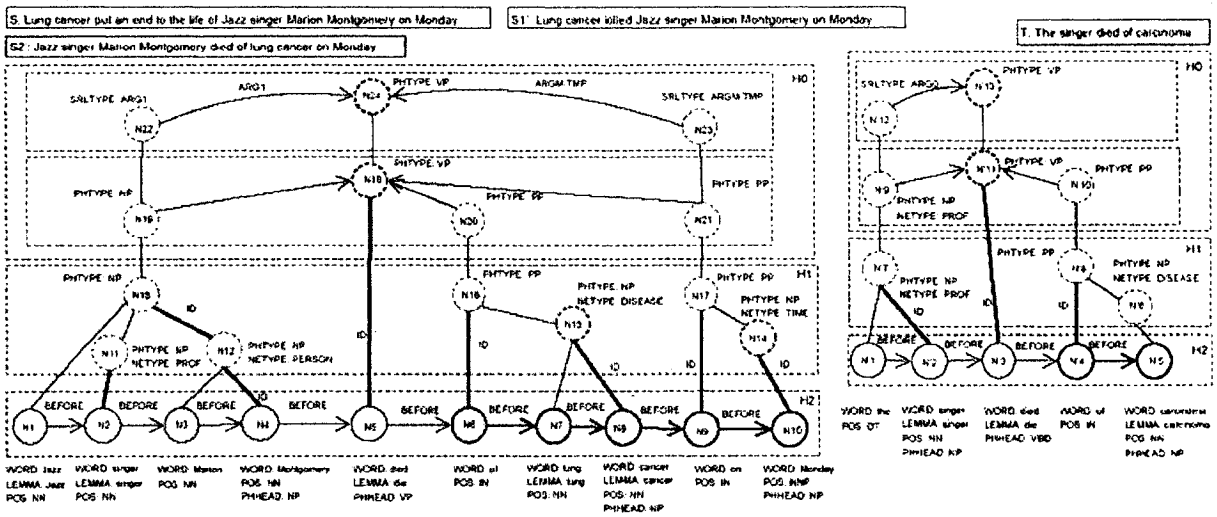

Figure 2.2: Concept graphs of the source and target sentences for meaning entailment.

will require a reference resolution, which makes the work more challenging.

For this linguistic work however, no evaluation result is provided.

2.1.3 Parse-tree matching

In the domain of language, graphs provide a natural way to express the dependencies between words and phrases in a sentence. Graph matching provides a framework for structural matching of phrases that would be difficult to resolve at the lexical level.

The University of Illinois Semantic Entailment System

Salvo et al. [dSBGP+05] introduce a hierarchical knowledge representation for Meaning Entailment: a sentence is entailed by a paragraph if its concept graph can be unified with that of the paragraph. A cost function determines the goodness of a unification. Unified nodes must be at the same level in the hierarchy, and the cost of unifying nodes at higher levels dominates those of the lower levels. Nodes in both hierarchies are checked for subsumption in a top-down manner. Figure 2.2 shows an example. Here, the hierarchy level H_0 consists of verbs that unify if they are synonyms based on WordNet and their constituent phrases at H_1 level unify. The hierarchy set H_2 corresponds to word-level nodes. As can be seen, syntax is used only at the topmost level H_0. As we will see later, subject and object relations are considered to be critical in our matching algorithm.

Semantic Entailment at the University of Stanford QA System

Haghighi et al. [HNM05] have tried to learn the cost of matching the hypothesis parse tree to the text in the Textual Entailment Recognition problem.

They parse a sentence using a slightly modified Collins' parser and then collapse collocations and named entities (e.g., "Jeff Bezos" is replaced by "Bezos") and also collocations listed in WordNet, including verbs and their adjacent particles (i.e., "blow off" is transformed to "blow"). Additionally, they fold certain dependencies (such as modifying prepositions) so that modifiers become labels connecting the modifier's governor and dependent directly ("in 1982" will become a *temporal* relation). Finally, verb arguments are semantically labeled; specifically, each predicate adds an arc labeled with the appropriate semantic role to the head of the argument phrase. This helps to create links between words which share a deep semantic relation not evident through the surface syntax.

Having such an augmented graph, the matching cost function is defined as $Cost(M) = \alpha VertexCost(M) + (1 - \alpha)RelationCost(M)$ where,

$$RelationCost(M) = \frac{1}{Z}\Sigma_{e \in H_E} w(e) PathSub(e, M(e))$$

and $PathSub(e, M(e))$ is the cost of substituting a direct relation $e \in M$ for its counterpart, $M(e)$, $w(e)$ is the weight of each hypothesis edge, based on the edge's label, Z is a normalization factor and finally, α shows the relative contribution of each feature to the final result value.

Minimizing this function is computationally hard, since if the PathCost model assigns zero cost only for preserving edges, then $RelationCost(M) = 0$ if and only if H is isomorphic to a sub graph of T. As an approximation, they try to find the matching M^* which minimizes the $VertexCost$. They then perform a local greedy hill-climbing search, beginning from M^* to approximate the minimal matching. The suggested optimization approach has not yielded weights which improves performance very much over the hand-set initialization values. That is due to the presence of several local maxima and ridges in the parameter space. Figure 2.3 shows an example graph matching between *"Bezos established a Company"* and *"Jeff Bezos found Amazon.com in 1991."*.

The Graph Matching system fits a single global threshold / features from the development set and has shown an accuracy of 56.8% for the task.

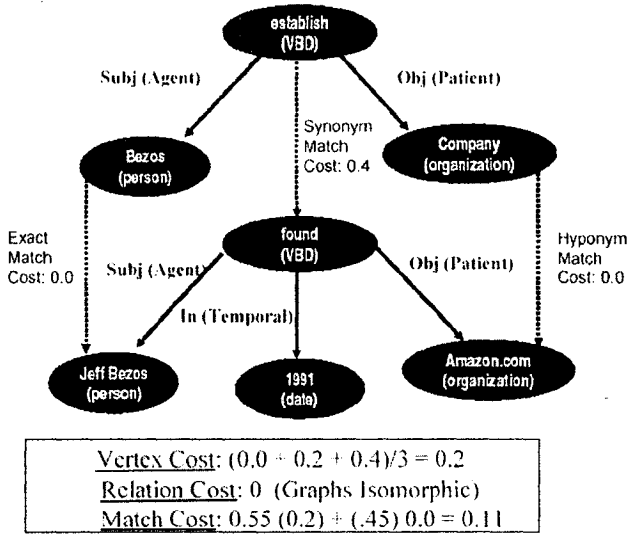

Figure 2.3: An example graph matching ($\alpha = 0.55$). Dashed lines represent mappings [HNM05].

2.2 Open Domain Statistical Approaches

Knowledge-poor techniques are fast and applicable to large amounts of (unstructured) data. Moreover, in contrast to knowledge-rich approaches, when dealing with complex syntactic structures in the text or abundance of information (on the web), knowledge-poor approaches can provide an approximate solution.

On the other hand, knowledge-poor approaches ignore any a priori knowledge in the text; for example, data represented in tables in web pages or sometimes the ordering of words are ignored.

2.2.1 Parse-tree based

Although the approaches we mention in this section use linguistic tools, they use statistics in critical answer extraction or candidate ranking tasks. These statistics are based on observations from a sample of question/answer pairs used for learning beforehand.

Dependency Tree Mapping by UIUC QA Application

Punyakanok et al. [PRtY04] try to match the parse tree of the question with the ones from candidates in order to find the best match as well as mark the answer itself. Inspired by work in machine translation, they select the mapping that results in the minimum *edit distance.*

They reformulate each question into a statement form by replacing the question word (e.g.

'what', 'when', or 'where') with a special token *ANS*, that stands for the answer phrase that will be extracted. The Collin's parser [Col97] is used to parse this modified question and the candidate sentences, before the tree mapping method is performed on them.

The following penalty values are determined experimentally for *delete, insert* and *change* operations in a labeled tree transformation algorithm:

```
delete: = 5, if w is a stop word,
        = 200, otherwise.
insert: = 200, if w is a stop word,
        = 5, otherwise.

change of w1 to w2:
if w1 is *ANS*,
        = 5, if w2 matches the expected answer type,
        = 200, otherwise.
else
        = 0, if word w1 is identical to word w2,
        = 1, if w1 and w2 have the same lemma form,
                        = 200, otherwise.
```

In other words, deleting or inserting a content word from/in the candidate parse tree is costly (200 points) while these operations on a stop word in order to modify the tree to match the question tree is allowed without a significant cost. Modifying a word to another is very costly, unless those words are from the same lemma or semantic type.

Their evaluations on 500 questions from TREC-2002 shows an MRR of 36.6% that is just a little bit higher than a simple bag of words method (the baseline) of 30.2%. This evaluation is done at the document level. The error in finding the actual position of the sentence and the answer phrase in that sentence will lower these numbers.

The MULDER QA System

Figure 2.4 shows the architecture of the MULDER QA system. Receiving a question, a Maximum Entropy Inspired (MEI) Parser[3] parses the question in order to compose high-quality keyword queries (reformulation) of the question [Cha00]. Search for the produced reformulations on the web is done in parallel for increasing processing speed.

Having the parse tree of the question, a question classifier module assigns one of the *Nominal,*

[3]The PC-KIMMO lexical analyzer was embedded into the parser by the MULDER developers to compensate for its limited vocabulary. The limited vocabulary initiates some slow processes, and also increases the search space [KEW01].

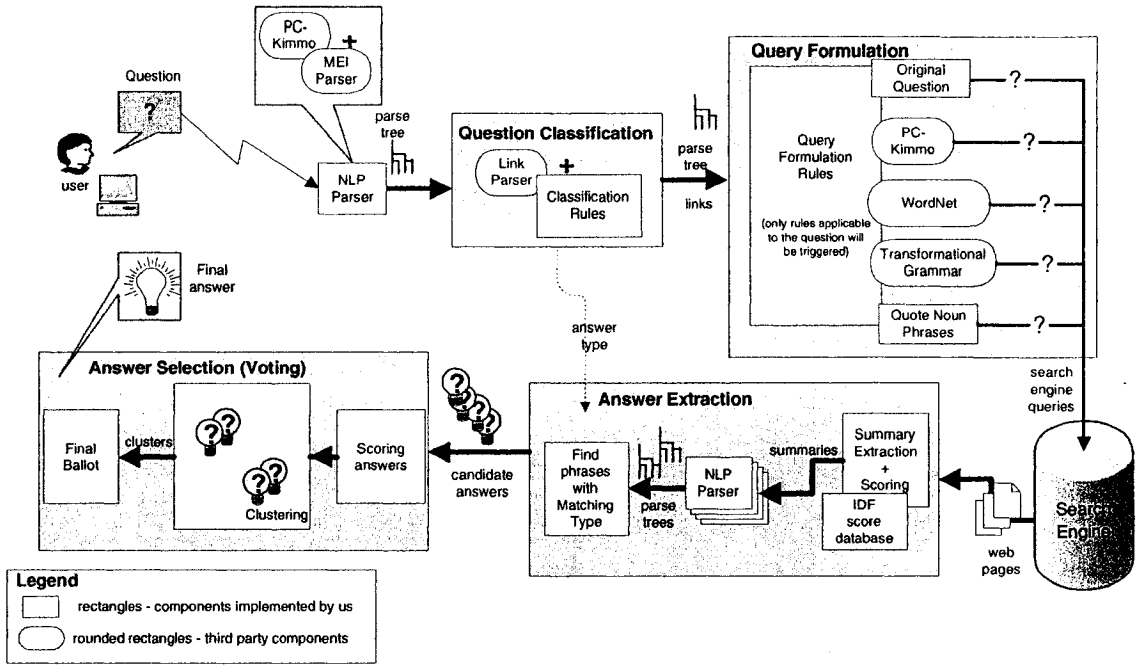

Figure 2.4: The MULDER system architecture [KEW01].

Numerical and *Temporal* types to the question. They correspond to Noun Phrase, Number and Date answers: "This simple hierarchy makes classification an easy task and offers the potential of very high accuracy [KEW01]".

MULDER then uses the verb-object links (produced by LinkParser [ST91]) to find the object in the question. It also consults WordNet to determine the semantic type of the object. The following reformulation strategies are considered in the order of general to specific:

Verb Conversion: In *"When did Nixon visit China?"*, we expect the target sentence to be *"Nixon visited China ..."* rather than *"Nixon did visit China ..."*. To form the verb *"visited"*, they use PC-KIMMO's word synthesis feature.

Query Expansion: In wh-adjective questions such as *"How tall is Mt. Everest?"*, the answers may occur in sentences such as *"the height of Everest is ..."*. For these questions, WordNet is used to find the attribute noun of the adjectives. MULDER composes a new query by replacing the adjective with its attribute noun.

Noun Phrase Formation: Atomic noun phrases which are composed of more than one word, are quoted in the produced queries, in order to obtain more relevant web pages.

Transformation: Predicting the format and order of constituents in the answer helps in a more qualitative search. Some grammatical conversion rules are used in this module for producing alternative ways of expressing the answer.

For answer extraction, first the snippets of text (summaries returned by Google) are marked that are likely to contain the answer. MULDER ranks them and selects the N best ones. The scoring function prefers the summaries that contain more important keywords in the vicinity of one another. To determine the importance of a keyword, they use inverse document frequency (IDF) as their criteria. Each summary is then parsed using the MEI parser and obtains phrases of the expected answer type.

For answer selection, the candidates provided by the previous phase are clustered, in order to put similar answers together. Then a voting procedure is run, and the cluster with highest score wins, and is returned as the best answer[4].

Experiments show that MULDER outperforms both AskJeeves and Google text snippets. MULDER consistently requires less user effort to reach an answer than either system at every level of recall. AskJeeves has the lowest level of recall; its maximum is roughly 20%. Google requires 6.6 times more total effort than MULDER at 64.5% recall (Google's maximum recall). Since they use the web as the only information source, their results are not comparable to TREC systems.

The IBM QA System

Although the core of the IBM QA system is mainly knowledge-poor, it includes a number of linguistic techniques, which we will describe. These include question focus matching using WordNet and query expansion [IFR01].

Taking a two-pass QA approach, the IR retrieves passages from the corpus in the first pass. The words in these paragraphs are then used as the local context to expand the query. These new words are used in turn for a better ranking of passages in the second pass[5]. In the IBM QA system, each query is expanded by a set of twenty words. Finally, the answer selection model uses maximum entropy and new features are defined as whether to include the new words in the query or not.

The second pass analyses the paragraphs:

[4]Clustering has several advantages, such as reducing noise, allowing alternative answers, and helps in separating facts from fiction.

[5]In this re-ranking, half of the IDF of new words is added to the score of each sentence.

Focus Matching The first word in most question sentences, such as *Why, What, Who, Which, etc.* is called the *'Question Word'*. A word or a sequence of words occurring after this *'Question Word'* is considered to be the question focus. If such an optional focus exists for a question, the sentences which have words in *Hypernym* or *Hyponym* of this head have their score boosted by a *'Focus Score'* defined as:

$$S_f = (F + (3 - d_f) * D_f)$$

where F is the focus boost (usually set to 10% of the total IDF weight of the question words), d_f is the semantic distance between the head word and its counterpart in the candidate sentence, and finally D_f is the distance penalty for the focus.

For example, in the question *"What river in the US is known as Big Muddy?"*, the focus is found to be *river* and in WordNet, we see that the *"Mississippi River"* (a candidate answer) has a hyponymy relationship with 'River'. The value of d_f would therefore be 1 in this case. D_f is usually selected to be 4. This score is added to the sentence score, as *'Focus Score'*.

Exploiting Syntactic Dependencies Considering simple syntactic relations would result in effective improvements in information retrieval. For example, *'information retrieval'*, *'retrieval of information'*, *'retrieve more information'*, *'information that is retrieved'*, etc. are all reduced to *'retrieve+information'* pair, where *'retrieve'* is a head or operator, and *'information'* is a modifier or argument [SLPCW97].

Constraints are constructed on parse trees set up by shallow parsing. The dependency tree is mainly used for the following purposes [IFR01]:

- Constrain the match of branches;
- Analyze dependencies coming together with the question word in a question;
- Give credit to the proper named entity in a sentence.

  Using a parse structure gives the system the capability to score higher those answer candidates that have the words in a similar structure [IFR01].

For a simple example, consider the question together with its dependency syntax tree in Figure 2.5 asking about the inventor of the "paper clip". Now consider the sentence " *'Like the guy who*

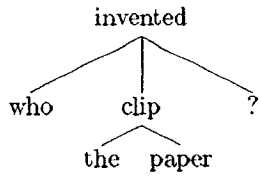

Figure 2.5: Dependency graph for the question *"Who invented the paper clip?"*.

invented the safety pin, or the guy who invented the paper clip', David says". By constructing the parse tree for this sentence, although *"invented the paper clip"* gets full credit by matching perfectly with the parse tree of the question, *'David'* gets no credit, since the dependency structure of the sentence does not satisfy the dependency derived from the question [IFR01].

IBM obtained a very high MRR of 39.0% on the TREC-10 questions. They believe that most of the answer selection algorithms still has space for improvement, while the performance of their IR engine equipped with a query expansion module is quite high.

2.2.2 Statistical and Pattern based Methods

Most of the time, in IR techniques, text is treated as a bag of words in statistical QA systems; however, simple extensions on this simple technique have resulted in more precise systems. For example, Mladenic [Mla98] reports an improvement in text classification only by extending the bag-of-word representation by using a word sequence of length 2. In this section, we review a few statistical methods for question answering in open domain.

IBM's Statistical QA System

Researchers in IBM have used statistical approaches to predict the class of the answer (answer tagging) and named entity tagging [IFR01] . There is also an effort to maximize the word-by-word overlap between question words and answer words. They apply a *Maximum Entropy* formulation for answer selection; the problem is to find an answer a to a question q that maximizes the conditional probability of $p(a|q)$. However, since the training data was not sufficient for directly learning $p(a|q)$, they proposed a *correctness distribution* criteria ($p(c|a,q)$), which measures the correctness (c) of an answer for a question.

A hidden variable, e, is introduced that represents the class of the answer. The classes can be:

Name Expressions: person, salutation, organization, location, country, product.

Time Expressions: date, date-reference, time.

Number Expression: money, percent, cardinal, ordinal, age, measure, duration.

Earth Entities: geological objects, areas, weather, plant, animal, substance, attraction.

Human Entities: events, organ, disease, occupation, title-of work, law, people, company-roles.

Unknown Phrase: selected when no category can be recognized.

The following formula measures the correctness of an answer for a question [IFR01]:

$$p(c|q,a) = \sum_e p(c,e|q,a) = \sum_e \frac{p(c,e,q,a)}{p(q,a)}$$

$$= \sum_e \frac{p(c|e,q,a)p(e,q,a)}{p(q,a)}$$

$$= \sum_e \frac{p(c|e,q,a)p(e|q,a)p(q,a)}{p(q,a)}$$

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

The term $p(e|q,a)$ represents the answer tagging problem, and $p(c|e,q,a)$, the answer selection problem. For using a maximum entropy distribution, a number of features are required. They are mostly derived from answer candidate strings and answer candidate sentences. IBM uses the following 31 features:

- the document rank as returned by the IR engine,
- twelve features from the answer sentence, and
- eighteen features from other answer candidate strings.

The training data for the answer selection model is drawn from question number 251 to 450 from TREC-9 and 200 questions from TREC-8. As mentioned before, this system achieves an MRR of 39.0% at TREC-10.

RMIT/CSIRO QA System: Pure Statistical IR

The RMIT/CSIRO QA system [FKK+00] computes the lexical similarity of a candidate to the question using experimentally obtained formulas. For example,

$$sim(s,q) = \frac{1}{w_s w_q} \sum_{t \in s \cap q} w_{s,t} \times w_{q,t} + count(t \in s \cap q)$$

where, s is the candidate sentence, q is the question, t is a term which exists both in the question and in the candidate, and finally, $w_{q,t}$ is the weight of the term t in the question q computed as:

$$w_{q,t} = log(number\ of\ times\ t\ appears\ in\ q + 1) \times log(\frac{total\ no\ of\ docs}{no\ of\ docs\ where\ t\ occurs} + 1)$$

and

$$w_q = (\sum_{t=1}^{n} w_{q,t}^2)^{1/2}$$

As the authors noticed, multi-word terms are over favored in such a scoring. They therefore suggest to use linguistic parsing to fix this problem.

QUANTUM

QUANTUM [PLK02] uses pre-defined lexical/part of speech based patterns called *extraction functions* to locate and extract candidate answers. Each question is decomposed in three parts by QUANTUM:

- 'Question Word',
- 'Question Focus',
- 'The Discriminant', which typically is the remaining part of the question, which is less bounded to the answer, than the 'question focus'.

The question is then classified in one of about 40 hand-written patterns based on lexical form, grammatical and noun phrase tags. Each function considers the lexical and syntactic structures of all the possible English questions which can fall in one category.

The answer is assumed to be a noun phrase. Some features are recognized for the answer noun phrase, its relation with the question word, and some syntactic or semantic relation in the components of the answer passage, by the most appropriate extraction functions; for example, if the answer is a measurement, it should contain a number, or if it is a location name, it should start with a capital letter. More formally, $C = f(\rho, \varphi)$, where f is the extraction function, ρ is a passage, φ is the question focus, and C is the list of candidates found in ρ.

The *Okapi* IR system [RWHB+92] retrieves the relevant passages, searching the corpus. In addition to corpus search using *Okapi*, a number of question reformulations are sent to web. The main idea in this module of QUANTUM [PLK02] is to anticipate the context and formulation of the answer for a question. For example, for *'Who is the prime minister of Canada?'*, an answer can be *"The prime minister of Canada is* $< PERSON - NAME >$*"*, so proposing the idea to search for the exact phrase *"The prime minister of Canada is"*, and then, in the found candidates, checking whether the next phrase is a $PERSON$ (using a name entity tagger). The Yahoo! web search engine was used to retrieve a number of passages which were treated like *Okapi*'s output.

Since redundancy of the answer is one of the credibility features in web search, this module increases the score of a candidate answer when it is repeated in text snippets.

After finding an answer on the web, *Okapi* is used to search the corpus to find a document to accompany the answer. An NP-chunker and an NE-tagger are run on these passages, to derive features for the retrieved passages for further checks and filtering. As an example, in

$CLAUSE$ *colonized Hong-kong.*

$CLAUSE$ is supposed to be any noun phrase which is a country name as well; a validity check is performed on the second criteria.

Simple semantic relations of Hypernymy and Hyponymy are considered for answer candidates, if current specifications are not useful. Table 2.3 shows some examples of what extraction functions look for, among which some semantic relations are used.

Apparently, not using semantic relations in the IR phase, skips any answers semantically similar to the question keywords. Using *Okapi* score, and extraction score, which determines how well each passage satisfies anticipated features of the answer, candidates are ranked. The overall performance of QUANTUM is around 16% on TREC 2002 factoid questions, placing them below average.

Extraction function	Example of Criteria
$definition(\rho, \varphi)$	Hypernyms of φ
$specialization(\rho, \varphi)$	Hyponyms of φ
$cardinality(\rho, \varphi)$	Pattern: NUMBER φ
$measure(\rho, \varphi)$	Pattern: NUMBER UNIT φ

Table 2.3: Sample of extraction mechanisms for each extraction function.

2.3 Closed Domain QA Systems

The problem in comparing closed-domain systems is that the textual genre/complexity differ significantly across work. We are fortunate to have a colleage working on the same corpus and question/answer set, focusing on re-ranking the list of paragraphs returned by the Okapi [RWHB+92] IR system. Okapi can be seen as a naive QA system since it returns paragraphs, instead of a list of documents. Doan et al. [DNK06] apply a new scoring method which increases the score of candidates containing occurrences of domain terms found in the corresponding question. They use domain specific vocabulary to characterize the relevance of a candidate to the corresponding question better than the IR engine does. Their results show an MRR of 42.3% for Okapi (with MRR-1=30%) and 54.7% after their re-ranking[6] (MRR-1=50%).

In another work, in a reading comprehension task done by Diego Molla, et al. [Mol03], sentences are short and easy to parse: each document (story) has 5 questions related to it. They report an MRR of around 40% in the best system configuration for the reading comprehension task.

We have selected a few closed domain QA systems to present different techniques in the field. In addition to the final accuracy they achieve, for each system we analyze the type and complexity of questions and documents they deal with.

QA for the Construction Sector

Zhang et al. [ZSD+04] extend the general QA approach to domain-specific questions in the construction sector by embedding the Canadian Thesaurus of Construction Science and Technology hierarchy of concepts. The size of their collection is about 8M bytes, containing 240 articles from the Canadian Building Digest. As the test set, one hundred (100) actual, expert questions are collected with one

[6]This precision is at the paragraph level and is generally higher than our final results in closed-domain which are at the sentence level

paragraph manually associated with each one as the correct answer. 42% of them are named entity questions (e.g., *"What is the address of the Educational Facilities Laboratories Inc.?"*). They report an MRR value of 66.6% for Keyword search using Okapi, while the MRR value of their NE and semantic category search after post-processing the results is 76.9%. Another 40% of the questions belong to Category questions (e.g. *"What product is used to remove the stains caused by water?"*). The MRR value of keyword search is 47.9%, whereas the MRR for Category search is 55%. The remaining 18% do not belong to the two previous types, they are keyword questions and are simply evaluated by keyword search. For all the 100 questions, the MRR value of QA with their thesaurus post-processing is 65.5%.

WEBCOOP

Benamara in the WEBCOOP system [Ben04] uses three types of relaxation in order to cope with complex questions in the touristic domain: relaxation based on cardinality, the type of the question focus, and finally the constants. These relaxations are hard-coded in the QA system. Although they cover a large proportion of queries for touristic domain, they may not work for the categories that are not predicted by the system developers. Even the evaluation task can be subjective and be done on different subsystems (e.g. Templates in WEBCOOP.)

QA at Korea University

Chung, et al. [CSH+04] have built a high precision QA system for answering practical questions such as weather information, stock quote, TV broadcasting schedule, traffic information etc. via a home agent robot. This QA system processes semi-structured text data from the Internet about weather forecasts and stores it in the form of a relational database. A domain specific hand-coded ontology is manually built that contains weather terms and cities; it is used during question analysis and the inference process.

Their QA engine transforms the question into an SQL database query, executes the query and transforms the results into natural language form. Since they know the structure of specific web pages they use as their source of information, they can accurately and easily extract the information from the semi-structured web pages, compared unstructured data.

Their question analyzer extracts several keywords that describe the question, such as event word, date, time, and location, by using a dependency parser, and the user question is represented only by these extracted keywords. Using a profile of the user, this module infers the missing information such as the date, time, or location not expressed in the user's request.

They assume there is a finite number of expected question topics, each modeled as a *query frame*. For example, the following frames are used for precipitation forecast for the next day:

```
[PRECIPITATION_TOMORROW]
[DIURNALRANGE_TODAY]
[WINDDIRECTION_CURRENT]
[TEMPERATURE_CURRENT]
```

Each frame has a rule for SQL generation. For example, PRECIPITATION_TOMORROW has the following SQL generation rule:

```
SELECT date, amprecpr, pmprecpr FROM
forecast_tbl WHERE $date $city
```

The words 'date', 'amprecpr' and 'pmprecpr' are field names in the database table 'forecast_tbl', which mean date, the precipitation probability of morning and afternoon of the day.

Analyzing a question means selecting a query frame in this system. They use a hand-coded decision tree-like classifier for selecting an appropriate query frame for the extracted question keywords.

For transforming the information to a natural answer sentence, each query frame has an answer generation pattern. For example, DIURNALRANGE_TODAY has the following generation pattern.

```
The diurnal temperature range of $date($1) is $2 C
```

$1 and $2 are the first and second field value of the queried result and finally, $date() is the function that converts a normalized date expression to a natural language expression.

For evaluation, they have collected 50 weather questions from 10 graduate students. Precision and recall rates are 90.9% and 75.0% respectively. The low recall rate is due to some questions related to invalid date and topic (for example there's no information on the forecasting web pages for questions about later than 7 days later.)

The primary reason for the wrong answer is the failure of invalid topic rejection. It is due to the insufficient of weather prediction domain ontology data. For example, from the question *"What is the discomfort index calculated from the todays weather?"*, the keyword 'discomfort index' was not extracted while 'weather' was extracted, because the former was not in the ontology [CSH+04].

Jupiter

The Jupiter system [ZSG+00] is a conversational system that provides weather information over the phone. Jupiter recognizes user question over the phone, parses the question with the TINA language understanding system [Sen92]. It parses both the query and the sentences from weather reports[7] using a manually constructed grammar that encodes both syntactic and semantic information into the parse tree. They learn probabilities for a full parse and a robust parse solution on a large corpus of utterances using a fully automatic procedure. Interestingly, they prefer a full parse tree compared to a robust parse forrest with a higher score for the candidate. Having the question's semantic frame, an SQL query and finally the answer in natural language is generated by their GENESIS system.

For evaluations, they discard irrelevant questions with a precision of 60%. They use a training set of more than 2000 utterances tagged with 90% mutual judgment agreement for this learning. Jupiter's accuracy is evaluated on 2238 test utterances. Approximately 5% of the utterances are rejected because of much high recognition error rate. Based on the voice recognition output, 80% of the utterances are answered correctly; 13.1% of them had at least one word incorrectly understood, and 33.9% of them had an error in sentence recognition. Being consistently maintained, their parser has achieved an accuracy of 99% over a period of 11 weeks.

The Halo QA System

Mulkar et al., [MHH+07] describe the Halo logic proving based QA system in the Chemistry and Biology domains. They use two subsections of a high school chemistry text book and various paragraphs describing the human heart as their corpus. Parser problems are avoided by manually simplifying the sentences' syntactic structure where necessary by

- splitting sentences into two when joined by a conjunction,

[7] National Weather Service reports for example are available as text written by expert forecasters.

- removing appositive structures and writing it as a separate sentence.

Domain dependent axioms are crafted manually to enrich shallow logical forms with deep semantic information such as verb argument names such as *object-of*, *destination-of*, *agent-of* etc. in addition to simple *instance-of* relation extractable from a lexicon. Patterns capture three types of knowledge from sentences:

- General Facts such as *"An H+ ion is a proton."*,
- Causal Facts such as *"When bases are added to acids, they lower the amount of acid."*,
- Reaction Theory such as *"NaOH dissociates into Na+and OH- ions when it dissolves in water"*.

Sentences might need advanced forms such as existential and universal quantifiers.

Evaluations shows that correct logical forms were generated for 91 out of 133 sentences from the chemistry book. Parsing errors by the Charniak parser [Cha00], error in linking of modifiers with the syntactic head are reported as the major causes of error.

2.4 Conclusion

In this chapter, we reviewed linguistic and statistical systems in both open and closed domain. In open domain, some statistical systems such as [SJT+04] (Section 2.1.1) use linguistic features and process the semantic label of verb arguments, or [dSBGP+05] (Section 2.1.3) construct and map concept graphs of the question and candidate sentences and obtain very good results in open domain.

Pure linguistic criteria for measuring the similarity of sentences, however, impose very strict syntactic constraints that result in high precision, but low recall. This problem has been observed by researchers in the field, such as Renxu Sun et al. [SJT+04]. Some statistical systems, such as in the work of Milen Kouylekov et al. [KT04] and Nyberg et al. [DHKN03a] (Section 2.1.2), have tried to relax syntactic constraints; they learn to look for important syntactic links and only score these links. Such methods, however, are very lenient in considering the relative importance of primary roles (such as subject and object) over less important roles (such as a determiner or a modifier). The most interesting effort towards improving this syntactic measure is weighting the matching

links (that have similar head, relation and tail) according to their Inverse Document Frequency (IDF) [IFR01] (Section 2.2); rare link types have more information content than frequent relation types. This effort has not significantly improved the recall problem; in the end, most parse tree based techniques perform poorly compared to syntactically blind statistical methods.

In closed domain, since redundancy of answers does not exist, statistical QA systems are not likely to perform as well as in open domain. Most current closed domain QA systems, such as Zhang et al. [ZSD+04] and Benamara et al. [Ben04] (Section 2.3), embed world knowledge in the form of axioms or structured data in order to improve the QA accuracy. This prevents these QA systems to be portable to other domains.

In the following chapter we introduce characteristics of two document collections and their associated question sets used to develop and evaluate our question answering system.

Chapter 3

Document Collections

As mentioned in the introduction, this work initially targets closed domain questions about telecommunications. Later on, we will analyze the applicability of our method in open domain in order to see how biased it is to this specific domain and also evaluate our method on a standard question set. For this purpose, we will use the TREC QA factoid questions from the years 2003 to 2006. In this chapter we describe the development and test question/answer sets and the document collections we used for evaluation in both cases.

3.1 The Bell Corpus

To prepare our question/answer data set, we asked 15 students to assume themselves to be Bell Canada customers and compose questions relevant to the company's Web pages. The document collection is made up of 250 web pages, internal documents and a few technical manuals of Bell Canada. In total, it accounts for 500KB of text. Around 15 documents were assigned to each student. In general, the questions collected varied in style, length and complexity.

In order to have an idea of the complexity of our text and put it into perspective with the state of the art, we compiled a table of simple text complexity features and readability measures for our corpus, news articles from AQUAINT used in the standard open domain question answering (TREC QA) [VT99], short stories and grade 5 reading comprehension text often used in QA [Mol03] (Table 3.1).

Statistics	Bell Qs	TREC Qs	Bell	Short Stories	AQUAINT	Grade 5
Number of words	344	396	10,947	9,672	10,088	1,525
Number of sentences	39	59	551	460	637	174
Avg. # characters/word	4.36	4.35	4.89	4.44	4.66	4.21
Avg. # syllables/word	1.43	1.46	1.71	1.49	1.55	1.36
Avg. # words/sentence	7.82	6.25	19.87	21.03	15.84	8.76
Readability Measures						
Gunning Fog Idx	6.55	6.73	13.83	12.08	10.95	4.79
Flesh Kincaid Grade level	4.73	3.93	12.30	10.18	8.93	3.85
Automated Readability Idx	3.53	2.18	11.55	9.99	8.44	2.78
SMOG	8.04	7.78	13.64	11.94	11.24	6.74
Flesch Reading Ease	76.88	78.06	42.27	59.56	59.26	83.05

Table 3.1: Complexity of the Bell corpus compared to other genres of text.

As we expect in a closed domain (see Chapter 1) most questions are long, and complex (compare 'Bell Qs' and 'TREC Qs' columns in Table 3.1) and include mostly 'what' and 'how' type questions. For answering, some questions need knowledge of the domain, acronyms or synonyms, while some are very similar, lexically and syntactically to the answer sentence. In total, we randomly chose 45% of these questions for development, and kept the rest for testing.

The most popular *Readability Measures* [Gre04] show that the Bell texts have on average longer sentences and hence are of lower reading ease compared to *short stories*[1], news articles from the *AQUAINT* corpus, and grade 5 reading comprehension texts. Table 3.1 shows the relative complexity of randomly chosen 60KB of text from each of these collections. The *Gunning Fog index* indicates the number of years of formal education that a person requires in order to easily understand the text on the first reading. This criteria computes 13.83 years for our text and 10.95 for AQUAINT articles. The *Flesh Kincaid*, *Automated Readability Index (ARI)* and *SMOG* are approximate representations of the U.S. grade level needed to comprehend the text. These measures consistently judge the Bell text more complicated than the other genres.

Finally the *Flesch Reading Ease* measure indicates how understandable a text is. A text with the score in the 90 to 100 range is considered easily understandable by an average 5th grader; 8th and 9th grade students easily understand texts with a score of 60-70 (which is the case for Short Stories and *AQUAINT*), and texts with results of 0-30 are best understood by college graduates[2].

In order to break down the complexity of closed domain question answering and systematically

[1] We took 5 classic short stories from `http://www.bnl.com/shorts/`

[2] Reader's Digest magazine has a readability index of about 65, Time magazine scores about 52, and the Harvard Law Review has a general readability score in the low 30s.

analyze our results, we categorize the *question/answer sentence* pairs based on their lexical and syntactic similarity into three categories: *simple, average* and *challenging.* This way, we can concentrate on the *simple* category, and gradually build up on the NLP techniques/rules to solve the *average* and finally *challenging* questions. The criteria used for each category is presented in the following sections.

3.1.1 Simple Questions

Simple questions have most of the *question keywords* appearing in the answer sentence; so no a priori knowledge about technical terms, acronyms or synonyms is required. Both the question and answer sentences are in simple and conventional grammatical structures; lexical and syntactic similarity make it easy to analyze the syntactic similarity of a candidate sentence to the question.

Here is an example of a *simple* question/answer pair:

```
<question> What does IP Messaging Enterprise Standard service include? </question>
<answer> IP Messaging Enterprise Standard service includes POP3, IMAP and
    Webmail mailbox access. </answer>
<doc> B-GE-EBusiness-IPMessaging.txt3 </doc>
```

In this sample, all of the question keywords ('IP', 'Messaging', 'Enterprise', 'Standard', 'service' and 'include') appear in the answer sentence. The phrase *"IP Messaging Enterprise Standard"* is the 'subject' in both question and answer sentences (Figure 3.1); so the 'object' in this sentence is exactly what is asked for (by the question word 'What'). As we will see later, the exact lexical form for proper nouns is considered while for other content words, only their stem need to match. Note that we cannot say that simple information retrieval is enough to answer this type of questions: containing all question keywords is not sufficient for a sentence to answer the question. Syntactic analysis is an important task to be performed to decide if a candidate sentence with lexical overlap conveys the same meaning as the question. Also note that in our view of closed domain, it is more informative to return a sentence as an answer. A sentence provides some explanation about the answer for the user.

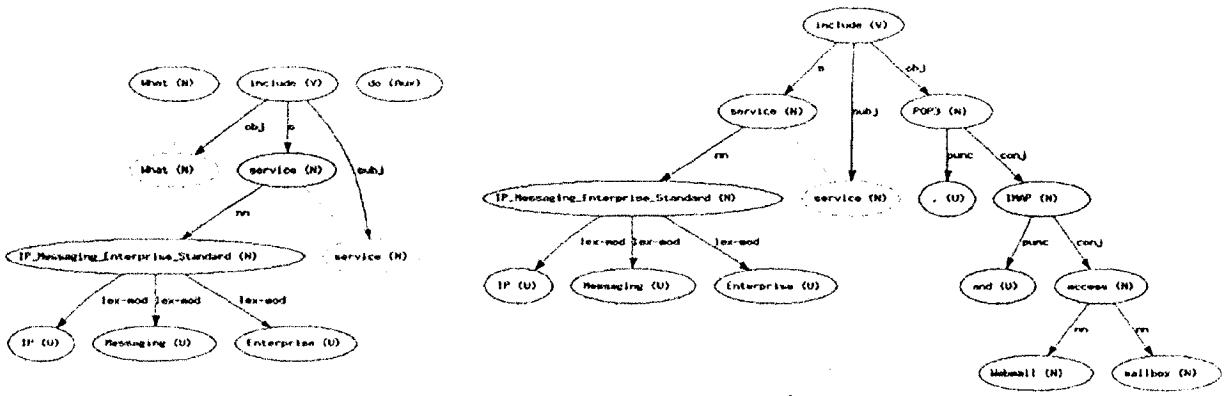

Figure 3.1: Parse tree similarity between a simple question (left) and its answer sentence (right).

3.1.2 Average Questions

Average questions are longer and have a more complex parse tree, compared to simple questions. The syntactic structure of the answer sentence is not exactly the same as the question and question keywords might not appear in the answer sentence in the exact same lexical form, requiring a lexical knowledge base such as WordNet [Lin98], or a dictionary for finding the semantic relation between them. For example, synonymy of the verbs *'allow'* and *'provide'* is needed before deciding the relevance of the following candidate to the question:

```
<question> What does Virtual FaxTM provide? </question>
<answer> Virtual FaxTM allows you to send and receive faxes anywhere
    as quickly and easily as regular e-mail, using your PC or wireless
    device, such as a PDA or RIM Blackberry, through a web based or client
    based e-mail interface.  </answer>
 <doc> B-I-VirtualFax.txt3 </doc>
```

Looking up the verb 'allow' in WordNet, shows that its fifth sense is a hyponym of the verb 'provide':

```
allow: Sense#5
leave, allow for, allow, provide for
   => yield, give, afford
   => supply, provide, render, furnish
```

Finally, in the following example, knowing that *"Bell Canada"* is the same entity as *"Bell Mobility"* is needed to match the subject of the question with that of the candidate answer (Figure 3.2).

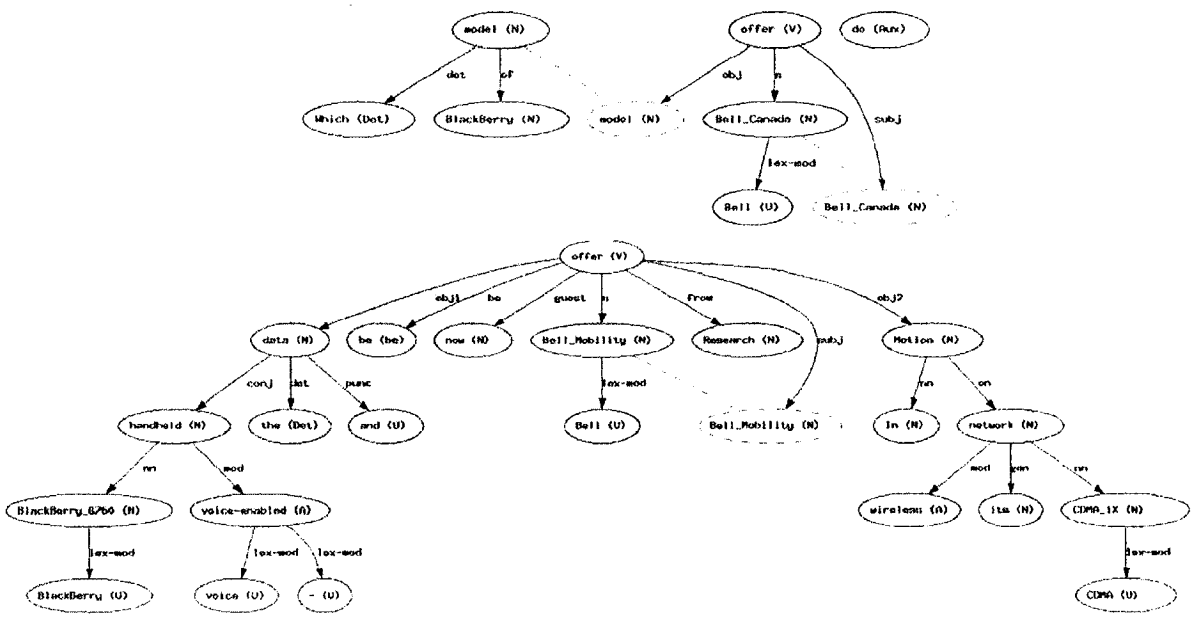

Figure 3.2: Parse tree similarity between an average question (top) and its answer sentence (down).

```
<question> Which model of BlackBerry does Bell Canada offer? </question>
<answer> Bell Mobility is now offering the data and voice-enabled BlackBerry
    6750 handheld from Research In Motion on its CDMA 1X wireless network.
</answer>
<doc> AB-PressReleases4Mobile.txt3 </doc>
```

3.1.3 Challenging Questions

Advanced NLP techniques such as coreference resolution and pragmatic knowledge are required to answer challenging questions.

In the following example, coreference resolution will show that *'it'* refers to *"Alphanumeric pagers"* that is referred to as the direct 'object' in the question.

```
<question> How can I customize an alphanumeric pager? </question>
<answer> Alphanumeric pagers provide you the ability to view a text message
    and/or numeric message on a pocketsize pager. Customize it with personal
    voicemail or subscribe to our Pagelink' service. </answer>
<doc> B-W-Paging.txt3 </doc>
```

Finally, in the following sample, detecting synonymy of the phrase *"no answer transfer"* and the sentence *"you don't answer"* is a challenge.

```
<question> What is no answer transfer? </question>
<answer> When someone calls your mobile, it will ring there first.
    If you don't answer, it will ring through to a second number predetermined
    by you. It can be your office, your home, your pager, or another mobile. You
    decide when and where you want to be reached. And if you can't pick up there,
    your caller can leave you a message.  </answer>
<doc> P-W-MFeatures-AnswerTransfer.txt3 </doc>
```

Note that this classification was done manually and although we used the lexical and syntactic similarity between the question and the answer sentence for classification, another annotator with a different solution perspective might introduce other criteria for their classification and might not necessarily agree on our classification.

3.2 The AQUAINT Corpus

For testing our candidate ranking method in open domain, we used the factoid questions from the TREC 2003, TREC 2004, TREC 2005 and TREC 2006 sets. In total, they comprise 1290 questions. We kept the first 50 questions from the 2005 set for our open domain development. With our syntactic perspective, we realized that questions that have a copula main verb have so much flexibility in connecting sentence entities to convey the same meaning. Equivallent syntactic structures are very hard to recognize from unequivallent transformations for this verb. Therefore, we categorized these questions based on their main verb type into copulative (having a 'to be' verb) versus non-copulative questions (having a 'content' verb). The question *Q-150.6:"What year was the program first broadcast?"* is an example of a copulative and *Q-150.2:"What network aired the show?* is an example of a non-copulative question about target number 150, *"television show Cheers"*, in the TREC 2006 question set.

Additionally, questions posed in our closed domain are mostly non-copulative (more than 70%) as opposed to the factoid questions in TREC which include around 33% non-copulative questions. Distinction between copulative versus non-copulative verbs will help in analyzing our closed domain ranking method on each category and pinpoint areas that it can handle and where it needs to be improved. We believe that in other domains as well there are more non-copulative questions than copulative ones because of the non-factual nature of closed domain questions. So, we will categorize them based on this syntactic criteria as well for our investigations. On the other hand, to categorize

the 1280 open domain questions based on their complexity into simple, average and challenging, we need the sentence in the corpus that answers each question, in addition to expert annotators for tagging and resolving possible conflicts. We decided to use our closed domain questions with the tagged level of complexity for our development. Analyzing the complexity of open domain questions in order to break down the complexity of building specific syntactic characteristics for each category is left as a future work.

The TREC data set also provides the answer to each question in the form of a regular expression and a set of documents that support it. For example,

```
81.3 (Bob )?Baffert APW19990515.0082 APW19990819.0158 APW19990217.0009
```

gives *"Baffert"* or *"Bob Baffert"* as correct answers to question 81.3 with three supporting documents for them.

The TREC question sets for the years 2004 to 2006 consist of question series where each series asks for information regarding a particular target. Targets includes people, organizations, events and other entities. Targets and questions were developed by NIST assessors after searching the document set to determine if there was an appropriate amount of information about each target.

The document set used for answering these questions is the AQUAINT document set. This collection consists of 1,033,461 documents, about 3GB in size, taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. Each file is named to reflect the source and date, and contains a stream of SGML-tagged text data presenting the series of news stories reported on the given date as a concatenation of DOC elements. The following is a sample from this collection.

```
<DOC>
<DOCNO> NYT19990430.0001 </DOCNO>
<DOCTYPE> NEWS STORY </DOCTYPE>
<DATE_TIME> 1999-04-30 00:01 </DATE_TIME>
<HEADER>
    A8974 &Cx1f; taf-z
    u s &Cx13; &Cx11; BC-BOX-TVSPORTS-COLUMN-N 04-30 0809
</HEADER>
<BODY>
<SLUG> BC-BOX-TVSPORTS-COLUMN-NYT </SLUG>
<HEADLINE>
    SPORTS COLUMN: A MARCIANO DOCUDRAMA GETS MUCH OF IT WRONG
```

```
</HEADLINE>

    (ATTN: Iowa) (rk)
    By RICHARD SANDOMIR
    c.1999 N.Y. Times News Service
<TEXT>
  <P>
    Toward the end of ''Rocky Marciano,'' a coming Showtime
    docudrama, the retired boxer flies to Denver on the final day of
    his life to visit Joe Louis in a psychiatric hospital. When he
    leaves, Marciano hands the hospital administrator a bag full of
    cash to upgrade Louis' care.
  </P>
  ...
</TEXT>
</BODY>
<TRAILER>
    NYT-04-30-99 0001EDT &QL;
</TRAILER>
</DOC>
```

In this chapter we presented the characteristics of the two document collections and the question sets that we will use for evaluating our work. In the following chapters, we present our linguistic candidate ranking method.

Chapter 4

Syntactic Unification for Candidate Ranking

In this chapter, we present the core of our work: a syntactic scoring method for ranking candidate answers of a question. In order to score each candidate sentence, we find the best subtree of each candidate and match it against the parse tree of the question. The purpose of this matching is to analyze the syntactic consistency of each candidate with the question. In other words, in addition to having important common key terms, the terms in a candidate sentence must have similar syntactic relations with one another in order to convey a meaning similar to the question's. Although this is not a sufficient condition, with current linguistic analysis available, we assume that similarity in critical syntactic relations imply similar semantics[1].

Technically speaking, to choose an appropriate subtree of the candidate sentence, this method applies a strict lexical constraint followed by a semantic constraint. Having guessed the relevant subtree of the candidate sentence (the target subtree), our method catches approximate syntactic similarity of this subtree to the question. This method is robust to minor syntactic differences and parsing errors; we consider wrong modifier or propositional attachment as minor errors while misplaced subject or object relations as a major parse error, because most of the times, entities in these two roles play the most important roles in conveying the meaning of the sentence. For this

[1] In the same spirit as *syntax driven semantic analysis*.

purpose, we strictly map subject, object and attaching subtrees in the target to their counterparts in the question. Then, a lenient syntactic measure in addition to a bag of words measure compute the unifiability of each of the subject, object and modifier subtrees with their counterpart. In the later process, the number of matching *word stems* in subtrees introduce a strong linguistic feature, while the number of matching *syntactic links* boosts the unification score. In this way, we contribute syntax while not being dependent on having a perfect parse tree.

Although both of these features have been investigated individually in different types of texts (e.g. [SJT+04, KL03]), to our knowledge, this novel combination of syntactic and lexical criteria is unique and new to the field. In the following, we describe this syntactic unification method in detail.

4.1 Choosing an Appropriate Subtree

First, we narrow down the search to the part of the sentence that is relevant to the question. Essentially, we believe that the best subtree in the candidate parse tree is the one that has a similar verb to the question's main verb as well as equivalent arguments that map on the argument of the question's main verb. Performing this test on every combination of verbs will find the most appropriate subtree of the candidate sentence (the target subtree). For long sentences having multiple verbs (which are numerous in closed domain), computing multiple verb similarities is expensive in terms of processing time. To avoid going through this comprehensive test, we exploit another feature for the target subtree. A strong verb similarity should co-occur with an essential entity match (question head) in the target subtree. Therefore, we consider the question head as an anchor to locate relevant subtrees, and mark the one(s) that have a semantically similar main verb to the question as the target subtree.

4.1.1 Finding the Question Head

To choose the question head, we rank all the noun phrases in the question and pick the one that contains the most valuable question keywords. Two factors contribute in deciding if a word is important in the question: the part-of-speech of that word, and the number of modifiers it has. Part-of-speech tagging is done through the Gate infrastructure which uses the Brill POS tagger [Bri92], with Penn Treebank notations. Specifically, we consider content words with the following part of

Category	Brill Part of Speech	Weight
Proper Noun	NNP	3.0
Common Noun	NN, NNS	1.0
Verb	VB, VBD, VBN, VBZ, VBG	0.75
Adjective	JJ, JJS	0.5
Adverb	RB, RBS	0.25

Table 4.1: Part of Speech weights used in ranking the keywords.

speech tags: NN, NNP, NNS, JJ, JJS, VB, VBG, VBN, VBD, VBZ, RB, RBS.

To rank question content words, we use two criteria: the number of modifiers of each word, and its part-of-speech. This strategy is based on the hypothesis that if a word has more modifiers, it acts as a central idea in the question and is therefore more important [LKR07]. For example, in the question *Q76.4- "What is the title of his all-time best-selling record?"*, the noun 'record' has three modifiers ('his', 'all-time' and 'best-selling') and therefore is considered very important in the question. Question keywords are ranked based on the following heuristic function [LKR07]:

$$Score(kw) = (\#modifiers + 1) \times Score_{POS}(kw)$$

As the equation shows, ranking also takes into account the part of speech of a word. Indeed, $Score_{POS}$ favors proper nouns, then common nouns, verbs, adjectives, and finally adverbs, in order to boost the distinctive words with significant meaning that do not convey multiple meanings to the top of the keyword list. The values we determined experimentally with our closed domain training set (see Chapter 6) are provided in Table 4.1. As part of evaluations, in Chapters 6 and 7, we will show that our scoring method is not very sensitive to these values, as long as the relative order of these classes is preserved.

The rationale behind these values is to boost proper nouns in the list, since they convey a unique meaning within the text. Verbs, on the other hand, are more ambiguous [Lin98] and can have more synonyms, so they are slightly pushed down the list. Adverbs usually modify verbs or adjectives, and not the nouns, so they receive the lowest rank.

Having marked the question head, we search for if in the candidate sentence parse tree. If this head phrase is found in the candidate sentence, it becomes an anchor to find the relevant verb. We then move up from this noun phrase in the candidate parse tree to reach the first parent verb. For

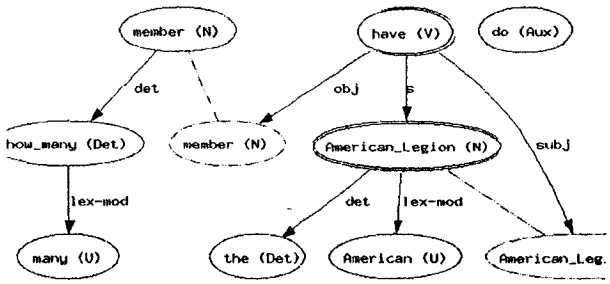

Figure 4.1: Parse tree of the question *"How many members does the American Legion have?"*

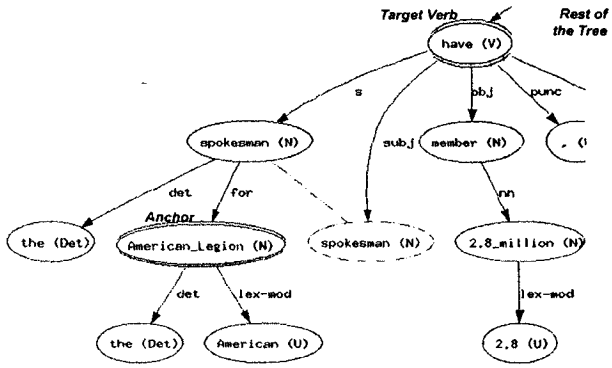

Figure 4.2: Parse tree of the sentence *"...said Phil Budahn, spokesman for the American Legion, which has 2.8 million members."*

example, in the question *Q98.5-"How many members does the American Legion have?"*, *'American Legion'* is chosen as the question head since it contains two proper nouns (see Figure 4.1). In the candidate sentence, *NYT19990316.0071-"...said Phil Budahn, spokesman for the American Legion, which has 2.8 million members."*, this anchor is found in the left subtree of the verb *'have'* (see Figure 4.2). Moving up from this anchor, we skip the noun node *'spokesman'* and mark the verb node *'have'* as the seed point for starting the unification (target subtree). In such a long candidate sentence as this one, using the anchor reduces the candidate verbs to only the ones that include the question head (or a reference to it).

4.1.2 Semantic Similarity of Verbs

Since the main action specified in a non-copulative question is typically realized by a verb, our first step is to verify the semantic relatedness of the question's main verb to the verb in the target subtree.

WordNet::Similarity [PPM04] provides six measures of similarity which use the information found in an *is-a* hierarchy of concepts (or synsets), *has-part*, *is-made-of*, and *is-an-attribute-of* relations and quantify how much concept A is similar to concept B. Three of the six measures of similarity

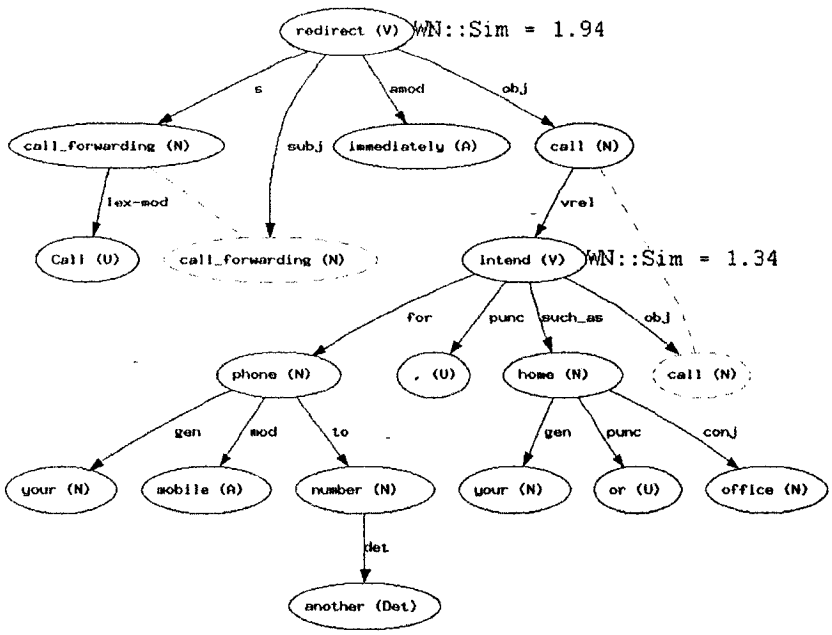

Figure 4.3: Finding a semantically similar verb in the sentence *"Call Forwarding immediately redirects calls intended for your mobile phone to another number, such as your home or office."* to the question's main verb 'work'.

are based on the information content of the least common subsumer (LCS) of concepts A and B. Information content is a measure of the specificity of a concept, and the LCS of concepts A and B is the most specific concept that is an ancestor of both A and B. These measures include *res*, *lin*, and *jcn* [PPM04].

The *lin* and *jcn* measures augment the information content of the LCS with the sum of the information content of concepts A and B themselves. The *lin* measure scales the information content of the LCS by this sum, while *jcn* takes the difference of this sum and the information content of the LCS.

The other three similarity measures are based on path lengths between a pair of concepts: *lch* (Leacock and Chodorow), *wup* (Wu and Palmer), and *path*. The *lch* measure finds the shortest path between two concepts, and scales that value by the maximum path length found in the *is-a* hierarchy in which they occur. The *wup* measure finds the depth of the LCS of the concepts, and then scales that by the sum of the depths of the individual concepts. The depth of a concept is simply its distance to the root node. The measure *path* is a baseline that is equal to the inverse of the shortest path between two concepts.

We chose the *lch* (Leacock and Chodorow) similarity measure from these six WordNet::Similarity

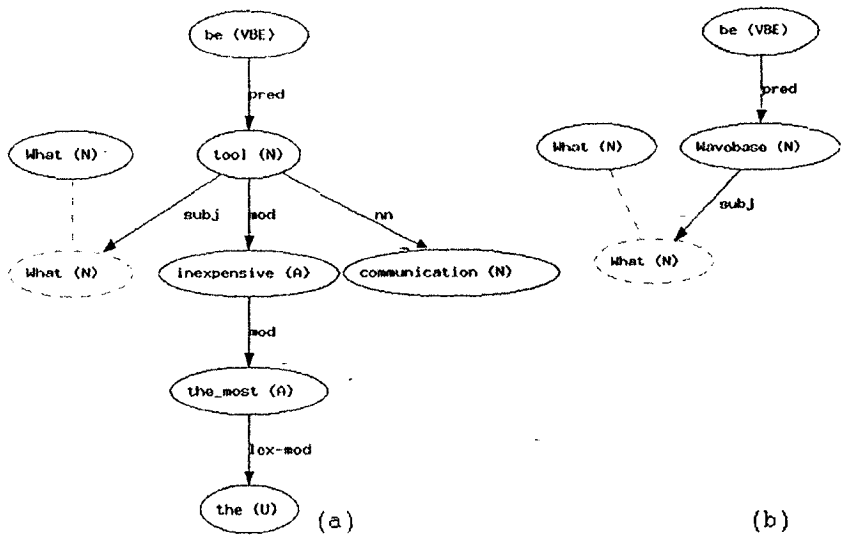

Figure 4.4: Parse tree of the questions (a) *"What is the most inexpensive communication tool?"* and (b) *"What is Wavebase?"*

measures for scoring the relatedness of two verbs. This measure identifies the best if two verbs are synonym in both our closed and open domain training sets (see Chapter 3) among these six semantic similarity measures.

As an example, Figure 4.3 shows the parse tree of a candidate answer for the question *"How does Call Forwarding work?"*. The verb *'redirect'* has a similarity of 1.94 to the verb *'work'*, the question's main verb. This subtree (which happens to be the whole parse tree in this example) includes the question head (*"Call Forwarding"*); so, we can proceed with unification by checking whether their main verbs relate the same entities (subject and object in particular). A fuzzy statistical method evaluates the similarity of the two *subject* subtrees, and likewise for *object* or *modifier* subtrees, if any. We will look at this method in detail in Section 4.2.

4.1.3 Equivalent Copulative Structures

Straight forward argument mapping does not work well with the sentences that have copulative verbs. Copulative questions such as *"What is Wavebase?"*, *"What is the most inexpensive communication tool?"* and *"At what speed is the network compatible with?"* have a particular structure. They convey a state and not an action and therefore, their argument structure is more flexible. For example, although the first two questions are syntactically similar (see Figure 4.4), their answers come in different structures: *'ANS1'* in *"ANS1 are the most inexpensive communication tool."* has

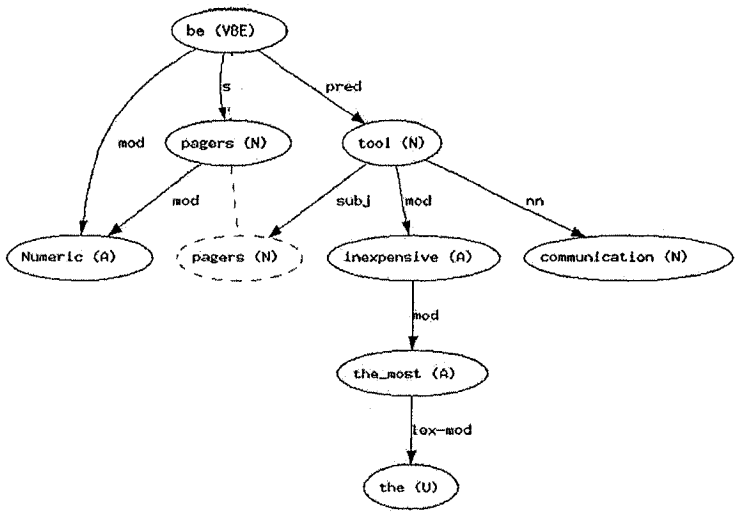

Figure 4.5: Parse tree of the sentence *"Numeric pagers are the most inexpensive communication tool."*

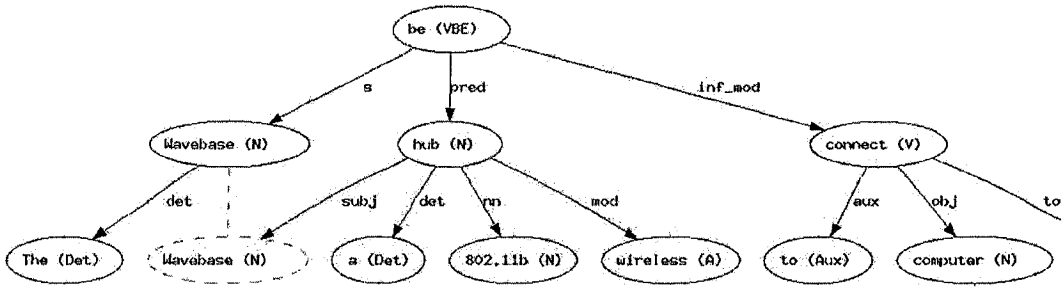

Figure 4.6: Parse tree of the sentence *"The Wavebase is a 802.11b wireless switched hub with 4 ports to connect up to 4 computers to ..."*

a *subjective* role (Figure 4.5), while *'ANS2'* in *"Wavebase is ANS2"* comes in the predicate of a relative clause (*'pred'* subtree in Figure 4.6). This phenomenon led us to toggle *subject* and *predicate* arguments when unifying two copulative structures.

4.1.4 Inter-type Syntactic Mapping

When the question or the answer sentence is copulative while the other one has a non-copulative main verb, our linguistic verb selection does not work: our experiments show that all WordNet::Similarity measures return a high score for similarity of a 'to be' verb to almost any content verb. Additionally, arguments of a copulative verb cannot be blindly mapped on to the arguments of a non-copulative verb: precise syntactic mapping for each specific verb should be applied. In this situation, multiple mapping cases can happen; for example, the answer to a copulative question might appear as a noun modifier or a propositional attachment in a non-copulative sentence. For example, the

copulative question *"What is the Bell Symptatico High-Speed Internet Service?"* is answered by the non-copulative sentence *"The Sympatico High-Speed Internet service gets you online with high-speed access to the Internet at an affordable price."*. The answer to the non-copulative question *Q109.2-"How many countries does it operate in?"* about "Telefonica of Spain" is answered by the propositional attachment in the copulative sentence *APW20000317.0087-"Telefonica is the largest supplier of telecommunications services in the Spanish and Portuguese speaking world with operations in 17 countries and over 62 million customers."*.

Modeling of all possible mapping cases manually is difficult and will not cover many cases with our question answer sets. This should be done automatically and with a larger data set in order to significantly improve the results. For this reason, we leave this task as a future work (see Section 8.3).

4.2 Unifying Two Subtrees

After a candidate subtree is chosen based on the semantic similarity of its head verb, we proceed with unification by checking whether the target verb relates the same entities as the question. Our heuristic syntactic method evaluates how similar the two *subject* subtrees are, as well as the two *object* and *modifier*, subtrees if any. These similarity scores in addition to the IR score of the document that the candidate is selected from, are summed to produce the final score of a candidate:

$W_1 \times \Sigma_{i:Counterpart\ Subtree} UnificationScore(Question_i, Candidate_i) +$

$W_2 \times WN :: Similarity(Verb_q, Verb_T) +$

$W_3 \times Score_{IR}(Candidate)$

where, (W_1, W_2, W_3) are weights and they are set to (1, 1, $\frac{1}{3}$) in our best setting. They determine sentence's syntactic and semantic similarity and its document relevance contributions to the final score, respectively. Experimenting with different weights can show the importance of each feature. We postulate that the semantic feature should have more importance; however, in order to receive a good similarity measure, verb senses must be specified; this is very difficult to achieve with current state of the art word sense disambiguation algorithms [LDK04]. Currently, we are satisfied with this model, since it does not provide unreasonable values for the syntactic nor semantic contributors. To guarantee minimum similarities for the candidates that get a score, we experimentally fixed minimum thresholds for the semantic and syntactic features, $T_{semantic} = 1.8$ and $T_{syntactic} = 0.8$

for the unification sum of subtrees[2].

To compute unifiability score of two counterpart subtrees (*subject, object* or *modifier* marked by the linguistic selection method as verb argument) we apply a heuristic process. This step uses two criteria: the number of overlapping words based on a bag-of-words approach and the number of overlapping links weighted by their importance:

$$UnificationScore(Question_i, Candidate_i) = \beta \times WordOverlap + (1 - \beta) \times LinkOverlap$$

Weighted Bag-of-words Overlap

Common words are a strong hint for equivalent phrases. For example, the noun phrase *"the American Legion"* in the sample question shown in Figure 4.1, should be unified with *"the spokesman for the American Legion"* in the answer sentence depicted in Figure 4.2. As we justified in Section 4.1.1, when weighting the question keywords, we give more importance to proper nouns, nouns and verbs, and consider a lower score for adjectives and adverbs. In this case for example, a score of 6.25 is returned by matching the words ('the'=0.25, 'American'=3.0 and 'Legion'=3.0).

In addition to computing the lexical similarity of two phrases, we measure the syntactic similarity of two phrases to boost their unification score if the matching words have similar relations to one another.

Expecting strict syntactic similarity results in low recall when facing meanings that are put into words using a different syntax: *"president of Russia, Jeltsin"* and *"the Russian president, Jeltsin"* for example are two equivalent phrases, with different syntactic relations between perfectly matching words. Therefore, as long as there is word overlap, we unify the two subtrees. Note that syntactic relations inside arguments are not critical like the subject, object and modifier syntactic relations, and will boost the score.

Syntactic Links Overlap

By analyzing a few unification cases, we realized that matching different types of links should have a variable contribution to the final unification score. Compare a modifier (*'mod'*) link matching in the candidate *"wireless network"* as opposed to a determiner (*'det'*) link in the candidate *"a*

[2]Each subtree can be seen as a paraphrase, since its focus is an entity (noun) with possibly some modifiers.

Category	Minipar Relation	Weight
Lexical modifier	lex-mod	1.0
Adjective/Nominal modifier	mod, pnmod, pcomp-n, nn	0.5
(pre)Determiner	(pre)det	0.25
Possessives	gen	0.25

Table 4.2: Weights of different syntactic links used in scoring the similarity of two phrases.

network" matching with the phrase *"... a wireless network ..."* in the question. The first case shows a stronger similarity since it narrows down the meaning of the noun (*'network'*). To account for this, we weight links differently: i.e., a lexical modifier link (shown as "lex-mod" in Minipar) has the highest weight of 1.0 because it connects two proper nouns, while a determiner has the lowest score. Table 4.2 shows the classes of equivalent links we selected (by intuition) and the values we obtained experimentally for each class. These values can also be learned given a tagged set of equivalent, but syntactically different phrases, such as an appropriately selected subset of the *"Equivalent sentence pairs with minor differences in content"* from the Microsoft Research Paraphrase corpus [DBQ05]. In Chapters 6 and Chapter 7, we perform a sensitivity analysis for these valuses and show that our scoring method is not highly sensitive to these values, as long as the relative order of the classes of syntactic relations is preserved.

For the previous example (Figure 4.2), the value of the *LinkOverlap* feature will therefore be $1.0 + 0.25 = 1.25$ (for the lexical modifier link in *"American Legion"* and the determiner link in *"the Legion"*).

Combining Similarity Features

The parameter β shows the relative importance of the two features: $\beta = \frac{1}{2}$ assigns equal importance to either feature, while $\beta = \frac{1}{3}$ (our configuration) considers the link overlap feature to be twice as important as the bag-of-words feature. Note that the *LinkOverlap* feature subsumes *WordOverlap* because, the head and tail words need to match in order to have a link match. The purpose of this feature is to reflect the syntactic similarity of lexically matching words. Also note that the absolute value of the final score is not important since the scores are used only to rank the candidates and pick the best one. This combination results in a total score of $\frac{1}{3} \times 6.25 + (1 - \frac{1}{3}) \times 1.25 = 2.89$ for the matching of these two phrases (subject subtrees) in the previous example.

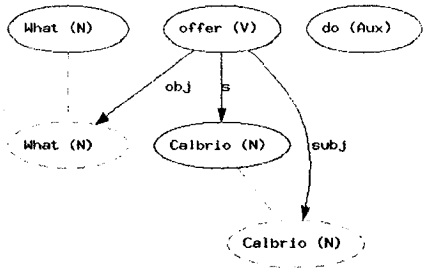

Figure 4.7: Parse tree of the question *"What does Calbrio offer?"*

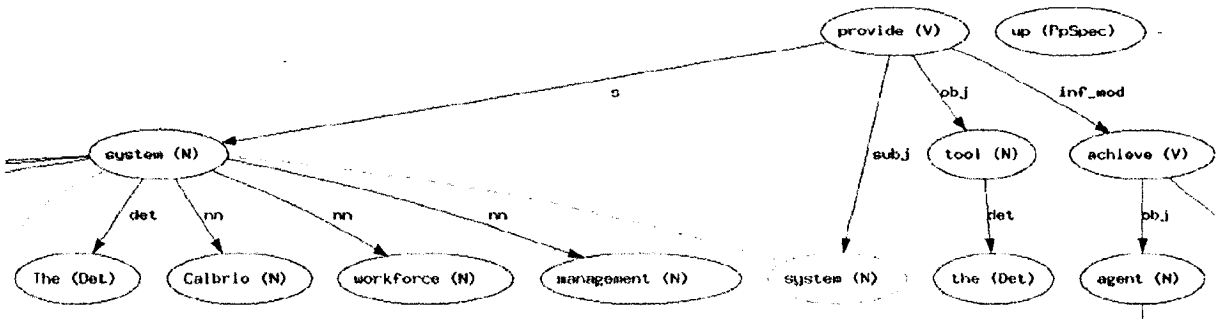

Figure 4.8: Parse tree of the sentence *"The Calbrio workforce management system, made up of fully integrated software modules, provides the tools to achieve optimal agent staffing ..."*

As another example, the word *'Calbrio'* in the question *"What does Calbrio offer?"* (see Figure 4.7) is the only noun phrase in the question. It is found in the left subtree of the verb *'provide'*. Moving up from this anchor, skips the noun node *'system'* and marks the verb node *'provide'* as the seed point for performing the unification. In such a long candidate sentence as this, using an anchor reduces the candidate verbs to the ones that include the question head or a reference to it.

Additionally, the noun *'Calbrio'* appears as *"The Calbrio workforce management system"* in the answer sentence, depicted in Figure 4.8. Here, a score of 3.0 is returned by matching the words 'Calbrio' and 0.0 for syntactic link overlap, since there is no common relation in the two subject subtrees. So, the final matching score for the subject will be $\frac{1}{3} \times 3.0 = 1.0$.

Recall that the reason we relax our linguistic constraints at this stage is that we are focusing on the target subtree that conveys a similar event as the question; only a clue about similarity of counterpart arguments is sufficient to conclude that the target verb modifies the same entities as the question. Syntactic differences of verb arguments (subtrees) should not critically affect our judgment.

4.3 Tuning for Open Domain

In this section we explain two adjustments to our scoring method to make it applicable for factoid open domain questions. Namely, instead of returning a sentence, we identify and return a phrase and we embed Web statistics in our method to increase its performance in open domain.

4.3.1 Answer Phrase Extraction for Factoid Questions

In closed domain, our objective is to return a sentence as the answer. In open domain though, the answer is a word or a phrase. We need identyfy and extract the answer from the best candidate sentence when answering a factoid question. To do this, we extract the noun (phrase) that has the same semantic type as the expected answer type, predicted when analyzing the question (Section 5.3). The answer types that we handle are Date, Person, Organization, and Location. We use the default Gate NE Tagger to label noun phrases in a candidate answer. This plug in has basic rules for tagging nouns and can be replaced with future versions that might be more accurate.

When extracting the answer from a long candidate sentence, more than one phrase might be tagged by the NE Tagger as the target type we are looking for. The benefit of our linguistic method in such cases is that it can locate and choose the noun phrase positioned in the answer subtree marked by the unifier. If because of an imprecise parsing or wrong identification of the answer subtree no noun phrase or more than one noun phrase are found in the answer subtree, the closest noun phrase to the target subtree is returned as the answer; if the target subtree is the same as the candidate's parse tree, we choose the phrase that has a higher rank in the list of Aranea QA's suggested answers (discussed in the following section).

Precision of this answer extraction method highly depends on the accuracy of the NE Tagger. Gate's NE Tagger however, has a very low precision (based on our observations on our development set). In the following Section we present our final answer extraction method using Aranea's suggested answers.

Rank	1st	2nd	3rd	4th	5th	6th	7th	8th
Accuracy (MRR)	34.71	43.53	46.11	51.07	52.72	54.65	56.93	57.76

Table 4.3: Accuracy of Aranea answers at ranks 1 to 8.

4.3.2 Applying Statistics from the Web

Aranea [LK03] is a statistical QA system developed by Jimmy Lin[3] and made publicly available for academic use. It is a Web-based factoid question answering system that uses a combination of data redundancy and database techniques. Its performance in TREC 2002, TREC 2003, and TREC 2004 was very competitive.

Aranea extracts answers from the Web using two different techniques: *knowledge annotation* and *knowledge mining*. Knowledge annotation is an approach to answering large classes of frequently occurring questions by utilizing semi-structured and structured Web sources. Knowledge mining is a statistical approach that leverages massive amounts of Web data to overcome many natural language processing challenges. It basically submits some answer patterns specific to each question to the Web and performs a statistical analysis and voting on the n-grams computed from the generated texts. Aranea's policy is to obviate the need to understand both the structure and meaning of language by using simple pattern matching techniques on the vast Web that redundantly provides the information:

> With enough data, there is a good chance that an answer will appear as a simple reformulation of the question [LK03].

We modified Aranea to handle the 2006 TREC question format. When running the modified Aranea [KBLR06] on TREC 2005 questions, we observed that 52.07% of correct answers are returned in the top 4 positions. When we take the top 20 answers, it results in a maximum precision of 61.57%, while excessively increases the processing time (see Table 4.3).

One simple way of embedding statistics in our linguistic QA system is to feed it with the top answers given by Aranea. At the same time, this allows us to verify the answers given by Aranea against the document collection and re-rank them syntactically, we to our QA system. Specifically, Aranea's answers are initially used to expand the information retrieval query and later on in order

[3]Available at `http://www.umiacs.umd.edu/ jimmylin/downloads/inde.g.html` for download.

to boost unification scores.

To expand the IR, we basically added m documents from the IR result for the extended query

$$AranAnswers\ AND\ QuestionKeywords$$

to the regular document list we have for the *QuestionKeywords* query. Later on, candidates that include Aranea's answers are boosted based on the position of that target answer in the syntactic tree. In the ideal case, an Aranea's answer fills in the role that is asked for in the question by the question word. However, since such a perfect mapping rarely occurs, we implemented a heuristic to compute high relevance of the suggested answer with the question words using the parse tree as opposed to using the linear form of the sentence. This heuristic boosts the score of the candidates that contain one of Aranea's answers as opposed to other candidates that contain none of Aranea's suggested answers.

We assume that the candidate is highly relevant if one of Aranea's top 4 suggested answers appears in the target subtree of the candidate. If this is the case, then the normalized score for that Aranea's answer is added to the candidate's unification score. Although this heuristic does not guarantee correctness of that answer, with the level of detail we understand the meaning of a sentence, such an assumption is reasonable. In other cases when one of Aranea's answers occurs inside the parse tree but outside the target subtree, $\frac{1}{D} \times Score(Cand_{Aranea})$ is used for boosting, where D is the path distance of the Aranea's answer from the root of the target subtree: the effect of this is that it considers the closer positions as more relevant. For example, 17 is suggested by Aranea as the second possible answer to the question *Q125.4-"How many opening season performances did Enrico Caruso have at the Met?"*. The score of the candidate *"Garuso sang his 17 opening nights during just 18 seasons, from his Met debut as the Duke..."* (Figure 4.9) is boosted from 3.0 to 3.7, which puts it in the top of the candidate ranked list.

The final answer is the word (or noun phrase) among Aranea's suggested answers or identified by the Gate NE Tagger that is found in the best candidate sentence (after boosting sentence scores and reranking the candidate sentence set).

As we will see in the evaluations (Chapter 7), this heuristic improved our results from 17.1% to 20.6% for the TREC 2003 to 2006 factoid questions (see Section 7.4.2).

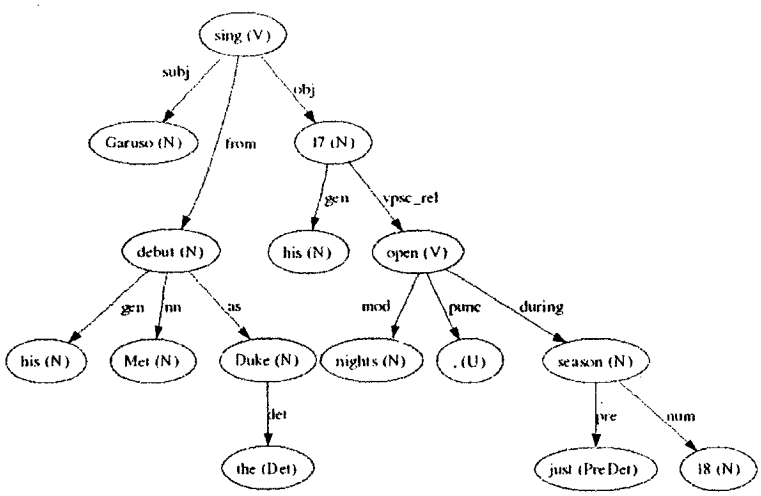

Figure 4.9: Boosting a candidate score using Aranea's suggested answers.

In this chapter, we explained our syntactic ranking method for answering questions with a main content verb. This candidate scoring method uses the question head as an anchor to narrow down the search to a subtree in the parse tree of a candidate sentence (the target subtree). Semantic similarity of the action in the target subtree to the action asked in the question is then measured using WordNet::Similarity on their main verbs. In order to verify the syntactic similarity of this subtree to the question parse tree, syntactic restrictions as well as lexical measures compute the unifiability of critical syntactic participants in them. We showed how to use suggested answers from a statistical QA system to make our method return a word (or phrase) instead of a sentence in open domain. Also we used Web statistics information to increase its performance on open domain TREC factoid questions.

In the following chapter, three processes involved in composing the set of candidate sentences are presented.

Chapter 5

Candidate Sentence Selection

In this chapter, we describe the processes involved in extracting sentences from the corpus that have lexical similarity to the question and are likely to contain the required answer (a.k.a. the candidate answers). Although a variety of clues can be used for limiting the search to a few documents such as title, date, or the location of a document in the directory structure, we use salient words in the question (question keywords) for this purpose. Our question analyzer (the first process in Figure 1.1) recognizes important question keywords by analyzing the parse tree of the question and the parts of speech of its words.

A limited number of documents in the corpus that contain all (or the most important) question keywords are retrieved using an information retrieval engine (the second process in Figure 1.1). These relevant documents are then processed in order to see if those keywords occur in the vicinity of one another, preferably in one sentence (the third process in Figure 1.1), or to decide if their appearance in that document is a coincidence and unlikely to be related to one event. The final result after running these three processes is a set of sentences called *candidate sentences.*

Since most of the algorithms we provide are based on the parse tree of the sentences, we start off by introducing the parser that we used.

5.1 The Minipar Parser

In order to choose an appropriate parser for our work, we considered the following factors:

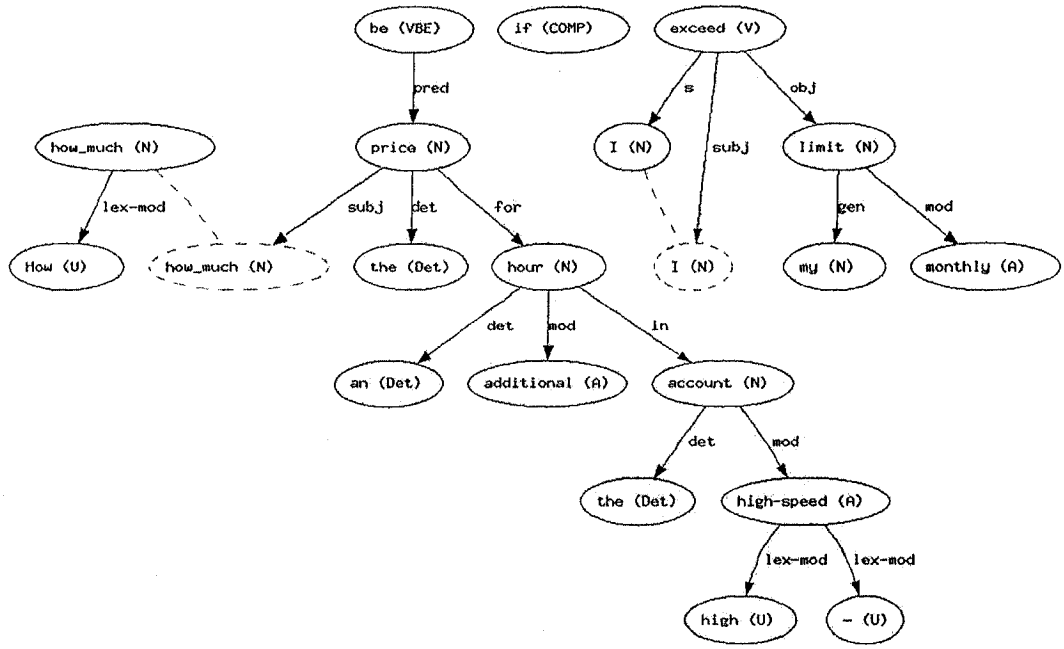

Figure 5.1: Parse tree produced by Minipar for the question *"How much is the price for an additional hour in the high-speed account if I exceed my monthly limit?"*

- Parsing accuracy,
- Parser's ability to provide partial parse structures for syntactically complex and long sentences,
- Parser's ability to track references to previous words that are referred to using wh-pronouns such as 'which', 'that', 'who', 'whom', etc.

We examined three popular parsers: Link parser [ST91], Collins parser [Col97] and Minipar [Lin93] parser. Link parser's output is complex and contains much more information than what we need for our purpose. It differenciates between eight categories of comparative structures, or much details about *transitive*, *intransitive* and *optionally transitive* verbs. Therfore, the output of this parser should be processed to interpret syntactic roles (ex. subject, object, adjunct). Furthermore, this parser can only consider links of a certain length and fails in parsing very long sentences. It also rejects sentences with spelling errors, such as *"I saw bob"* because the word 'bob' is recognized to be a verb according to the parser's dictionary. Moreover, it cannot handle conversational text style, which we encounter for a few Bell Canada customer service questions.

Collins parser's is a statistical parser trained on the Penn Wallstreet Journal Treebank. This parser's output also requires to be further processed to label syntactic roles.

Minipar is a shallow, broad coverage dependency parser that has achieved about 88% precision and 80% recall with respect to dependency relationships on the SUSANNE corpus [Sam95]. Comparative experiments with Link, Collins and Minipar parsers on a subset of our training set showed that in addition to directly providing the dependency relations, Minipar is more robust when parsing long sentences (that is sometimes actually more than one sentence that is incorrectly tagged to be a sentence). Finally, availability of a Minipar wrapper in Gate[1], our implementation framework, was an additional incentive to choose it.

In its shipped version in Gate, Minipar takes as input, a document with tokens and sentences marked by the Gate Tokenizer and Sentence Splitter processes beforehand. It then determines the dependency relations between the words in each sentence in that document. Finally, the type of the dependency relationship between the words that are referred to as head and child is produced. Unlike some other parsers, Minipar only produces one parse tree for a sentence. We improved the Minipar wrapper by processing the *virtual* nodes that are generated by the parser engine, but ignored in the wrapper. A virtual node provides a reference to an actual node in the parse tree for wh-pronouns such as 'that', 'which', etc. This node plays the role of the referred node and makes the parse tree more understandable and easy to analyze.

Figure 5.1 shows the parse tree produced by Minipar for the sentence *"How much is the price for an additional hour in the high-speed account if I exceed my monthly limit?"*. Figure 5.2 shows output annotations produced by the Minipar wrapper in Gate for this sentence.

Another feature of Minipar we tried to exploit is that it recognizes all the terms that start with a capital letter as nouns. So, before parsing a long sentence in our closed domain, a technical gazetteer can be used to analyze each sentence and highlight all nouns that are found in the technical terms repository or in the domain ontology (together called the gazetteer). This pre-processing step should result in a better parse tree. For example, *'loop'* in the term *'u loop'* is tagged as a verb, while when its first letter is capitalized, it is correctly tagged as a noun modified by *'U'*. Preliminary experiments show that although this idea improves the accuracy of the parse tree for some sentences, it introduces a significant amount of false positives by wrongly capitalizing the verbs that have an equivalent noun in the gazetteer (such as 'plan' and 'guide'). Given a close to perfect accuracy for the part of speech tagger in Gate, we will be able to improve the parse accuracy of long closed

[1]General Architecture for Text Engineering, available at `http://www.gate.ac.uk/`

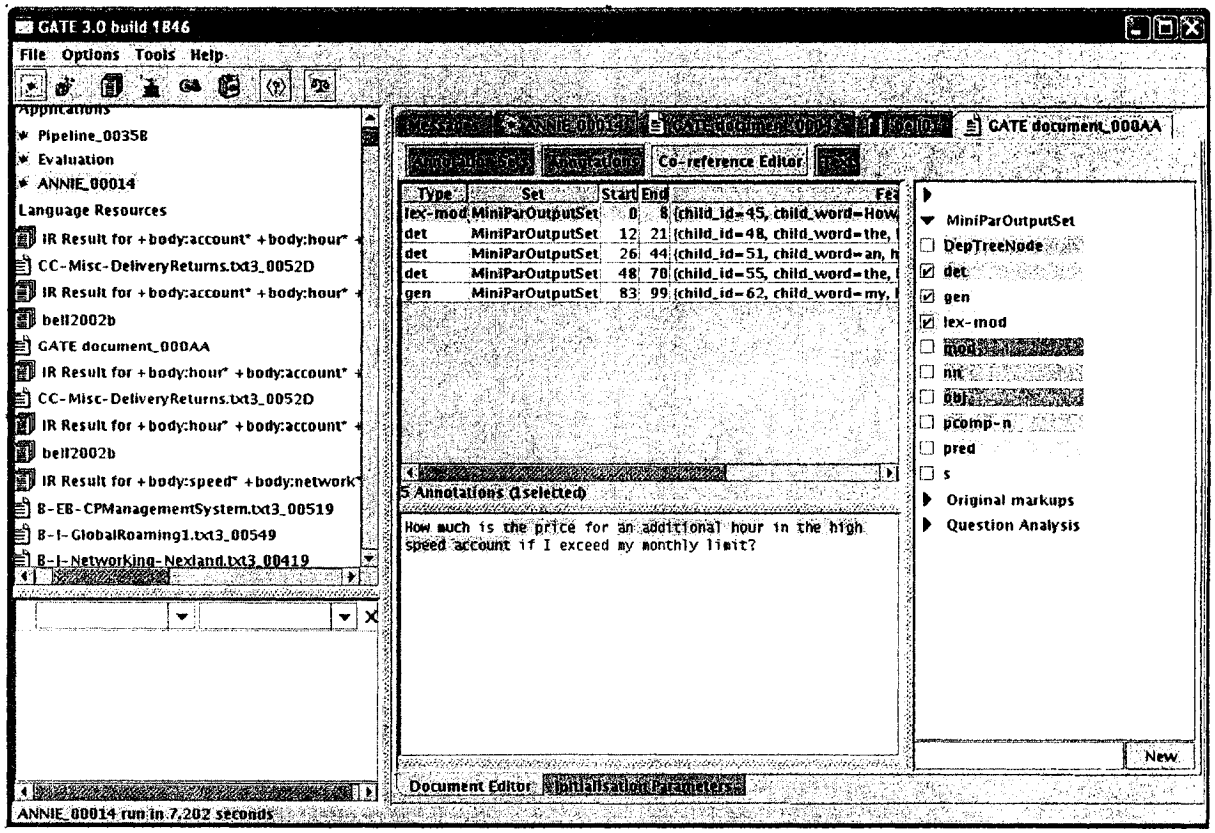

Figure 5.2: Annotations produced by Minipar in Gate for the question *"How much is the price for an additional hour in the high-speed account if I exceed my monthly limit?"*

domain sentences. Currently, we parse our questions without performing any lexical modifications.

Since our method does not depend on a precise parse tree, we use the partial parse trees that are produced by Minipar for long sentences. In the following, we explain how the parse tree of a question is used to mark important keywords.

5.2 Information Retrieval

Some QA systems use a general IR system to return complete documents or relevant passages only (e.g. QUANTUM [PLK02], MULDER [KEW01] described in Chapter 2), while others use their own specialized IR methods for QA purpose (e.g. IBM [IFR01]). In this section we describe and justify our choice of two information retrieval engines to select the documents relevant to the question (the second process in Figure 1.1): Lucene and BioKI. Lucene is based on the vector space model. It is an state of the art information retrieval engine, written in Java (with appropriate wrappers in Gate environment) and BioKI [LDKB07] exploits low frequency words and their distance in the text to catch relevant text chunks that are not recognized using the frequency statistics.

5.2.1 The Lucene IR Engine

For information retrieval in both open and closed domain, we use the Lucene IR engine[2]. It supports the use of wild cards in the query. For example, since the word *'charity'* in the question might not exist in the exact same lexical form in the corpus, we append the '*' wildcard to its stem, *'charit'* before sending it to the engine.

The IR engine might not find any document that contains all the query keywords: lexically different words which are synonyms can express one meaning. In order to increase recall in such a situation, less important keywords should be discarded in order to relax the query and find relevant documents. Having a ranked list of keywords as the query, we designed a feedback-loop to relax the query until a minimum number of documents are retrieved. For this purpose, the IR engine is run iteratively, each time removing the least important keyword if no relevant document is found in the corpus (the minimum is specified by a threshold variable, two documents in our best configuration). For our example question *Q98.4- "What organization has helped to revitalize Legion membership?"*,

[2]Apache Software Foundation, Lucene 1.4.3 API `http://lucene.apache.org/java/docs/api/`

the last keyword, *'organiz*'*, is removed from the query *"Legion" and "membership*" and "revit*" and "help*" and "organiz*"* before we get one document returned by the Lucene IR engine. This relaxation will be later compensated by our strict candidate ranking module (Chapter 4) that filters out irrelevant candidates.

Evaluation results (Chapters 6 and 7) show that F-measure in extracting the correct document is low and has space for improvement. Using another IR method might increase recall, specially for more than one third of the questions in our closed domain. We tried another IR technique in order to possibly find a better method for document retrieval.

5.2.2 The BioKI IR Engine

BioKI is an information retrieval system that was designed to find low-frequency information in a document collection. It was originally developed as a literature search tool, for interactive queries through a web interface, but has been successfully adapted for use in different scenarios, such as the TREC Genomics track [HCRR06]. Applying BioKI on a large corpus such as AQUAINT is costly and so not feasible in the time limits of our work in open domain. Therefore, we use it as an alternative to Lucene only in closed domain.

BioKI works by splitting documents into segments, scoring each segment and re-ranking the segments based on their score. This allows the system to retrieve relatively short bits of text that are very focused on the query words. Originally, the segmentation was done using the TextTiler algorithm [Hea93], which splits the text into groups of consecutive paragraphs that exhibit lexical coherency. While this approach was successful for literature search, other IR tasks, such as TREC, benefited from a simpler strategy, namely splitting on paragraph boundaries [LDKB07].

BioKI has many modes for scoring paragraphs; the one that was used in our experiment is the *lenient* scoring. It tries to find all of the paragraphs containing all the search terms. These paragraphs are then ordered, so that paragraphs with all the search terms co-occurred within a short span are ranked higher than those with the keywords spread across a longer span. After that, it ranks the paragraphs that have some, but not all, of the search terms. Those that have a higher number of keywords present rank higher than those with fewer keywords. Again, all the paragraphs with a given number of keywords present are ordered by the length of the span of text containing

the keywords, so that a shorter span will rank higher than a longer span.

We performed an experiment with the BioKI information retrieval engine [BSDL06] in closed domain to see if changing the IR method would improve our document retrieval accuracy. Using terms with low frequency in closed domain (with low redundancy available) seems more appropriate for improving recall at document level.

5.3 Question Analyzer

Question analyzer extracts the *expected answer type* and a ranked list of *question keywords* to be fed to Lucene or BioKI IR engines (the second process in Figure 1.1). *Question keywords* will be used for retrieving the documents and passages relevant to the question, based on the assumption that relevant passages contain words in common with the question. The *expected answer type* is used as a constraint for extracting the answer word (or phrase). For example, the answer type for a question starting with 'Who' is *PERSON*; after tagging a candidate answer, the word that is tagged as *PERSON* is the answer. We use the existing work done in the Aranea QA system [LK03] to extract the expected answer type for a question.

Important words in the question are marked as question keywords. As we justified in Section 4.1.1 two factors contribute in deciding if a word is important: the part-of-speech of that word, and the number of modifiers it has. To identify question keywords to be used for information retrieval we process *important* parse links provided by the Minipar parser [Lin93]: depending on on the type of the parse link seen, both 'head' and 'tail' or only the 'tail' word is kept:

- For a nominal complement of a preposition such as *"for convenience"* and determiner relation such as *"the network"* (shown as *'pcomp-n'*, *'det'* respectively, in the Minipar notations), only the tail (*'convenience'* and *'network'*) is marked as a question keyword.

- For an adjunct modifier link, a lexical modifier such as *"electric guitar"*, a conjunction, a subject or object links, a noun complement and a passive verb modifier of a noun such as *"the service provided..."* (shown as *'mod'*, *'lex-mod'*, *'conj'*, *'subj'*, *'obj'*, *'nn'*, and *'vrel'* in Minipar), both head and tail words are considered.

For example, the following ranked list of keywords is computed for the question *Q98.4-"What organization has helped to revitalize Legion membership?"*:

```
1-Legion
2-membership*
3-revit*
4-help*
5-organiz*
```

Finally, the content words that are not involved in any syntactic relation of our interest, are added to the query, with a value of 0 as their number of modifiers (hence at the bottom of the list). We use them as low-ranked keywords that can constrain the number of documents returned in the IR phase[3].

5.4 Candidate Sentence Extraction

As we saw so far, the Lucene IR engine returns a ranked list of documents. Now, we need to extract the most relevant sentences from these documents (see the third process in Figure 1.1). To do so, we first considered a *keyword-weighted* sentence selection method. It selects all sentences that contain valuable question keywords, given that the score of each question keyword is computed by the question analyzer beforehand. We sum up the scores of all the question keywords that appear in a sentence and select that sentence if the result is above a predefined threshold[4].

As a second approach, we used the simple *boolean vector* selection method. It selects sentences that contain α percent of the question keywords. Comparing the precission and recall of the two methods, the boolean vector selection is more precise and selects correct candidates for significantly more questions in our development set; so, we performed all our experiments with this sentence selection method.

Experiments with the threshold for the *boolean vector* sentence selection show that varying the similarity threshold (from 35% to 75%) only affects the candidate ranking time (last process in Figure 1.1): on average, the number of candidates retrieved decreases from 55 (with a standard deviation

[3]It is interesting to note that PiQASso [ACF+01] ranks question keywords based on their depth in the parse tree, which is akin to our method of using the number of modifiers in the parse tree.

[4]The documents in the retrieved list are tokenized and their sentence boundaries are marked beforehand using pre-existing language tools in the Gate framework.

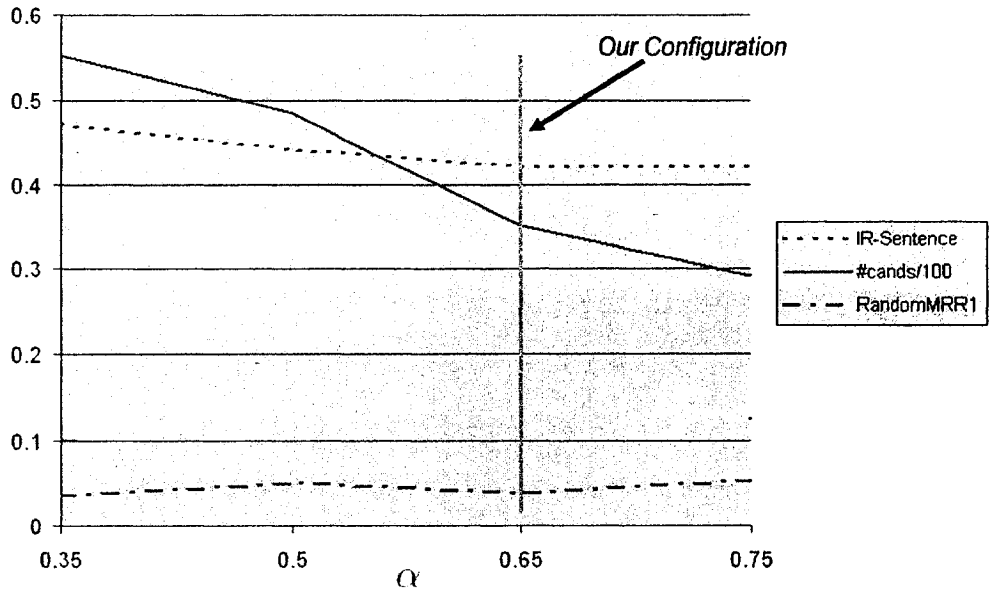

Figure 5.3: Effect of the parameter α on recall at the sentence level for our closed domain development question set.

of 23) to 25 (std.dev. 16) when increasing the threshold from 35% to 75% for our development set in open domain[5].

Figure 5.3 shows how applying a more strict sentence selection reduces the number of candidates selected ('#cands/100' line) for our development set. We fixed the sentence selection threshold (α) to be 0.65 in order to compromize between recall and precision. Although we will see that the ranking method is not highly sensitive to noise, high noise ratio in general makes the ranking process difficult and decreases the final results. In this diagram, *IR-Sentence* (recall at sentence level) shows the fraction of questions for which a correct candidate sentence exists in their candidate sentence set. We have a relatively low recall at this level because no advanced IR mechanism, such as query expansion, was implemented in this work. We will analyze the effect of the error in each of these processes on our test sets in Chapters 6 and 7.

In open domain TREC questions, to improve the IR performance, the list of returned documents is filtered using the PRISE top document list; the later list is compiled by the NIST organization, running the PRISE IR engine on question keywords plus the answer, so it uses more information in choosing the documents. The candidate answer selection process receives the top n documents that

[5]On an Athlon AMD 3500+ 64bit CPU with 4GB of RAM, the time needed to rank these candidates increases from 110 seconds to 200 seconds per question.

are returned by Lucene and filtered against the PRISE document list. In this way we increase the precision of the returned document list.

In this chapter we described three preliminary processes involved in selecting candidate sentences to be ranked by the syntactic unifier. These processes are the question analyzer, information retrieval, and finally, candidate sentence selector. In the following chapter, we will evaluate our syntactic ranking method in closed domain.

Chapter 6

Evaluation in Closed Domain

In this chapter, we compute the accuracy of our QA system in extracting the correct answer sentence from a closed set of documents about telecommunications. In Chapter 7, we will evaluate our method in open domain and compare its accuracy with other open domain QA systems for the TREC QA sets.

The best evaluation of our closed domain candidate ranking method is to run it over a standard corpus such that its accuracy can be compared to other systems run on the same question/answer collection. In this way, parameters such as text genre, the level of difficulty of questions and possible bias of a method towards the corpus will have no effect on the evaluation. However, since such a standard corpus exists only in open domain, in closed domain we are only able to compare our method to a baseline.

To evaluate the performance of our unification method, we will first focus on the error introduced by individual processes that extract candidates beforehand. The accuracy of the input to the unifier will provide an upper bound on its accuracy. We will first evaluate the performance of our information retrieval module at the document and then at the sentence level. We will see that, although we tried two different IR method, retrieving the correct document is the major source of error. Finally, we report the contribution of syntactic, semantic and statistical features we used in our scoring method separately on our development set.

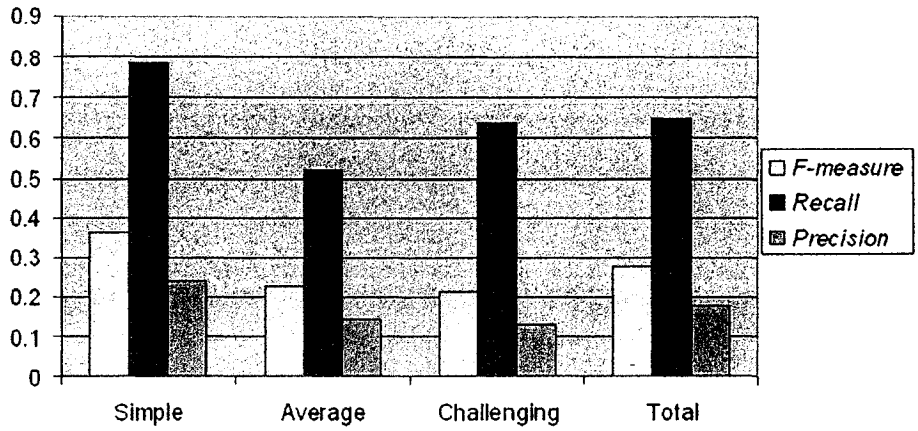

Figure 6.1: Precision and recall at the *document* level for the Bell test question set.

6.1 Information Retrieval

As we described in Chapter 5, query keywords are extracted and ranked using the parse tree of the question. We will first evaluate the performance of information retrieval at the document level. For our closed domain training set, usually less than 10 documents are retrieved for a question and in few cases that we get more than 10 documents, they do not contain the correct document. Therefore, we set up our experiments based on the 10 top documents.

Figure 6.1 shows the precision and recall measures for evaluating the Lucene IR engine at the document level on the Bell test question set. This diagram shows an overall F-measure of 29.2% at level 10 for our closed domain test set.

From this figure, we conclude that retrieving the correct document is easier for 'simple' questions (F-measure=36.9%) compared to the 'average' (F-measure=25.6%) and 'difficult' categories (F-measure=21.4%). The reason can be lack of query expansion; in harder questions, less question keywords appear in the answer sentence; instead, synonyms are used, which are lexically different from the original keywords. However, there are still debates how query expansion should be done in order to improve IR [MSB98].

Since we perform an iterative IR to increase recall when there are no documents returned by IR, the contribution of our keyword ranking method was computed by performing an experiment while the keyword removal feature was disabled. In this configuration, recall at document level drops to 47.7% compared to 64.9%. Therefore, we believe that the factors we considered for ranking the keywords are appropriate and important.

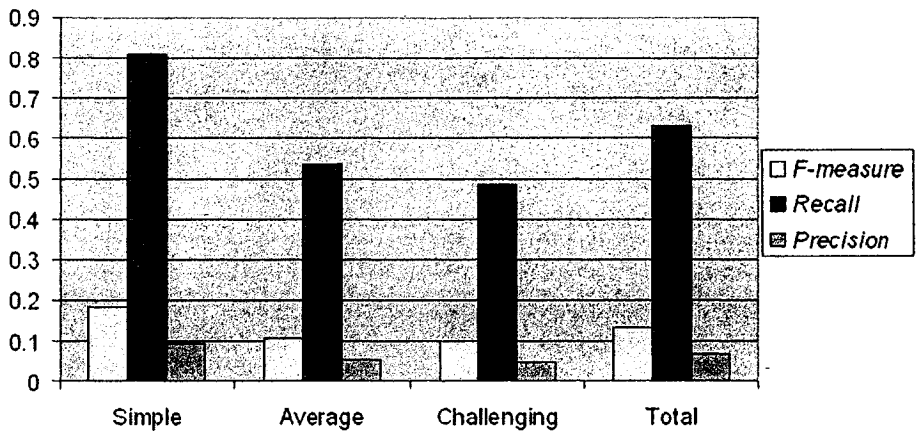

Figure 6.2: Precision and recall using the BioKI IR engine at the *document* level for the Bell test question set.

To see if changing the IR engine increases our retrieval accuracy, we performed an experiment with the BioKI information retrieval engine [BSDL06] in closed domain. By applying a low frequency information retrieval method, we try to increase recall, and possibly fill in the accuracy gap between our IR method and the state of the art IR for question answering: in open domain, LCC's CHAUCER QA system [HHW+06] for example has reached a recall of 86.1% at TREC 2006 (see Chapter 2).

For this experiment, we provided BioKI with the query and obtained a ranked list of paragraphs from the telecommunications document collection. Multiple paragraphs from the same document might occur in the list. Sentences in the paragraphs are analyzed by the candidate answer extractor using the same method and selection threshold.

The results of this new IR engine show that precision at the document level drops significantly from 17.8% to 7.5% (Figure 6.2 compared to Figure 6.1). Hence, a less precise document list is returned for a question compared to when Lucene is used. We observed that more than one paragraph is returned from a document; this sometimes makes multiple instances of a wrong document to be placed in the top 10 position, which makes the competition for the correct document(s) difficult. In the following section, we analyze whether this lower F-measure at the document level results in a lower F-measure at the sentence level as well.

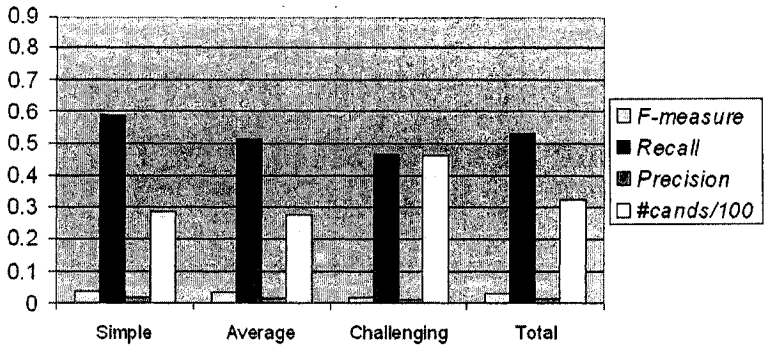

Figure 6.3: Precision and recall at the *sentence* level for the Bell test question set.

6.2 Candidate Sentence Extraction

Given the list of documents returned by the IR engine, extracting candidate sentences from them is an important step that we evaluate separately. Note that sentences extracted at this level are the ones sent to the unifier for ranking.

As we described in Section 5.4, we use a boolean model to choose sentences that contain $\alpha = 0.65$ of question keywords. A low value for this parameter results in a loose restriction that selects many sentences as candidates, while a high threshold marks only the sentences with very high lexical similarity to the question (see Figure 5.3). We accept a slight error in precision by choosing 0.65 for this parameter in favor of passing more candidates to the unifier and higher recall. On average, around 32 candidates are selected for ranking, more for challenging questions and less for easy and average difficulty questions. We will show later that cutting down the size of the set to half by removing irrelevant sentences does not improve much the ranking results. In other words, our syntactic ranking method is not very sensitive to the size of the set, but apparently, it faster analyzes a smaller set. On the other hand, whether a set includes the correct answer or not is important (recall) and puts a high bound on the final ranking result.

Figure 6.3 shows the accuracy of candidate extraction from the documents returned by Lucene. Recall shows the percentage of questions for which at least one correct candidate sentence is retrieved. Precision shows the ratio of correct candidates in the set. The '#cand/100' bar shows the number of candidates that are selected for each question category in closed domain (divided by 100). For simple questions, on average 28.6 candidates are selected, while when we move to challenging questions, deciding if a sentence is relevant or not is not that straight forward: we need to go through more

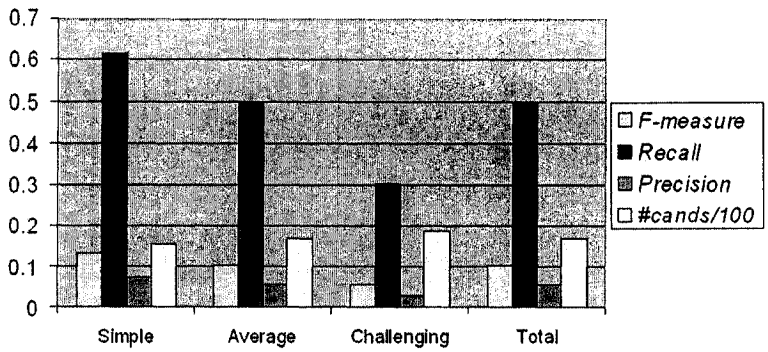

Figure 6.4: Precision and recall using the BioKI IR engine at the *sentence* level for the Bell test question set.

iterations and drop more keywords before retrieving a minimum number of relevant documents and candidate sentences. Consequently, this relaxation results in more candidates for harder questions (46 for challenging questions).

Although precision at document level is low, BioKI filters some irrelevant paragraphs from the relevant documents. Lower noise should help in finding the answer among a smaller set of candidates.

Processing the paragraphs returned by BioKI results in a more precise set of candidates (5.7% shown in Figure 6.4 vs. 1.7% in Figure 6.3). Recall, however, does not drop significantly. Besides, the size of the candidate set is half after filtering irrelevant paragraphs by BioKI. Under these circumstances, we expect an increase in the ranking accuracy (which can be seen in Figure 6.5).

6.3 The Syntactic Unifier for Candidate Ranking

We rank candidate sentences based on their syntactic and semantic similarity to the question. Before evaluating the performance of our ranking method, we looked at the baseline for this task. The baseline for a configuration (α) is the accuracy resulted by randomly selecting one candidate from the set of candidate sentences as the answer[1]. As mentioned in the introduction, accuracy of a question answering system is reported as the Mean Reciprocal Ranking (MRR) score which is equal to the inverse of the position of the first correct answer in the list [VT99]:

$$Score(answer\ list) = \frac{1}{Rank(first\ correct\ answer)}$$

[1]Taking any type of information about candidates, such as their document score, or their lexical similarity to the question might be a more informed baseline. However, using such a type of information can be considered as a preliminary QA system.

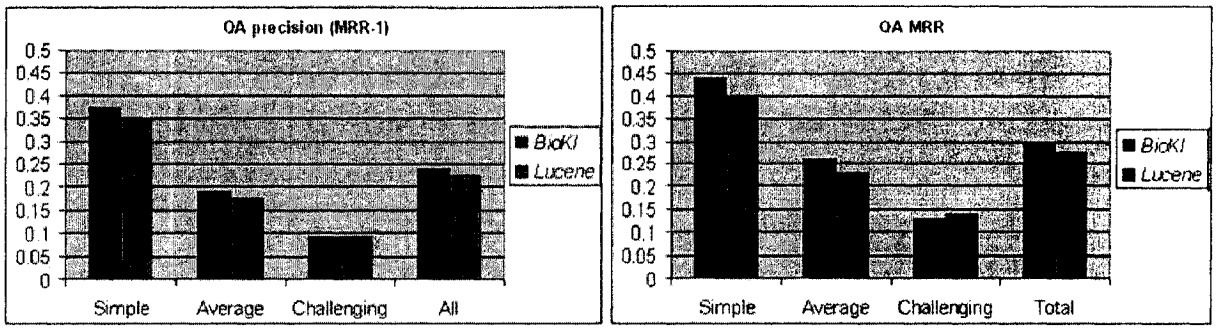

Figure 6.5: Final precission (left) and accuracy (right) of the QA system running Lucene or BioKI on the Bell test set.

Having x candidates in the list, MRR is defined as the average score over all possible positions:

$$MRR = \frac{1}{x}\Sigma_{i=1}^{x}\frac{1}{i}$$

In our closed domain, the size of the candidate list is around 32; so, if we select a candidate at random, MRR should be 12.7%. However, experimentally, the baseline is $MRR = 9.2\%$ because of high standard deviation in the size of candidate sets.

Figure 6.5 shows a maximum overall MRR of 29.4% for our QA system on the Bell closed domain test set, when running the BioKI IR engine, and 27.3% when running Lucene. Finally, we achieve an MRR of 43.7% for simple questions running BioKI and 39.3% with Lucene. This shows successful application of our ranking method on questions with simple syntactic structures, without any major difference with their answer sentence.

Figure 6.5 (left) shows the final precision (MRR-1) of 24.0% when running BioKI; to compute MRR-1, only the correctness of the answer at rank one is considered. Theoretically, the correct answer would occur on average in the middle of the list which results in the MRR-1=$\frac{1}{x/2}$. The average size of our candidate sets is around 32, that results in a theoretical precision of 6.25%. However, experiments show a random MRR-1 of 5%.

As we expected, feeding candidate sentences extracted by the BioKI IR engine produces a higher precision (Figure 6.5 (left)). Excluding the error of candidate sentence extraction leaves an accuracy of $\frac{24.0}{50.0} = 48.0\%$ for the unifier given BioKI candidates, and $\frac{22.4}{53.1} = 42.2\%$ for Lucene. Denominators are recalls at sentence level extracted from BioKI in Figure 6.4 and Lucene in Figure 6.3, respectively. Our unifier is favored in closed domain however, because the students who composed the questions, saw the documents beforehand. This may have unintentionally led them to choose a syntactic

Feature	Weight	Contribution	MRR
Lucene IR Score	$W_3 = \frac{1}{3}$	2.5%	30.5%
Semantic WN::Similarity Score	$W_2 = 1$	5%	28%
Syntactic Unification Score	$W_1 = 1$	23%	23%

Table 6.1: Approximate contribution of each feature in the final ranking accuracy (MRR).

structure for their questions that is close to the structure of the answer sentence.

6.4 Sensitivity Analysis of Weights in Closed Domain

In this section, we analyze the usefulness of our scoring features and the sensitivity of the optimum weights we experimentally assigned for answering our closed domain questions.

Recall that we used the following function to score candidate sentences (Section 4.2):

$W_1 \times \Sigma_{i:Counterpart\ Subtree} UnificationScore(Question_i, Candidate_i) +$

$W_2 \times WN :: Similarity(Verb_q, Verb_T) +$

$W_3 \times Score_{IR}(Candidate)$

Table 6.1 shows the MRR achieved on the development set by adding each feature. As the table shows, using only the syntactic similarity measure achieves an MRR of 24%. Adding semantic similarity information to choose the target subtree, the MRR increases by approximately 5%. In closed domain, similar nouns appear with multiple verbs (ex. "...*provided* in POP3", "...*introduce* POP3", "customers who *use* POP3", etc.) and even an average performing similarity measure can reject a considerable number of irrelevant sentences. Additionally, the fact that the variation of verbs is limited to one topic helps in getting a more reliable semantic similarity measure from WordNet::Similarity. Finally, adding the document score feature slightly increases the MRR by 2.5%. We expected more contribution from this feature; however, query words are not as distinctive as in open domain, since they appear in many documents and usually more than once (especially frequent nouns and proper nouns in our corpus, such as 'Bell', 'Sympatico', 'wireless', 'cell', 'phone', etc.). Therefore, keyword statistics are not very helpful.

Recall that the syntactic score is itself composed of two features:

$$UnificationScore(Question_i, Candidate_i) = \beta \times WordOverlap + (1 - \beta) \times LinkOverlap$$

In order to analyze the sensitivity of the syntactic features, we changed the *part of speech* and *parse link* weights by 50% from the values we used in Tables 4.1 and 4.2, respectively. As long as varying these values preserves the order of categories, the MRR decreases by only 4%. However, changing the order that we linguistically justified, will drastically lower the final results.

As we saw in Figure 5.3, we experimented with several values of sentence-selection threshold (α) from 0.35 to 0.75 in closed domain. The MRR varied within a range of 4%, with its minimum at $\alpha = 0.35$, and close to maximum at $\alpha = 0.75$. The optimal value for this parameter is 0.65, a compromise between the number of candidates selected and sentence level recall.

6.5 Conclusion

In this chapter, we presented the contribution of our ranking method in closed domain. We showed that the factors we considered for ranking the question keywords are appropriate and our syntactic keyword ranking method has a significant effect (23%) in retrieving more relevant document; by performing an iterative IR and removing low ranked keywords when necessary, recall at document level increases from 47.7% to 64.9% when Lucene is used.

Candidate answer selection and candidate ranking are the two phases in finding the best sentence in a document collection for a question. Candidate answer extraction is the major source of error in the final results. It introduces an error of around 50%. These are not two independent phases though: improving the recall of the candidate answer selection part results in selecting more candidate sentences, hence results in a more difficult ranking task for the unifier. However, we expect the overall accuracy of question answering to increase, because of our closed domain observations with the two IR engines: doubling the number of candidates when switching from BioKI (Figure 6.4) to Lucene (Figure 6.3) very slightly decreased our final QA accuracy (from 24.0% to 22.4% shown in Figure 6.5). Hence, if a better candidate answer extractor marks more candidates, it will not drastically decrease the unifier's accuracy in finding the best candidate among them.

Having a small corpus / question set can result in building a biased method that works only with this question set. In the following chapter, we evaluate our unifier's accuracy in the TREC open domain question sets in order to see how general our ranking method is and whether it can answer standard factoid questions.

Chapter 7

Evaluation in Open Domain

Successful application of our ranking method in closed domain encouraged us to analyze its performance for open domain question answering as well. In this way, we compare our performance with other state of the art QA methods on a standard question set, and also show that this syntactic ranking method is not domain dependent and can be applied for factoid questions about other topics than telecommunications. In this chapter, the accuracy of our candidate ranking is evaluated on TREC 2003 to 2006 factoid open domain questions. We start by providing a short history of the TREC factoid QA evaluation and show that features we used for candidate ranking suits open domain as well.

7.1 TREC Evaluation of Factoid Questions

The purpose of a question answering system is to return a final answer word or phrase, rather than a ranked list of documents, in response to a question. Since 1999, NIST has been organizing a question answering track to encourage and evaluate work in QA. They provide a corpus and a set of questions and a word or phrase represented by a regular expression as the answer key for each question. Human judges hired by NIST compose answer keys. In the QA track, the task definition requires systems to retrieve small snippets of text that contained an answer for open-domain, closed-class questions (i.e., fact-based, short-answer questions that can be drawn from any domain). At TREC-8 and TREC-9, strings of 250 characters and 5 strings of 50 characters were required at TREC-10; however, in the

following years, an exact answer phrase or word was required for each question.

A response is composed of a single string and a document id (or the string "NIL" if a system decides that no answer for the question exists in the corpus). The "NIL" string will be judged correct if there is no answer known to exist in the document collection; otherwise it will be judged as incorrect. If an [answer-string, docid] pair is given as a response, the answer-string must contain nothing other than the answer (since TREC-10), and the docid must be the id of a document in the collection that supports the answer string as an answer. For question answering, the unit judged is the entire string that is returned by the system. Different QA runs very seldom return exactly the same answer strings, and it is very difficult to determine automatically whether the difference between a new answer string and a judged string is significant with respect to the correctness of the answer.

In the early years of TREC QA (1999 to 2002), the QA task runs were evaluated using mean reciprocal rank (MRR): the score for an individual question was computed as the reciprocal of the rank at which the first correct answer was returned or 0 if no correct response was returned. The score for the entire run was then computed as the mean over all the test questions.

To emphasize on the precision of systems, starting at TREC-11, only one answer was allowed for each question. This phrase should contain the exact answer, and nothing more. Finally, in the last TREC QA conference in 2006, three measures are reported in the scores by the referees:

1. "incorrect": the answer-string does not contain a correct answer or the answer is not responsive

2. "globally correct": the answer-string consists of exactly a correct answer, that answer is supported by the document returned, and there are no later documents that contradict the answer.

3. "unsupported": the answer-string consists of exactly a correct answer, but the associated document id is incorrect. So, that document does not support the answer.

4. "locally correct": the answer-string consists of exactly a correct answer and that answer is supported by the document returned, but a later document contradicts the answer.

5. "inexact": the answer-string contains a correct answer and the document supports that answer, but the string contains more than just the answer (or is missing bits of the answer);

Year	median	best
TREC-12 (2003)	17.7%	70.0%
TREC 2004	17.0%	77.0%
TREC 2005	17.9%	71.3%
TREC 2006	18.6%	57.8%

Table 7.1: Best and median accuracy for TREC 2003 to 2006 QA main tasks.

Being *responsive* means such things as including units for quantitative responses (e.g., $20 instead of 20) and answering with regard to a famous entity itself rather than its replicas or imitations [DLK06].

It is also interesting to note that future plans for the QA track might require an answer to include possible disambiguations and justifications of why the answer is correct, or use ad-hoc text in blogs.

In terms of numbers, the best and median accuracy values for the last four TREC QA main tasks are given in Table 7.1.

7.2 Information Retrieval

To evaluate the precision and recall of our information retrieval at the document level, we count the number of questions that have the answer document retrieved in the top n rank by Lucene[1]. The ratio of these hits over the total number of questions will be the recall at level n. The threshold was set to $n = 10$ for closed domain and is $n = 35$ for open domain tests. These thresholds were chosen according to the size of the corpus and the number of documents that can include the answer for a question. In our closed domain, the corpus size is very small and only one document answers the question in most cases (see Chapter 3), while the AQUAINT corpus contains more than one million documents and usually more than one document answers a question. Therefore, in open domain we process 35 documents from the top of the list of documents returned by IR. Our development experiments show that increasing this level to more than 35 introduces more noise than correct candidates.

Figure 7.1 shows the precision and recall of our information retrieval for 426 non-copulative questions in the TREC 2003 to 2006 data sets (50 non-copulative questions from TREC 2005 were kept for development). Precision and recall at the document level (at level 35) are around 10% and

[1]Running BioKI in open domain is not feasible in its current configuration, since an efficient document/keyword index is not yet implemented for it. Currently, BioKI loads and process an excessive number of documents for each question.

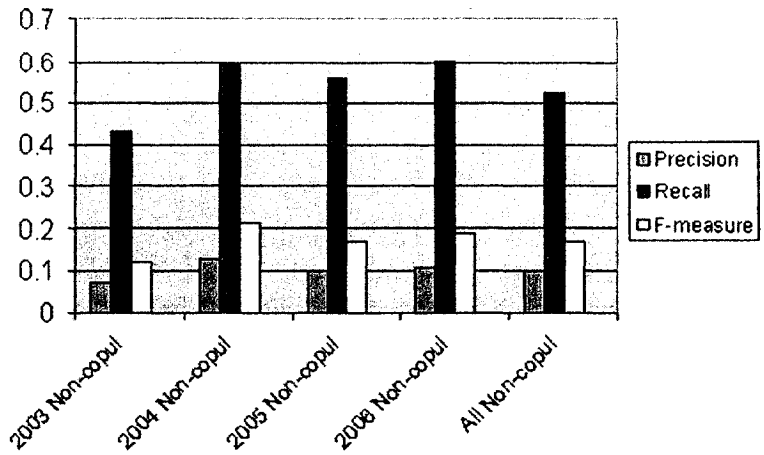

Figure 7.1: Precision and recall at the *document* level for TREC non-copulative test questions.

53% respectively (except for the 2003 where most participants performed significantly more poorly). Compared to the state of the art in IR for open domain QA systems (86.1% [HHW+06]), our IR method has a lot of space for improvement.

When the keyword ranking feature is taken out and question keywords are randomly discarded during IR iterations, recall drops to 43.6% with the TREC training questions. For many questions, discarding proper nouns (especially those from the target) drastically lowers the relatedness of the documents returned to the question.

7.3 Candidate Sentence Selection

Figure 7.2 shows how applying a more strict sentence selection reduces the number of candidates selected ('#cands/100' line) and the final QA precision (MRR-1) for our open domain development set.

In this diagram, *Document Recall* shows the fraction of development questions for which a correct document is returned by information retrieval. We have a relatively low recall at this level because no advanced IR mechanism, such as query expansion, was implemented in this work. *Sentence Recall* shows the fraction of questions for which a correct candidate sentence is selected in their candidate sentence set. As we discussed before, increasing our sentence selection keyword overlap threshold (α) results in less candidates with higher precision (but lower recall). A smaller set is easier to rank in order to find the best sentence (as shown in the *Baseline MRR-1* line. Based on this experiment

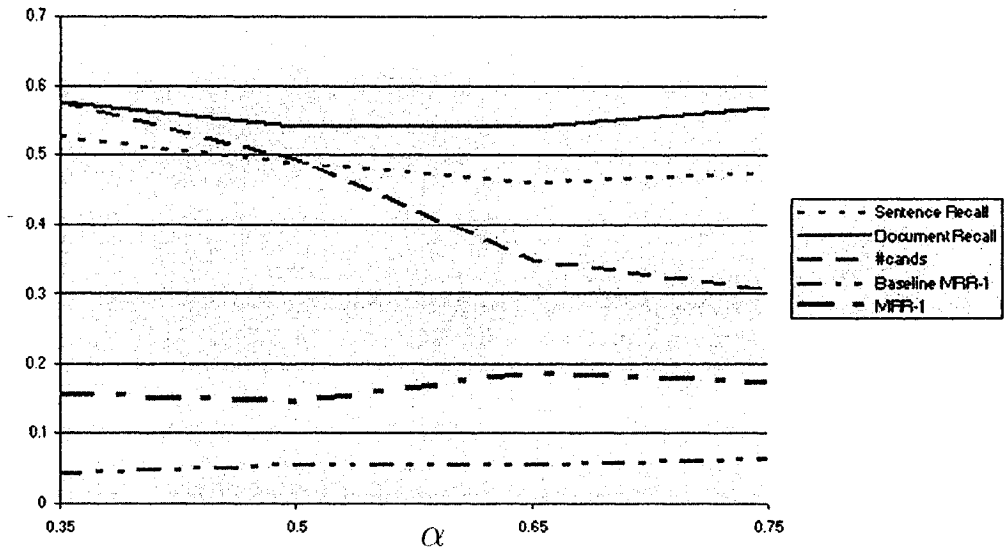

Figure 7.2: Effect of the parameter α on recall at the sentence level for our open domain development set.

on our development set, we fixed the sentence selection threshold (α) to be 0.65, the optimal point on the diagram.

Figure 7.3 shows the input accuracy at the sentence level. As shown in this diagram, sentence level recall drops to around 42% from 53% at the document level. The precision obtained for this recall level in candidate sentence extraction is 2.9%.

7.4 The Syntactic Unifier for Candidate Ranking

The theoretical baseline for this task is the precision of randomly selecting a candidate as the answer:

$$MRR = \frac{1}{x}\Sigma_{i=1}^{x}\frac{1}{i}$$

where x is the number of candidate answers. On average, we have 1.7 correct answers in a set of $x = 30$ candidates; so we can assume one correct answer in a set of 17 candidates. This results in a theoretical baseline MRR of 16.7%. However, the experimental baseline ranking accuracy is 11.2% for open domain, because of high deviation in the size of candidate sets.

If only the top ranked candidate is considered in scoring (as applied recently in TREC QA evaluations), the probability of picking the correct answer is MRR-1=$\frac{1.7}{30}$ = 5.7% while experiments

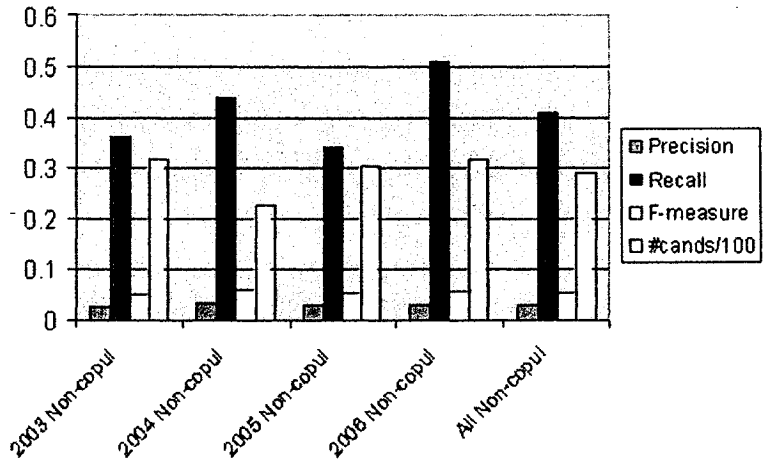

Figure 7.3: Precision and recall of candidate sentence extraction for the TREC non-copulative test questions.

show MRR-1=5.0% for open domain questions. Apparently, this baseline decreases as we relax the candidate selection threshold (α).

The "Ranking Precision" column in Figure 7.4 shows the unifier's accuracy when the error in the IR's output is excluded from the final result. The results show a high performance for the candidate ranking algorithm especially for non-copulative questions (twice as high compared to copulatives). Low accuracy for the copulative TREC questions (without a main content verb) shows the important role of the main verb in our method. As we mentioned in Chapter 3, around two third of open domain questions are copulative.

7.4.1 Error Analysis

To better understand where the system goes wrong, we manually analyzed the errors in the 139 non-copulative questions from TREC 2006. As we mentioned earlier, lack of a query expansion method prevents our system to extract candidates that have different wordings from the question. Low IR recall causes more than 50% of the error by missing their candidate answers to start the unification with.

The most frequent error source is when the answer to a non-copulative question is given in a copulative sentence (nine cases or 6.5%). As we mentioned earlier, many major syntactic differences can exist when mapping a copulative question on a non-copulative answer and vice versa. Manual modeling of multiple mapping cases is difficult and will not cover much cases. This should be done

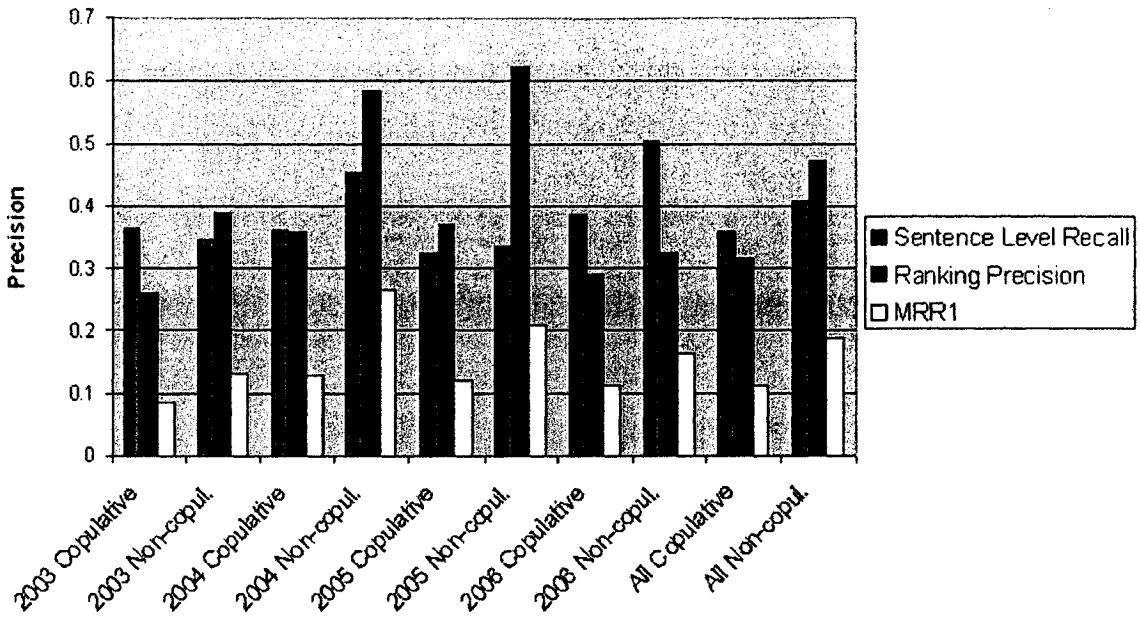

Figure 7.4: The accuracy of our syntactic ranking method on the TREC test question sets.

automatically and in large scale in order to significantly improve the results.

The *lcs* semantic similarity measure does not return a correct value for main verbs in eight correct answer sentences. We do not specify the sense of verbs, while WordNet::Similarity has the capability of accepting the sense numbers in order to compute a more precise semantic relation between verbs.

Improving the semantic named entity tagger will help answering eight more questions by boosting their unification score. An 'Organization' tag for the sentence *"Crumb, who founded the volunteer Hospice of Clallam County..."* for example would help move this candidate for the question *Q120.3- "What organization did she found?"* about 'Rose Crumb' to the first rank.

Three questions needed resolving the date of an event by adding the year the article is written in, to the answer. Answers found by our system for three questions were correct, but are not included in the judgement file. And finally, bad parse tree for the correct candidate resulted in missing three more questions. In the sentence *"Since the group formed in 1990, Duritz has been providing grainy, winsome prose about people who are needed or in need."* for example, that answers the question *Q97.2- "What year did the group form"*, the verb 'formed' is wrongly tagged as adjective:

```
0---mod--- IN:Since(Prep)
    2---pcomp-n--- NN:group(N)
 1---det--- DT:the(Det)
 3---pnmod--- VBN:formed(A)
```

```
      4---mod--- IN:in(Prep)
    5---pcomp-n--- CD:1990(N)
6---punc--- ,:,(U)
```

In conclusion, compared to the performance of other QA systems (in closed or open domain), IR is a major bottleneck of our system. Harabagiu et al. [HHW+06] for example, have an accuracy of 86.1% at the document level. Excluding this error however, shows that our syntactic ranking method has a high precision of 47% for the non-copulative question sets.

7.4.2 Syntactic Verification of Aranea's Answers

As mentioned in the previous chapter, we used the output of Aranea on the TREC data to improve our IR result. We added m documents from the IR result for the extended query

$$AranAnswers\ AND\ QuestionKeywords$$

to the regular document list we have for the *QuestionKeywords* query. Additionally, candidates that include Aranea's answers are boosted based on the position of that target answer. Figure 7.5 shows the final accuracy that our QA system achieves. MRR-1 shows the final QA precision, without receiving answer hints from Aranea. As expected, in MRR-1 (with Aranea), the unifier's accuracy consistently improves when receiving answer hints from Aranea: overall, for non-copulative TREC questions, it improves from 18.7% to 20.3%.

The "All Systems" column shows how other TREC participants performed on copulative versus non-copulative questions. Note that based on the performance of previous TREC submissions, the questions in TREC 2003 were harder to answer, with an average accuracy of MRR=12.2% for the year 2003, compared to the precision MRR-1=15.5% in the year 2004 and MRR-1=16.7% in 2005: TREC QA systems tend to work slightly better on copulative questions.

7.5 Accuracy on Copulative versus Non-copulative

In Section 3.2, we saw that more than 70% of our closed domain questions are non-copulative. To investigate if this is a factor in getting high precision in closed domain, we computed the unifier's precision for each set. Figure 7.6 shows our findings: the unifier accurately finds the best candidate

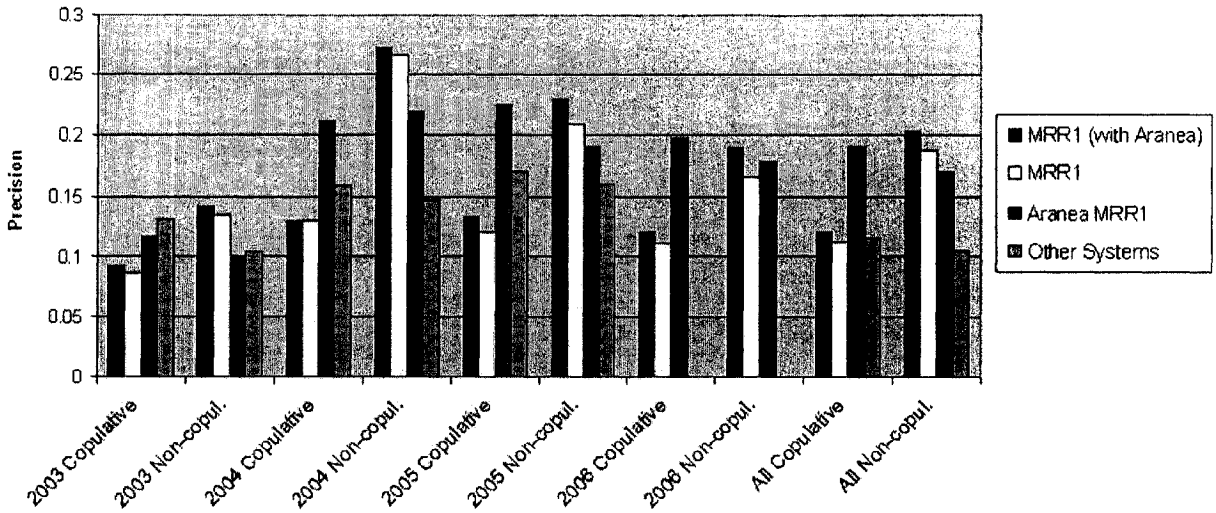

Figure 7.5: Comparison of the accuracy of the unifier with the modified Aranea and other QA participants on the TREC factoid questions.

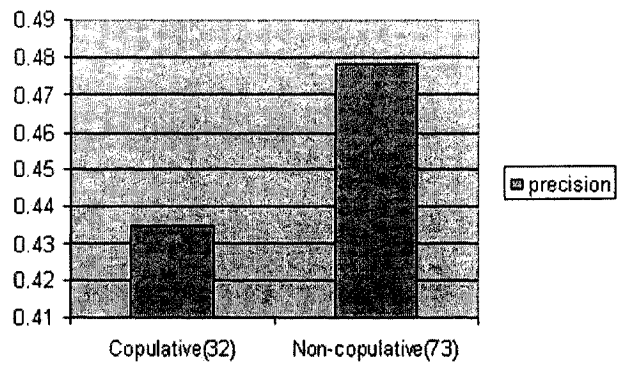

Figure 7.6: Accuracy of the unifier on copulative vs. non-copulative questions in closed domain.

answer for 47.8% of non-copulative questions and 43.5% of the copulative ones. Please note that we disregard the questions for which no correct candidate exists in the list returned by the IR. So, 73 non-copulative and 32 copulative candidates are legitimate for this analysis.

Higher relative accuracy on non-copulative questions is not as much as in open domain (Figure 7.4). However, it does seem to show again a better accuracy for non-copulative questions. Recall that generally, QA systems perform better on copulative questions in open domain (see Figure 7.4). This may mean that copulative questions are somehow easier to answer in open domain. If we assume that this is also true in closed domains, our higher accuracy on non-copulative questions becomes more significant. However, we did not have another QA system to run on our closed domain in order to verify this claim.

Category		#Q	Aranea	Overall	Aranea+Unifier	Overall
2003	copulative	171	11.7%	11.0%	11.7%	12.7%
	non-copulative	119	10.1%		14.1%	
2004	copulative	149	21.1%	21.4%	21.1%	23.2%
	non-copulative	81	21.9%		27.2%	
2005	copulative	230	22.5%	21.2%	22.5%	22.7%
	non-copulative	137	19.1%		23.0%	
2006	copulative	264	19.7%	19.0%	19.7%	19.5%
	non-copulative	139	17.7%		19.0%	

Table 7.2: Performance of the Aranea QA system vs. Aranea+Unifier on the TREC question sets.

7.6 Collaboration with a Typical QA System

Most current open domain QA systems use redundancy from the Web and the corpus to rank their candidate answers. By combining such a list with the syntactically ranked candidate list returned by our method, we have a chance to apply one's information to the other, specially, if their areas of expertise are known and do not overlap, the improvement in the combined accuracy is guaranteed.

In our experiments, Aranea answers *date* and *person* questions considerably better and performs poorly on *numeric* questions, compared to our syntactic QA system. On the 2004 question set, only 25% of the questions are commonly answered correctly by the two systems; this shows the potential complimentary feature of these two systems. In our work, we simply follow a syntactic perspective for distinguishing between the areas of expertise of each system. Precise categorization of questions based on their linguistic, content and other significant features are left as one of our future works.

We observed that not only Aranea, but other QA systems also tend to answer copulative questions slightly better, while our syntactic unifier works better for non-copulative ones. The reason is our syntactic method's dependency on a main content verb (Figure 7.5). This promises that a combination of our unifier with a typical QA system based on the question type, can result in a significant improvement in the combined accuracy. To see the effect of combining our unifier with a typical QA system, we performed an experiment with the Aranea QA system [LK03].

Table 7.2 shows the accuracy of Aranea as well as the combination of Aranea and our QA system. For this experiment, Aranea is run on copulative questions and our QA system on the non-copulative ones. The improvement ranges from 11.0% overall accuracy (the weighted average over the copulatives and non-copulatives) to 12.7% for 2003, from 21.4% to 23.2% for 2004, from 21.2% to 22.7% for the 2005 and finally from 19.0% to 19.5% for the 2006 factoid questions.

Feature	Weight	Contribution	MRR
Aranea's Suggested Answers		5%	27%
Lucene IR Score	$W_3 = \frac{1}{3}$	4%	22%
Semantic WN::Similarity Score	$W_2 = 1$	2%	18%
Syntactic Unification Score	$W_1 = 1$	16%	16%

Table 7.3: Approximate contribution of each feature in the final MRR in open domain.

7.7 Sensitivity Analysis of Weights in Open Domain

Table 7.3 shows the contribution of each feature in the final MRR in our open domain development set. Initially, the syntactic measure results in an MRR[2] of 16%. It is quite interesting that so many non-copulative questions have the same verb and very strong syntactic similarity to their candidate answer. By using semantic similarity to choose the target subtree, the MRR increases by approximately 2%, much less than in closed domain (5%). We suspect that this is because in closed domain, the variation of verbs is less and limited to the telecommunications topic, compared to open domain, in which verbs are about almost any topic with even different senses in different contexts. Finding semantic similarity of two general verbs is harder than when they are from a closed set and related to a given topic.

Adding the document score to sentence scores, increases the MRR by 4%. Compared to closed domain (Table 6.1), information retrieval has a more significant contribution in open domain (4% versus 2.5%). It again confirms that statistical word frequency information is more helpful when having a large document collection.

Finally, embedding statistical information from Aranea improves our results considerably by 5%. Recall from Section 4.3.2 that Aranea's answers expand the information retrieval query and also boost the unification scores. So, this much improvement in the final result is expected as a result of improving two processes. This also confirms that answer redundancy information from the Web is very important for answering factoid open domain questions.

In open domain also, we evaluated the sensitivity of the sentence-selection threshold (α) from 0.35 to 0.75 (Figure 7.2). Other values, lower the final MRR by 5%. Ranking accuracy is affected more in open domain, because we deal with many more documents. Although a lower threshold increases the sentence level recall, it introduces much more noise in the set of candidate sentences

[2]We report the MRR instead of MRR-1 because it is more robust and informative by providing a score for the list even if the correct sentence is not positioned in the top of the list. Additionally, we will be able to compare it with feature contributions in closed domain, reported as MRR.

($\alpha = 0.35$ has the lowest MRR). Interestingly, the optimal value for this parameter is the same as in closed domain (0.65), so that not too many candidates are selected, and at the same time, sentence level recall stays at an acceptable level.

7.8 Conclusion

In closed domain (Chapter 6), we saw that our unifier ranked the set of candidate sentences with an accuracy of 48.0%. We also saw that incorrect sentences in the candidate set when using the Lucene IR engine, do not affect significantly this accuracy (less than 6%). In this chapter we computed the accuracy of our method in open domain factoid question, which deals with a huge document collection, and much more noise. By providing our syntactic and semantic method with the Web answer redundancy information, we achieved the same level of accuracy in open domain as well. The set of candidate sentences is larger and includes on average 1.7 correct sentences though. These accuracy levels show the usefulness of syntactic mapping and semantic information for answering non-copulative questions. We also showed that our method is slightly sensitive to the sentence-selection threshold, however, the optimum value (0.65) is the same with the huge AQUAINT corpus. We also showed that using Aranea's suggested answers computed statistically from the Web for boosting the score of candidate sentences increases the final MRR-1 consistently by around 2.5%.

We saw that the syntactic feature has significant contribution in both open and closed domains; possibly because of the annotators bias towards composing questions in a similar syntactic structure as the answer sentence, it helps more in our closed domain. Verb semantic similarity contributes more in closed domain according to less variation in the vocabulary and topic. IR score on the other hand, has more contribution because query keywords are more discriminative in open domain.

Having the answer to a non-copulative question in a copulative sentence from the document collection, incorrect semantic named entity tags, and imprecise semantic similarity of main verbs were the major sources of error in our ranking method.

Finally, we showed that our syntactic method's expertise in answering non-copulative open domain questions can be exploited by building a multiplexer that runs a Web based statistical QA system on copulative questions and our syntactic unifier on the non-copulative ones. This combination resulted in a QA system that performs better than both Aranea or our Unifier alone.

Chapter 8

Conclusion and Future Work

At the beginning of our work, we started within the closed domain scenario and analyzed the questions posed in the telecommunications domain. These questions are not usually factual: they tend to ask for properties, procedures, or conditions, hence their answers are long and complex. We showed how imposing simple linguistic constraints on selecting only the candidates that refer to the same event and participating entities that the question asks for, succeeds in finding a correct answer (for 64.7% of the questions that have at least one correct answer in their candidate set.)

Later on, we evaluated our unifier's accuracy in open domain in order to see if having a small corpus / question set have resulted in building a biased method that works well only in closed domains. We suggested a syntactic classification of open domain factoid questions into copulatives, and non-copulatives. Copulative questions ask about a state with a 'to be' verb, while non-copulative questions contain at least one main non-copulative verb. The later category, that includes more than one third of the TREC questions, are similar in structure to the closed domain questions. So, we applied our method and showed that the syntactic unification method suits this subset of factoid questions and ranks their candidate sentences with an accuracy of around 45%. However, since our method relies on the semantic similarity of the question's main verb with an appropriate verb in the candidate, it does not perform as well on the copulative questions. These questions have a relatively free syntactic structure that makes the manual modeling of mapping rules between a copulative question and candidate sentences very difficult.

8.1 Conclusion

Simple syntactic mapping of the question's main verb and its arguments on candidate sentences finds the correct candidate with an acceptable accuracy of 48.0% for the questions that have a main content verb (non-copulative). The more flexible the candidate selector method is, the more candidates it will extract which makes the task of ranking them more difficult. However, we expect the overall accuracy to increase if the candidate answer extractor works more precisely; we have the evidence that in closed domain, doubling the number of candidates by switching from BioKI IR engine (Figure 6.4) to Lucene (Figure 6.3) IR engine very slightly decreased our QA system's accuracy (from 24.0% to 22.4% shown in Figure 6.5). Hence, if a better candidate answer extractor marks more candidates, it should not drastically decrease the unifier's accuracy in finding the best candidate among them.

Open domain factoid questions can be categorized based on their main verb type into copulative (having a 'to be' verb) versus non-copulative questions (having a 'content' verb). Most non-copulative questions have a typical syntactic structure: a content verb, a subject, one or more objects and modifiers; one of these syntactic constituents is replaced by the question word. Syntactic mapping works very well for closed domain questions too. We believe that immigrating this syntactic ranking to other domains is straightforward, because of the non-factual nature of closed domain questions that makes most of them be non-copulative.

Additionally, reranking the syntactically scored list of candidate answers by statistically computed scores from the Web results in a slight improvement in the ranking accuracy.

Finally, our syntactic method's expertise in answering non-copulative open domain questions can be exploited by building a multiplexer that runs a Web based statistical QA system on copulative questions and our syntactic unifier on the non-copulative ones. This combination improved the results of the compatitive Aranea QA system.

8.2 Contributions

Question Keyword Ranking Our question keyword ranking strategy is based on two factors to decide the importance of words: the number of its modifiers and its part-of-speech. The first feature

is based on the hypothesis that if a word has more modifiers, it acts as a central idea in the question and is therefore more important. We process *important* parse links provided by the Minipar parser for identifying the question keywords: based on the type of the parse link seen, both 'head' and 'tail' or only the 'tail' word is kept. Additionally, using the second feature, this method boosts proper nouns towards the front of the keyword list, since they convey a unique meaning. Verbs, on the other hand, are more ambiguous and can have more synonyms, so they are slightly pushed down the list. Adverbs usually modify verbs or adjectives, and not the nouns, so they receive the lowest rank.

This keyword ranking method has a significant effect in retrieving more relevant documents compared to when keywords are not ranked.

Syntactic Question Categorization The syntactic ranking method we propose is suitable in nature for answering the question with a main content verb. Therefore, we categorized our questions based on their main verb type into copulative (which have a 'to be' verb) versus non-copulative questions (which have a 'content' verb). Most non-copulative questions have a typical syntactic structure: a content verb, a subject, one or more objects and modifiers; one of these syntactic constituents is replaced by the question word.

Open domain results show that our ranking method outperforms the average accuracy of current QA systems in answering non-copulative questions.

Syntactic Ranking by Computing the Unification Score In order to score each candidate sentence, we find the best subtree of each candidate and match it against the parse tree of the question. The purpose of this matching is to analyze the syntactic consistency of each candidate with the question. In other words, in addition to having important common key terms, the terms in a candidate sentence must have similar syntactic relations with one another in order to convey a meaning similar to the question's.

We first find the relevance of the question's main verb to candidate's verbs using the *Leacock and Chodorow* similarity measure from the WordNet::Similarity. The arguments of the matching verb are then mapped on the arguments of the question. We relax our linguistic constraints at the subtree level (verb arguments), because the focus is already on a sentence that conveys a similar event or state to the question; only a clue about similarity of its verb arguments (phrases) is sufficient

to conclude that its verb modifies the same entities as the question. Syntactic differences of verb arguments (subtrees or phrases) should not critically affect the judgment.

Fuzzy Phrase Equivalence Detection We introduced a syntactic phrase equivalence recognision that usese syntax while not being dependent on having a perfect parse tree. Actually, each subtree can be seen as a phrase, since its focus is an entity (noun) and the whole phrase plays a syntactic role such as subject or object in the sentence.

To detect two equivalent phrases, we use two criteria:

- the *number of overlapping words* based on a bag-of-words approach,
- the number of *overlapping links* weighted by their importance.

When combining them, the overlapping links feature is given twice as importance compared to the bag-of-words feature.

Although both features have been investigated individually in different types of text (e.g. [SJT+04, KL03]), to our knowledge, this novel combination of syntactic and lexical criteria for detecting equivalent phrases is unique and new to the field.

8.3 Future Work

In this section, we summerize the research topics that this work has opened for further investigation, specifically in computing the similarity of sentences.

Query Expansion for IR Query expansion will help to remove the major source of error in our QA system. One of the controversial topics in information retrieval is whether adding synonyms of words to a query improves the results. The problem in query expansion is the ambiguity of words: without first disambiguating a word, one has to add synonyms of all the senses of a word to the query; this often introduces more noise than selecting relevant documents [Voo94].

Improvement of the Syntactic Mappings In our work, we tried to answer non-copulative questions using only non-copulative sentences. However, copulative sentences exist in the corpus that answer a non-copulative question and vice-versa. Finding an appropriate mapping from a copulative

parse tree to a non-copulative parse tree would be interesting for the non-copulative questions that are answered by a copulative sentence. We mapped the subject or the predicate subtrees (shown as *'pred'* in Minipar) in the answer tree on the question's subject and object subtrees and picked whichever scores higher.

Coreference Resolution Sometimes the answer and its supporting context appear in two consecutive sentences (see Section 3.1.3 for an example). Coreference resolution is required to identify the answer in such cases.

More importantly, resolving reference to the target in questions that have more than one pronoun needs to be studied. Additionally, questions that refer to answers from previous questions complicate the processing even more. For example, in the target series 197, the answer to the question *Q197.1- "What animal was the first mammal successfully cloned from adult cells?"* is used in *Q197.2- "What year was this animal born?"*, and answers from both these is further required as a context for *Q197.3- "At what institute was this procedure done?"*.

Improving the Semantic Similarity Criteria By Word Sense Disambiguation, we will be able to specify the sense of verbs, in order to obtain a more informed similarity measurement. WordNet::Similarity works with specific synsets. Having the sense numbers it can compute a more precise semantic relation between the correct senses of the given words.

Analyzing whether applying the semantic WordNet::Similarity measure on nouns will improve the subtree matching would be another direction of work. For an acceptable accuracy, these similarity measures need the sense number of the pair of words. In other words, for an accurate similarity values, each word needs to be sense disambiguated first, based on its context and the grammatical structure that includes them. Even with improvements in word sense disambiguation, we will still need to find a way to keep the number of comparisons needed to minimum: computing semantic similarity of a word in the question to one in the candidate sentence is a costly task since it involves looking up dictionary indices several times for the words themselves and the ones involved in their adjacency.

Embedding Statistical Criteria in the Ranking For open domain, combining the prevalent answer redundancy with our linguistic method would add a very important statistical feature to our system. Special attention should be given to parsing the question; for example, converting the question into affirmative or using more than one parser to detect incorrectly parsed questions should be studied.

Automatic Learning of the Mapping Rules Automatic learning of the best weights for the features used will provide the opportunity to introduce more features, find their relative importance and compare it to the fixed weights we currently use, and finally, a broader possibility for improvement; for example, finding a decent mapping between copulative sentences, or from a non-copulative to a copulative would not be out of reach. By involving the syntactic features we can expect more powerful patterns than POS patterns learned for question answering.

TREC Questions along with the sentences that include their answers can be compiled as positive samples while co-occurrence of the question keywords without the answer phrase will introduce negative samples for this task. Since the syntactic parse tree is not always precise, however, these samples will eventually need human verification.

One idea would be to use a number of different QA systems in parallel, and learn for what type of questions which system performs better, and so, adjust appropriate weights for the answers by each QA system accordingly. Therefore, defining and analyzing more linguistic features for the question is a future direction of research. For example, LCC System performs well for factual questions, or our QA system performs better than Aranea for questions that ask for numbers. Therefore, after the learning phase, large weights may be derived for this type of questions, and smaller weight for qualitative, temporal, or other inferential questions. Learning the capabilities of each system would give an ultimate solution to tackle open domain question answering.

SavvySearch [HD97] uses the number of links explored by the user as a direct feedback to change the effectiveness weight of a search engine for a specific query type.

Some features of the question which can be considered in learning which system to rely more for answering that specific question can be the following:

- Question type: e.g. factual, how or why questions.

- The corpus which is used (if it can be specified as an input parameter to the QA system).
- Accuracy of question analysis; i.e. if the question head, the parse tree, and other syntactic structures have been determined with confidence, knowledge-rich systems would probably perform better[1].
- The question context: there may be specialized QA systems which can provide answers to some specific domain. So, association of questions to QAs can also be a parameter to learn. Context can be determined by looking up the question head (and other important NPs) in WordNet and following hypernym relations to reach broad categories such as 'manner', 'occupation', 'natural object', 'act' , 'move', etc.

I hope enough motivation and justification is provided throughout this thesis, so these main directions of future work will be investigated upon the findings of this work.

[1] Here we assumed that the question analysis phase is done externally, which is very unlikely to be applicable, since each system does its own analysis of the question.

Bibliography

[ABC+01] Kisuh Ahn, Johan Bos, Stephen Clark, James R. Curran, Tiphaine Dalmas, Jochen L. Leidner, Matthew B. Smillie, and Bonnie Webber. QED: The Edinburgh TREC-2003 Question Answering System. In *Proceedings of the Twelfth Text REtrieval Conference (TREC-12)*, pages 631–635, Gaithersburg, MD, 2001.

[ACF+01] Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi, and Alessandro Tommasi. PiQASso: Pisa Question Answering System. In *Proceedings of the Twelfth Text REtrieval Conference (TREC-12)*, 2001.

[Ae96] G. Adorni and M. Zock (editors). *Trends in Natural Language Generation-An Artificial Intelligence Perspective.* Springer, Berlin, Heidelberg, 1996.

[BCC+01] John Burger, Claire Cardie, Vinay Chaudhri, Robert Gaizauskas, Sanda Harabagiu, David Israel, Christian Jacquemin, Chin-Yew Lin, Steve Maiorano, George Miller, Dan Moldovan, Bill Ogden, John Prager, Ellen Riloff, Amit Singhal, Rohini Shrihari, Tomek Strzalkowski, Ellen Voorhees, and Ralph Weishedel. Issues, Tasks and Program Structures to Roadmap Research in Question & Answering (Q&A). 2001.

[Ben04] Farah Benamara. Cooperative Question Answering in Restricted Domains: the WEB-COOP Experiment. In *ACL Workshop: Question Answering in Restricted Domains*, Spain, 2004.

[BFL98] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet

project. In Christian Boitet and Pete Whitelock, editors, *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics*, pages 86–90, San Francisco, California, 1998. Morgan Kaufmann Publishers.

[Bri92] Eric Brill. A Simple Rule-based Part of Speech Tagger. In *Proceedings of the Third Annual Conference on Applied Natural Language Processing, ACL*, pages 152 - 155, Italy, 1992.

[BSDL06] Sabine Bergler, Jonathan Schuman, Julien Dubuc, and Alexandr Lebedev. BioKI, A General Literature Navigation System at TREC Genomics 2006. In *Proceeding of the TREC 2006 Conference*, pages 379–382, Gaithersburg, Maryland, 2006. National Institute of Standards and Technology (NIST).

[Cha00] Eugene Charniak. A Maximum-Entropy-Inspired Parser. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pages 132–139, Seattle, Washington, 2000.

[Col97] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics*, pages 16–23, Madrid, Spain, 1997.

[CSH+04] Hoojung Chung, Young-In Song, Kyoung-Soo Han, Do-Sang Yoon, Joo-Young Lee, Hae-Chang Rim, and Soo-Hong Kim. A Practical QA System in Restricted Domains. In *ACL Workshop on Question Answering in Restricted Domains*, Spain, 2004.

[DBQ05] Bill Dolan, Chris Brockett, and Chris Quirk. Microsoft research paraphrase corpus, 2005.

[DHKN03a] Benjamin Van Durme, Yifen Huang, Anna Kupsc, and Eric Nyberg. Towards light semantic processing for Question Answering. In *Proceedings of the HLT-NAACL 2003 Workshop on Text Meaning*, pages 54–61, NJ, USA, 2003.

[DHKN03b] Benjamin Van Durme, Yifen Huang, Anna Kupsc, and Eric Nyberg. Towards light semantic processing for question answering. In *Proceedings of HLT-NAACL Workshop on Text Meaning*, pages 54–61, Edmonton, Canada, 2003.

[DLK06] Hoa Trang Dang, Jimmy Lin, and Diane Kelly. Overview of the TREC 2006 Question Answering Track. In *Proceeding of the TREC 2006 Conference*, pages 52–67, Gaithersburg, Maryland, 2006.

[DNK06] Hai Doan-Nguyen and Leila Kosseim. Using Terminology and a Concept Hierarchy for Restricted Domain Question Answering. In *Research on Computing Science, Special issue on Advances in Natural Language Processing*, 2006.

[dSBGP+05] Rodrigo de Salvo Braz, Roxana Girju, Vasin Punyakanok, Dan Roth, and Mark Sammons. An inference model for semantic entailment in natural language. In *AAAI05*, Illinois, USA, 2005.

[(ed04] Diego Molla (editor). ACL Workshop on Question Answering in Restricted Domains. Spain, 2004.

[Fel98] Christiane Fellbaum. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

[FKK+00] Michael Fuller, Marcin Kaszkiel, Sam Kimberley, Corinna Ng, Ross Wilkinson, Mingfang Wu, and Justin Zobel. The RMIT/CSIRO Ad Hoc, Q&A, Web, Interactive, and Speech Experiments at TREC-8. In *Proceedings of TREC-8 Conference*, pages 549–564, Gaithersburg, Maryland, 2000.

[GLS95] Dennis Grinberg, John Lafferty, and Daniel Sleator. A Robust Parsing Algorithm for Link Grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*, pages 111–125, Prague, Czech Republik, 1995.

[Gre04] Jerry Greenfield. Readability Formulas For EFL, 2004.

[HCRR06] William Hersh, Aaron M. Cohen, Phoebe Roberts, and Hari Krishna Rekapalli. TREC 2006 Genomics Track Overview. In *Proceeding of the TREC 2006 Conference*, pages 68–87, Gaithersburg, Maryland, 2006.

[HD97] Adele E. Howe and Daniel Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, 1997.

[Hea93] Marti A. Hearst. Texttiling: A quantitative approach to discourse segmentation. Technical Report S2K-93-24, 1993.

[HHW+06] Sanda Harabagiu, Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Ying Shi, and Bryan Rink. Question Answering with LCC's CHAUCER at TREC 2006. In *Proceedings of the Text REtrieval Conference (TREC 2006)*, pages 283–292, Gaithersburg, MD, 2006.

[HNM05] Aria D. Haghighi, Andrew Y. Ng, and Christopher D. Manning. Robust textual inference via graph matching. In *HLT-EMNLP*, pages 387–394, Vancouver, Canada, 2005.

[IFR01] Abraham Ittycheriah, Martin Frans, and Salim Roukos. IBM's Statistical Question Answering System. In *Proceedings of TREC-10 Conference*, pages 258–264, Gaithersburg, Maryland, 2001.

[KBLR06] Leila Kosseim, Alex Beaudoin, Abolfazl Keighobadi Lamjiri, and Majid Rasmara. Concordia university at the trec-qa track. In *Proceeding of the TREC 2006 Conference*, Gaithersburg, MD, November 2006.

[KEW01] Cody Kwok, Oren Etzioni, and Daniel S. Weld. Scaling question answering to the web. In *Proceedings of the tenth international conference on World Wide Web*, pages 150–161, Hong Kong, 2001.

[KKM+06] Diane Kelly, Paul Kantor, Emile Morse, Jean Scholtz, and Ying Sun. User-centered evaluation of interactive question answering systems. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*, pages 49–56, New York, NY, USA, 2006.

[KL03] Boris Katz and Jimmy Lin. Selectively Using Relations to Improve Precision in Question Answering. In *Proceedings of the EACL 2003 Workshop on Natural Language Processing for Question Answering*, Hungary, 2003.

[KSW06] Michael Kaisser, Silke Scheible, and Bonnie Webber. Experiments at the University of Edinburgh for the TREC 2006 QA Track. In *Proceedings of the Text REtrieval Conference (TREC 2006)*, Gaithersburg, MD, 2006.

[KT04] Milen Kouylekov and Hristo Tanev. Document filtering and ranking using syntax and statistics for open domain question answering. In *Proceedings of ESSLLI 2004 Workshop on Combining Shallow and Deep Processing for NLP*, pages 21–30, Nancy, France, 2004.

[LDK04] Abolfazl Keighobadi Lamjiri, Osama El Demerdash, and Leila Kosseim. Simple features for statistical word sense disambiguation. In *Proceedings of the 3rd International ACL Workshop on Workshop on Word Sense Disambiguation (Senseval-3)*, pages 37–44, Barcelona, Spain, 2004.

[LDKB07] Abolfazl Keighobadi Lamjiri, Julien Dubuc, Leila Kosseim, and Sabine Bergler. Low-frequency indexing low frequency information for answering complex questions (to appear). In *Proceeding of the RIAO'2007 - Large-Scale Semantic Access to Content (Text, Image, Video and Sound)*, Pittsburgh, USA, 2007.

[Len95] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[Lin93] Dekang Lin. Principle-based Parsing without Overgeneration. In *Proceedings of ACL-93*, pages 112–120, Ohio, USA, 1993.

[Lin98] Dekang Lin. Review of WordNet: An electronic lexical database, 1998.

[LK03] Jimmy Lin and Boris Katz. Question Answering from the Web Using Knowledge Annotation and Knowledge Mining Techniques. In *Proceedings of CIKM'03*, pages 116 - 123, Louisiana, USA, 2003. ACM.

[LKR07] Abolfazl Keighobadi Lamjiri, Leila Kosseim, and T. Radhakrishnan. A hybrid unification method for question answering in closed domains. In *Proceedings of the 3rd International Workshop on Knowledge and Reasoning for Answering Questions (KRAQ07)*, pages 36–42, Hyderabad, India, 2007.

[MBT06] Dan Moldovan, Mitchell Bowden, and Marta Tatu. A Temporally-Enhanced PowerAnswer in TREC 2006. In *Proceedings of the Text REtrieval Conference (TREC 2006)*, pages 275–282, Gaithersburg, MD, 2006.

[MCH03] Dan Moldovan, Christine Clark, and Sanda Harabagiu. COGEX: A Logic Prover for Question Answering. In *Proceedings of HLT-NAACL 2003*, pages 87–93, Edmonton, Canada, 2003.

[Me05] Diego Molla and Jose Luis Vicedo (editors). Special Issue of Computational Linguistics on Question Answering in Restricted Domains. 2005.

[MHH+07] Rutu Mulkar, Jerry R. Hobbs, Eduard Hovy, Hans Chalupsky, and Chin-Yew Lin. Learning by reading: Two experiments. In *Proceedings of the 3rd International Workshop on Knowledge and Reasoning for Answering Question (KRAQ'07)*, pages 3–6, Hyderabad, India, 2007.

[Mla98] Dunja Mladenic. *Machine Learning on non-homogeneous, distributed text data.* PhD thesis, University of Ljubljana, Slovenia, 1998.

[MM01] Rada Mihalcea and Dan I. Moldovan. eXtended Wordnet: progress report. In *Proceedings of the NAACL-2001 Workshop on WordNet and Other Lexical Resources*, Pittsburgh, USA, 2001.

[MN02] Dan Moldovan and Adrian Novischi. Lexical Chains for Question Answering. In *Proceedings of COLING*, pages 674–680, Taiwan, 2002.

[Mol03] Diego Molla. Towards semantic-based overlap measures for Question Answering. In *Proceedings of Australasian Language Technology Workshop (ALTW)*, Melbourne, 2003.

[MSB98] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving Automatic Query Expansion. In *Annual ACM Conference on Research and Development in Information Retrieval*, pages 206–214, Melbourne, Australia, 1998.

[NMC+03] Eric Nyberg, T. Mitamura, J. Callan, J. Carbonell, R. Frederking, K. Collins-Thompson, L. Hiyakumoto, Y. Huang, C. Huttenhower, S. Judy, J. Ko, A. Kupse,

L. Lita, V. Pedro, D. Svoboda, and B. Van Durme. The JAVELIN Question Answering system at TREC 2003: A Multi-Strategy approach with dynamic planning. In *Proceedings of the Twelfth Text REtrieval Conference (TREC-12)*, Gaithersburg, MD, 2003.

[PGK05] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles, 2005.

[PLK02] Luc Plamondon, Guy Lapalme, and Leila Kosseim. The QUANTUM Question Answering System at TREC-11. In *Proceeding of the TREC-11 Conference*, pages 750–757, Gaithersburg, Maryland, 2002.

[PPM04] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet::Similarity - measuring the relatedness of concepts. In *Proceedings of the Fifth Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-04)*, Boston, USA, 2004.

[PRtY04] Vasin Punyakanok, Dan Roth, and Wen tau Yih. Natural language inference via dependency tree mapping: An application to question answering. In *Computational Linguistics, Vol. 6, No. 9*, 2004.

[PWH+04] Sameer Pradhan, Wayne Ward, Kadri Hacioglu, James Martin, and Dan Jurafsky. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association of Computational Linguistics (HLT/NAACL)*, Boston, MA, 2004.

[RWHB+92] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. Okapi at TREC. In *Proceeding of the First TREC Conference*, pages 21–30, Gaithersburg, Maryland, 1992.

[Sam95] Geoffrey Sampson. *English for the Computer: the SUSANNE corpus and analytic scheme.* Oxford University Press, 1995.

[Sch05] Karin Kipper Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon.* PhD thesis, University of Pennsylvania, PA, USA, 2005.

[Sen92] Stephanie Seneff. TINA: A natural language system for spoken language applications. In *Computational Linguistics*, volume 18 (1), pages 61–86, 1992.

[SJT+04] Renxu Sun, Jing Jiang, Yee Fan Tan, Hang Cui, Tat-Seng Chua, and Min-Yen Kan. Using Syntactic and Semantic Relation Analysis in Question Answering. In *Proceedings of the Thirteenth Text REtrieval Conference (TREC-13)*, Gaithersburg, MD, 2004.

[SLPCW97] Tomek Strzalkowski, Fang Lin, Jose Perez-Carballo, and Jin Wang. Building effective queries in natural language information retrieval. In *Proceeding of the Fifth Conference on Applied Natural Language Processing (ANLP)*, pages 299–306, Washington DC., 1997.

[ST91] Daniel D.K. Sleator and Davy Temperley. Parsing English with a Link Grammar. In *Technical Report CMU-CS-91-196, Carnegie Mellon University, School of Computer Science*, Pittsburgh, PA, 1991.

[TKM+05] Hristo Tanev, Milen Kouylekov, Bernardo Magnini, Matteo Negri, and Kiril Simov. Exploiting Linguistic Indices and Syntactic Structures for Multilingual Question Answering: ITC-irst at CLEF 2005. In *CLEF-2005 Working Notes*, pages 21–23, Vienna, Austria, 2005.

[Voo94] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69, NY, USA, 1994. Springer-Verlag New York, Inc.

[VT99] Ellen Voorhees and Dawn Tice. The TREC-8 Question Answering Track Evaluation. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 83–106, Gaithersburg, MD, 1999. National Institute of Standards and Technology (NIST).

[ZSD+04] Zhuo Zhang, Lyne Da Sylva, Colin Davidson, Gonzalo Lizarralde, and Jian-Yun Nie. Domain-Specific QA for the Construction Sector. In *Information Retrieval for Question Answering (IR4QA) Workshop in 27th ACM-SIGIR 2004*, pages 64–70, Sheffield, UK, 2004.

[ZSG+00] Victor Zue, Stephanie Seneff, James Glass, Joseph Polifroni, Christine Pao, Timothy J. Hazen, and Lee Hetherington. Jupiter: A telephone-based conversational interface for weather information. In *IEEE Transactions on Speech and Audio Processing*, volume 8 (1), 2000.