From Document Retrieval to Question Answering

From Document Retrieval to Question Answering

Christof Monz

ILLC Dissertation Series DS-2003-4

ILLC Dissertation Series DS-2003-4

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Plantage Muidergracht 24
1018 TV Amsterdam
phone: +31 20 525 6051
fax: +31 20 525 5206
e-mail: illc@science.uva.nl
home page: http://www.illc.uva.nl

From Document Retrieval to Question Answering

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.mr. P.F. van der Heijden
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in de
Aula der Universiteit
op donderdag 11 december 2003, te 12.00 uur

door

Christof Monz

geboren te Haan, Duitsland.

Promotie commissie:

Promotores:
Prof.dr. F.M.G. de Jong
Prof.dr. R. Scha

Co-promotor:
Dr. M. de Rijke

Overige leden:
Prof.dr. C. Clarke
Dr. K. Sima'an
Prof.dr. M. Stokhof
Prof.dr. B. Webber

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 612-13-001 and 220-80-001.

Cover design by Christof Monz.
Typeset in Palatino using pdfLaTeX.
Printed and bound by Print Partners Ipskamp, Enschede.

ISBN: 90-5776-116-5

Für meine Eltern,
Christina und Karl-Heinz Monz

Contents

List of Tables

List of Figures

Preface

Information is one of the most valuable assets in modern society. With the pervasive presence of computers, storing huge amounts of data has become both efficient and inexpensive. We are now in a position where we have unprecedented amounts of information at our finger tips. How do we access these large amounts of data in order to find the information we are interested in? Some data is stored in a database, a form that is designed to facilitate accessing the data in a number of ways. However, most textual data is not available in a structured database, but is only available in an unstructured format, such as plain text. Transforming unstructured data into a database format can be a very laborious process, depending on the complexity of the anticipated database. As a consequence, most data remains stored in an unstructured format.

The search for relevant information in large amounts of unstructured data calls for automatic means that aid in this process, as manual inspection of all data is practically infeasible. The issue of developing methods and tools for finding automatically relevant information is addressed by the research area of information retrieval. In recent decades, sophisticated document retrieval systems have been developed. These systems allow a user to submit a query, that is, a number of keywords describing the user's information need, to a retrieval system, and they return a list of relevant documents, such as newspaper articles, legal or medical documents, web pages, or patents. As the name suggests, document retrieval systems return *full documents* to satisfy a user's information need. However, often, an information need is more specific and much more appropriately expressed as a question instead of a set of keywords. Imagine, for instance, that you want to know when the Titanic sank. You probably prefer being able to ask the question *When did the Titanic sink?* to a retrieval engine, and getting a date back from the system, over submitting the keyword-based query *Titanic sank* and skimming through the documents returned by the system in the hope that one of them contains the date on which the Titanic sank.

Information retrieval systems that allow for users to pose natural language ques-

tions are known as *question answering systems*. Such systems have been developed since the 1950s, but originally they were mainly restricted to narrow domains, such as baseball statistics or lunar soil samples. In the late 1990s, however, the question answering task was integrated into the annual Text REtrieval Conference (TREC), which gave a significant impulse to question answering as a research area, boosting it into the direction of open-domain question answering.

Open-domain question answering systems very often use document retrieval techniques to identify documents that are likely to contain an answer to the question asked by the user. And this is where the focus of this thesis lies: the role of document retrieval in the context of question answering. The aim of this thesis is to identify document retrieval approaches that are particularly useful for identifying documents which contain an answer to a question.

About this Thesis

This PhD thesis is the outcome of a curvy education. After graduating from high school, and having served community service, I started studying German literature in my home town Wuppertal in Germany. Right away, in my first semester, I had to take mandatory courses in linguistics, which, at that time, I imagined to be a nuisance rather than an interesting research area. One of those linguistics courses was given by Lisa Rau, who convinced me that linguistics is indeed an interesting field. I was so intrigued by it that I changed my major subject to linguistics within that very same semester. During my basic studies in linguistics I became more and more interested in formal semantics.

After having finished my basic studies, I continued my studies at the Institute for Natural Language Processing (IMS), at the University of Stuttgart, Germany, which hosted a large number of well-known formal semanticists at the time. My study at the IMS introduced me to the research field of computational linguistics, and I am indebted to many of the institute's researchers for providing an extremely stimulating research environment. In particular, I am grateful to Hans Kamp, Uwe Reyle, and Esther König-Baumer.

In 1997, I spent a year as an exchange student at the Institute for Logic, Language and Computation (ILLC) at the University of Amsterdam, in The Netherlands. During that period I worked closely together with Maarten de Rijke on applying theorem proving to computational semantics; I would like to thank Maarten for being a great companion ever since.

After having received my degree in computational linguistics from the University of Stuttgart in 1999, I returned to the ILLC to do my PhD in computer science, focusing on information retrieval and question answering. This shift of research area was due to my urge to do more applied natural language processing research.

While it has been a curvy road, I am glad that it has not been a straight lane, as it has shown me many facets of human language, which is an intriguing phenomenon whose scientific understanding requires insights from many research areas.

Acknowledgments

Doing my PhD at the Institute for Logic, Language and Computation (ILLC) was a great experience. I would like to thank Maarten de Rijke, who offered me the opportunity to pursue a PhD at the ILLC. I have learned a great deal from him and I am very indebted to him for his continuous support.

Although the numer of people at ILLC that were interested in information retrieval was rather limited in the beginning, it grew steadily over the years, especially after the launch of the Language & Inference Technology (LIT) group in 2001. The discussions with many of the researchers in the LIT group and at ILLC have been very fruitful for my research. From the many peope I have talked to over these years, I would like to thank in particular Börkur Sigurbjörnsson, Carlos Areces, Caterina Caracciolo, Detlef Prescher, Gilad Mishne, Gabriel Infante-Lopez, Gabriele Musillo, Harald Hammarström, Jaap Kamps, Jeroen de Knijf, Jon Ragetli, Juan Heguiabehere, Karin Müller, Khalil Sima'an, Maarten de Rijke, Marco Aiello, Raffaella Bernardi, Stefan Schlobach, Valentin Jijkoun, Vera Hollink, and Willem van Hage.

Much of the work I have published over the years was written together with some colleagues, and the discussions with them have proved to be very insightful. For this, I would like to thank Börkur Sigurbjörnsson, Carlos Areces, Gilad Mishne, Hans de Nivelle, Jaap Kamps, Jon Ragetli, Leon Todoran, Maarten Marx, Maarten de Rijke, Marcel Worring, Marco Aiello, Oren Tsur, Stefan Schlobach, Valentin Jijkoun, and Vera Hollink.

This thesis benefited a lot from the contributions of my supervisors, Franciska de Jong, Maarten de Rijke, and Remko Scha, and I would like to thank them for their efforts. Bonnie Webber, Charlie Clarke, Khalil Sima'an, and Martin Stokhof acted as members of the approval committee for this thesis, and I am very grateful for their comments.

I would also like to thank the people that provided administrative support during my PhD studies: Ingrid van Loon, Marco de Vries, Marjan Veldhuisen, and Ria Rettob.

On a more personal level, I would like to thank Angela Ward, Christiaan Kooman, Gabriel Infante-Lopez, and Marco Aiello, for a great time in Amsterdam.

Last but certainly not least, I would like to acknowledge the financial support that I received from the Netherlands Organization for Scientific Research (NWO), under project numbers 612-13-001 and 220-80-001. Without NWO's support the research that lead to this thesis would not have been possible.

Chapter 1

Introduction

This introductory chapter discusses the main issues that are addressed in this thesis, and it provides the essential backgound to the research area of question answering. The architecture of a prototypical question answering system is given, and its main components are discussed. We also provide some details on the TREC question answering evaluation campaign, which is a driving force for much of the ongoing research on question answering, including the work described in this thesis.

Document retrieval systems have become part of our daily lives, mostly in the shape of internet search engines, such as GOOGLE (Google) or ALTAVISTA (AltaVista). Although document retrieval systems do a great job in finding relevant documents, given a set of keywords, there are situations where we have a more specific information need. For instance, imagine you want to know when the story of Romeo and Juliet took place. One possible solution could be to search for *Romeo and Juliet* and hope that the returned documents contain the date at which the story took place. But of course, it would be much nicer if you could simply ask the question *When did the story of Romeo and Juliet take place?* and got back the answer *13th century*. The virtue of question answering systems is that they allow the user to state his or her information need in a more specific and natural form, viz. as a natural language question, and that they do not return full documents which have to be skimmed by the user to determine whether they contain an answer, but short text excerpts, or even phrases.

Developing systems that are able to answer natural language questions automatically has been a long-standing research goal. Building systems that enable users to

access knowledge resources in a natural way (i.e., by asking questions) requires insights from a variety of disciplines, including, Artificial Intelligence, Information Retrieval, Information Extraction, Natural Language Processing, and Psychology.

Over the years, many question answering systems have been developed, for a variety of purposes: Some systems are intended to provide database access in very specific domains, such as rocks and soil samples that were collected on the Apollo 11 lunar mission (Woods, 1977), while others are more open-domain oriented, aiming to answer general trivia-like questions.

The context in which a question answering system is used, i.e., the anticipated user, the type of questions, the type of expected answers, and the format in which the available information is stored, determines the design of the system. Two basic types of question answering systems can be distinguished: systems that try to answer a question by accessing structured information contained in a database, and systems that try to answer a question by analyzing unstructured information such as plain texts. Of course, many actual systems are hybrids of both types, and combinations of both types of systems will be discussed later.

For question answering systems that use databases to find an answer, the main challenge is to transform a natural language question into a database query. Often, systems of this type are also referred to as natural language interfaces to database systems, rather than stand-alone systems (Androutsopoulos et al., 1995). Since database question answering systems use knowledge bases that are structured (or at least semi-structured), they exploit that structure to match elements from the question with database entries of the appropriate type, and finally identify a database entries which are of the type the question was asking for. Since the manual construction of databases is a laborious process, and the automatic construction of databases is mainly confined to information that can be easily captured by automated means, database question answering systems tend to be restricted to rather narrow domains. Well-known database-oriented question answering system are BASEBALL (Green et al., 1963), which answers questions about results, locations, and dates of baseball games, the aforementioned LUNAR system allows a user to ask questions about lunar rock and soil material that was compiled during the Apollo 11 moon mission, and PHLIQA1 (Bronnenberg et al., 1980; Scha, 1983), which was designed to answer short questions against a data base containing fictitious data about computer installations in Europe and companies using them. Although each of the systems was working 'pretty well' (unfortunately, no formal evaluations are available), they were restricted to their respective domain, and expanding them to domains other than the ones they were initially intended for is a non-trivial process, requiring a substantial amount of expertise in the areas to which the system should be expanded to. This restriction to narrow domains, and the problems that were encountered when adapting database-oriented question answering systems to new domains, are probably the main reasons for the rather modest impact these systems had on commercial applications in information processing.

The other type of question answering systems are text-based systems. Textual

question answering systems do not require their knowledge bases to be in a particular format, instead they aim to find an answer to a question by analyzing documents in plain-text format, such as newspaper/newswire articles, manuals, and encyclopedias. Textual question answering systems match the question with text units, e.g., phrases or sentences, in the document collection, and within those units, identify the element the question is asking for. The task of identifying elements of the appropriate type is closely related to the research area of information extraction, and in fact, some systems do integrate insights from information extraction into their question answering approach, see e.g. (Srihari and Li, 1999). The intricate part is to identify text units that are likely to contain an answer, as they can express this information in a way that is very different from the original question. These differences may pertain to syntactic structures, different wording, or a combination of both. To some extent these differences can be compensated for by the amount of data that is searched for an answer: The more data is available, the higher the chance that there are occurrences where this information is expressed in a way similar to the question. On the other hand, increasing the size of the data used for finding an answer, also increases the computational costs of finding an answer. Therefore, an appropriate balance has to be found between the level of sophistication of the answer identification strategies and the amount of data that is inspected.

Current document collections contain hundreds of thousands of documents, and searching through all of them for an answer takes much too long to be useful for real applications. Therefore, most, if not all, textual question answering systems use a document retrieval system to identify documents that are likely to contain an answer to the original question. This restricted set of documents is then analyzed further by using more sophisticated tools to find an actual answer.

The pre-selection of documents that are considered for further analysis is a critical step in the whole question answering process. This pre-selection acts like a filter for the document collection to select documents that are likely to contain an answer. Selecting too many documents might increase the computational costs to an extent which hurts the system's usefulness. It might also fail to reduce sufficiently noise, which may in turn hurt the performance of later modules in the question answering pipeline. Selecting too few documents might have the effect that none of them contains an answer to the original question, while there are documents in the collection that do contain an answer. The research issue at this point is to identify appropriate ways of ranking the documents in the collection with respect to their likelihood of containing an answer, such that documents containing an answer are high-ranked. This allows the subsequent analysis steps to be restricted to a small number of documents, which allows for a more focused analysis.

The question is whether techniques that have proved to be effective for traditional document retrieval are equally effective for retrieval as pre-fetching for question answering. More specifically, what retrieval techniques should be used? This thesis compares some of the traditional and also new retrieval techniques in the context of question answering.

Textual question answering systems date back to the 1960s, see e.g., ORACLE(Phillips, 1960), PROTOSYNTHEX (Simmons et al., 1963), and ALA (Thorne, 1962). Until the early 1990s, there were few further research efforts in the area. In recent years however, question answering witnesses a true renaissance. The re-emerging interest in textual question answering is largely due to the Text REtrieval Conference (TREC) initiative, which has featured a textual question answering track since 1999 (Voorhees, 2001c). At TREC, participating groups evaluate and compare their question answering systems with respect to some standard set of questions. This allows for an objective comparison of question answering techniques and the rapid interchange of ideas to further the research in that area. As we will see below, the TREC question answering data sets play an important role throughout this thesis.

The remainder of this chapter is organized as follows: The next section describes the architecture of standard textual question answering system. Section 1.2 then provides some information on the TREC question answering evaluation campaign, in particular on the document collections, the type of questions, and the evaluation criteria. Section 1.3 discusses the main research questions that are addressed in this thesis. Section 1.4 gives a short overview of the thesis, and the material that is covered by later chapters.

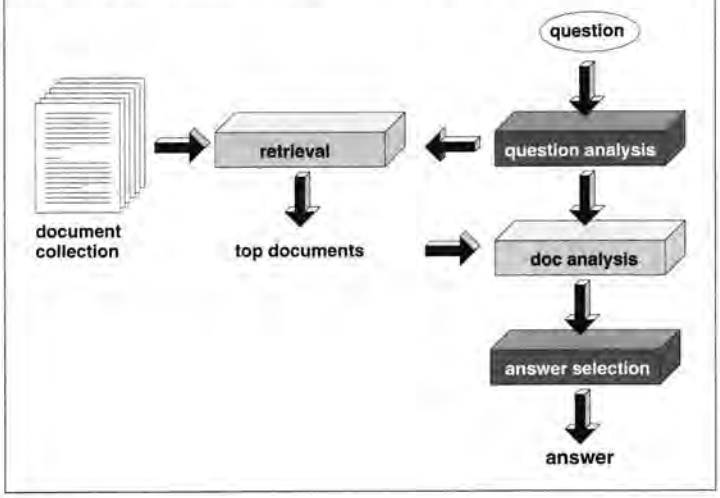
1.1 Textual Question Answering System Architecture

1.1.1 The General Architecture

Currently, there are dozens of textual question answering systems described in the literature. In 2002, 34 research groups participated in the question answering track of the annual Text REtrieval Conference (TREC), each group having implemented their own system. These systems cover a wide spectrum of different techniques and architectures, and it is impossible to capture all variations within a single architecture. Nevertheless, most systems also have a number of features in common, which allows us to give a general architecture of a prototypical question answering system. Figure 1.1 displays the main components, of such a general architecture, and the ways in which they interact. The prototypical system has four components: *question analysis*, *document retrieval*, *document analysis*, and *answer selection*. Each of these components is discussed in more detail later in this section. At this point we only give a brief overview of the whole architecture.

Given a natural language question posed by a user, the first step is to analyze the question itself. The question analysis component may include a morpho-syntactic analysis of the question. The question is also classified to determine what it is asking for, i.e., whether it is asking for a date, a location, the name of a person etc. Depending on the morpho-syntactic analysis and the class of the question, a retrieval query is formulated which is posed to the retrieval component. Some of this information, such as the question class and a syntactic analysis of the question, are also sent to the document analysis component.

Figure 1.1: General architecture of a textual question answering system



The retrieval component is generally a standard document retrieval system which identifies documents that contain terms from a given query. The retrieval component returns a set or ranked list of documents that are further analyzed by the document analysis component.

The document analysis component takes as input documents that are likely to contain an answer to the original question, together with a specification of what types of phrases should count as correct answers. This specification is generated by the question analysis component. The document analysis component extracts a number of candidate answers which are sent to the answer selection component.

The answer selection component selects the phrase that is most likely to be a correct answer from a number of phrases of the appropriate type, as specified by the question analysis component. It returns the final answer or a ranked list of answers to the user.

Let us now take a closer look at each of the four components.

1.1.2 Question Analysis

The main function of the question analysis component is to understand the purpose of the question, i.e., the kind of information the question is asking for. To identify the purpose of a question, the question is analyzed in a number of ways. First, the question is assigned a class, or a number of classes. Table 1.1 shows a number of question classes that are used in our textual question answering system TEQUESTA (Monz and de Rijke, 2001a; Monz et al., 2002). Although this is the set of classes of

a particular system, it bears a strong resemblance with the question classes of many current question answering systems. Note that some of the classes in table 1.1 are

Table 1.1: Examples of question types

question type	description	
`agent`	name or description of an animate entity causing an action	
	Who won the Oscar for best actor in 1970?	(topic id: 1424)
`aka`	alternative name for some entity	
	What is the fear of lightning called?	(topic id: 1448)
`capital`	capital of a state or country	
	What is the capital of Kentucky?	(topic id: 1520)
`date`	date of an event	
	When did the story of Romeo and Juliet take place?	(topic id: 1406)
`date-birth`	date of birth of some person	
	When was King Louis XIV born?	(topic id: 1880)
`date-death`	date of death of some person	
	When did Einstein die?	(topic id: 1601)
`expand-abbr`	the full meaning of an abbreviation	
	What does NASDAQ stand for?	(topic id: 1531)
`location`	location of some entity or event	
	Where did Golda Meir grow up?	(topic id: 1818)
`thing-ident`	a thing identical to the description	
	What is the atomic number of uranium?	(topic id:1547)
`what-np`	an instance of the NP fitting the description	
	What college did Allen Iverson attend?	(topic id:1484)

hierarchically ordered. For instance, the question class `date-death` is a subclass of the class `date`.

Assigning question classes can be accomplished in a variety of ways. One of the simplest, and yet quite effective, ways is to apply pattern matching to the question to identify its type. Table 1.2 lists some of the patterns that are used to classify questions. Classification is sensitive to the order in which the patterns are applied. For instance, the more specific patterns `date-birth` and `date-death` are applied first, before the more general pattern `date`. Note that there is no pattern to classify `what-np` questions, as these require more syntactic information. As an alternative to pattern matching there are much more sophisticated means for question classification, Suzuki et al. (2003); Zhang and Lee (2003) use support vector machines, a machine learning approach. Hermjakob (2001) fully parses questions and then apply a large number of rules to the parse tree to classify questions. Li (2002) uses language models for question classification.

In parallel, a morpho-syntactic analysis of the words in the question is carried out. This assigns to each word in the question a part-of-speech tag, indicating whether a word is a verb, singular noun, plural noun, etc. After having assigned part-of-speech tags to words, it is possible to classify questions as `what-np`

Table 1.2: Sample patterns for question classification

Question class	Example patterns
agent	`/[Ww]ho /,/ by whom[\.\?]/`
aka	`/[Ww]hat( i|\')s (another|different) name /`
capital	`/[Ww]hat is the capital /,/[Ww]hat is .+\'s capital/`
date	`/[Ww]hen /,/[Ww](hat|hich) year /`
date-birth	`/[Ww]hen .* born/,/[Ww](hat|hich) year .* born/`
date-death	`/[Ww]hen .* die/,/[Ww](hat|hich) year .* die/`
expand-abbr	`/stand(s)? for( what)?\s*?/,/the abbreviation .+ mean\s*?/`
location	`/[Ww]here(\'s)? /,/ is near what /`
thing-ident	`/[Ww](hat|hich)( wa| i|\')s the /`
what-np	-

questions, simply by checking whether the question is of the form `(What|Which) (ADJ|NOUN)* NOUN`.

In addition to classifying the question, the question analysis component has to formulate the query that is posed to the retrieval component. In order to do so, each word is first normalized to its morphological root. Typically, this is done by using a rule-based stemmer, such as the Porter stemmer (Porter, 1980), or by looking up the morphological root in a machine readable dictionary. The morphologically normalized words are used to pose the query to the retrieval engine.

There are many ways to formulate the query, depending on the functionality of the retrieval engine. E.g., some engines allow structured queries, where terms are connected by certain operators, such as the `proximity`-operator, which requires the terms in its scope to occur close to each other in the document. Here, we simply assume *bag-of-words* queries, where a query is an unordered list of single terms.

The quality of the question analysis component has far-reaching consequences for later stages in the question answering process. For instance, if a question is incorrectly classified, the document analysis module will try to find phrases of the wrong type.

1.1.3 Document Retrieval

We now turn to the document retrieval component, which is the main topic of this thesis.

The function of the document retrieval component is not to find actual answers to the question, but to identify documents that are likely to contain an answer. This process of pre-selecting documents is also known as *pre-fetching*. Because the information need in question answering is much more specific than in traditional retrieval, many systems use a *boolean* retrieval system, which gives more options to formulate a query, or *passage-based* retrieval which emphasizes the fact that answers are normally expressed very locally in a document. Using a passage-based retrieval approach instead of a full-document retrieval approach, has the additional advan-

tage that it returns short text excerpts instead of full documents, which are easier to process by later components of the question answering system.

Document retrieval has a long tradition and many frameworks have been developed over the years, resulting in sophisticated ways to compute the similarity between a document and a query (Salton and Buckley, 1988; Zobel and Moffat, 1998). However, these approaches have been tailored to queries that in the majority of cases express a more general information need than actual questions. For instance, a document retrieval information need from the TREC collection asks for documents about Nobel Prize winners, regardless of the field or the year it was awarded, whereas one of the information needs from the question answering data set asks *Who won the Nobel Peace Prize in 1992?*. Hence, we need to address the issue whether retrieval approaches that perform well for traditional retrieval are equally well-suited for question answering. Or, whether we need new retrieval approaches that are particularly tailored to question answering.

Depending on the retrieval engine that is actually used, the retrieval component returns either an unordered set of documents that are likely to contain an answer, or a ranked list of documents, where the documents are ranked with respect to their likelihood of containing an answer.

Although document retrieval is just one of the components of the whole question answering process, its effectiveness is critical to overall performance of a question answering system. If the document retrieval component fails to return any document that contains an answer, even optimally functioning document analysis and answer selection components will inevitably fail as well to identify a correct answer.

1.1.4 Document Analysis

The document analysis component searches through the documents returned by the retrieval component to identify phrases that are of the appropriate type, as specified by the question analysis component. To this end, a named-entity recognizer is used to assign semantic types to phrases in the top documents. The set of named entities includes person names, organization, dates, locations, temporal and spatial distances, etc. If a phrase is of the appropriate type, it has to be linked to the information need expressed by the question, in order to consider it a potential answer, or candidate answer. Linking a candidate answer to the question is a non-trivial process, and there are a number of ways to do this. For certain types of question, parse trees or parse dependency graphs can be used to determine whether the phrase occurs in the right syntactic position. For instance the answer phrase to question (1.1), can be expressed as the subject of a relative clause (1.2.a), or the noun phrase which is modified by an apposition (1.2.b).

(1.1) Who developed the vaccination against polio? (topic id: 911)

(1.2) a. Dr. Jonas Salk who developed a polio vaccine ...

b. Dr Albert Sabin, developer of the oral polio vaccine, ...

In some cases these syntactic relationships can also be approximated by patterns, although pattern matching will lose some of the flexibility of using a deeper analysis like parsing.

For other types of questions, on the other hand, pattern matching is a simple and effective means to find answers. Consider question (1.3), and the text snippet (1.4).

(1.3) What year was Mozart born? (topic id: 1225)

(1.4) ...Mozart (1756–1791).

Here, a candidate answer can simply be identified by applying a pattern such as `NAME (YEAR_BIRTH-YEAR_DEATH)`. Soubbotin and Soubbotin (2001, 2002) showed that pattern matching can lead to a well-performing question answering system, see also (Ravichandran and Hovy, 2002; Ravichandran et al., 2003) for ways to automatically generate answer matching patterns.

Sometimes, linking a phrase to the question is much more difficult, and involves complex reasoning on the lexical definition of a word. The following example is taken from (Harabagiu et al., 2001). Consider question (1.5) and the text excerpt in (1.6), which contains the answer phrase *Harvard*.

(1.5) Where did Bill Gates go to college? (topic id: 318)

(1.6) ...Bill Gates, Harvard dropout and founder of Microsoft, ...

The fact that Bill Gates has attended Harvard can be intuitively inferred from the noun *dropout*. However, drawing this inference automatically can be very difficult. Machine readable dictionaries, such as WORDNET (Miller, 1995), do contain more information about the meaning of the word *dropout*. The WORDNET entry for dropout is *someone who quits school before graduation*, but this leaves us with another subproblem. We have to draw the inference that the verb *quit* presupposes a prior phase of attending, which unfortunately cannot be extracted from WORDNET. This example just illustrates that many inferences that are intuitively rather easy are often hard to automatize, see (Harabagiu et al., 2001) for a discussion of more examples.

If it is not possible to establish an explicit link between a phrase of the appropriate type and the question, be it via the syntactic structure, pattern matching, or lexical chaining, then linear proximity is often used as a fallback strategy to link the phrase to the question. As a proximity restriction it is often required that the candidate answer phrase occurs in the same sentence as some of the query terms, or in the preceding or following sentence.

The document analysis component passes on the list of candidate answers to the answer selection component, together with the way in which each candidate answers was linked to the question, i.e., whether it was due to analyzing the syntactic structure, application of pattern matching, lexical chaining or proximity constraints.

1.1.5 Answer Selection

The final component selects the phrase that is most likely the answer to the original question from the candidate answer phrases coming from the document analysis component. Note that the selection component does not necessarily have to return a single final answer to the user, but it can also return a ranked list of answers, where answers are ordered with respect to the confidence the system has in each of them.

Similar to the other components, there is a variety of ways to select or rank candidate answers. The first criterion for preferring one candidate answer over another one, is the way they were identified as candidates by the document analysis component. If a candidate answer is linked to the question by applying a rather strict pattern or by its position in a parse tree or dependency graph, it is more likely to be a correct answer than a candidate answer that is linked to the question because it occurs in the proximity of words from the question. If lexical chaining is involved in establishing a link, the length of the chain and nature of its elements, whether it is an ISA relation, or part of a word definition, play a role in estimating the correctness of that candidate answer.

In addition to—or in combination with—the way in which the candidate answer is linked to the question, the frequency of a candidate answer can also be considered as a criterion for answer selection. The frequency of a candidate answer is the number of occurrences it was linked to the question. Using frequencies to select an answer is also known as *redundancy-based answer selection*, see, e.g., (Clarke et al., 2002a,b; Dumais et al., 2002). Counting these frequencies can be restricted to the set of documents that were considered in the document analysis component, but it can also be extended to a larger set. Some question answering systems use the whole document collection to count how often a candidate answer co-occurs with terms from the question. Other systems even go beyond the actual document collection and use the world wide web to get these frequencies, cf. (Magnini et al., 2002).

These approaches, considering the way the candidate answer is linked to the question, and the number of times it could be linked, or a combination of both, allow a system to rank the candidate answers. If a system is required to return a single final answer, the highest-ranked candidate answer is simply chosen.

If the document analysis component does not provide any candidate answers, or only candidates that are merely linked to the question by proximity and only linked with a low frequency, the answer selection component can decide to jump back to the question analysis component and try to reformulate the retrieval query by adding or deleting terms in order to get a different set of documents that are used to identify candidate answers, see (Harabagiu et al., 2001).

1.2 Question Answering at TREC

Since 1992, the annual Text REtrieval Conference (TREC)[1] organized by the National Institute of Standards and Technology (NIST) provides a forum for researchers to compare the effectiveness of their systems in information retrieval related tasks, such as document retrieval, document filtering, spoken document retrieval, video retrieval, cross-lingual retrieval, and question answering. For each of these subareas, called *tracks* in TREC terminology, NIST provides a document collection, which is used by all participants of that particular track. Shortly before the conference the participants get a set of information needs, called *topics* in TREC terminology. The topic in (1.7), is taken from the TREC-8 document retrieval track, and the topic in (1.8) is taken from the TREC 2002 question answering track.

(1.7)
```
<top>
<num> Number: 403
<title> osteoporosis
<desc> Description: Find information on the effects of the
dietary intakes of potassium, magnesium and fruits and
vegetables as determinants of bone mineral density in elderly
men and women thus preventing osteoporosis (bone decay).
<narr> Narrative: A relevant document may include one or more
of the dietary intakes in the prevention of osteoporosis. Any
discussion of the disturbance of nutrition and mineral metabolism
that results in a decrease in bone mass is also relevant.
</top>
```

(1.8)
```
<top>
<num> Number: 1397
<desc> Description:
What was the largest crowd to ever come see Michael Jordan?
</top>
```

After the topics have been released, each group has a limited amount of time (one week in the question answering track) to submit their results to NIST. If a group is participating in the document retrieval track, the result is a set of relevant documents, and if they are participating in the question answering track, the result is a set of answers.

Upon receipt of the results, they are manually inspected by NIST employed assessors, who judge whether a document is relevant for a given topic, or, in the case of the question answering track, whether the submitted answer is indeed correct. The resulting set of relevant documents or correct answers, is called the set of *qrels* or *judgments*.

[1] All TREC proceedings, guidelines, etc. are publicly available from `http://trec.nist.gov`.

Question answering has been part of TREC since TREC-8, held in 1999. Over the years, the question answering track has undergone a number of changes, which we will briefly discuss.

At TREC-8 (Voorhees and Tice, 2000a,b), the document collection consisted of approximately 528,000 newspaper and newswire articles, and 200 questions. The questions were fact-based, short answer questions, and were guaranteed to have at least one document in the collection that contains an answer to it. The questions were to a lesser extent taken from the FAQ FINDER (Burke et al., 1997) query logs, and mainly taken from questions that were manually created by TREC participants, the NIST TREC team, and the NIST assessors. For each question, participants returned a ranked list of five pairs of the form <document-id, answer-string>. The answer-string was limited to either 50 or 250 bytes (characters). The NIST assessors inspected each answer-string and decided whether it contained a correct answer in the context provided by the document. An answer was counted correct only if the document from which it was taken allows the assessor to draw the conclusion that it is indeed a correct answer. These answers are also referred to as *supported* or *justified* answers. If an answer string is identical to a supported answer, but the document it was extracted from does not support it, the answer is considered *unsupported* or *unjustified*. Individual questions received a score equal to the reciprocal of the rank at which the first correct answer was returned. If none of the five responses contained a correct answer, it was set to 0. The overall score of a system is computed as the mean reciprocal rank (MRR), which is the mean of the individual question scores.

At TREC-9 (Voorhees, 2000a), question answering was done against a larger document collection, consisting of all newspaper and newswire articles from previous TREC collections, resulting in a document collection containing approximately 978,000 articles. The other change with respect to TREC-8, was to use questions from actual users. To this end, questions were taken from the log of Microsoft's Encarta system (Encarta) and questions from the log of the EXCITE web search engine (Excite).

At TREC-10 (Voorhees, 2001b), or rather TREC 2001 as the TREC organizers have changed the naming convention,[2] two things were changed. First, the answer strings were limited to 50 bytes only. Second, the set of questions also included questions that were known not to have an answer in the document collection, and the correct answer to that question was to indicate that it does not have an answer. The questions at TREC 2001, were taken from the MSNSEARCH logs (MSN Search) and ASKJEEVES logs (Ask Jeeves).

At TREC 2002 (Voorhees, 2002), again a number of things were changed. First, the AQUAINT corpus, a new document collection was used.[3] The AQUAINT corpus contains approximately 1,033,000 newspaper and newswire articles, which cover more recent years than the document collections previously used.

[2]Throughout this thesis, the TREC 2001 conference will often be referred to as TREC-10, and the TREC 2002 conference will often be referred to as TREC-11.

[3]LDC: `http://www.ldc.upenn.edu/`.

A more substantial change in the question answering track was the requirement that answers now had to be exact answers instead of 50 byte text snippets. For instance, consider question (1.9). At TREC-8, TREC-9, and TREC 2001, both answers (1.10.a) and (1.10.b), would have been judged as correct answer, assuming that the document they were extracted from supports these answers.

(1.9) How far is it from Denver to Aspen? (topic id: 894)

(1.10)a. away as about 200 miles away. Now an estimated

b. 200 miles

At TREC 2002, however, only answer (1.10.b) would have been assessed as correct. Answers that also contain text that does not, strictly speaking, contribute to the answer were judged as *inexact*.

Another change at TREC 2002 was that participants were limited to return only one answer, as opposed to five for TREC-8, TREC-9, and TREC 2001.

Finally, at TREC 2002, participants were asked to return their answers ordered with respect to their system's confidence that this answer is correct. I.e., if the answer to question n has a higher confidence than the answer to question m, but the answer to question n is actually incorrect, the system's overall score will drop to a larger extent than compared to the situation where the answer to question m has had a higher confidence than the answer to question n.[4]

1.3 Research Questions

Question answering systems tend to be rather complex, having several modules, as we saw in section 1.1. Each of these components, and each of the techniques that they employ, has a certain impact on the overall performance of a question answering system. Even in a PhD thesis, it is very difficult to thoroughly investigate all aspects of a question answering system. Therefore, certain boundaries have to be set. In this thesis, we will focus on the retrieval component, and its effect on question answering.

As discussed above, the role of retrieval as a pre-fetch to question answering is to identify documents that are likely to contain an answer to a given question. Since the information needs in question answering are much different from the information needs in traditional document retrieval the question arises what retrieval techniques should be employed to optimize the performance of the retrieval component. This general issue can be subdivided into a number of more specific research questions:

[4] At the time of writing, the TREC 2003 question answering track is ongoing, and again a few things were changed. As we do not use the TREC 2003 data set throughout this thesis, we dispense with a further discussion of this track.

1. Do retrieval techniques that are known to perform well for document retrieval perform equally well when searching for documents that contain an answer to a question?

2. What can be gained from tailoring a document retrieval engine to the task of finding documents that contain an answer to a question?

3. To what extent does the retrieval component affect the overall performance of a question answering system?

In order to answer these questions in a general way, it is necessary to abstract from a particular question answering system. In this thesis, we will compare the effectiveness of retrieval systems purely on the basis of their ability to identify documents that contain an answer. Whether these answer-containing documents are documents that allow for easy extraction of the answer depends on the specifics of the document analysis and the answer selection modules. These components might prefer different documents containing an answer, than the ones delivered by the retrieval module, because the answer is expressed in such a way that it can be more easily detected by them. Taking these aspects into account would limit the generality of the conclusions that can be drawn, and it is questionable whether they can be applied to question answering systems different from the ones that were used in establishing these results.

1.4 Outline of the Thesis

Chapter 2. In this chapter we discuss some of the earlier approaches to question answering, ranging from philosophy to database theory. Looking at earlier approaches is not only of historical value, but also reveals general issues in question answering and ways in which these issues have been addressed over the years. The purpose of this chapter is to identify key issues in question answering by considering a number of previous approaches. These issues include the way in which the question answering process should be modeled and what the elementary analysis steps are, how the appropriateness of an answer can be defined, and how the analysis steps can be automatized.

Chapter 3. In this chapter, we compare the effectiveness of some common retrieval techniques with respect to their ability to find documents that contain an answer to a question. The techniques discussed in this chapter include morphological normalization, blind relevance feedback, and passage-based retrieval.

Chapter 4. This chapter introduces a new proximity-based retrieval method and applies it to question answering. This approach is a more flexible alternative to

passage-based retrieval and exhibits significant improvements over standard retrieval techniques as described in the previous chapter. In addition, the proximity-based method automatically identifies smaller text excerpts within a document that are likely to contain an answer to a question, thus reducing the amount of text which has to be processed by the answer selection component.

Chapter 5. When using a retrieval system as a pre-fetch for question answering, the question arises which words from the question should be used to formulate the query, as the performance strongly depends on the query used. By analyzing all possible combinations of query words, optimal queries can be identified. Query words are represented as a number of features, including part-of-speech tag, whether it occurs in question focus, type of question, etc. The M5′ regression tree learning algorithm is used to learn weights indicating the importance of a certain word for a given question.

Chapter 6. Although expanding queries with semantically related terms has not proved to be effective in ad hoc retrieval, certain question types can benefit from expanding the query with terms that are expected to be part of the answer. In particular, pre-fetch queries generated from questions that ask for measures such as height, age, distances, etc., can be expanded with a closed class of words expressing units such as miles, years, feet, stories, etc.

Chapter 7. This chapter considers how some of the retrieval approaches investigated in this thesis affect the way in which actual answers can be extracted from relevant text excerpts delivered by the retrieval systems. To this end we use a particular question answering system. As mentioned above, the conclusions that can be drawn from these experiments are less general than the conclusions formulated in the other chapters, but nevertheless it provides some useful insights in the interaction between the retrieval module and the other components for a concrete system.

Chapter 8. In the last chapter, we draw some overall conclusions for the key issues that are addressed in this thesis. We also formulate a number of remaining research questions.

Chapter 2

Theoretical and Practical Approaches to Question Answering

Question answering has a long tradition, involving many disciplines, ranging from philosophy to database theory. Depending on the discipline different aspects of the question answering process are investigated. Philosophical and psychological approaches focus more on theoretical aspects, whereas artificial intelligence and database approaches investigate how a practical question answering system can be engineered. Looking at earlier approaches to question answering is not only of historical value, but also reveals general issues in question answering and ways in which these issues have been addressed over the years.

Although question answering received a great deal of attention in recent years, from areas such as artificial intelligence, natural language processing, database theory, and information retrieval, the field itself is not new. Simmons (1965) already reviewed as many as 15 implemented and working systems for question answering. Psychological approaches to question answering date back to the 1930's, and philosophical discussions of issues involved in question answering can even be traced back to Aristotelian times.

Giving a comprehensive survey of previous work in the area of question answering is far beyond the purpose of this chapter, but we do want to review a selection of approaches so as to identify a number of central problems, and discuss the ways in which they have been dealt with in the different approaches.

The purpose of this chapter is to identify key issues in question answering by considering a number of previous approaches. These issues include the way the question answering process should be modeled and what the elementary analysis steps are, how the appropriateness of an answer can be defined, and how the anal-

ysis steps can be automatized.

In this chapter, three general perspectives on question answering are considered. In the first section, we have a brief look at philosophical approaches which aim to define a semantics for questions and the relationships that can hold between questions and answers. In section 2.2, a psychological model of symbolic question answering is discussed. Finally, in section 2.3, we review a number of working systems which have been developed throughout the past decades.

2.1 Formal Semantics of Question Answering

Compared to the other approaches that we will discuss in this chapter, philosophical approaches have a very different take on the subject of question answering. The prevalent difference is that philosophical approaches mainly focus on the semantics of questions, and their answerhood, i.e., the relationships that can hold between a question and an answer. For instance, an answer can be *correct*, *incomplete*, or *uninformative*. In contrast, practical approaches—and to some extent also psychological approaches—are mainly driven by the issue of *how* to get an answer to a given question. Nevertheless, in practical approaches the concept of *answerhood* plays an important role too: In order to maximize the effectiveness of a system, one has to consider whether an answer is, e.g., correct or uninformative. One could say that philosophical theories of question answering provide a formal specification of the post-conditions which have to be satisfied by answers that were generated by practical systems.

The discussion in this section is mainly based on (Harrah, 1984), but see also (Ginzburg, 1995) and (Groenendijk and Stokhof, 1997) for further surveys.

2.1.1 Hamblin's Postulates

In many philosophical approaches, the problem of defining a semantics for questions is reduced to defining the meaning of a question as the complete set of its answers. This approach is referred to as *set-of-answers reduction*, and is based on three postulates proposed by Hamblin (1958):

P1. An answer to a question is a statement.

P2. Knowing what counts as an answer is equivalent to knowing the question.

P3. The possible answers to a question are an exhaustive set of mutually exclusive possibilities.

Several linguists and philosophers have argued against adopting these postulates, as they are empirically questionable. For instance, P1 does not seem to cover cases where the answer is a noun phrase, or a simple *yes* or *no*, but it can be argued that

those answers are abbreviations of full sentences. Postulate P2, is even more controversial, and is also strictly dependent on P1. In Hamblin's approach, the meaning of a question is the set of all possible answers. Therefore, a yes-no question is represented as a set containing the assertion of the question and the negation of that assertion. For wh-questions, such as (2.2), all persons that could possibly be the Secretary General of the United Nations can be used to formulate answers. For example, the three assertions (2.2.a–c) do belong to the set of propositions that represent the meaning of question (2.2).

(2.1) Who is the Secretary General of the United Nations?

(2.2) a. Kofi Annan is the Secretary General of the United Nations.
b. Yasser Arafat is the Secretary General of the United Nations.
c. Magic Johnson is the Secretary General of the United Nations.

Hamblin's approach does not consider the actual context in which a question is asked, and therefore every imaginable context is used for formulating the propositions that count as possible answers.

Karttunen (1977) proposed a more restrictive approach, considering only those answer-propositions that are correct in the context in which the question is asked for representing the meaning of a question. I.e., in the current context (September 15, 2004), the meaning of question (2.2) is the singleton containing only proposition (2.2.a).

But even when adopting P1, and using Karttunen's more restrictive version of P2, it is easy to come up with counterexamples to P2.

(2.3) a. Who did John Hume share the 1998 Nobel Peace Prize with?
b. Who did David Trimble share the 1998 Nobel Peace Prize with?

(2.4) a. John Hume shared the 1998 Nobel Peace Prize with David Trimble.
b. David Trimble shared the 1998 Nobel Peace Prize with John Hume.

Because the relation *share* is symmetric in the examples (2.3) and (2.4), the answers (2.4.a) and (2.4.b) make the same statement, and both are correct answers to question (2.3.a) and to question (2.3.b), but the two questions are certainly not identical. Of course, it is legitimate to argue that in a particular context, by choosing (2.4.a) or (2.4.b) as an answer to (2.3.a) or (2.3.b), the *topic-focus* structure of the answer will change, and therefore the answers will be uttered with a different intonation. But incorporating *topic-focus* distinctions requires a more complex form of representation, which is beyond the scope of this discussion.

Although the idea that questions can be fully defined in terms of their answers might be too simplistic, it is interesting to see to what extent this idea is realized in current practical question answering systems. One way to analyze this is to look at semantically equivalent questions and check whether they are answered in the

same way. Such an analysis illustrates to what extent a practical system actually 'understands' a question.

The issue of semantically equivalent questions received some attention in the TREC-9 question answering track (Voorhees, 2000a). For 54 of the questions additional variants were generated by paraphrasing the original question. For each set of variant questions, the union of all answer documents was built, and for each document in the union, the documents that appeared as an answer document to each variant question were counted. One set of variant questions was discarded as none of the participating systems was able to return a correct answer to any of the variants. For the remaining 53 question sets, which contained 4.6 variants on average, only 38% of the documents appeared as an answer to each variant.

The rather modest overlap of answer documents indicates that many of the systems participating in TREC-9 were susceptible to changes of the surface structure requiring further research aimed at deeper understanding of questions.

2.1.2 Completeness and Distinctness

Belnap and Steel (1976) present a formal framework for representing questions that have quantificational constraints on possible answers. These representations are of the form $?(^{n}_{m}\ C\ D)\phi$, where:

- ϕ is a description constraining the entities that answer the question,
- m is the minimal number of entities satisfying the description,
- n is the maximal number of entities satisfying the description. n can also be left unspecified by setting it to $-$,
- C specifies completeness and can take the value $\forall$ if the answer has to be complete, i.e., it has to enumerate all entities satisfying the description, or C can be set to $-$ if this is not necessary,
- D specifies distinctness and can take the value $\neq$ if the entities satisfying the description have to be distinct, or $-$ if this is not necessary.

This allows one to distinguish between certain types of questions:

- Single-example questions (*Name a ...*): $?(^{1}_{1}\ -\ -)\phi$
- Some-examples questions (*Name some ...*): $?(^{-}_{1}\ -\ -)\phi$
- n-distinct-examples questions (*Name n different ...*): $?(^{n}_{n}\ -\ \neq)\phi$
- All-distinct-examples questions (*Name all different ...*): $?(^{-}_{1}\ \forall\ \neq)\phi$

This formalization covers simple identity questions, such as example (2.5), but has a stronger bearing on enumeration- or list-questions, such as example (2.6).

(2.5) What is the tallest building in New York City? : $?(^{1}_{1}\ \forall\ -)\phi$ (topic id: 280)

(2.6) Name 21 Godzilla movies. : $?(^{21}_{21}\ -\ \neq)\phi$ (topic id: 47)

Example (2.6) is taken from the TREC-11 question set of the question answering list track. Voorhees (2002) explains the evaluation criteria that were applied when assessing answers to list questions. Those criteria share many characteristics with Belnap and Steel's formalization. For instance, a correct list of answers should contain exactly as many items as the question is asking for, and all individual answers in the list should be pairwise distinct.

2.1.3 Informativeness

Even though a question can have a number of correct answers, this does not mean that all answers are equally appropriate. In particular, some answers can be more specific or contain more information than others.

Assuming an approach where the meaning of a question (formally expressed as $[\cdot]$) is expressed as a set-of-answers, one can easily define the notion of *informativeness*.

- ϕ is a more informative answer to $?\psi$ than ϕ' iff $\phi, \phi' \in [?\psi]$ and $\phi \Rightarrow \phi'$ and $\phi' \not\Rightarrow \phi$.

Where $\phi \Rightarrow \phi'$ means that ϕ logically implies ϕ'.

To see an example where two answers differ with respect to informativeness, consider question (2.7), from the TREC-11 data set.

(2.7) How many chromosomes does a human zygote have? (topic id: 1404)

(2.8) a. 46

b. 23 pairs

The answers (2.8.a) and (2.8.b) were automatically generated by two systems participating in TREC-11. Both answers were judged correct, but (2.8.b) is more informative as it carries the additional information that the chromosomes are organized in pairs.

Summing up, one can say that philosophical approaches to question answering formalize many of evaluation criteria that are applied to practical systems. Although such a formalization is unlikely to find its way into open-domain systems due to the difficulties of operationalizing many of the formal concepts for arbitrary data, it has been operationalized on more restricted domains, see, e.g., (Scha, 1983).

2.2 Psychological Modeling of Question Answering

There is a vast amount of literature on the psychology of human language understanding, leading to the erection of the discipline of psycholinguistics. Parts of these research activities are devoted to the understanding of mental procedures that humans execute when they answer questions. In particular, the work of Arthur Graesser and his colleagues covers a broad spectrum of the psychology of question answering. Their work has evolved over many years and is discussed in a series of publications. Here, we focus on one of their overview articles (Graesser and Murachver, 1985), which gives a comprehensive description of their approach. We chose the approach by Graesser and his colleagues, because it is one of the best-developed and most general approaches in the field of psychology.

Most psychological approaches to question answering are tied to the area of story understanding. A short story is given to a number of human subjects and after having read it they are asked to answer some questions about the content of the story. These question can range from simple fact questions to more complex questions asking for motivations or procedural aspects.

Graesser and Murachver (1985) sketch a general architecture for symbolic question answering, which is depicted in figure 2.1. The basic steps of the question answering process are:

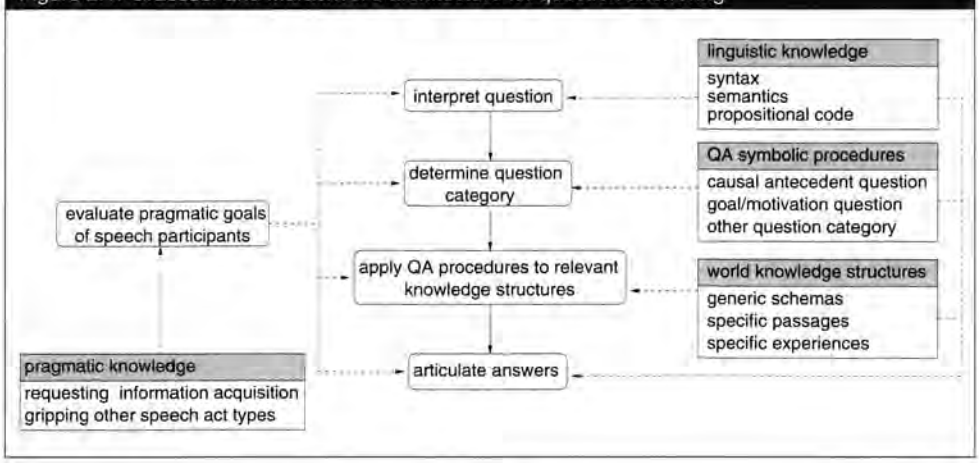
- interpret question,
- determine question category,
- apply QA procedures to relevant knowledge structures, and
- articulate answers.

Each of these steps consults additional knowledge bases, including linguistic and world knowledge. All steps are guided by pragmatic knowledge, covering extra-linguistic knowledge, such as certain idiosyncrasies and traditions, which can be decisive in answering questions appropriately. For example, it is absolutely inappropriate, though in principle correct, to reply to a question like *Do you have the time?* by simply saying *yes*. The issue of appropriateness is rather intricate and beyond the scope of this discussion, but see (Levinson, 1983; Searle, 1969) for further details.

2.2.1 Question Interpretation

The first step of the question answering process is to interpret the question. In the terminology of Graesser and Murachver (1985), this means that a question is represented in an underspecified way. Although they refer to this component as *question interpretation*, it is actually just a first step in understanding the question. In their question answering framework, the product of question interpretation is an expression with a *question function*, a *statement element*, and a *knowledge structure element*. For instance, consider question (2.9).

Figure 2.1: Graesser and Murachver's architecture for question answering



(2.9) Why did Reagan cut the education budget?

The outcome of the question interpretation would be:

$$\text{WHY}(< \text{Reagan cut the education budget} >, < ? >)$$

where WHY is the question function, < Reagan cut the education budget > is the statement element, and the knowledge structure element is left unspecified (i.e., < ? >) at the stage of question interpretation. The number of question functions is fixed, and any of the following six functions can be assigned to a question: WHY, HOW, CONS (i.e., what is the consequence of), WHEN, WHERE, and SIG (i.e., what is the significance of). Other question functions such as WHO or WHAT are not considered in the approach of Graesser and Murachver (1985).

The statement element can also contain more than one statement. Consider question (2.10).

(2.10) Why were the teachers depressed after Reagan cut the education budget?

The outcome of the question interpretation would be:

$$\text{WHY}(< \text{teachers were depressed} > < \text{Reagan cut the education budget} >, <?>)$$

If the statement element is more complex, the statement which is the *focus* of the question has to be determined. In question (2.10), the focus is < teachers were depressed >. Note that the temporal relationship *after* is neglected in the representation of the question.

2.2.2 Question Categorization

The next step in the question answering process is to select the question category. This is a further specification of the question representation that was generated by the question interpretation module. Categorizing a question enables later components to apply the appropriate knowledge extraction strategies to generate an answer. Graesser and Murachver (1985) use 21 question categories, which are based on Lehnert's classification (Lehnert, 1978). Since Lehnert's work is discussed in more detail later on, here we only mention the two categories which are relevant to cover questions (2.9) and (2.10):

- Goal orientation (Why did Reagan cut the education budget?)
- Causal antecedent (What caused the teachers to become depressed?)

One can see that simply using the question function, which is WHY for both questions, is not enough to distinguish between (2.9) and (2.10). Therefore, Graesser and Murachver (1985) also classify the focus statement of the statement element into three categories: *state, event,* and *action.* States are ongoing characteristics of an entity (*the teachers are depressed, the tree is tall*). Events are state changes (*the tree fell, the student passed the exam*). Actions involve an agent who does something to achieve a desired state or state change (*the student sold his car, Reagan cut the education budget*). The combination of the question function and the category of the focus statement form the final question category. For instance, question (2.9) is classified as WHY-action, whereas question (2.10) is classified as WHY-state.

2.2.3 Knowledge Structure Procedures

After the question has been interpreted and a category has been assigned to it, the third step in the question answering process is to apply question answering procedures to relevant knowledge structures . During this step, knowledge structures are accessed and manipulated to identify an answer to a question.

As presented in figure 2.1, there are three kinds of knowledge structures: *generic schemas, specific passages,* and *specific experiences.* The latter two are knowledge structures that were built by reading a certain passage or having a certain experience, respectively. In contrast to these specific structures, there are also general structures that are abstractions from specific structures. For instance, a PLAYING FOOTBALL schema includes actions such as *throwing the football, catching the football,* and *lining up,* and a COWBOY schema includes properties such as *wears a hat,* and *rides a horse.* World knowledge structures are represented as conceptual graphs. The nodes in the graph are categorized statement nodes that are connected by directed, relational arcs. The categories of the statement nodes include *goal, physical event,* and *internal state,* and the kinds of relationships between the nodes include *consequence, reason,* and *property.*

In order to answer a question, at least one knowledge structure must be accessed, but it is also possible to access multiple structures. Consider again question (2.9), it can be answered by using a generic schema, e.g., NATIONAL BUDGET, or a specific schema, e.g., a Washington Post article on the defense budget, which results from reading a particular newspaper article:

WHY-action(< Reagan cut the education budget >, < NATIONAL BUDGET >)

WHY-action(< Reagan cut the education budget >, < Washington Post article >)

Using the general NATIONAL BUDGET schema could result in an answer like *to save money*, whereas the more specific Washington Post article schema could result in an answer like *to free money for raising the defense budget*.

Once a schema has been chosen, the symbolic question answering procedures are called. Their execution consists of three stages:

- *Stage 1: Node matching.* Find an entry node in the knowledge structure that matches the queried statement node.
- *Stage 2: Arc search procedure.* Generate nodes that are connected via a path to the entry node.
- *Stage 3: Checking node constraints.* The candidate answer nodes are evaluated as to whether they satisfy specific node constraints.

The node matching stage may involve inexact matches. For instance, the NATIONAL BUDGET schema may not contain the node *Reagan cut the education budget*, but the node *president cut the education budget*. In that case, node matching would involve argument substitution, which is constrained by lexical and world knowledge.

Arc searching is guided by the question category. Depending on the category, only certain types of arcs are traversed, and the direction in which arc searching proceeds, i.e., using incoming arcs vs. outgoing arcs, also depends on it. Since question (2.9) is categorized as WHY-action, arc searching starts at the entry node following outgoing arcs that lead to a node stating that *Reagan cut the education budget*. In contrast, question (2.10) is categorized as WHY-state, and arc searching follows incoming arcs to identify a node that links to the entry node stating that *teachers are depressed*.

After arc searching has identified a number of candidate answer nodes, they are evaluated as to whether they satisfy certain constraints. In particular, the answer nodes have to be of the appropriate type. Questions of the category WHY-action seek for nodes of the type *goal*, and questions of the category WHY-state require answer nodes to be of the type *state*.

2.2.4 Answer Articulation

The final step in the question answering process is to articulate the answer. Answer articulation covers the ways in which the answer is expressed in natural language.

This process is of minor concern to Graesser and Murachver (1985), and only a few general aspects are outlined, such as the fact that answer articulation depends on the question category. For example, questions categorized as WHY-state, would require answers to use connectives like *because*, whereas questions categorized as WHY-action, would require connectives like *in order to* or *so that*.

2.2.5 Discussion

Graesser and Murachver (1985) present an intuitively appealing framework for modeling the mental aspects of question answering. But certain parts of the model are described in a rather abstract way, and it is questionable whether those parts can be concretized to a level where practical question answering systems can benefit from the overall framework. Especially the deployment of knowledge structures, which is essential for actually answering questions, seems to be the Achilles' heel of any potential implementation of their approach. It is not only unclear what the underlying heuristics are that have to constrain the arc searching procedures in large knowledge structures, or how partial node matching can be reliably accomplished, but most importantly, how these knowledge structures can be built in an automatic fashion, in the first place. It appears that they have to be generated automatically, given their specificity and number. Even for very restricted domains, manual construction seems to be too laborious a process and therefore doomed to be infeasible.

It seems also that knowledge structures cannot be used independently of each other. In the case of partial node matching other knowledge structures have to be consulted in order to decide whether a partial match is legitimate, see e.g., page 25 where *Reagan* is matched with *president*. This again makes the need for more controlled arc searching even more obvious.

Having pointed out the problems that any realization of the approach of Graesser and Murachver will face, it should be emphasized that many of their ideas can be found in current question answering systems. First of all, the four processing steps, and the order in which they are arranged, are respected by most systems. Question categorization has become a standard technique and all systems use it in one form or another. In most cases, the classification schemes bear a stronger resemblance to the schemes proposed by Lehnert (1978) or Graesser and Huber (1992). Most systems also focus on particular question categories and use more sophisticated subclassifications for those categories, but the choice for focusing on certain categories is mainly due to the type of questions that are considered by current evaluation fora such as the TREC question answering tracks.

The distinction between the focal statement element and the other statement elements is current practice in many QA system nowadays, to guide the answer extraction process.

Graesser and Huber (1992) address the problem of partial node matching only in passing. This problem still remains one of the harder challenges in QA. Their proposed solution to use lexical knowledge to resolve these cases is certainly right,

but unfortunately they do not provide a more concrete approach how this problem could be solved.

From all current QA systems, the work by Harabagiu and Moldovan (1997) is probably the one that comes closest to the model outlined above. Although they do not seem to be using generic schemas, WORDNET (Miller, 1995), a general lexical knowledge base, is exploited to identify more opaque conceptual relations when building a conceptual graph for a piece of text. Also, path-finding strategies are used to select nodes in the graph as candidate answer nodes.

2.3 Practical Approaches to Question Answering

A practical question answering system takes a question posed in natural language as input, accesses a knowledge base, and returns an answer, and all stages of this process are executed automatically, without any human intervention (except for posing the question, of course). The first practical QA systems date back to the late 1950's, and a large number of systems have been developed throughout the years. Especially the mid 1960's and early 1970's mark a very lively period in the area. In the 1980's many of the problems in practical QA became apparent and system development efforts were going back. It was not until the late 1990's that intensive research on practical QA was resumed.

In this section we will discuss a number of early QA systems, and their general ideas and shortcomings. The discussion of systems that were developed before the 1970's is based on Simmons (1965, 1969), and the interested reader is referred to his surveys for a more detailed discussion. Throughout this section, we distinguish between three types of systems: *data base-oriented*, *text based*, and *inference based*. For each type of system a number of implementations are discussed below.

2.3.1 Database-Oriented Systems

Database-oriented question answering systems use a traditional data base to store the facts which can be questioned. The data base can be queried by natural language questions which are translated into a data base language query, e.g., SQL. These types of systems are often referred to as front-end systems, because they do not address the problem of answer extraction, but leave this to standard data base techniques.

BASEBALL

The BASEBALL system (Green et al., 1963) answers English questions about the scores, teams, locations, and dates of baseball games. Input sentences have to be simple, and not contain sentential connectives, such as *and*, *or*, *because*, etc., or superlatives, such as *most* or *highest*. The data about baseball games are stored in a data base in attribute-value format:

```
month      :July          team1 :Red Sox
place      :Boston        team2 :Yankees
day        :7             score1:5
serial no.:96             score2:3
```

These database entries are manually constructed. The questions are transformed into the same format, but in an automatic way. First, questions are partially parsed to identify phrases. Using a dictionary, certain phrases are mapped to attribute-value pairs, e.g., *Yankees* is mapped to `team:Yankees`. Wh-words mark the attribute the questioner is interested in, e.g., *who* creates an entry of the form `team:?`, or *where* an entry of the form `place:?`.

After the attribute-value structure for the question has been built, the actual answer search takes place. In some cases, this simply requires matching the question structure with a structure in the database and substituting the blank item of the question structure with the corresponding value in the database structure. In other cases, when the wh-phrase is *how many* or the question contains quantifiers such as *every*, answer searching is a rather complicated searching and counting procedure.

The major shortcoming of the BASEBALL system, with respect to open-domain question answering, is the database, which was constructed manually, but it is not inconceivable that this process could be automatized, resulting in a more self-contained system. It also seems that this database-oriented approach is tied to specific domains, where the attribute-value structures can be uniform, and the types of questions are limited.

LUNAR

The LUNAR system (Woods, 1977) was developed at Bolt Beranek and Newman Inc. (BBN), to enable lunar geologists to conveniently access, compare, and evaluate the chemical analysis data on lunar rock and soil material, that was compiled during the Apollo moon missions. The system contains two data bases: a 13,000 entry table of chemical and age analyses of the Apollo 11 samples, and a keyphrase index to the entry table. The entries in the analysis table specify the concentration of some constituent in some phase of some sample, together with references to research articles, where these facts were established.

Natural language questions are analyzed automatically with a transition network parser (Woods, 1970), and translated into a data base query language. For the translation step, a dictionary is consulted which contains syntactic and morphological information about a word, and a number of partial data base query language constructions. For instance, question (2.11.a) will be translated as the query (2.11.b.).

(2.11)a. Does sample S10046 contain olivine?

b. `(TEST (CONTAIN S10046 OLIV))`

The `TEST` function results from recognizing the question as a yes/no question, the transitive verb *contain* triggers the two-place predicate `CONTAIN`, and the noun *olivine*

refers to the internal designator OLIV. Depending on the question, the resulting data base queries can reach a high level of complexity.

Another interesting feature of the LUNAR system is the ability to deal with sequences of questions, such as the sequences in (2.12) and (2.13).

(2.12)a. How many breccias contain olivine?

b. What are they?

(2.13)a. Do any samples have greater than 13 percent aluminum?

b. What are those samples?

Actually, questions (2.12.b) and (2.13.b) are easy to answer, once (2.12.a) and (2.13.a) have been answered, respectively, because in order to do so, all instances satisfying the first database query are retrieved, and the follow-up question is just a request to enumerate them.

The LUNAR system is one of the few early practical QA systems where at least some form of evaluation has been carried out. During a demonstration of a prototype, 111 questions were asked by geologists. 10% failed due to parsing errors, and 12% failed due to dictionary coding errors. After fixing the dictionary coding errors the system answered 90% of the remaining 78% of the questions correctly.

The LUNAR system shows that automatic question answering can be appealing to users in a real-world setting, albeit in a very restricted domain. In this project, immense efforts were put in constructing the dictionary, and transferring the system to a different domain appears to be a very laborious process—potentially even requiring a different database format.

PHLIQA1

The PHLIQA1 system (Bronnenberg et al., 1980; Scha, 1983) was developed at Philips Research Laboratories in Eindhoven, the Netherlands. It was designed to answer short questions against a data base containing fictitious data about computer installations in Europe and companies using them. Questions are translated into a formal language which is then used to access the data base. Translation is divided into three stages:

- *English-oriented Formal Language (EFL).* At this level, words are morphologically normalized, and content words are identified. No domain specific knowledge is applied at this stage.
- *World Model Language (WML).* The content words of a question are disambiguated and categorized using domain specific knowledge.
- *Data Base Language (DBL).* The content words of a question are mapped onto data base primitives.

The translation of a question into EFL uses standard NLP techniques, and is independent from the domain and the final data base format. Translating the EFL representation into WML requires a domain specific ontology which assigns categories to certain content words. E.g., *Intel* is recognized as company, *Belgium* as a country, *January* as a month, etc. Although the domain is largely determined by the data base, it does not coincide with it. For instance, *country* and *month* do not have any corresponding data base primitives. The final step is to translate WML representations into DBL representations. This step is independent from the language in which the original question was posed, i.e., it could have been English, Dutch, or French, but it is strictly dependent on the data base format. Figure 2.2 displays the three levels of representing the question *Is each cpu in Eindhoven a P1400?*.

Figure 2.2: Question translation steps of Phliqa1

Question: Is each cpu in Eindhoven a P1400?

EFL:

```
(forall:(head:CPUS,
         mod:(λc:IN(<c,EINDHOVEN>))),
 holds:(λx_S:(forsome:P1400S,
              holds:(λx_O:BE(<x_S,x_O>)))))
```

WML:

```
(forall:(head:GS_cpu,
         mod:(λc:F-SITE-CITY(F-CONF-SITE(F-CPU-CONF(c)))=EINDHOVEN)),
 holds:(λx_S:F-CPU-CPUMODEL(x_S)=P1400))
```

DBL:

```
(forall:(head:GS_conf,
         mod:(λe_1:F-SITE-CITYNAME(F-CONF-SITE(e_1))=''EINDHOVEN''
                   ∧F-COUNTRY-NAME(F-SITE-COUNTRY(F-CONF-SITE(e_1)))
                    =''NETHERLANDS'')
 holds:(λz_0:F-CPUMODEL-NAME(F-CONF-CPUMODEL(z_0))=''P1400'')))
```

On all three levels, the question is represented as a universal quantification. In the EFL representation, the variables `c`, x_S, and x_O are still untyped, i.e., uncategorized. The `mod`-field and the `holds`-field take sets as values, which are built by lambda abstraction, where $\lambda x \phi(x)$ is the set of instantiations `d` of `x` such that substituting `d` for `x` in ϕ results in a true statement. In the WML representation, the variables `c` and x_S are categorized as `cpu`. The relations `IN` and `BE` are further specified by a series of functions. At the final DBL representation level, the variables e_1 and z_0 are of type `conf` (configuration). This type mapping is necessary as there are no `cpu` primitives in the data base, but there are `conf` primitives. The DBL representation is then used to actually query the data base.

The PHLIQA1 system and the LUNAR system share a number of features. The most striking similarity is the use of the lambda calculus. The LUNAR system is implemented in LISP, a functional programming language, which has the lambda calculus as its underlying paradigm. An appealing aspect of the PHLIQA1 system is the division of the translation from natural language questions to data base queries into three stages. This modularization should enable one to transfer the system to different data bases or different languages more easily. Another feature, which is nowadays a standard component of question answering systems, is to classify certain content words or phrases along a (domain dependent) ontology. Almost all current QA systems perform a similar step by applying a named entity recognizer, e.g., IDENTIFINDER (Bikel et al., 1999), to the questions and the data to identify persons, company names, locations etc.

2.3.2 Text-Based Systems

Text-based systems do not assume the data to be pre-formatted. The data used to answer questions is plain machine readable text. Text-based systems have to analyze both, the question as well as the data, to find an appropriate answer in the text corpus.

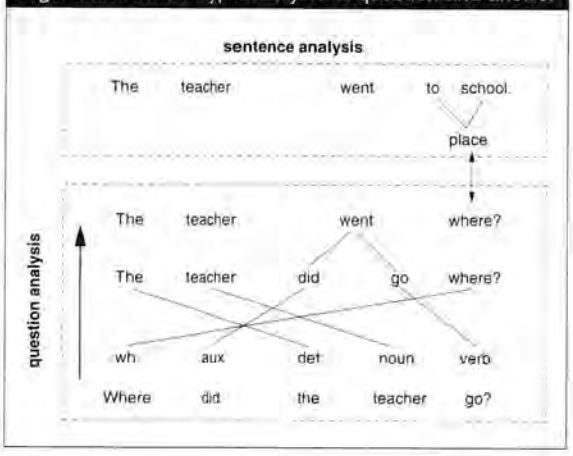
ORACLE

The ORACLE system (Phillips, 1960) produces a syntactic analysis of both the question and a text corpus which may contain an answer. This analysis transforms the question and the sentences in the corpus into a canonical form, marking the subject, object, verb, and time and place indicators. The analysis is limited to simple sentences and completely fails if sentences are more complex, e.g., contain more than two objects.

The first processing step is to assign to each word a part-of-speech tag, which is looked up in a small dictionary. Words such as *school*, *park*, and *morning*, etc. receive an additional tag marking them as time or place indicators. Whereas the analysis of the text corpus can be done offline, the question has to be analyzed at query time. The question is marked up analogously to the sentences in the corpus, but in addition, it is also transformed into a declarative sentence, which involves reordering of the words and combining auxiliary verbs with their head verbs. An example analysis including the transformation step is shown in figure 2.3.

The ORACLE systems exemplifies what is probably the simplest form of automated question answering. Its major shortcoming is the restriction to simple sentences. Also, the way a (transformed) question is compared to a potential answer sentence, by simply comparing the identical words and the word order will miss many answers, in case the sentence and the question use different words which are semantically equivalent. Nevertheless, ORACLE-type systems currently witness a renascence, because they offer a simple and yet effective approach, if the text corpus

Figure 2.3: Oracle-type analysis of question and answer



is very large. Many current QA systems use the internet as a text corpus for finding sentences containing an answer, see e.g., Attardi et al. (2002); Brill et al. (2001); Buchholz and Daelemans (2001); Lin et al. (2002); Xu et al. (2002). If the corpus is large enough, the sheer amount of data will increase the probability that there is an answer sentence in the same wording as the question, and therefore outbalance the need to perform a more sophisticated semantic and syntactic analysis.

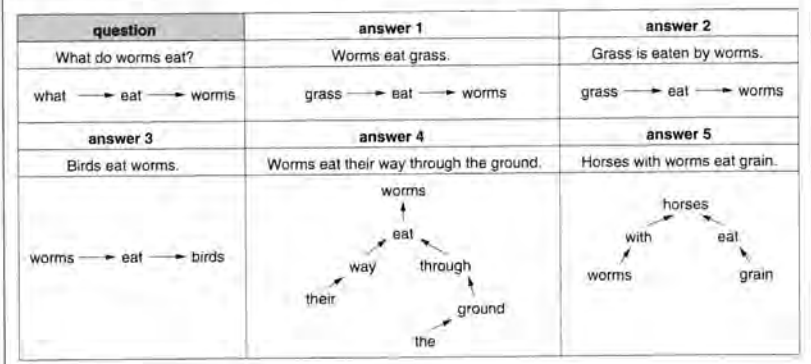
PROTOSYNTHEX

The PROTOSYNTHEX system (Simmons et al., 1963) attempts to answer questions from an encyclopedia. The encyclopedia is indexed and by using a simple scoring function, sentences or paragraphs resembling the question are retrieved. The words in the index are the stemmed forms of the words in the encyclopedia, i.e., *govern*, *governor*, *government*, *governing*, etc., are reduced to *govern*. Before retrieval, a lexical look-up expands the questions with words of related meaning.

The question and the text are then parsed using a modification of the dependency logic developed by Hays (1962). Figure 2.4 shows the dependency graphs for the question *What do worms eat?* and some potential answers. Although all potential answers contain the words *worms* and *eat*, only the answers whose dependency graph can be matched onto the graph of the question are kept. In figure 2.4, answer 1 and 2 have a complete agreement of dependencies, 4 and 5 agree partially, and 3 has no agreement. Comparing the degree of agreement, *grass* would be ranked highest, followed by *their way* and *through the ground*, and the lowest rank would be assigned to *grain*.

PROTOSYNTHEX also has a learning component, in which dependency parses are

Figure 2.4: Dependency structures of a question and potential answers



corrected by a human operator. This helps the system to deal with syntactic ambiguities in natural language. The most distinguishing feature of the PROTOSYNTHEX system is the use of dependency graphs. These graphs have the advantage that they are less susceptible to syntactic subtleties and allow for more flexible matching between question and potential answer representations. For the very same reason, dependency parsers are frequently used in modern QA systems, see e.g., Attardi et al. (2001); Harabagiu et al. (2001); Katz et al. (2001).

AUTOMATIC LANGUAGE ANALYZER

The AUTOMATIC LANGUAGE ANALYZER (ALA) system (Thorne, 1962) translates questions and sentences into an intermediate language which is strongly related to dependency graphs. Both are also augmented with semantic codes from Roget's Thesaurus (Roget, 1946).[1] The degree of matching between a question and a piece of text is used to select the best answers. Each word in a sentence or question is assigned a weight indicating whether it is the subject of a sentence, the verb or a modifier. If a sentence and the question share the same subject this adds more to the similarity score than if the subject and the object are identical. In addition, more opaque semantic relations like semantic correlation between words are also considered:[2]

$$\text{semantic correlation} = \frac{n_{ab}}{\sqrt{n_a \cdot n_b}}$$

[1]The 1911 version of Roget's thesaurus is freely available in machine-readable format at `http://www.ibiblio.org/gutenberg/titles/roget_s_thesaurus.html`.

[2]Semantic correlation bears a strong resemblance with the notion of mutual information in Fano (1961), which is defined as $I(a,b) = \log_2(P(a,b)/P(a)P(b))$.

where n_a and n_b are the number of thesaurus clusters in which word a and word b occur, respectively, and n_{ab} is the number of clusters in which both occur.

The AUTOMATIC LANGUAGE ANALYZER (ALA) system tries to exploit lexical knowledge, and computing the semantic correlation adds numerical weights to the links between words in a thesaurus, whereas most thesauri simply contain purely binary information saying that two words are similar or not.

WENDLANDT & DRISCOLL

Wendlandt and Driscoll (1991) describe a system which answers questions about the NASA Space Shuttle, using NASA plain text documentation maintained by the public affairs department.

In the first stage of the system, document retrieval is applied to identify a number of paragraphs that contain the content words mentioned in the question. The paragraphs are ranked with respect to a similarity weight function. To identify a paragraph that actually contains an answer, the top ten n are further analyzed. This analysis focuses on recognizing thematic roles and attributes occurring in the question and the top n paragraphs.

Wendlandt and Driscoll's repertoire of thematic roles is based on Fillmore (1968), containing roles such as *agent* (the thing which causes an action to happen), *object* (the thing affected by an action), *instrument* (the thing with which an action is performed), and *location* (where an action occurs). Attributes are abstract categories for certain words or phrases, including *heat*, *amount*, *size*, *order*, etc. For both thematic roles and attributes a dictionary of trigger words was manually constructed, where a word can trigger several roles or attributes. Table 2.1 lists a few trigger words and the corresponding roles and attributes. In order to reduce ambiguity, each trigger word is adorned with a probability distribution indicating the probability of a thematic role or attribute, given that trigger word. The total number of thematic roles and attributes is approximately 60.

Table 2.1: Trigger words of Wendlandt & Driscoll

Trigger word	Corresponding thematic roles and attributes
area	location
carry	location
dimensions	size
in	destination, instrument, location, manner, purpose
into	location, destination
on	location, time
of	amount
to	location, destination, purpose

After the thematic roles and attributes in the question and the top n paragraphs have been recognized, the paragraphs are reordered by computing a similarity score

based on the common roles and attributes.

Wendlandt and Driscoll (1991) evaluated their system using 21 questions. If just paragraph retrieval based on content words is applied, 60 paragraphs have to be considered in order to find a document containing an answer, whereas reordering the paragraphs by considering thematic roles and attributes, this number drops to 38, which is a decrease of approximately 37%.

The idea of using thematic roles and attributes for retrieving relevant paragraphs is closely related to the predictive annotation approach proposed by Prager et al. (2000), where the pre-fetching process uses similar categories to identify documents that contain phrases of the same type the question is asking for.

MURAX

The MURAX system (Kupiec, 1993) answers general fact questions using an online version of Grolier's Academic American Encyclopedia (Grolier, 1990), containing approximately 27,000 articles. The encyclopedia is accessed via an information retrieval system to select articles which contain an answer. The returned articles are analyzed further to identify answer candidates.

The question categories used by MURAX are shown in table 2.2. Kupiec (1993) focuses on these question types, because they are likely to allow for short answers which can be expressed in a noun phrase, whereas why- or how-questions require a more elaborate answer.

Table 2.2: Question categories in Murax

Question type	Answer type
Who/Whose	Person
What/Which	Thing, Person, Location
Where	Location
When	Time
How many	Number

Questions are linguistically analyzed by a part-of-speech tagger and a lexico-syntactic pattern matcher. Noun phrases are identified simply by using patterns which are defined in terms of part-of-speech tags. More specific phrases are identified by also considering lexical information. Simple noun phrases (NPs) and main verbs are first extracted from the question, as illustrated in question (2.14).

(2.14) Who was the [$_{NP}$Pulitzer Prize]-winning [$_{NP}$novelist] that [$_{V}$ran] for [$_{NP}$mayor] of [$_{NP}$New York City]?

The phrases are used in constructing the boolean query which is used to retrieve articles from the encyclopedia which are likely to contain an answer to the question. From the retrieved articles, sentences are selected which contain many of the query

terms. The noun phrases in these sentences which do not occur in the question are considered answer hypotheses.

For each answer hypothesis, MURAX tries to verify whether it is an argument of the relations stated in the question, e.g., an answer to question (2.14) has to be a novelist, and has to be someone who ran for mayor in New York City. To establish those relations, pattern matching based procedures are applied. However, this information does not necessarily have to be expressed in the same document from which the answer hypothesis has been taken, but might be contained in a different encyclopedia article. To this end, secondary queries are formed, containing the answer hypothesis plus words from the relation one would like to establish. The retrieval procedure for secondary queries is analogous to the retrieval procedure outlined above, and pattern matching is also applied to sentences from secondary document matches.

One of the major problems the MURAX system addresses is formulation of the retrieval queries, which is essential for returning documents that indeed contain an answer. MURAX uses a boolean retrieval system, which is especially sensitive to query formulation, because boolean systems have the tendency to return either too many or too few documents, cf. Baeza-Yates and Ribeiro-Neto (1999). As a solution, Kupiec proposes not to rely on a single retrieval run, but to wrap the retrieval and answer selection process in an outer loop. The loop stops if answer selection was successful, and otherwise it modifies the retrieval query appropriately. If a number of query formulations have been tried, but answer selection is still unsuccessful, the loop terminates.

Harabagiu et al. (2001) showed that this technique of query (re-)formulation and validation of the answer selection step, which they refer to as *feedback loops*, can be very effective in a corpus-based QA system.

2.3.3 Inference-Based Systems

Similar to data based-oriented systems, most inference-based systems require the data to be pre-formated. Although this is not an essential requirement, it eases the process of inference drawing. The focus of inference-based systems is to infer relationships that are not explicitly stated between entries in the knowledge base on the one hand, and the question and the knowledge base on the other hand.

SPECIFIC QUESTION ANSWERER

The SPECIFIC QUESTION ANSWERER (SQA) was developed at Bolt Beranek and Newman Inc. (BBN), to find short answers to simple natural language questions (Black, 1964). The system can only extract brief specific answers that are either directly stated in a corpus, or can be deduced by applying certain inference rules. The corpus consists of a number of inference rules and declarative statements. The problem of syntactic analysis of the questions is only rudimentarily addressed and

in most cases, exact structural matching between the question and possible answers in the corpus is required.

The inference rules in the corpus are conditional statements of the form *If A then B*, and the declarative statements are simple sentences such as *Mercury is a planet.* Technically, declarative statements are the consequents of a conditional statement with an antecedent which is always true. Typically, a corpus can consist of entries such as the following:

(1) Mercury is next smaller than Pluto.
(2) Pluto is next smaller than Mars.
(3) Mars is next smaller than Venus.
(4) If X is next smaller than Y, then X is smaller than Y.
(5) If X is next smaller than Y and Y is smaller than Z, then X is smaller than Z.
⋮

Inferences are done by matching the consequent of a conditional, and instantiating the variables appropriately. The next reasoning task is to check whether the antecedent(s) of the conditional can be satisfied in the corpus.

For example, the question *What is next smaller than Pluto?*, can be trivially answered by using declarative (1). If the question were *Pluto is smaller than what?*, it can match the consequents of (4) and (5), generating two inference chains. The first chain is successfully terminated by (2), and the second chain is successfully terminated by matching (2), then (4), and finally (5).

The way inference is realized in the SPECIFIC QUESTION ANSWERER is basically the inference mechanism of the logic programming language PROLOG (Colmerauer, 1978). Although it is appealing, it is quite questionable to what extent such an approach can be integrated into any realistic setting.

SEMANTIC INFORMATION RETRIEVER

The SEMANTIC INFORMATION RETRIEVER (SIR) system (Raphael, 1964) implements a limited formalization of the relational calculus. Similar to the SPECIFIC QUESTION ANSWERER, this system also avoids the complexities of syntactic analysis, by limiting itself to 20 fixed simple sentence formats for both, questions and declaratives. For these fixed formats logical translation procedures are provided. E.g., *every boy is a person* will be translated into `SETR(boy,person)`, meaning that `boy` is a subset of `person`. If a sentence or question does not fit any of the 20 formats, further analysis is terminated. Figure 2.5 shows some example inputs and their corresponding data structures.

Given a number of input statements, and the question *How many fingers are on John?*, it can be deduced that a finger is part of a hand, and that any person (including John) has two hands. Since the information how many fingers are on a hand is not specified, a computer response asks for this additional information. Once the additional

Figure 2.5: SIR examples session	
Input statements	Formalization
Every boy is a person.	`SETR(boy,person)`
John is a boy.	`SETR(john,boy)`
Any person has two hands.	`PARTRN(hand,person,2)`
A finger is part of a hand.	`PARTR(finger,hand)`
Question	
How many fingers are on John?	`PARTRN(finger,john)`
Computer response	
How many fingers per hand?	
Input statement	
Every hand has five fingers.	`PARTRN(finger,hand,5)`
Answer	
The answer is 10.	

statement has been added, it can be inferred that the answer is 10. The SEMANTIC INFORMATION RETRIEVER (SIR) system uses a limited number of predicates, such as (numerical) part-of, subset, is-a, etc., and inference rules are provided for those predicates, similar to the inference rules of the SPECIFIC QUESTION ANSWERER system.

One question that comes up immediately is whether the set of relations can be expanded to a level where the system can also answer questions beyond simple toy examples. In the previous decade, WORDNET (Miller, 1995) has established itself as a very prominent knowledge base, encoding some of the aforementioned relations for a larger vocabulary. But direct reasoning on WORDNET has been shown to be far from trivial, due to problems such as lexical ambiguity, and the inevitable incompleteness of the knowledge base. It seems that a purely inference-based approach to question answering will always be restricted to very specific domains, where lexical ambiguity is less prominent and manual construction of the knowledge base can reach a satisfactory level of completeness.

QUALM

The QUESTION ANSWERING LANGUAGE MECHANISM (QUALM) system (Lehnert, 1978, 1981) is hard to classify, as its purpose is twofold: (i) to provide a psychological model of question answering and (ii) to implement a computer simulation of the model. While (i) suggests to classify QUALM as a psychological approach to question answering, we decided to discuss it in the context of practical systems, because it illustrates the challenges of symbolic question answering systems.

QUALM was implemented as a language-independent question answering module, which can be integrated into other natural language processing applications. It is not a stand-alone system. Lehnert (1978) mentions four NLP systems which make

use of QUALM for story understanding: SAM, PAM, ASP, which were developed at Yale University, and COIL, which was developed at Xerox Palo Alto Research Center.

Before QUALM starts processing a question it requires that the question has been parsed, and a conceptual graph has been built. Analogously, the information against which a question is to be answered has to be represented as a conceptual graph. The conceptual graphs underlying QUALM conform to Schank (1975).

Four stages of the question answering process are distinguished:

- Conceptual categorization,
- Inferential analysis,
- Content specification, and
- Answer retrieval heuristics.

During conceptual categorization, the conceptual graph representation of a question is assigned a question category. The thirteen categories used in Lehnert (1978) are listed in table 2.3. Category assignment is accomplished by a number of case distinction rules on the conceptual graph.

Table 2.3: Qualm question categories

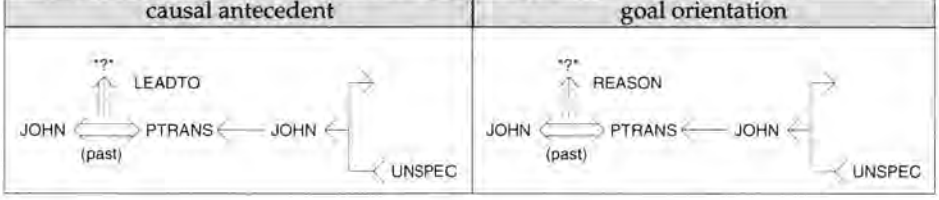
Question type	Example question
causal antecedent	How did the glass break?
goal orientation	Mary left for what reason?
enablement	How was John able to eat?
causal consequent	What happened after John left?
verification	Did John leave?
disjunctive	Was John or Mary here?
instrumental/procedural	How did John go to New York?
concept completion	What did John eat?
expectational	Why didn't John go to New York?
judgmental	What should John do to keep Mary from leaving?
quantification	How many dogs does John have?
feature specification	How old is John?
request	Will you take out the garbage?

Some questions cannot be assigned a unique category, e.g., *Why did John leave*, can be interpreted as a question asking for the causal antecedent, i.e., *What caused John to leave?*, or the goal orientation, i.e., *For what purpose did John leave?*. This difference will be expressed in the conceptual graph representation of the question, see figure 2.6.

After question categorization, further inferential analysis can impose additional constraints on potential answers. Inferential analysis considers the context of a question as well as certain pragmatic principles. This additional step is required to rule out question answer pairs such as example (2.15).

Figure 2.6: Qualm conceptual graph representations for question

Representation of the question *Why did John leave?* as a causal antecedent question and as a goal orientation question. PTRANS indicates a transfer of physical location. The source and the target of the movement are not specified.



(2.15)a. What haven't I added? (before baking a cake)

b. A pound of dog hair and an oil filter.

The content specification step assesses the way questions should be answered in order to satisfy the questioner. The criteria include the level of elaboration and the level of reliability. Given a low level of reliability, the system will return an answer, even if it is not sure whether this is a correct answer, whereas in case of a high level of reliability, the system will not respond with saying that it was unable to find an answer. Note that the issue of a system's confidence in a particular answer recently became part of the TREC evaluation criteria, see Voorhees (2002).

The final step in QUALM's question answering procedure is to search the internal representation of a story for an answer. There are three levels of story representation in which answers may be found:

- the causal chain representation,
- script structures, and
- planning structures.

When searching the causal chain representation, a matching procedure looks for a conceptualization having everything that the question concept has, and perhaps having additional material not found in the question concept. But many questions can only be answered appropriately when considering the context. Script structures represent the different sub events of a story. Each sub event is represented by a conceptual graph, and some of the graphs are connected to each other representing the way the story evolves. For instance, consider a story about John taking the bus to New York, visiting some friends, and returning back home. The graph representing the question *Why did John go to New York?*, will match the description of John's bus tour to New York. Since the question is a goal orientation question, the answer searching procedure continues along the destination path of the script structure, returning *to visit some friends* as an answer.

Planning structures are consulted if the answer cannot be extracted from the script structures. A plan explicates why a person does certain things, and how he or she normally tries to achieve some goal. Plans cannot be built from the story itself but have to be taken from some world knowledge module.

Unfortunately, no brief overview of the QUALM system, does justice to such a complex system, but nevertheless a few conclusions can be drawn. Lehnert (1978) describes the intricacies of question answering in a detailed way that is—to the best of our knowledge—unparalleled in the literature.

Abundant scenarios are discussed to illustrate the many facets that play a role in answering a question appropriately. Lehnert argues convincingly that conceptual graphs provide a framework that is general enough to capture many of these aspects. Although representational issues and general principles of finding answers are discussed in great length, Lehnert falls short in providing concrete algorithms that accomplish this task. It is also not clear how the system behaves with respect to unseen data, i.e., short stories or questions it has not been fine-tuned for. Lehnert (1994, page 151) explains that "question answering heuristics had not been tested on questions other than the ones that were presented" and that the system "was carefully engineered to handle the input it was designed to handle and produce the output it was designed to produce."

To summarize, Lehnert (1978) identifies and addresses many important problems, but in order to evaluate their effectiveness much more work has to be put in testing them against a larger data set.

2.4 Discussion

In this chapter we have discussed a variety of approaches to question answering, ranging from philosophical, over psychological to practical approaches. Philosophical approaches are substantially different from the latter two, because they do not touch on the issue of *how* to find an answer to a question at all, but focus on formalizing the semantics of a question and on the relationships that can hold between an answer and a question. Although, the philosophical findings might not seem to be pertinent to practical question answering at first glance, they do provide insights into formalizing the appropriateness of answers.

Contrary to philosophical approaches, psychological approaches do investigate the process of finding answers. Their main objective is to provide a model that approximates the way human beings answer a question. Graesser and Murachver (1985) present an intuitively appealing architecture capturing essential aspects of human question answering. Although their model was built to explain the cognitive processes that are involved, it also bears a strong resemblance to underlying architectures of many practical question answering systems.

From the myriad of practical question answering systems that have been implemented over the last decades, we could only discuss a small number, but we hope to have captured some of the most prevalent features present in many implemented

systems. Despite the diversity among the systems, a number of commonalities crystalize. Below, some of the techniques that are used by several systems, including modern QA systems are listed:

Information retrieval. Most of the text-based systems (ORACLE, PROTOSYNTHEX, and MURAX) use information retrieval techniques to identify text units that are likely to contain an answer, before answer selection methods are applied.
Question categorization. Although question categorization has become a standard technique present in many current systems, only two of the systems discussed above (QUALM and MURAX) classify questions to guide the answer selection process.
Parsing. Some systems (PROTOSYNTHEX and ALA) use a dependency parser to analyze the question and potential answer sentences to select and answer. The MURAX system uses a partial parser for formulating the query sent to the retrieval system.
Thematic roles. The WENDLANDT&DRISCOLL system is the only system which explicitly uses thematic relations and is based on Fillmore's (1968) linguistic theory. The ALA system also uses the syntactic function of a word, e.g., whether it is the subject of a sentence, which often indicates a certain thematic role, but does carry out any proper role identification.
Named entity recognition. In order to decide whether a candidate answer is of the appropriate type, the PHLIQA1, ORACLE, and WENDLANDT&DRISCOLL system recognize and classify named entities, where the classification can be rather domain dependent as exemplified by the PHLIQA1 system.
Surface matching. The ORACLE system uses syntactic transformation rules to match the transformed question string to sentences that potentially contain an answer. In the days the system was developed this approach turned out to be too simplistic, but it recently received again some attention due to the availability of large amounts of data, such as the Internet.
Taxonomic reasoning. The SIR system uses isa and part-of relations of a taxonomic hierarchy to deduce the answers to a question. Many current QA system use WORDNET to carry out similar reasoning steps, but it has also become evident that existing knowledge bases have to be automatically extendable in one way or another to be useful for systems which are not restricted to a specific domain.

Many of these techniques are present in modern QA systems, and the quality of each of these techniques obviously affects the overall performance of a QA system. However, there is barely any systematic evaluation at this point indicating the impact of a particular technique.

Chapter 3

Document Retrieval as Pre-Fetching

Current question answering systems rely on document retrieval as a means of providing documents which are likely to contain an answer to a user's question. A question answering system heavily depends on the effectiveness of a retrieval system: If a retrieval system fails to find any relevant documents for a question, further processing steps to extract an answer will inevitably fail as well. In this chapter, we compare the effectiveness of some common retrieval techniques with respect to their usefulness for question answering.

Document retrieval systems aim to return relevant documents to a user's query, where the query is a set of keywords. A document is considered relevant if its content is related to the query. Question answering systems, on the other hand, aim to return an answer to a question.

Since question answering systems are generally rather complex, consisting of several modules, including natural language processing (part-of-speech tagging, parsing), document retrieval, and answer selection, disentangling some components and evaluating them separately can help to gain a better insight in the way the performance of one component affects the others. In this chapter and the remainder of this thesis, we will focus on the retrieval component and its effect on answer selection.

Most, if not all, current question answering systems first use a document retrieval system to identify documents that are likely to contain an answer to the question posed, see, e.g., (Hovy et al., 2000; Kwok et al., 2001b; Burger et al., 2002; Na et al., 2002). This pre-processing step, also referred to as *pre-fetching*, is mainly motivated by feasibility considerations. Question answering requires a deeper analysis

of the documents, e.g., syntactic parsing, synonym linking, pattern matching, etc. It is impossible to do this for a complete collection of documents of reasonable size in an efficient manner. Therefore document retrieval is used to restrict the whole collection to a subset of documents which are probable to contain an answer, and then the actual process of answer selection is carried out on this subset.

The information needs for ad hoc retrieval on the one hand and document retrieval as a pre-fetch for question answering on the other hand are quite different, viz. finding documents that are on the same topic as a query and documents that actually contain an answer to a question. The question is whether techniques that have proved to be effective for ad hoc document retrieval are equally effective for retrieval as pre-fetching for QA. More specifically, what retrieval techniques should be used (e.g., boolean vs. vector space), should morphological normalization, such as stemming, be applied, is passage-based retrieval more effective than retrieval with full documents?

The importance of these questions lies in the strong impact of the effectiveness of a document retrieval system on the overall performance of the answer selection module: If a retrieval system does not find any relevant documents for a question, even a perfect answer selection module will not be able to return a correct answer. The PRISE retrieval system (Prise) was used by NIST (for TREC-10 and TREC-11) to provide participants in the QA track with potentially relevant documents, in case a participating group did not have a retrieval system. For example, using a cut-off of 20, which is in the vicinity of the cut-offs used by many participants in TREC QA tracks, PRISE failed to return any relevant documents for 28% of the questions of the TREC-11 data set. This affected not only questions which can be considered difficult by the current state of the art in QA, or questions which did not have an answer in the collection, but also relatively 'easy' questions such as (3.1) and (3.2).[1]

(3.1) What year did South Dakota become a state? (topic id: 1467)

(3.2) When was Lyndon B. Johnson born? (topic id: 1473)

Our objective is to investigate what retrieval techniques enhance document retrieval when used as a pre-fetch for QA. This includes the comparison of existing techniques in the current chapter, but also the introduction of a new retrieval approach in the next chapter.

The remainder of this chapter is organized as follows: The next section reviews some earlier work on document retrieval as a pre-fetch for QA. Section 3.2 explains the test data and retrieval techniques that are investigated. Also some issues related to evaluation are discussed. Section 3.3 presents the results of the experiments. Finally, section 3.4 gives some conclusions and an outlook on future work.

[1] Here, *easy* means that many participants of the QA track were able to return a correct answer.

3.1 Related Work

To the best of our knowledge, there is little systematic evaluation of document retrieval as pre-fetching for question answering. This is somewhat surprising considering the number of QA systems employing document retrieval in one form or another. The earliest work focusing on this issue is (Llopis et al., 2002), where the impact of passage-based retrieval vs. full document retrieval as pre-fetching is investigated.

Roberts (2002) also compared passage-based retrieval to full-document retrieval as a pre-fetch for question answering. In addition, he evaluated the impact of passage length on the overall performance of the University of Sheffield question answering system (Scott and Gaizauskas, 2000). He reports a slight increase in documents that contain an answer (+2.8%) when using two-paragraph passsages instead of full-document retrieval.

Tellex (2003); Tellex et al. (2003) compare the impact of several passage-based retrieval strategies that were used by TREC participants. The different approaches are compared with respect to the overall performance of a version of the MIT question answering system (Tellex, 2003). Within their approach, only different passage-based retrieval systems were compared to each other, but they were not compared to other document retrieval strategies, in particular full-document retrieval.

Clarke and Terra (2003) compare their own passage-based retrieval approach to full-document retrieval using an implementation of the OKAPI retrieval system (Robertson et al., 1998; Robertson and Walker, 1999). Their results indicate that full-document retrieval returns more documents that contain a correct answer, but that passage-based retrieval might still be useful in the context of question answering as it returns shorter excerpts that might ease the process of identifying an actual answer.

Moldovan et al. (2002, 2003), which is more remotely related to our work, gives a detailed failure analysis of their question answering system, showing that 37.8% of the errors are due to the retrieval module. Their retrieval module consists of several smaller modules contributing differently to this error rate: Keyword selection (8.9%), keyword expansion (25.7%), actual retrieval (1.6%), and passage post filtering (1.6%). The reason that keyword expansion has such a strong impact is probably due to their use of surface forms for indexing, where no form of stemming is applied, and one of the tasks of keyword expansion is to add morphological variants, cf. Paşca (2001). The impact of keyword selection is likely to be due to the fact that they use a boolean retrieval model which is much more sensitive to query formulation than, for instance, vector space models. Although at first glance the impact of retrieval on the overall effectiveness seems to be rather small for their system (1.6%), it shows that other retrieval issues such as stemming, and the choice of the retrieval model, viz. boolean vs. vector space or probabilistic, still have a strong impact on the overall performance of their QA system.

Summing up, although most of the literature on question answering discusses

the choices that were made for preferring certain retrieval techniques over others, those decisions are rarely explicated by comparative experimental findings.

3.2 Experimental Setup

In this section we introduce the data sets that are used to compare different retrieval approaches experimentally.

3.2.1 Test Data

We used the TREC-9, TREC-10, and TREC-11 data sets consisting of 500 questions each with 978,952 documents for TREC-9 and TREC-10 from the TIPSTER/TREC distribution and 1,033,461 documents for TREC-11 from the AQUAINT distribution. Recall from chapter 1 that at TREC-9 and TREC-10, participants were required to return up to five answer-document-id pairs for each question, where the answer can be any text string containing maximally 50 characters, and the document-id refers to the document from which the answer was extracted. At TREC-11, participants were required to return one answer-document-id pair for each question, where the answer had to be the exact answer.

In addition, we used the judgment files which were provided by NIST as a result of their evaluation.[2] A judgment file, which is comparable to a qrel file in ad-hoc retrieval, indicates for each submitted answer-document-id pair, whether the answer is correct and whether the document supports, i.e., justifies, the answer. The justifying documents form the set of relevant documents against which we evaluate the different document retrieval approaches for pre-fetching. If none of the participants returned a supported answer, that topic was discarded from our evaluation. This also included questions that did not have an answer in the collection, which can be the case since TREC-10.

The final evaluation sets consist of 480, 433, and 455 topics for TREC-9, TREC-10, and TREC-11, respectively. The original question set for TREC-9 actually contained 693 questions where 193 questions were syntactic variants of 54 of the remaining 500 questions. Here, we did not use the variants, but if a relevant document for a variant was included in the judgment file, it was added to the set of relevant documents of the original question. Variants were removed to avoid repetition of topics, which could bias the overall evaluation. We also included 10 topics of the TREC-11 question set, where, although none of the participants found a relevant document, NIST assessors 'coincidentally' recognized a document containing an answer during their evaluation.

This way of building the qrel sets is known as *pooling* (Sparck Jones and van Rijsbergen, 1975), where for each query the top n documents (usually $n = 100$) of each submitted run are added to the pool and manually assessed for relevance.

[2]The judgment files are available from the TREC web site: `http://trec.nist.gov`.

Documents that were judged irrelevant and all documents that were not in the pool are considered not relevant. Of course, the resulting evaluation sets are not flawless, because there might still be a number of documents in the collection that contain an answer to a question, but are not listed in the qrel file. This is a well-known uncertainty in information retrieval on large document collections, where manual construction of the qrels is infeasible.

A particular problem in the current setting is the pool depth, which is 5 for TREC-9 and TREC-10 and only 1 for TREC-11. It is not clear to what extent this affects evaluation, although Zobel (1998) reports that moving from a pool depth of 100 to a depth of 10 changed the relative performances of only a few systems, and Keenan et al. (2001) conclude that systems that are likely to be effective at larger pool depths will also distinguish themselves at lower pool depths.

Another issue is whether the resulting evaluation sets are biased towards a particular retrieval system which contributed to the pool. NIST made available the top 1000 documents that were retrieved by the SMART retrieval system (Buckley and Walz, 1999) for TREC-9 and the Prise retrieval system (Prise) for TREC-10 and TREC-11. Participating groups were allowed to use these top sets instead of relying on their own retrieval system. If the majority of the participating groups use the provided top documents, the resulting qrel sets could be biased towards the SMART or Prise retrieval system. We consulted the TREC proceedings[3] to see how many of the systems used the top sets provided by NIST. Table 3.1 shows the results. For instance, at TREC-9, 6 of the 28 participating systems used the top sets, 15 another retrieval system, which can be their own system or an off-the-shelf system, such as SMART or MG (Witten et al., 1999), 2 participating groups used a combination of both, and for 5 systems it is unclear because they did not provide any documentation.

Table 3.1: Retrieval systems used by TREC QA participants

		retrieval systems used							
data set	total	NIST		other		combination		unspecified	
TREC-9	28	6	(21%)	15	(53%)	2	(7%)	5	(18%)
TREC-10	36	10	(28%)	15	(41%)	3	(8%)	8	(22%)
TREC-11	34	7	(21%)	17	(50%)	0	(0%)	10	(29%)

It can be seen from table 3.1 that only a rather moderate portion of participants used the documents rankings provided by NIST. Nevertheless, when compared to the number of relevant documents that were found by all participants together, the SMART system found 96.8% (TREC-9) of them, and the PRISE system 91.4% (TREC-10) and 88.2% (TREC-11). If one compares these numbers to the percentages of relevant documents that were found by the best performing systems in the TREC-7 and TREC-8 ad hoc retrieval tracks, which are 71.7% and 70.7% respectively, they do

[3]Available from `http://trec.nist.gov`.

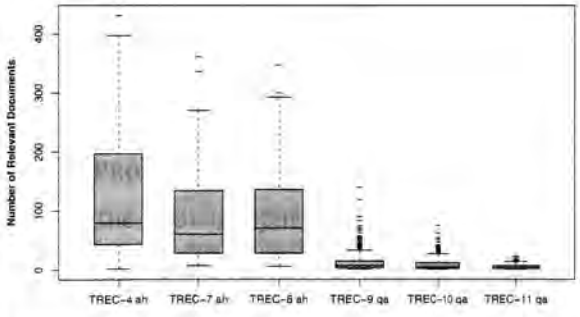
indeed seem rather high. On the other hand, it was possible to submit manual runs to the TREC ad hoc tracks, i.e., runs where the queries were manually constructed and interactively adapted by a human inquirer, which was not possible for the TREC QA tracks. For both TREC-7 and TREC-8, 24% of the relevant documents originated from manual runs only, cf. Voorhees and Harman (1998, 1999). In addition, it is not unusual in ad hoc retrieval evaluation efforts that a single system finds almost all relevant documents. For instance, at the CLEF 2001 monolingual task for French, one (automatic) system found 97.8% of all relevant documents, and for German, one system found 96.2%, cf. Peters et al. (2002, appendix).

One of the traits of the question answering data sets, compared to earlier ad hoc retrieval data sets, is the much smaller number of relevant or supporting documents. Figure 3.1 displays the statistical distribution of relevant documents over several data sets. As will be seen later on, this property does affect retrieval performance. The reason for the small number of relevant documents per topic can be

Figure 3.1: Number of relevant documents for different TREC data sets

(a) shows the Box-and-whiskers plots of the number of relevant documents per topic for ad hoc retrieval (ah) and question answering (qa) data sets. (b) displays the median number of relevant documents and the corresponding median absolute deviation (mad).

(a)



(b)

data set	median	mad
TREC-4 ah	74.0	89.2
TREC-7 ah	55.0	62.8
TREC-8 ah	68.5	60.1
TREC-9 qa	7.0	8.9
TREC-10 qa	5.0	6.6
TREC-11 qa	3.0	3.0

twofold. The information needs of question answering are more specific than for ad hoc retrieval, and the number of top documents that contribute to the pool is much smaller.

Voorhees (2000b) shows that relative system performance in the ad hoc retrieval task is quite immune to alternations in the qrel set, as long as the number of topics is large enough, and the documents that are marked as relevant are representative for all relevant documents and are not selected with respect to a certain property, e.g., highly relevant vs. relevant in general (Voorhees, 2001a). Unfortunately, topics with few relevant documents are problematic for a different reason. Buckley and Voorhees (2000) show that evaluation measures become unstable and small changes

in the document ranking can result in large changes in the evaluation score if the topics have only a few relevant documents. This instability makes it harder to reliably conclude that two systems differ significantly in their performance. But Buckley and Voorhees (2000) also indicate that increasing the number of topics can stabilize evaluation, and it is probably fair to assume that the topic sets in the current setting, which contain 480, 433, and 455 topics for TREC-9, TREC-10, and TREC-11, respectively, are large enough to compensate for the instability due to the small number of relevant documents. Voorhees (2003) estimates the stability of the evaluation scheme that was used at the TREC-11 QA track. Her results show that using a topic set of size 450, absolute differences in average precision between 0.07 and 0.08 result in an error rate of 5%, which is slightly larger than for ad hoc retrieval using 50 topics, where a difference of 0.051 results in an error rate of 5%. An error rate of 5% for a topic set of size n, means that if one compares two systems 100 times using different topic sets of size n, then on average we can expect 95 of those 100 sets to favor one system, and the remaining 5 to favor the other. For further details on the impact of the topic set size on the evaluation stability of document retrieval, the reader is referred to (Voorhees and Buckley, 2002).

3.2.2 Document Retrieval Approaches

In this subsection we introduce some techniques which are known to have a positive impact on the effectiveness of stand-alone document retrieval, and which have also been used by participants in TREC's question answering tracks. Of course, this is only a selection of retrieval techniques that can and have been applied to prefetching, and even the techniques we do discuss cannot be analyzed in full-depth as this is beyond the scope of this chapter. Nevertheless, we aim to discuss some techniques that are commonly used.

All retrieval techniques discussed in the remainder of this thesis use the FlexIR retrieval system (Monz and de Rijke, 2001b, 2002), which was developed at the University of Amsterdam. FlexIR is a vector-space retrieval system with several features including positional indexing, blind feedback, and structured querying.

Stemming

A stemmer removes morphological information from a word, e.g., *electing*, *election*, *elected*, are all reduced to *elect*. Stemming has a long tradition in document retrieval, and a variety of stemmers are available, see Hull (1996) for an overview. Here, we use the Porter stemmer (Porter, 1980), which is probably the most commonly used stemmer. Since the Porter stemmer is purely rule-based, it sometimes fails to recognize variants, e.g. irregular verbs such as *thought*, which is stemmed as *thought*. Therefore, we decided to also use a lexical-based stemmer, or lemmatizer (Schmid, 1994). Each word is assigned its syntactic root through lexical look-up. Mainly number, case, and tense information is removed, leaving other morphological derivations such as nominalization intact, e.g., the noun *election* is not normalized to its

underlying verb *elect*.

Some QA systems do not use stemming to avoid compromising early precision (Clarke et al., 2000a), while others use a hybrid approach where the index contains both the original word and its stem, and matching the stem contributes less to the document similarity score than matching the original word.

Blind Relevance Feedback

A retrieval system using relevance feedback allows the user to mark each of the top n (usually $5 \leq n \leq 10$) documents as either relevant or non-relevant. This information is used to formulate a new retrieval query, adding terms from the documents that were marked relevant, and excluding terms coming from documents that were marked as non-relevant.

If no actual user information is available, relevance feedback can be imitated by simply assuming that all of the top n documents are relevant. This approach is called *blind relevance feedback*.

Blind relevance feedback also analyzes the top n (again, usually $5 \leq n \leq 10$) documents from a preliminary retrieval run to add new terms, and to re-weight terms that were part of the original query. Blind feedback has become a standard technique in document retrieval because of its consistent and strong positive impact on retrieval effectiveness, cf. (Mitra et al., 1998; Robertson and Walker, 1999). On the other hand it is not used in the context of question answering, which might be because there is only a small number of relevant documents, see fig. 3.1, and it is known that blind feedback performs rather poorly under those circumstances. Nevertheless, we wanted to confirm this empirically in the context of question answering.

Our blind relevance feedback approach uses the top 10 documents and term weights were recomputed by using the standard Rocchio method. We allowed at most 20 terms to be added to the original query.

Passage-Based Retrieval

Passage-based retrieval splits a document into several passages, where passages can be of fixed length or vary in length, start at any position or at fixed positions, and overlap to a certain degree, see Kaszkiel and Zobel (1997, 2001) for a comprehensive overview. Passage-based retrieval has proved particularly useful for document collections that contain longer documents, such as the Federal Register sub-collection of TREC. Using passages instead of whole documents emphasizes that the information sought by a user can be expressed very locally. This probably also explains its appeal to question answering, where answers tend to be found in a sentence or two, and it is not surprising that many QA systems use passage-based retrieval instead of document retrieval, cf. Chu-Carroll et al. (2002); Clarke et al. (2002a); Greenwood et al. (2002); Vicedo et al. (2002); Xu and Zhang (2002).

From the broad spectrum of available passage-based retrieval techniques, we used the approach described in (Callan, 1994), because it is fairly standard and yet flexible enough to model a number of ways to realize passage-based retrieval. In Callan's approach, all passages are of fixed length and each passage starts at the middle of the previous one. The first passage of a document starts with the first occurrence of a matching term. Given a query q and a document d which is split into passages $pass_d^1, \ldots, pass_d^n$, the similarity between q and d ($sim(q,d)$) is defined as $max_{1 \leq i \leq n} sim(q, pass_d^i)$. This mapping of passages to their original documents is mainly for evaluation purposes, as the NIST judgments are based on document ids. When using a passage-based retrieval system in the context of an actual QA system one would probably like to return passages instead, as this allows the answer selection procedure to analyze smaller and more focused text segments.

3.2.3 Evaluation Measures

There are a number of evaluation measures that can be used to compare the performance of the different retrieval techniques discussed above. Each measure highlights a different aspect and using several measures to describe the performance of a system is more revealing than using a single measure. On the other hand, when comparing systems, it is often more convenient to use a single measure and the choice depends on the purpose of the retrieval system and the context in which it is used (Sparck Jones, 2001). For instance, it is common to use non-interpolated average precision, also referred to as mean average precision (MAP), in ad hoc retrieval, and p@n in web retrieval. Given a query q, its set of relevant documents REL_q and a ranking of documents ($rank_q : D \rightarrow \mathbb{N}$) resulting from the retrieval process, average precision of an individual query is defined as:

$$\text{avg\_prec}(q) = \frac{\sum_{d \in \mathrm{REL}_q} |\{d' \in \mathrm{REL}_q | \mathrm{rank}(d') \leq \mathrm{rank}(d)\}| / \mathrm{rank}(d)}{|\mathrm{REL}_q|}$$

The mean average precision is then simply the mean of all individual average precisions.

At first glance, the obvious way to compare the performance of different document retrieval approaches that are used for pre-fetching by some QA system, is to rank them with respect to the effectiveness of the complete QA system. If document retrieval approach x leads to a better performance of the QA system than retrieval approach y, then x should be ranked higher than y. Although this is a legitimate way to proceed, it does not allow one to generalize to situations where a different QA system is used, or the original QA system has been modified.

At TREC-11, 34 groups participated in the question answering track, each with their own system. Although many of the techniques that were used at least partially overlap with each other, there is a considerably broad spectrum. For instance, if a system uses rather strict pattern matching for selecting answers, it is more susceptible to generating false negatives (not finding an answer, although it is in the

document) than false positives (selecting an incorrect answer), and therefore document pre-fetching should opt for high recall. Since answers can be expressed in a variety of ways, finding as many relevant documents as possible, increases the probability that one of them is matched by the answer selection process. On the other hand, if a system makes extensive use of lexical relations, such as synonymy or more opaque semantic relations, it becomes more susceptible to generating false positives, and pre-fetching should opt for high precision. One could say that the way in which answer selection reacts to a variety of pre-fetching approaches, also reveals some characteristics of the answer selection process.

This leads us to consider the following two evaluation measures, where R_q is the set of documents that contain an answer to question q.

p@n: $|\{d \in R_q \mid \text{rank}(d) \leq n\}|/n$. The number of found relevant documents up to rank n divided by n.

r@n: $|\{d \in R_q \mid \text{rank}(d) \leq n\}|/|R_q|$. The number of found relevant documents up to rank n divided by the number of all relevant document for that question.

p@n measures the precision of a given retrieval system at rank n, whereas r@n measures the recall. Note that the internal order of the ranks up to rank n does not affect either of the two scores. Often, it is convenient to neglect the exact precision and recall scores and simply measure whether a system returns a relevant document:

a@n: 1 if $|\{d \in R_q \mid \text{rank}(d) \leq n\}| \geq 1$, and 0 otherwise.

Another reason for using a@n is that it is the measure used by Llopis et al. (2002), and it will allow us to compare some of our results to their findings later on. Note that a@n is also equivalent to the evaluation measure used by (Roberts, 2002), where he refers to it as %ABD (percentage of answer bearing documents).

An alternative way of selecting an appropriate evaluation measure is to compare the rankings of all individual measures to identify evaluation measures that rank different retrieval approaches similarly to many other evaluation measures. This way, evaluation measures can be identified that are more representative than others, see (Voorhees and Harman, 1998). From the measures generated by our evaluation script (which can be considered an extension of the `trec_eval` program[4]), 25 measures have been selected for further analysis: p@n, r@n, a@n, ($n \in \{5, 10, 20, 50, 100, 200, 500, 1000\}$), and mean average precision. In total, we compared 14 runs for each of the TREC collections, see section 3.3 for a more detailed description of the runs. For a given collection all runs were ranked by each of the 25 measures and each ranking was compared to all other rankings. Two given rankings were compared by computing the correlation between them using Kendall's τ (Kendall, 1938). Kendall's τ computes the distance between two rankings as the minimum number of pairwise adjacent swaps to turn one ranking into the other. The distance is normalized by the number of items being ranked such that two identical rankings produce a correlation of 1, the correlation between a ranking and its inverse is -1, and the expected correlation of two rankings chosen at random is 0.

[4] `trec_eval` is available from `ftp://ftp.cs.cornell.edu/pub/smart/`.

For a particular collection the τ score of a selected measure was computed by averaging over the τ scores between the ranking resulting from the selected measure and the rankings resulting from all other measures. Finally, we averaged the τ scores of each measure over the three collections.

The rationale behind using the average correlation between a particular measure and the remaining measures, was to investigate how representative a single measure is. The final ranking of all measures and their corresponding τ scores are displayed in table 3.2. It is notable that evaluating at lower cut-offs is more represen-

Table 3.2: Kendall's τ correlation between the different evaluation measures

rank	meas.	τ	rank	meas.	τ	rank	meas.	τ	rank	meas.	τ
1.	p@50	0.435	8.	a@100	0.399	15.	r@50	0.348	22.	p@100	0.298
2.	p@20	0.434	9.	a@50	0.376	16.	p@5	0.341	23.	r@1000	0.143
3.	MAP	0.433	10.	r@500	0.372	17.	p@500	0.334	24.	p@1000	0.137
4.	a@20	0.432	11.	a@10	0.369	18.	a@500	0.333	25.	a@1000	0.097
5.	r@10	0.420	12.	a@5	0.361	19.	r@200	0.328			
6.	r@5	0.418	13.	p@10	0.361	20.	p@200	0.315			
7.	r@20	0.411	14.	a@200	0.354	21.	r@100	0.313			

tative than using higher cut-offs. Also, mean average precision is ranked relatively high.

In the remainder of this chapter, a@n ($n \in \{5, 10, 20, 50\}$) is the principal evaluation measure, because it indicates the immediate effects on answer selection: A QA system using the top n documents of a retrieval system with an average a@n score of m will necessarily have an error rate of at least $1 - m$. In addition, p@n and r@n (using the same cut-offs) allow for a more detailed inspection of the changes in precision and recall; motivated by the discussion above. We do not use higher cut-offs, e.g. a@100 or r@500, although they are ranked relatively high in table 3.2, because most question answering systems seldomly consider more than the top 50 documents returned by the retrieval component. Also, we do not select mean average precision as principal evaluation measure, despite its high ranking in table 3.2, because it does not allow one to draw immediate conclusions with respect to the performance of the overall QA system (as opposed to a@n) and it conflates precision and recall, which can be helpful for predicting the performance of certain types of answer selection strategies.

3.2.4 Statistical Significance

When comparing the effectiveness of two retrieval approaches or methods, the question arises whether it is safe to say that one method is indeed better or more effective than the other one. Usually, such a judgment is based on considering a set of queries, where both methods are evaluated with respect to each query, and at the end, the average scores for both systems are compared. Assuming that method m_1 has a higher score than method m_2, one might be tempted to say that m_1 is more

effective than m_2. The problem is that the higher score of m_1 might be due to a few extreme instances and it is not clear whether this judgment is indeed valid and will carry over to unseen cases. What one is really interested in is whether the difference between them is *statistically significant* and not just caused by chance.

Significance testing aims to disprove a null hypothesis H_0. If one wants to test whether method m_1 is significantly better than method m_2, the null hypothesis will be that m_2 performs at least as good as m_1. The underlying idea is to show that the probability that the null hypothesis holds is so small that it is implausible and should therefore be rejected. Rejecting H_0 leads to accepting the alternative hypothesis H_1, saying that m_1 outperforms m_2. The difference in performance between m_1 and m_2 is expressed by the mean difference μ over the whole population, where μ is expressed with respect to some evaluation measure, for instance, mean average precision or p@n.

$$H_0 : \mu \leq 0$$
$$H_1 : \mu > 0$$

There are many techniques for drawing statistical inferences, and the paired t-test is probably the best-known technique (see, e.g., (Kitchens, 1998)). Many of the inference techniques make certain assumptions about the data to which they are applied. The most common assumption, which also underlies the paired t-test, is that the data is taken from a population which is normally distributed. In the setting of retrieval this means that for a number of queries, the differences between two methods are normally distributed. Whether this assumption holds for text retrieval has been frequently doubted in the literature on retrieval evaluation; see e.g., van Rijsbergen (1979, chapter 7). More recently, Savoy (1997) performed several *goodness-of-fit* tests, including the χ^2 and Kolmogorov-Smirnov tests, to further investigate this issue and concluded that in most cases, the assumption of an underlying normal distribution cannot be validated empirically.

These doubts resulted in a general avoidance of statistical inference in retrieval evaluation. In the early years of information retrieval research, statistical inference was approximated by a rule of thumb, where Sparck Jones (1974) calls absolute improvements of at least 5% *significant*, and improvements of at least 10% *material*. Later, weaker statistical inference tests, such as the sign test, the paired Wilcoxon test and the Friedman test, cf. (Conover, 1980; Hollander and Wolfe, 1973; Siegel and Castellan, 1988), were applied to retrieval, see (Hull, 1993). These tests are *non-parametric*, meaning that they do not assume the data to obey some underlying mathematical model, such as a normal distribution. The paired Wilcoxon test and the Friedman test both use rankings instead of the actual values, and the sign test only considers a binary distinction which indicates whether method m_1 was better than method m_2 or the other way around.

More recently, a powerful non-parametric inference test, the bootstrap method, which has been developed by Efron (Efron, 1979; Efron and Tibshirani, 1993), has been applied to retrieval evaluation, see e.g., (Savoy, 1997) and Wilbur (1994). Wilbur

(1994) compared the bootstrap method to the Wilcoxon and Sign tests, and rated bootstrapping as more powerful than the latter two. Bootstrap methods just assume the sample to be i.i.d. (independent and identically distributed), in other words, the sample should be representative of the whole population. Note that a more general alternative would be the randomization test, which does not even require the sample to be representative, see (Cohen, 1995), but considering that the questions were more or less arbitrarily drawn from query logs, see chapter 1, we can assume the sample to be representative.

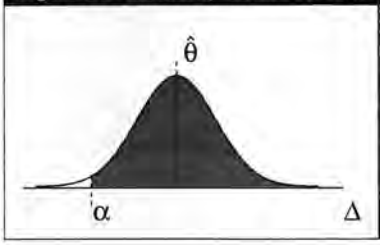
The basic idea of the bootstrap is a simulation of the underlying distribution by randomly drawing (with replacement) a large number of samples of size N from the original sample of N observations. These new samples are called *bootstrap samples*. For each of these samples, an estimator, usually the mean, is calculated. Given this set of means, the standard error can be approximated as follows:

$$\mathrm{SE}_{bootstrap} = \sqrt{\frac{\sum_{i=1}^{b}(\theta_i^* - \hat{\theta})^2}{b-1}}$$

where θ_i^* is the mean of the ith bootstrap sample, $\hat{\theta}$ is the mean of the original sample, and b is the number of bootstrap samples.

Having calculated the mean and the standard error of the bootstrap samples allows us to compute a confidence interval. There are different methods available to compute a confidence interval. One of these is the percentile method. Considering H_0 and H_1 from above, one-tailed significance testing computes a confidence interval of 95% using the 5th and 100th percentiles. If the left limit of the confidence interval is greater than zero, we can reject H_0 and affirm H_1 with a confidence of 95%, see Mooney and Duval (1993); Rietveld and van Hout (1993). Figure 3.2, shows the confidence interval for a normal distribution centered around the estimator $\hat{\theta}$.

Figure 3.2: Bootstrap confidence interval



Distribution of the bootstrap samples. The shaded part is the confidence interval beginning at α-th percentile, covering 1-α*100% of the area of the distribution. The x-axis ranges over the differences between two methods.

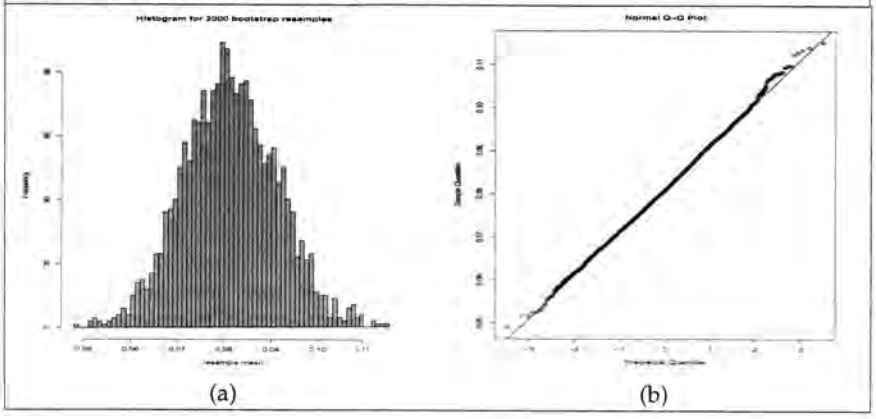
For the experiments in this chapter, we used the `boot` package from the statistical software R; see Davison and Kuonen (2002) for a short introduction to the boot package.[5] The number of bootstrap samples was set to 2000, which is higher than the standard size of 1000 samples (see Davison and Hinkley, 1997), but smaller

[5] R is freely available at `http://www.r-project.org`.

numbers of samples failed to yield a normal distribution of the bootstrap estimators. Figure 3.3 (a) shows the histogram of an actual bootstrap re-sample where the median lies at approximately 0.08, and figure 3.3 (b) shows the normal probability plot (or Q-Q plot) which maps the sample quantiles onto the quantiles of a normal distribution, indicating that the resulting bootstrap sample distribution strongly approximates a normal distribution. The distributions shown in figure 3.3 stem from

Figure 3.3: Distribution of the bootstrap re-samples

(a) shows the histogram of the bootstrap re-samples. (b) is the normal probability plot which shows how closely the bootstrap re-sampling approximates a normal distribution (the straight line).



(a) (b)

a particular bootstrap re-sample, but are representative for all bootstrap re-samples we will encounter later on.

In the sequel, we will indicate improvements at a confidence level of 95% with $^{\triangle}$, and improvements at a confidence level of 99% are marked with $^{\blacktriangle}$. Analogously, decreases in performance at a confidence level of 95% are marked with $^{\triangledown}$, and decreases at a confidence level of 99% are marked with $^{\blacktriangledown}$. No mark up is used if neither an increase nor decrease in performance is significant at neither a confidence level of 95% nor 99%.

3.3 Experimental Results

This section reports on the experimental results of the retrieval techniques discussed in section 3.2.2. Before we discuss the results for the different retrieval methods in detail, the first subsection discusses the weighting scheme that is used to compute the similarity between a document (or passage) and the query, and the second subsection addresses some issues that are relevant for the way the actual retrieval

queries are formulated.

3.3.1 Document Similarity

All experiments in this section use the Lnu.ltc weighting scheme, see (Buckley et al., 1995; Singhal et al., 1996). The Lnu.ltc weighting scheme is based on Salton and Buckley's weighting scheme, introduced in (Salton and Buckley, 1988). In their approach, the weighting scheme that is used to compute the similarity between a document d and a query q is of the form $F_d C_d N_d . F_q C_q N_q$. The weighting scheme has two parts: weighting parameters for the document (F_d, C_d, and N_d), and weighting parameters for the query (F_q, C_q, and N_q). The function of the weighting parameters are as follows:

$F_{d/q}$ computes the weight of a term based on its frequency in the document/query.

$C_{d/q}$ computes the weight of a term based on its frequency in the collection.

$N_{d/q}$ normalizes the document/query weights.

Given these weighting parameters, the similarity between a query q and a document d is then computed as:

$$sim(q,d) = \sum_{t \in q \cap d} \frac{(F_d \cdot F_q) \cdot (C_d \cdot C_q)}{N_d \cdot N_q} \tag{3.3}$$

The actual computation of $sim(q,d)$ depends of course on the instantiations of the parameters F_d, F_q, C_d, etc. Using the Lnu.ltc weighting scheme as an instantiation of the weighting parameters has become one of the standard ways to compute document similarity. The definition of the individual instantiations of the parameters is given below. Here, $\text{freq}_{t,d}$ is the frequency of term t in document d, N is the number of documents in the collection, and n_t is the number of documents in which the term t occurs.

$F_d =$ L: $\frac{1+\log(\text{freq}_{t,d})}{1+\log(\text{avg}_{t' \in d}\text{freq}_{t',d})}$. (The frequency of term t in document d is normalized with respect to the average frequency of all terms in that document.)

$C_d =$ n: 1. (The document frequency of term t is not considered for computing the weight of the term with respect to document d.)

$N_d =$ u: $(1-sl) \cdot pv + sl \cdot \text{uw}_d$. (The term weights in the document are normalized by the number of its unique words with respect to the average number of unique words of all documents in the collection.)

$F_q =$ l: $\frac{\text{freq}_{t,q}}{\max_{t' \in q}\text{freq}_{t',q}}$. (The frequency of term t in query q is normalized with respect to the most frequently occurring term in the query.)

$C_q =$ t: $\log\left(\frac{N}{n_t}\right)$. (The ratio of documents that contain the term t is used to compute its query weight.)

$N_q =$ c: $\sqrt{\sum_{t' \in q}(F_q \cdot C_q)^2}$ (The terms in the query are normalized with respect to the square root of their squared sums.)

The instantiation (u) of the document normalization parameter (N_d) is more intricate and requires a longer explanation. This form of normalization is known as *pivoted document length normalization*, was developed to account for the bias of standard cosine similarity which tends to prefer shorter documents over longer ones, see (Singhal et al., 1996). The basic idea of pivoted normalization is to use the average length of all documents in the collection as a reference point (the *pivot*), and combine it with the length of individual documents. The normalization of a specific document is the weighted sum of the pivot, which is constant for all documents in the collection, and the length of the document at hand, where the weight is also referred to as the *slope*. Whereas the pivot is based on the average length of the documents, the slope is not directly linked to any trait of the collection, and has to be determined experimentally. Based on past experience, we set the slope to 0.2, which is also in line with many experimental settings reported in the literature, see, e.g., (Buckley et al., 1995).

Once instantiated by the Lnu.ltc weighting scheme, the general similarity computation in (3.3) looks as follows:

$$sim(q,d) = \sum_{t \in q \cap d} \frac{\frac{1+\log(\text{freq}_{t,d})}{1+\log(\text{avg}_{t' \in d}\text{freq}_{t',d})} \cdot \frac{\text{freq}_{t,q}}{\max_{t' \in q}\text{freq}_{t',q}} \cdot \log\left(\frac{N}{n_t}\right)}{((1-sl)\cdot pv + sl \cdot \text{uw}_d) \cdot \sqrt{\sum_{t' \in q}\left(\frac{\text{freq}_{t',q}}{\max_{t'' \in q}\text{freq}_{t'',q}} \cdot \log\left(\frac{N}{n_t'}\right)\right)^2}} \tag{3.4}$$

This concludes our brief discussion of the computation of document similarity in general, and the Lnu.ltc weighting scheme in particular. For more details on similarity weighting schemes, the reader is referred to (Salton and Buckley, 1988; Buckley et al., 1995; Singhal et al., 1996; Zobel and Moffat, 1998).

3.3.2 Query Formulation

Query formulation was identical for all methods. We used a stop word list containing 70 words to remove uninformative terms such as *they*, *can*, *the*, etc. Questions words are lemmatized before they are compared to the entries in the stop word list. Methods using stemming, which includes case folding, apply it after stop word removal in order to avoid notorious errors such as removing the content word *US* (abbreviating *United States*) because it becomes the pronoun *us* after stemming, which is a stop word.

Table 3.3: Lemmas vs. porter a@n scores

Comparison of the ratios of questions with at least one relevant document (a@n) using lemmas vs. porter stemming.

	TREC-9			TREC-10			TREC-11		
a@n	lemma	+porter		lemma	+porter		lemma	+porter	
a@5	0.668	0.700	(+4.6%)▲	0.644	0.649	(+0.7%)	0.481	0.523	(+8.6%)▲
a@10	0.739	0.785	(+6.1%)▲	0.729	0.734	(+0.6%)	0.606	0.626	(+3.2%)
a@20	0.804	0.845	(+5.1%)▲	0.787	0.801	(+1.7%)	0.665	0.705	(+5.9%)▲
a@50	0.872	0.914	(+4.7%)▲	0.856	0.875	(+2.1%)	0.751	0.795	(+5.8%)▲

In question answering, it is common practice to categorize the questions. E.g., question (3.5) might be categorized as a location-question, (3.6) as a find-abbreviation question, and (3.7) as a date-question.

(3.5) In what country did the game of croquet originate? (topic id: 1394)

(3.6) What is the abbreviation for the London stock exchange? (topic id: 1667)

(3.7) What year did California become a territory? (topic id: 1694)

Often, questions contain words that are distinctive for a particular category, but in many cases, these words are not helpful in distinguishing the answer documents. For instance, *France* is the answer to question (3.5), but it is common knowledge that France is a country and therefore most of the time not explicated in the answer document. The same holds for question (3.7) where the context in the answer documents makes it clear that *1848* is a year. Sometimes it can even be harmful to include certain words, as in question (3.6), where *abbreviation* helps to clarify the information need, but many answer documents express this information in a different way, e.g., by having the phrase *London stock exchange* followed by the parenthetical word *(LSE)*. Since *abbreviation* is infrequent, and therefore has a relatively high idf-score, including it can steer retrieval in the wrong direction.

For these reasons, it seems sensible to have a category dependent stop word list in addition to a general one. Nevertheless, we did not use category dependent stop word removal, because the quality of this process can influence the quality of the retrieval methods, and it is the latter we want to focus on in this chapter.

3.3.3 Stemming

The first retrieval technique we investigated was stemming. In the literature stemming is sometimes described as recall-enhancing, e.g., Kraaij and Pohlmann (1996), and the question was whether retrieval as a pre-fetch to a question answering system can benefit from stemming; in particular, since pre-fetching should opt for early precision. Table 3.3 shows the a@n scores for lower cut-offs, and table 3.4 shows the corresponding p@n scores.

Table 3.4: Lemmas vs. porter p@n scores

Comparison of the precision at n (p@n) scores using lemmas vs. porter stemming.

	TREC-9			TREC-10			TREC-11		
p@n	lemma	+porter		lemma	+porter		lemma	+porter	
p@5	0.290	0.310	(+7.0%)▲	0.266	0.270	(+1.5%)	0.160	0.167	(+4.6%)△
p@10	0.221	0.238	(+7.6%)▲	0.205	0.212	(+3.7%)△	0.120	0.123	(+3.0%)
p@20	0.161	0.171	(+6.6%)▲	0.149	0.154	(+3.4%)△	0.079	0.084	(+5.7%)▲
p@50	0.096	0.102	(+6.5%)▲	0.086	0.088	(+3.0%)△	0.044	0.047	(+6.5%)▲

The use of stemming exhibits consistent improvements for all collections and all cut-off values. One can notice that the improvements for TREC-10 are much lower than for the other two collections. This could be due to the much larger portion of definition questions in the TREC-10 question set. Questions asking for a definition often contain foreign or technical terms, see (3.8), or proper names, see (3.9), where in both cases morphological normalization does not apply very well, if at all.

(3.8) What is amitriptyline? (topic id: 936)

(3.9) Who was Abraham Lincoln? (topic id: 959)

As could be expected, applying stemming also improves recall, see table 3.5.

Table 3.5: Lemmas vs. porter r@n scores

Comparison of the recall at n (r@n) scores using lemmas vs. porter stemming.

	TREC-9			TREC-10			TREC-11		
r@n	lemma	+porter		lemma	+porter		lemma	+porter	
r@5	0.213	0.234	(+9.5%)▲	0.225	0.233	(+3.3%)	0.221	0.227	(+2.3%)
r@10	0.292	0.326	(+11.7%)▲	0.308	0.329	(+6.5%)△	0.316	0.317	(+0.3%)
r@20	0.386	0.417	(+8.0%)▲	0.400	0.423	(+5.6%)▲	0.387	0.407	(+5.2%)△
r@50	0.508	0.541	(+6.5%)▲	0.532	0.552	(+3.7%)△	0.498	0.536	(+7.7%)▲

Here, TREC-11 shows smaller improvements in recall at lower cut-offs (r@5 and r@10), than TREC-9 and TREC-10. This can be explained by the smaller average number of relevant documents for TREC-11, see figure 3.1.

Summing up, we have noticed that applying stemming consistently improves precision and recall, although the extent depends on the question type (e.g., definition questions show lower improvements) and the specificity of the question, i.e., if there is only a small number of documents containing an answer. For these reasons, and because stemming has become a standard technique in document retrieval, stemming is applied to all experiments discussed below, including the Lnu.ltc baseline run.

3.3.4 Blind Relevance Feedback

The experimental results for blind feedback compared to plain retrieval are shown in table 3.6.

Table 3.6: One-pass retrieval vs. blind feedback a@n scores (top 10)
Comparing simple and blind feedback retrieval.

	TREC-9			TREC-10			TREC-11		
a@n	Lnu.ltc	+feedback		Lnu.ltc	+feedback		Lnu.ltc	+feedback	
a@5	0.700	0.612	(−12%)▼	0.649	0.528	(−18%)▼	0.523	0.400	(−24%)▼
a@10	0.785	0.712	(−9%)▼	0.729	0.602	(−17%)▼	0.626	0.492	(−21%)▼
a@20	0.846	0.783	(−7%)▼	0.787	0.706	(−10%)▼	0.705	0.582	(−17%)▼
a@50	0.915	0.860	(−6%)▼	0.856	0.819	(−4%)▼	0.795	0.707	(−11%)▼

These results confirm our suspicion that blind feedback is not appropriate in the context of question answering. All runs dramatically decrease in performance. Measuring p@n and r@n shows similar decreases. One might suspect that the bad performance of feedback is most likely due to the small number of relevant documents per topic. This could also explain why the results decrease from TREC-9 to TREC-11, as the average number of relevant documents also decreases, see figure 3.1. One way of adapting blind feedback retrieval to a situation where the average number of relevant documents is small, is to use a smaller number of top documents from the initial run to reformulate the query. Table 3.7 shows the results for using the top 5 documents. But using the top 5 instead of the top 10 documents decreases the performance even further.

Table 3.7: One-pass retrieval vs. blind feedback a@n scores (top 5)
Comparing simple and blind feedback retrieval for top 5 docs.

	TREC-9			TREC-10			TREC-11		
a@n	Lnu.ltc	+feedback		Lnu.ltc	+feedback		Lnu.ltc	+feedback	
a@5	0.700	0.531	(−24%)▼	0.649	0.527	(−19%)▼	0.523	0.333	(−36%)▼
a@10	0.785	0.648	(−17%)▼	0.729	0.610	(−16%)▼	0.626	0.447	(−29%)▼
a@20	0.846	0.723	(−14%)▼	0.787	0.688	(−13%)▼	0.705	0.548	(−22%)▼
a@50	0.915	0.827	(−9%)▼	0.856	0.783	(−9%)▼	0.795	0.672	(−16%)▼

To further analyze the relationship between the number of relevant documents per topic and the change in performance when applying blind relevance feedback, we computed Kendall's τ correlation between the two variables. We looked at a number of evaluation measures, but the results are more or less the same for all of them, namely that there is no statistical correlation between the number of relevant documents and retrieval performance. For instance, considering the mean average precision, the correlation with the number of relevant documents is 0.003 for the TREC-9 data, −0.046 for the TREC-10 data, and −0.021 for the TREC-11 data, which

clearly indicates that within the respective data sets there is no correlation at all.

On the other hand, the same blind feedback method is effective for the TREC-7 and TREC-8 ad-hoc retrieval task. Table 3.8 shows the results for both data sets, using the title field only (T) and using the title and description field (TD) of the topic to build the query.

Table 3.8: Feedback for ad-hoc retrieval
Comparing simple and blind feedback retrieval.

	TREC-7			TREC-8		
topic fields	Lnu.ltc	+feedback		Lnu.ltc	+feedback	
T	0.155	0.155	(± 0.0%)	0.195	0.190	(−2.63%)
T+D	0.179	0.197	(+10.1%)	0.221	0.227	(+2.71%)

Using the title field only does not result in any improvement; on the contrary, for the TREC-8 data set there is even a slight decrease in performance. Using the title and description field, results for both data sets improve, although both improvements are not statistically significant.[6] The difference between the title only and title plus description runs suggests that query length might have an impact on the effectiveness of applying blind feedback. Returning to the question answering data sets, we computed Kendall's τ correlation between question length and the change in effectiveness when using blind feedback. The correlation for the TREC-9 data set is -0.038, for TREC-10 it is -0.003, and for TREC-11 it is -0.024, again strongly indicating that they are not correlated.

Another reason for the ineffectiveness of blind feedback for question answering pre-fetching lies in the fact that in many answer documents the information that allows one to answer the question is expressed very locally, e.g., in a sentence or two, and the rest of the document is often rather remotely related to the question. In document retrieval, on the other hand, highly relevant documents are on a whole predominantly on the information need expressed by the query. In particular, the probability that blind feedback will add terms from the answer seems to be rather low, because the initial retrieval run does not accomplish early high precision: At a cut-off level of 5 or 10 there are only one or two relevant documents, which is not enough to conclude that terms occurring in them are relevant for the next retrieval loop, unless they also occur in some of the other documents that do not contain an answer to the question.

To sum up, it is not clear what causes the strong decrease in performance of blind feedback retrieval for question answering pre-fetching. Obviously, it is due to the fact that the information needs are very different, but this difference cannot be further explained in terms of query length or the number of relevant documents per topic.

[6] Improvements for the TREC-7 data set are weakly significant at a confidence level of 90%.

3.3.5 Passage-Based Retrieval

Passage-based retrieval is widely used in QA systems and is therefore worth analyzing in more detail. As mentioned in section 3.2.3, we chose to define passages in terms of windows, where each window is of fixed length and has a 50% overlaps with the previous one. Defining windows this way, exhibited rather consistent improvements in earlier work on ad-hoc retrieval Callan (1994). We experimented with 11 different window sizes: 10, 20, 30, 50, 70, 100, 150, 200, 250, 350, and 500 words. In all cases, the overlap ratio of 50% remained fixed.

The similarity between a query and passage was computed with the Lnx.ltc weighting scheme, which is similar to the Lnu.ltc weighting scheme except that document length normalization is not applied. Normalization was left out because all passages are of fixed length and therefore normalization is expected to make little difference.

Figure 3.4, shows the a@n scores for the three TREC collections, with $n \in \{5, 10, 20, 50\}$. In addition to the passage-based runs, the results for the base runs, using full-document retrieval, are shown.

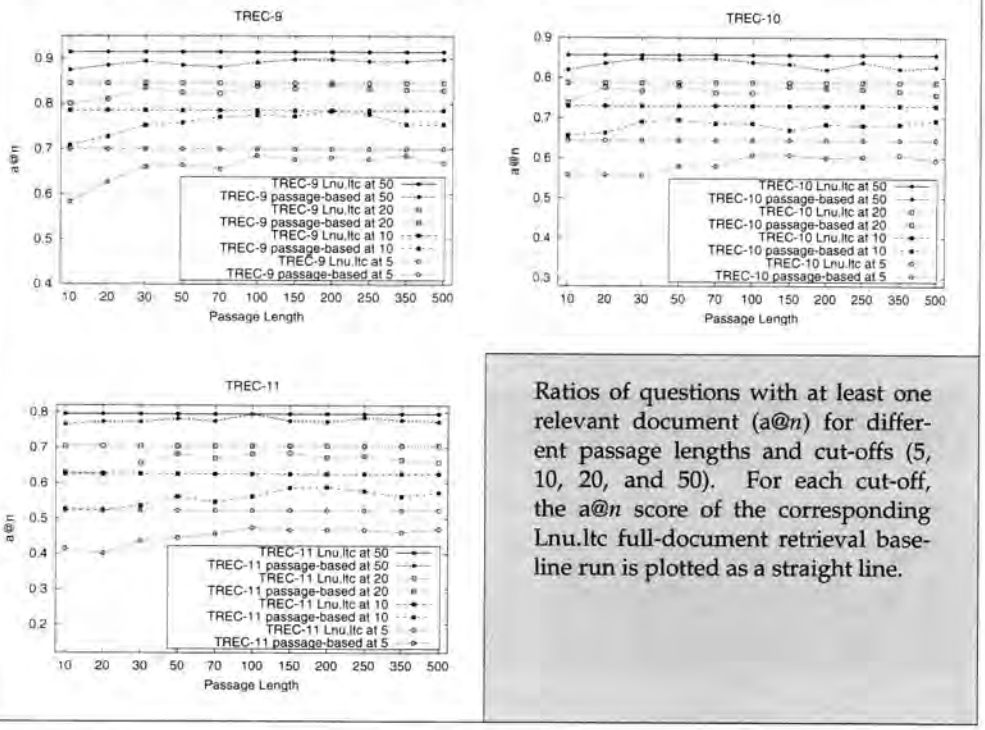
Contrary to what one might expect, all runs using passage-based retrieval perform worse than the respective full-document retrieval run, at any cut-off. In none of the cases, passage-based retrieval provides more questions with at least one relevant document than full-document retrieval. We expected passage-based retrieval to improve early precision by preferring documents that contain matching terms closer to each other and rank lower documents that do contain query terms but the terms are more distributed over the document. To analyze whether precision increased, we measured the p@n score and some of the findings are shown in table 3.9. As the results for other passages sizes do not yield additional insights, we chose not to include them. We did, however, make sure to select some window sizes that show the overall characteristics.

Although precision does increase in a few cases, in general, also precision scores drop when applying passage-based retrieval. However, an increase in precision does not mean that more questions are provided with relevant documents, as can be seen in figure 3.4, but that for some questions more relevant documents are found by passage-based retrieval than by full-document retrieval.

It is not obvious why passage-based retrieval performs worse than document retrieval. Especially since Llopis et al. (2002) report significant improvements for passage-based retrieval when used for question answering: a@5 +11.26%, a@10 +14.28%, a@20 +13.75% and a@50 +9.34%. These improvements are with respect to the results of AT&T's version of SMART on the TREC-9 data set. It is hard to compare their results directly to ours for two reasons: First, the AT&T run is significantly worse than our baseline, and, secondly, it is not clear how they dealt with question variants, as discussed in section 3.2.1.

In the approach by Llopis et al., documents are split into passages of n sentences ($n \in \{5, 10, 15, 20\}$), and each passage starts at the second sentence of the previous passage. Their improvements are probably not so much due to the fact that they use

Figure 3.4: Passage-based retrieval vs. baseline a@n scores



Ratios of questions with at least one relevant document (a@n) for different passage lengths and cut-offs (5, 10, 20, and 50). For each cut-off, the a@n score of the corresponding Lnu.ltc full-document retrieval baseline run is plotted as a straight line.

sentences instead of words to identify passage boundaries, but the fact that their passages have a much larger overlap ratio than the passages used here. Their best results are reported for passages containing 20 sentences, yielding an overlap ratio of approx. 95%—*approximately*, because sentences can differ in length—compared to an overlap of 50% used in our experiments.

Combining our results with the findings of Llopis et al., it can be concluded that passage-based retrieval can yield better results for document pre-fetching, but that passages should significantly overlap with each other. One way to proceed is to carry out more extensive experimentation to establish optimal parameter settings for passage length and overlap, see also (Monz, 2003). Another way is to apply retrieval techniques that are more flexible in using locality, which is the topic of the next chapter.

Table 3.9: Precision for passage-based retrieval
p@n scores for different passage sizes compared to full-document retrieval. Scores set in boldface indicate an improvement over the baseline.

			Passage Length			
	p@n	full	30	70	150	250
TREC-9	p@5	0.310	0.272 (−12.3%)$^{\blacktriangledown}$	0.275 (−11.4%)$^{\blacktriangledown}$	0.276 (−10.8%)$^{\blacktriangledown}$	0.275 (−11.4%)$^{\blacktriangledown}$
	p@10	0.238	0.208 (−12.7%)$^{\blacktriangledown}$	0.221 (−7.5%)$^{\blacktriangledown}$	0.219 (−8.0%)$^{\blacktriangledown}$	0.222 (−6.9%)$^{\blacktriangledown}$
	p@20	0.171	0.161 (−5.8%)$^{\triangledown}$	0.164 (−4.2%)$^{\triangledown}$	0.163 (−4.6%)$^{\blacktriangledown}$	0.163 (−5.0%)$^{\blacktriangledown}$
	p@50	0.102	0.102 (−0.2%)	0.102 (±0.0%)	**0.103 (+0.6%)**	0.101 (−1.3%)
TREC-10	p@5	0.271	0.226 (−16.5%)$^{\blacktriangledown}$	0.241 (−10.9%)$^{\blacktriangledown}$	0.248 (−8.4%)$^{\blacktriangledown}$	0.249 (−7.89%)$^{\blacktriangledown}$
	p@10	0.213	0.184 (−13.4%)$^{\blacktriangledown}$	0.189 (−11.4%)$^{\blacktriangledown}$	0.188 (−11.6%)$^{\blacktriangledown}$	0.189 (−11.0%)$^{\blacktriangledown}$
	p@20	0.154	0.139 (−9.9%)$^{\blacktriangledown}$	0.138 (−10.1%)$^{\blacktriangledown}$	0.145 (−5.9%)$^{\blacktriangledown}$	0.141 (−8.1%)$^{\blacktriangledown}$
	p@50	0.088	0.085 (−3.4%)	0.085 (−3.9%)$^{\triangledown}$	0.085 (−4.1%)$^{\triangledown}$	0.084 (−4.8%)$^{\blacktriangledown}$
TREC-11	p@5	0.167	0.141 (−15.5%)$^{\blacktriangledown}$	0.145 (−13.3%)$^{\blacktriangledown}$	0.150 (−9.9%)$^{\blacktriangledown}$	0.147 (−12.0%)$^{\blacktriangledown}$
	p@10	0.123	0.106 (−13.6%)$^{\blacktriangledown}$	0.108 (−12.2%)$^{\blacktriangledown}$	0.114 (−7.2%)$^{\blacktriangledown}$	0.109 (−11.3%)$^{\blacktriangledown}$
	p@20	0.084	0.080 (−5.0%)	0.078 (−6.7%)$^{\triangledown}$	0.079 (−6.3%)$^{\triangledown}$	0.078 (−6.8%)$^{\blacktriangledown}$
	p@50	0.047	**0.049 (+3.3%)**	**0.048 (+2.5%)**	**0.047 (+0.4%)**	0.046 (−1.4%)

3.4 Conclusions

In this chapter, we investigated three standard retrieval techniques and evaluated their performance in the context of question answering. Evaluation was done on the basis of a system's ability to return documents that contain an answer to a question. All experiments used questions and document collections from the TREC question answering track.

Applying stemming did result in statistically significant improvements in returning at least one relevant document at several cut-offs. Also with respect to precision and recall the application of stemming showed statistically significant improvements at several cut-offs.

Using blind relevance feedback to expand queries resulted in dramatic decreases in performance. The bad performance of feedback is most likely due to the small number of relevant documents per topic. Another reason could be the fact that in many answer documents the information that allows one to answer the question is expressed very locally, but our blind feedback approach used full documents to identify terms that are used for query expansion. One way to address the issue of locality is to use a local feedback approach such as local context analysis (Xu and Croft, 1996). Ittycheriah et al. (2001) have applied local context analysis to document retrieval for question answering, and report some interesting observations, but unfortunately they did not properly evaluate the impact of local context analysis.

Passage-based retrieval did not live up to the expected improvements. In fact, our approach resulted in minor improvements in precision in a few cases only, and overall performed worse than the baseline. This is in contrast to some other results in the literature and shows that the way passages are formed is an important issue.

Chapter 4

Minimal Span Weighting

For most questions, answers are expressed very locally, covering only a few sentences of a document. Taking into account the proximity between question terms is helpful in determining whether a document contains an answer to a question. In this chapter, we propose a new proximity-based approach to document retrieval, which combines full-document retrieval with proximity information. Experimental results show that it leads to significant improvements when compared to full document retrieval. Our approach also proves to be useful for extracting short text segments from a document, which contain an answer to the question asked. This allows answer selection to be focused on smaller segments instead of full documents.

One of the reasons passage-based retrieval is widely used as a pre-fetch in current question answering systems, is the intuition that the answers to most questions can be found in rather short text segments, occupying only a sentence or two. Of course, this depends on the type of question, as some types, e.g., procedural questions such as *How do I make spaghetti alla carbonara?*, require more extensive answers. The fact that most answers are expressed rather locally in a document has two consequences for retrieval as a pre-fetch to a question answering system. First, the retrieval method should take into account the proximity between query terms and rank documents where query terms occur close to each other higher than documents where this is not the case. Second, the retrieval method should return segments of the document which exhibit a high proximity between the query terms instead of full documents.

Both requirements are met by passage-based retrieval. However, the experiments discussed in the previous chapter did not show significant improvements of

passage-based retrieval over full-document retrieval when used as a pre-fetch to a QA system. On the contrary, in most cases it lead to a significant decrease in performance. This was in sharp contrast to the findings of Llopis et al. (2002), who report large improvements using passage-based retrieval. For a more detailed discussion of what might explain these differences, the reader is referred to section 3.3.5. Although we are reluctant to say that passage-based retrieval is indeed harmful in the context of question answering, it can be concluded that the parameters controlling passage-based retrieval, such as passage size, degree of overlap between passages, fixed length vs. variable length, etc., have to be carefully chosen, and might be highly collection and query dependent.

An alternative to passage-based retrieval that meets the two requirements mentioned above, is proximity-based retrieval. Other than for passage-based retrieval, parameters such as passage size, degree of overlap between passages, etc., do not need to be fixed. In passage-based retrieval, the proximity between a number of terms is determined by checking whether they occur in the same passage, which of course, depends on the size into which passages are split. In proximity-based retrieval, proximity is expressed as the distance between terms, i.e., the number of words occurring between them. Defining the proximity between two terms is trivial, but several approaches are possible if more than two words are involved.

The remainder of this chapter is organized as follows: The next section reviews previous approaches to proximity-based retrieval, some of which have also been applied in the context of question answering. Section 4.2 introduces our approach to proximity-based retrieval, and section 4.3 discusses the experimental results of our approach when used to identify relevant documents with respect to the different TREC question answering data sets. Section 4.4 reviews how useful our approach is when it returns text segments instead of full documents for further processing in a question answering system. Finally, section 4.5 provides some conclusions.

4.1 Related Work

Numerous approaches to proximity-based retrieval have been proposed in the literature. The intuition that the proximity between query terms in a document affects relevance dates back to 1958, when Luhn (1958) wrote:

> It is here proposed that the frequency of word occurrences in an article furnishes a useful measurement of word significance. It is further proposed that the relative position within a sentence of words having given values of significance furnishes a useful measurement for determining the significance of sentences. The significance factor of a sentence will therefore be based on a combination of these two measurements.

The first criterion, the *within-document frequency* of a term, has received a lot of attention, resulting in several weighting schemes, see e.g., Salton and Buckley (1988) and Buckley et al. (1995). The second criterion, considering the relative positions of

query terms, has more recently attracted some systematic investigation. Two problems motivate this interest. First, if the documents in a collection vary in length by several orders of magnitude, the matching terms in a long document can be widely spread and occur semantically unrelated to each other. Normalizing the similarity score by the document length is often used as a countermeasure. On the other hand, document length normalization has some unpleasant side effects, such as preferring shorter documents over longer ones, cf. (Singhal et al., 1996), and there is no clear consensus on how normalization should be carried out in a general way. Second, experimental research on the seeking behavior of human searchers using a web search engine (Jansen et al., 2000), has shown that most users only consider the top ten results neglecting everything else further down the ranked list of documents (links to web sites). This observation suggests that web retrieval systems should opt for early high precision, and proximity-based retrieval seems to be a natural way to accomplish this, cf., (Clarke et al., 2000b).

Keen (1992) was one of the first discussions where proximity-based retrieval approaches were evaluated in an experimental setting. The distances between all adjacent matching terms are computed and several ways of combining the distances to compute a similarity score are compared. According to his experiments, which use the LISA test collection,[1] the best method is to use the inverse of the sum of all distances. Further experiments which also considered the distances between terms from different sentences did not differ from experiments where only distances between terms from the same sentence are used. Using all pairs of terms instead of only adjacent terms did result in a small decrease in performance.

Hawking and Thistlewaite (1995, 1996) do not consider the distances between individual pairs of matching terms, but the minimal distance between all matching terms in a document. Terms are mapped into a concept space, where synonyms and morphological variants are represented by the same concept. In order to deal with partial spans, i.e., documents that do not contain all terms (concepts) from the query, a span is assigned a degree, indicating how many query terms are missing. E.g., a degree of 0 indicates that all terms from the query are in the span, a degree of 1 indicates that one term is missing, etc. Using this degree, a form of coordination level matching is implemented, where a span of degree $n+1$ always receives a lower score than a span of degree n, no matter what the sizes of the respective spans are. In all cases, the effect of the span size is dampened by taking its square root to slow down the decay with increasing sizes.

De Kretser and Moffat (1999a,b) propose a very different approach to proximity-based retrieval. Proximity is not expressed as the size of a span covering all query terms in a document, but as a complex function of the distances between all query terms in the document. Each occurrence of a query terms has a certain weight depending on the idf-score of that term, and this weight is distributed over the terms in its proximity, decaying as one moves away from the original term. The final relevance score of a document is computed by considering all occurrences of all

[1] Available from `http://www.dcs.gla.ac.uk/idom/ir_resources/test_collections/`.

query terms, where each occurrence adds the weight which is determined by its own weight (idf-score) but also by weights spread out from other terms occurring in its proximity. The appeal of this approach is the ability to account for different distributions of terms. For example, consider the following distribution of query terms, where t_i is an occurrence of a query term, and $*$ is an occurrence of a non-query term:

$$t_1 \ * \ * \ * \ * \ t_2 \ * \ * \ * \ * \ t_3$$
$$t_1 \ * \ t_2 \ * \ * \ * \ * \ * \ * \ * \ t_3$$

Note that span-based approaches such as Hawking and Thistlewaite (1996) cannot distinguish between the two distributions, because both cover the same span, although they do differ internally. In the approach of de Kretser and Moffat (1999a,b), on the other hand, the different distributions would result in a different similarity score.

Whereas the approaches described above applied proximity-based retrieval in the context of ad hoc document retrieval, Clarke et al. (2000a); Clarke and Cormack (2000) and Clarke et al. (2002a) used it as a method for pre-fetching and excerpt extraction in question answering. The main difference to the approach by Hawking and Thistlewaite (1996) is the use of all spans in a document. If a document contains n query terms, Hawking and Thistlewaite (1996) only considered spans that also contain n query terms, whereas Clarke et al. (2000a) also consider all spans that contain $m \leq n$ terms. Since proximity-based retrieval is used to identify text excerpts that are likely to contain answer to a question, the retrieval system returns a ranked list of spans instead of documents.

Kwok et al. (2000) also used proximity-based retrieval as a pre-fetch in their question answering system, but applied it only if all terms from the question did occur in a document, i.e., partial spans where not considered. This is a very strong restriction, and it requires the retrieval query to be formulated very carefully.

Rasolofo and Savoy (2003), apply proximity-based retrieval to collections of web pages. As mentioned above, web retrieval requires early high precision and proximity-based retrieval appears to be a natural way to accomplish this. Similar to Keen (1992), distances between pairs of terms are computed, without imposing the restriction of adjacency. The most interesting aspect of Rasolofo and Savoy (2003)'s approach is the combination of regular similarity computation based on the whole document using the Okapi similarity measure (Robertson et al., 1998) and proximity-based retrieval, which is much in the spirit of Luhn (1958)'s take on proximity-based retrieval, as cited above.

Although proximity-based retrieval is integrated into some question answering systems, e.g., Clarke et al. (2002a); Kwok et al. (2000), there is no experimental evaluation of its effectiveness as a pre-fetch for question answering. Cormack et al. (1999) used a proximity-based retrieval system as the question answering system for their participation in the TREC-8 250-byte task. They did not apply any question analysis, answer selection. Nevertheless, their top five responses contained a correct answer for 63% of the questions. To some extent this can be considered as an evaluation of

a proximity-based retrieval system, but the problem is the reliability of the TREC-8 data set. As mentioned in chapter 1, the questions in the TREC-8 data set were mostly back-formulations of sentences in the document collection which contained a correct answer (Voorhees, 2001c). This resulted in an unnaturally large word overlap between questions and answer sentences, which distorts many findings based on this data set.

4.2 Minimal Span Weighting

In this section, we introduce a new proximity-based approach to document retrieval, which is based on the minimal size of a text excerpt that covers all terms that are common between the document and the query, the number of common terms vs. the number of query terms, and the global similarity between the document and the query. The advantage of this approach over previous approaches to proximity-based retrieval, lies in the number of aspects that are taken into account, namely full-document similarity, ratio of matching terms, and the proximity of matching terms, and the parametrized way in which the different aspects are combined to compute the final document similarity score.

4.2.1 Definition of Minimal Span Weighting

Minimal span weighting takes the positions of matching terms into account, but does so in a more flexible way than passage-based retrieval. Intuitively, a minimal matching span is the smallest text excerpt from a document that contains all terms which occur in the query and the document. More formally:

Definition 4.1 (Matching span) Given a query q and a document d, where the function $\text{term\_at\_pos}_d(p)$ returns the term occurring at position p in d. A *matching span* (ms) is a set of positions that contains at least one position of each matching term, i.e. $\bigcup_{p \in \text{ms}} \text{term\_at\_pos}_d(p) = q \cap d$. ■

Definition 4.2 (Minimal matching span) Given a matching span ms, let b_d (the beginning of the excerpt) be the minimal value in ms, i.e., $b_d = \min(\text{ms})$, and e_d (the end of the excerpt) be the maximal value in ms, i.e., $e_d = \max(\text{ms})$. A matching span ms is a *minimal matching span* (mms) if there is no other matching span ms′ with $b'_d = \min(\text{ms}')$, $e'_d = \max(\text{ms}')$, such that $b_d \neq b'_d$ or $e_d \neq e'_d$, and $b_d \leq b'_d \leq e'_d \leq e_d$. ■

The next step is to use minimal matching spans to compute the similarity between a query and a document. Minimal span weighting depends on three factors.

1. *document similarity*: The document similarity is computed using the Lnu.ltc weighting scheme, see Buckley et al. (1995), for the whole document; i.e., positional information is not taken into account. Similarity scores are normalized with respect to the maximal similarity score for a query.

2. *span size ratio*: The span size ratio is the number of unique matching terms in the span over the total number of tokens in the span.

3. *matching term ratio*: The matching term ratio is the number of unique matching terms over the number of unique terms in the query, after stop word removal.

The msw score is the sum of two weighted components: The normalized original retrieval status value (RSV), which measures *global similarity* and the spanning factor which measures *local similarity*. Given a query q, the original retrieval status values are normalized with respect to the highest retrieval status value for that query:

$$\mathrm{RSV}_n(q,d) = \frac{\mathrm{RSV}(q,d)}{\max_d \mathrm{RSV}(q,d)}$$

The spanning factor itself is the product of two components: The span size ratio, which is weighted by α, and the matching term ratio, which is weighted by β. Global and local similarity are weight by λ. The optimal values of the three variables λ, α, and β were determined empirically, leading to the following instantiations: $\lambda = 0.4$, $\alpha = 1/8$, and $\beta = 1$. Parameter estimation was done using the TREC-9 data collection only, but it turned out to be the best parameter setting for all collections.

The final retrieval status value (RSV') based on minimal span weighting is defined as follows, where $|\cdot|$ is the number of elements in a set:

Definition 43 (Minimal span weighting) If $|q \cap d| > 1$ (that is, if the document and the query have more than one term in common), then

$$\mathrm{RSV}'(q,d) = \lambda\, \mathrm{RSV}_n(q,d) + (1-\lambda) \left(\frac{|q \cap d|}{1 + \max(mms) - \min(mms)} \right)^{\alpha} \left(\frac{|q \cap d|}{|q|} \right)^{\beta}$$

If $|q \cap d| = 1$ then $\mathrm{RSV}'(q,d) = \mathrm{RSV}_n(q,d)$. ■

Note that minimal span weighting only exploits minmal matching spans for documents containing more than one matching term, as proximity between terms is not defined for documents containing only one matching term. Therefore, the retrieval status value for documents containing only one matching term is equeal to the documents normalized retrieval status value as defined by its global document similarity.

At this point it might be helpful to further illustrate the definition by considering the following question:

(4.1) Who is Tom Cruise married to? (topic id: 1395)

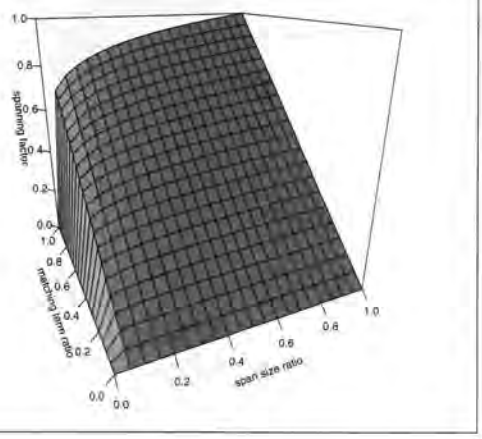
After stop word removal and applying morphological normalization, the query q={*cruise, marri, tom*}. Assume that there is a document d with terms matching at the following positions: $\mathrm{pos}_d(cruise) = \{20, 35, 70\}$, $\mathrm{pos}_d(marri) = \{38, 80\}$, and

$\text{pos}_d(tom) = \emptyset$. Then, the minimal matching span (mms) = $\{35, 38\}$, the span size ratio is $2/(1 + 38 - 35) = 0.5$, and the matching term ratio is $2/3$. Taking the latter two and the proper instantiations of α and β, the spanning factor is $0.5^{1/8} \cdot 2/3 = 0.611$. If the global (normalized) similarity between q and d is n ($0 < n \leq 1$), for instance $n = 0.8$, and $\lambda = 0.4$, the final msw-score for q and d ($\text{RSV}'(q, d)$) is $0.4 \cdot 0.8 + 0.6 \cdot 0.611 = 0.6866$.

To illustrate the behavior of the spanning factor, figure 4.1 plots the values of the spanning factor for all possible combinations of span size ratio and matching term ratio. One can see that, initially, the spanning factor decreases slowly as the

Figure 4.1: The spanning factor

3D plot of the spanning factor function, which is (span size ratio)$^{1/8}$· (matching term ratio).



span size ratio decreases, but then it drops sharply as the span size ratio falls below a certain threshold, approx. 0.05. Along the other dimension, the spanning factor decreases linearly with the matching term ratio.

4.2.2 Computing Minimal Matching Spans

The algorithm for computing the minimal matching spans is sketched in figure 4.2. The function `compute_mms` is applied to the sorted array `match_positions` which contains the positions at which a query term occurs in the document. Each position is added to a temporary buffer span which is implemented as a queue. Every time a position has been appended to the right of span (line 3) it is checked whether span is a matching span (line 4), see definition 4.1. If this is the case and it is the first time a matching span has been found or it is shorter than the shortest span found

Figure 4.2: The minimal matching span algorithm

`compute_mms` computes the minimal matching span for a document, given a sorted array (`match_positions`) of positions at which query terms occur.

```
1  compute_mms {
2     for(i=0; i<length(match_positions); i++) {
3        push(span,match_positions[i]);
4        if(matching_span(span)) {
5           if(undef(min_span_length)
6              || last(span)-first(span)+1<min_span_length) {
7              min_span_length=last(span)-first(span)+1;
8              min_span=span;
9           }
10          shift(span);
11       } else {
12          if(term_at_pos(first(span))==term_at_pos(last(span))
13             && length(span)>1) {
14               shift(span);
15          }
16       }
17    }
18    return min_span;
19 }
```

where:

- `length(a)` returns the length of array `a`
- `first(a)` returns the first element of array `a`, i.e., `a[0]`
- `last(a)` returns the last element of array `a`, i.e., `a[length(a)-1]`
- `push(a,e)` adds element `e` as last element to array `a`
- `shift(a)` removes the first element of array `a`
- `term_at_pos(p)` returns the term occurring at position `p`
- `matching_span(a)` returns 1 if the positions in `a` cover all terms occurring in the query and the document, and 0 otherwise.

so far, the current span becomes the new minimal matching span (line 5–8). Once a matching span has been stored in `span`, the first element of the queue is removed, turning `span` into an incomplete span, i.e., a span not containing positions for all query terms (line 10). If `span` was not a matching span (line 11), and the position that just has been appended to `span` (line 3) is an occurrence of the same term that occurs at the left-most position of `span` (line 12), the first element of `span` is removed (line 14), because any span with the current occurrence of that term will be shorter. When all positions have been considered, the function `compute_mms` returns an array

of positions containing the minimal matching span (line 18).

In order to compute minimal matching spans it is obviously required that the retrieval system keeps track of the positions of terms in the documents. The FlexIR system stores this information in the inverted index. Term-ordered retrieval systems (Kaszkiel and Zobel, 1998), consider each query term at a time, accessing the list of documents in which the term in question occurs at least once (the posting list). The entries in the posting list contain the term's within-document frequency and a list of its positions. Given the term's within-document frequency, its idf-score, and a number of other parameters, the matching score for this particular term and document is computed and added to a container (accumulator) which stores the similarity score of this document. In the course of processing the query, a document's accumulator increases if several query terms occur in this document. In minimal span matching it is also necessary that the document's accumulator says which query terms matched the document and what their respective positions in the document are. When all terms in the document have been processed, each accumulator is normalized with respect to document and query length according to the Lnu.ltc weighting scheme. At this point, the accumulators contain the retrieval status value (RSV) as used above.

Up to now, all steps in the retrieval process, except storing positional information, are standard and part of most retrieval system architectures, cf. Harman (1992); Witten et al. (1999). In a regular retrieval system the documents are sorted with respect to their retrieval status value and returned to the user. In our minimal span weighting system, each document accumulator is considered again in order to compute the document's minimal matching span, but this time the matching query terms and their positions are used. Before we can apply `compute_mms`, two preprocessing steps have to be carried out. Given a query q and a document d, we build an ordered list of positions at which a query term occurs:

$$\texttt{match\_positions} = \texttt{sort}(\bigcup_{t \in q \cap d} \text{pos}_d(t))$$

At the same time, the mapping `term_at_pos` is instantiated, storing the information which term occurs at which given position. Now, `compute_mms` can be applied to return the minimal matching span of document d. Once this has been done for all accumulators, minimal span weighting proceeds as described in definition 4.3. The only thing that remains to be done is to sort the accumulators with respect to the final retrieval status value RSV'.

The complexity of the `compute_mms` function depends on the implementation of the `matching_span` function which checks whether a span contains all n query terms that occur in the document. This can be done by using balanced binary trees whose complexity is never worse than $O(\log n)$, cf. (Musser and Saini, 1996). Completeness is checked for each matching term position, and therefore the complexity of `compute_mms` is $O(n \log n)$. Note that using hash tables instead of trees would not make a difference, because one of the preprocessing steps is to build an ordered

list of positions, which requires sorting, and this causes the overall procedure to have complexity $O(n \log n)$ anyway. The current implementation of minimal span weighting in the FlexIR system takes 0.5 CPU seconds on average to compute all minimal matching spans for a query, and this includes ranking the final retrieval status values.

4.3 Experimental Results

The experimental setting for evaluating the effectiveness of minimal span weighting is identical to the setting for our comparison of standard retrieval approaches in the previous chapter.

Recall from our discussion in chapter 3 that the results of using passage-based retrieval as a pre-fetch for question answering were somewhat inconclusive. More generally, this raises the question to what extent considering proximity does improve retrieval? The minimal span weighting (msw) approach offers a flexible way of integrating positional information into the weighting scheme. Figure 4.3 compares the results of minimal span weighting to the Lnu.ltc baseline and the top-ranked documents provided by NIST. The minimal span weighting scheme outperforms both, Lnu.ltc weighting and the NIST rankings. In particular at lower cut-off levels, minimal span weighting performs much better. One can also see that Lnu.ltc weighting is a well-performing and representative baseline, as it performs better than AT&T's SMART version for the TREC-9 data set and roughly identical to the PRISE system for the TREC-10 and TREC-11 data sets; both systems being standard modern retrieval systems. Table 4.1 provides more details on the differences in performance between minimal span weighting and the Lnu.ltc baseline. The msw

Table 4.1: Comparison of the a@n scores of msw retrieval runs to baseline runs

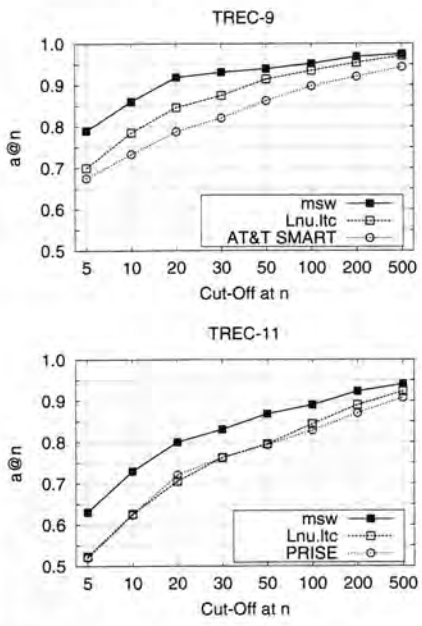
	TREC-9			TREC-10			TREC-11		
a@n	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
a@5	0.700	0.789	(+12.8%)▲	0.649	0.736	(+13.5%)▲	0.523	0.630	(+20.5%)▲
a@10	0.785	0.860	(+9.5%)▲	0.734	0.829	(+12.9%)▲	0.626	0.729	(+16.4%)▲
a@20	0.845	0.918	(+8.6%)▲	0.801	0.873	(+8.9%)▲	0.705	0.800	(+13.4%)▲
a@50	0.914	0.939	(+2.7%)▲	0.875	0.903	(+3.1%)▲	0.795	0.868	(+9.1%)▲

approach significantly improves retrieval for all three collections compared to the baseline. Improvements are especially high at lower cut-offs.

Taking a closer look at the precision at a given cut-off level n (p@n) reveals even higher improvements, see table 4.2. The drop in absolute precision at n for the TREC-11 data set (as compared to the TREC-9 and TREC-10 data sets), at all cut-off levels, is probably due to the fact that the questions were more difficult than questions of the TREC-9 and TREC-10 data sets, and, which is more likely, to the smaller average number of relevant documents.

Table 4.3 shows the recall for different cut-off levels. Whereas the a@n and p@n

Figure 4.3: Comparison of msw to Lnu.ltc weighting and NIST rankings



Comparison of the percentages of questions that have at least one relevant document when considering the top-n ranks (a@n), for msw runs, baseline runs and runs provided by NIST, where $n \in \{5, 10, 20, 50, 100, 200, 500\}$.

Table 4.2: Comparison of p@n scores of msw retrieval runs to baseline runs

	TREC-9			TREC-10			TREC-11		
p@n	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
p@5	0.310	0.377	(+21.5%)▲	0.270	0.322	(+19.1%)▲	0.167	0.226	(+34.8%)▲
p@10	0.238	0.293	(+22.9%)▲	0.212	0.255	(+20.0%)▲	0.123	0.167	(+35.2%)▲
p@20	0.171	0.214	(+25.1%)▲	0.154	0.186	(+20.6%)▲	0.084	0.114	(+35.1%)▲
p@50	0.102	0.124	(+21.9%)▲	0.088	0.105	(+19.1%)▲	0.047	0.060	(+26.3%)▲

Table 4.3: Comparison of the r@n scores of msw retrieval runs to baseline runs

	TREC-9			TREC-10			TREC-11		
r@n	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
r@5	0.234	0.297	(+27.0%)▲	0.233	0.302	(+29.5%)▲	0.227	0.309	(+36.4%)▲
r@10	0.326	0.401	(+22.8%)▲	0.329	0.412	(+25.4%)▲	0.317	0.424	(+33.5%)▲
r@20	0.417	0.517	(+23.9%)▲	0.423	0.525	(+24.2%)▲	0.407	0.532	(+30.6%)▲
r@50	0.541	0.639	(+18.0%)▲	0.552	0.647	(+17.1%)▲	0.536	0.655	(+22.1%)▲

scores for the TREC-11 data set are lower than for the other data sets, recall remains roughly the same. All improvements of using minimal span weighting instead of

Lnu.ltc weighting are significant at a confidence level of 99%.

It should be pointed out that the improvements in p@n and r@n are not only caused by the larger number of questions for which the msw run succeeded to return at least one relevant document. For instance, extrapolating the baseline's p@5 score for TREC-9 by adding 12.8% (which is msw's improvement for a@5), yields a p@5 score of 0.3489, compared to which the msw run is still 8.05% better.

Finally, we measured mean average precision, which combines precision and recall for all cut-offs, and the results are shown in table 4.4. As could be expected from

Table 4.4: Comparison of mean average precisions (MAP) of msw retrieval runs to baseline runs

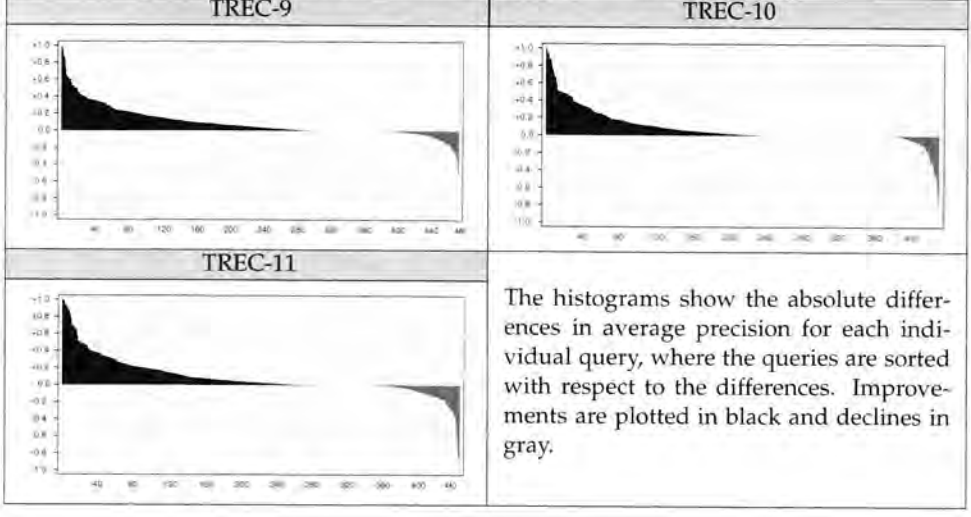
	TREC-9			TREC-10			TREC-11		
	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
MAP	0.280	0.366	(+30.8%)▲	0.279	0.358	(+28.3%)▲	0.214	0.296	(+37.7%)▲

the previous results, msw improves significantly compared to the Lnu.ltc baseline run.

4.3.1 Individual Query Performance

Despite the significant improvements of the minimal span weighting scheme over Lnu.ltc weighting, it does not improve for all queries. Figure 4.4 shows the histograms for the respective TREC collections, measuring the absolute difference in average precision between the Lnu.ltc baseline and minimal span weighting for each query.

Table 4.4: Histograms for absolute differences in average precision



The histograms show the absolute differences in average precision for each individual query, where the queries are sorted with respect to the differences. Improvements are plotted in black and declines in gray.

All three data sets exhibit a similar distribution of increases and decreases in effectiveness of minimal span weighting for individual queries. In most cases, the retrieval performances of the individual queries are affected positively, but for some queries msw performs slightly worse, and for a few queries performance drops dramatically. In order to see whether the impact of minimal span weighting depends on some characteristic of the query, we looked at the individual queries. If one could find such a characteristic, the λ factor in the msw scheme (see definition 4.3) could be easily instantiated in such a way that the effect of span matching is controlled appropriately. Unfortunately, it is very hard to find such a characteristic, and it might be possible that such a trait simply does not exist. Earlier work on predicting the hardness of an information need (Voorhees and Harman, 1997), which is loosely related to the current problem, has shown the difficulties in finding features in the topic that predict the behavior of a retrieval system.

Here, we only looked at one factor that could affect the performance of minimal span weighting: query length. We assume that the longer the query is, the harder it is to find a short span. To compute the correlation between query length and average precision, we used Kendall's τ measure, which resulted in a correlation of -0.056, strongly suggesting that query length and average precision are randomly related.

Just looking at the questions and their respective average precisions unfortunately did not suggest any prevalent characteristics of the question that might be indicative for predicting the retrieval system's performance. On the other hand, there are many more aspects of a question than its length that might play a role for the effectiveness of minimal span weighting, but a thorough investigation of these aspects is a very involved enterprise and remains an issue for future research.

4.3.2 The Effect of Coordination Level Matching

As stated in definition 4.3, the minimal span weighting scheme depends on three weighted factors, the document similarity, the span size ratio, and the matching term ratio. In the experiments discussed in this chapter, the span size ratio is taken to the power of $1/8$, in order to dampen the effect of differences in span size when the span size ratio is large, i.e., the matching query terms occur close to each other in a document. On the other hand, the matching term ratio is taken to the power of 1, i.e., left unchanged. Since the weight of the span size ratio is much smaller than the weight of the matching term ratio, the question arises whether considering the span length has a significant effect at all, or whether the improvement of the msw scheme are mainly due to the matching term ratio? Or to put it differently, how does the retrieval performance change, if the span size ratio is neglected, i.e., α in definition 4.3 is set to 0? If the improvements of the msw schema over the Lnu.ltc baseline are only marginally due to the span size ratio, this has important practical implications: Neglecting span size means that one does not have to keep track of the positions of a term in a document, which severely reduces the size of the inverted

index, and increases the efficiency of the retrieval system.

If the span size ratio factor is removed from the msw scheme, document similarity only depends on two factors: global document similarity, which is computed using the Lnu.ltc weighting scheme, and the matching term ratio. The technique of using the matching term ratio to compute document similarity is also referred to as *coordination level matching* or *ranking*, cf. (Salton and McGill, 1983). Coordination level matching ranks documents in such a way such that all documents containing $n+1$ query terms are ranked higher than documents containing n query terms.

In order to see whether the span size ratio does make a significant contribution to the performance of the msw scheme, we conducted experiments where the span size ratio factor was neglected, i.e., α was set to 0. Another parameter that needs to be fixed is the weight of the document similarity factor (λ). If $\lambda = 0$, documents are ranked by coordination level matching only. Having experimented with different instantiations of λ, ranging from 0 to 0.8, a value of 0.6 turned out to give the best results for all three TREC data sets. Table 4.5 compares the msw weighting scheme with the following instantiations $\lambda = 0.6$, $\alpha = 0$, and $\beta = 1$ to the Lnu.ltc baseline. Retrieval results improve significantly for all data sets at almost all cut-off levels,

Table 4.5: Comparison of the a@n scores of clm retrieval runs to baseline runs

	TREC-9			TREC-10			TREC-11		
a@*n*	Lnu.ltc	clm		Lnu.ltc	clm		Lnu.ltc	clm	
a@5	0.700	0.733	(+4.7%)▲	0.649	0.688	(+6.0%)▲	0.523	0.569	(+8.8%)▲
a@10	0.785	0.827	(+5.4%)▲	0.734	0.769	(+4.8%)▲	0.626	0.686	(+9.6%)▲
a@20	0.845	0.879	(+4.0%)▲	0.801	0.834	(+4.1%)▲	0.705	0.750	(+6.4%)▲
a@50	0.914	0.921	(+0.1%)	0.875	0.887	(+1.4%)	0.795	0.826	(+3.9%)▲

confirming the reputation of coordinate level matching to have a positive impact on early high precision. The impact of coordination level matching becomes particularly apparent for questions such as (4.2).

(4.2) When did Hawaii become a state? (topic id: 898)

Here, the corresponding idf-scores are *Hawaii* (7.73), *become* (2.54), and *state* (1.55). Clearly, *Hawaii* is dominant in any retrieval approach which is mainly based on idf-scores, such as Lnu.ltc weighting. But in the context of question answering also the more frequent terms *become* and *state* are essential for finding an answer, because the fact that a document just contains the term *Hawaii* is not very indicative as to whether it contains an answer to question (4.2). Coordination level matching treats all query terms in the same way, regardless of their respective idf-scores, and in combination with Lnu.ltc weighting the dominance of query terms with a high idf-score can be dampened.

However, the results for coordination level matchingare still worse than the results obtained by the original minimal span weighting scheme. In order to test whether including the span size ratio yields further significant improvements, we

Table 4.6: Comparison of the a@n scores of msw retrieval runs to clm runs

	TREC-9			TREC-10			TREC-11		
a@n	clm	msw		clm	msw		clm	msw	
a@5	0.733	0.789	(+6.7%)▲	0.688	0.736	(+7.0%)▲	0.569	0.630	(+10.7%)▲
a@10	0.827	0.860	(+4.0%)▲	0.769	0.829	(+7.8%)▲	0.686	0.729	(+6.3%)▲
a@20	0.879	0.918	(+4.4%)▲	0.834	0.873	(+4.7%)▲	0.750	0.800	(+6.6%)▲
a@50	0.921	0.939	(+2.0%)▲	0.887	0.903	(+1.8%)△	0.826	0.868	(+5.1%)▲

compared the clm retrieval results to the original msw results, see figure 4.6. In all cases, including the span size ratio does indeed result in significant improvements over the runs that did not use it.

Wilkinson et al. (1995) compared weighted coordination level matching (wclm) to regular coordination level matching, as discussed above, and their results show that weighted coordination level matching performs slightly better. Weighted coordination level matching considers the idf score of the matching query terms as is defined as

$$\mathrm{wclm}(d,q) = \frac{\sum_{t \in q \cap d} \log_2(N/df_t)}{\sum_{t \in q} \log_2(N/df_t)}$$

where N is the number of documents in the collection, and df_t is the number of documents in which term t occurs. We found that the results for using weighted coordination level matching are almost identical to those for using unweighted coordination level matching, and therefore we are dispensing with further details on the individual scores for the different data sets .

Summing up, the span size ratio makes a significant contribution to the improvements of the minimal span weighting scheme. But also using coordination level matching only in combination with global document similarity results in a significantly better performance than the Lnu.ltc baseline. This technique might be more appealing if efficiency or disk space issues are dominant factors in choosing a retrieval method for question answering pre-fetching.

4.4 Spans and Answerhood

Passage-based retrieval is widely used as a pre-fetch for question answering for two reasons. First, the answer to a question is normally expressed very locally, and using passages instead of whole documents takes the aspect of locality better into account. Whether passage-based retrieval is indeed more effective than document-based retrieval remains questionable as our experimental results in section 3.3.5 did not show any improvements. Second, returning passages instead of documents, allows later components of the QA system, such as answer extraction, to work on smaller and more focused text excerpts, thus reducing computational costs.

Similar to passage-based retrieval, minimal span weighting computes a text excerpt (a minimal matching span) which is used to re-weight the document it was

extracted from. In addition, it is also possible to return the minimal matching span instead of the document and have later components process the minimal span. The question is how useful is the minimal matching span for answer extraction, or to put it differently, how often does it contain a correct answer to a question?

Definition 4.2 of a minimal matching span, simply uses the positions of terms in a document, neglecting any kind of textual structure, such as sentence or paragraph boundaries. When using minimal span weighting for document retrieval this is indeed irrelevant, but when using the minimal matching spans for further processing one would like to have them obey at least sentence boundaries, which increases readability and enables them to be analyzed by a full parser. Additionally, it may happen that the answer is just to the left or right boundary of the minimal matching span, and the returned span would not include the answer, although the answer is in the same sentence as one of the span boundaries. In order to accomplish this, we extend each minimal matching span such that the left boundary is moved to the first word of the sentence in which it occurred, and the right boundary is moved to the last word of the sentence in which it occurred. Such an extended span is called a minimal sentential span, and it is formally defined as follows:

Definition 4.4 (Minimal matching sentential span) Let F_d be the set of positions of a first words of a sentence in document d, L_d be the set of positions of a last words of a sentence in document d and $mms_{q,d}$ is the minimal matching span in d for a query q, with the left boundary $b = \min(mms_{q,d})$, and right boundary $e = \max(mms_{q,d})$. The *minimal matching sentential span* is $(mms_{q,d} - \{b, e\}) \cup \{b', e'\}$, where $b' \in F_d$ and there is no $b'' \in F_d$ such that $b'' \leq b$ and $b'' > b'$, and where $e' \in L_d$ and there is no $e'' \in L_d$ such that $e'' \geq e$ and $e'' < e'$. ■

In practice, the extraction of a minimal matching sentential span also depends on the accuracy of the identification of sentence boundaries. A number of sentence splitters are available, see e.g., Palmer and Hearst (1994) and Reynar and Ratnaparkhi (1997). Here, we use our own sentence splitter, which uses the TreeTagger (Schmid, 1994) part-of-speech tagger to annotate the document. TreeTagger's tag set includes a sentence boundary tag, but in some cases sentence boundary tagging is incorrect and a number of manually constructed rules have been applied to correct this.

Returning to the use of minimal matching spans in the context of question answering, we reconsider the experiments discussed above, where minimal matching spans were used to rank documents, see section 4.3. For each of the top documents we know what the minimal matching span is and given that information, we computed the respective minimal matching sentential span. Before turning to the issue to what extent the minimal matching sentential spans contain answers to questions, their average lengths should be considered, because if the spans tend to be very long, the argument that they allow for a more focused analysis would be severely weakened. Table 4.7 shows the average and median number of words and bytes (characters) of the minimal matching sentential spans for different cut-off levels.

The first thing that jumps out is the large difference between average and me-

Table 4.7: Minimal matching sentential span lengths

The average (avg) and median (med) minimal matching sentential span lengths for the different TREC collections at cut-off levels 5, 10, 20, and 50, counted in words and bytes (characters).

	TREC-9				TREC-10				TREC-11			
	words		bytes		words		bytes		words		bytes	
	avg	med	avg	med	avg	med	avg	med	avg	med	avg	med
@5	65	36	396	225	56	34	345	215	77	39	467	236
@10	71	37	427	230	59	35	362	220	79	39	480	240
@20	72	38	435	233	62	36	378	221	84	40	506	247
@50	77	39	464	238	69	36	420	223	93	41	561	254

dian lengths; the former being roughly twice as large as the latter. This is due to a number of outliers with extremely long spans. Nevertheless, both average and median lengths are rather small and hence do allow for a focused analysis. Note that the numbers in table 4.7 roughly correspond to an average span length of 2–4 sentences and a median span length of 1 sentence.

The next question is to check how often the minimal matching sentential span does contain a correct answer. In order to evaluate this, one has to look at each span and decide whether this is the case. Obviously, this is a very laborious process and practically almost impossible, if done manually. One way to automatize this is to collect the known correct answers and simply apply pattern matching to see whether the minimal matching sentential span does match one of the correct answers. NIST provided a set of regular expressions that characterize the correct answers for the TREC-9 data set and Ken Litkowski did the same for the TREC-10 and TREC-11 data sets.[2] Table 4.8 lists some patterns from TREC-11. Some questions have simple string patterns (e.g., topic 1395) whereas some patterns are more complex (e.g., topic 1471). Many questions require a number of patterns allowing for small semantic differences (e.g., topic 1433) but other questions can also have a number of completely different answers (e.g., topic 1516).

Unfortunately, using these patterns to decide whether a minimal matching sentential span contains an answer is certainly not infallible. The span might contain an answer but not allow one to draw the conclusion that this is indeed a correct answer. For instance, consider question (4.3): Both minimal matching sentential spans, (4.4.a) and (4.4.b), contain the correct answer, but only (4.4.b) justifies it.

(4.3) Who is Tom Cruise married to? (topic id: 1395)

(4.4) a. This is the late Stanley Kubrick's swan song, but it'll be remembered as the film in which *Nicole Kidman* and Tom Cruise appear nude as married therapists pushing the envelope of sexual obsession.

b. Married actors Tom Cruise and *Nicole Kidman* play a loving, upscale married couple in Manhattan who are troubled by carnal temptations.

[2]The sets of answer patterns are available from the TREC web site: `http://trec.nist.gov`.

Table 4.8: Answer patterns for TREC-11

topic id	answers patterns
1395	Who is Tom Cruise married to?
	`Nicole Kidman`
1404	How many chromosomes does a human zygote have?
	`(46|23 pairs)`
1433	What is the height of the tallest redwood?
	`(367\.5|367 1/2)`
	`370\s?-\s?foot(-tall)?`
1454	How much money does the U.S. supreme court make?
	`\$\s*175,400`
	`175,400( U\.?S\.?)? dollars?`
1471	How fast does a cheetah run?
	`105 kilomet(er|re)s? per hour`
	`60 (m\.?p\.?h\.?|miles? per hour)`
1504	Where is the Salton Sea?
	`Calif(\s?\.|ornia)`
1516	What does CPR stand for?
	`cardio\s?-?\s?pulm(o|i)nary resuscitation`
	`Contraceptive Prevalence Rate`

Automatically evaluating whether a text snippet contains a correct answer is a notorious problem in building a reusable data collection for question answering, see, e.g., Breck et al. (2000); Maybury (2002); Voorhees and Tice (2000a). Voorhees (2000a) compared the ranking of QA systems at TREC-9 based on applying pattern matching with the official ranking that was based on human assessments and found a correlation, expressed as Kendall's τ, of 0.94 for the 250-byte runs and only 0.89 for the 50-byte runs. Nevertheless, at the current stage and in the current setting, using patterns to decide whether a text excerpt contains an answer to a question is the best approximation in automated evaluation.

In order to evaluate minimal matching sentential span extraction, two aspects have to be considered: First, does the span originate from a relevant document, and second, does the span contain a correct answer? The set of relevant documents for a question is defined as in section 3.2, and the set of spans containing a correct answer is identified by pattern matching. Given a cut-off level of n ($n \in \{5, 10, 20, 50\}$), R^+ (R^-) refers to the total number of relevant (non-relevant) documents for all questions, and S^+ (S^-) refers to the total number of spans containing (not containing) a correct answer. R^+S^+/R^+ is the number of relevant documents where the extracted span contains a correct answers divided by the total number of relevant documents. R^+S^+/R^+ indicates the ability of the span extraction to identify a text excerpt containing a correct answer, given a document that is known to contain a correct answer. On the other hand, R^-S^+/R^- is the ratio of spans from non-relevant docu-

ments that contain a correct answer. Table 4.9 shows the R^+S^+/R^+ and R^-S^+/R^- numbers for the different TREC collection at different cut-off levels. All in all, the

Table 4.9: Minimal matching sentential spans containing a correct answer

Percentage of minimal matching sentential spans from relevant documents (R^+S^+/R^+) and non-relevant documents (R^-S^+/R^-) containing a correct answer for different TREC data sets measured at cut-off levels 5, 10, 20, and 50.

	TREC-9		TREC-10		TREC-11	
	R^+S^+/R^+	R^-S^+/R^-	R^+S^+/R^+	R^-S^+/R^-	R^+S^+/R^+	R^-S^+/R^-
@5	70.0%	5.9%	65.3%	6.4%	64.1%	9.3%
@10	68.5%	6.5%	65.6%	8.1%	67.4%	9.1%
@20	70.2%	6.8%	68.0%	8.3%	67.1%	8.9%
@50	71.8%	7.5%	68.9%	8.7%	68.8%	7.9%

minimal matching sentential span is a relatively good starting point for answer extraction, because it contains the correct answer in 64.1–71.8% of the cases, but of course we hasten to add that this is still far from perfect. One can also see that in 5.9–9.3% of the cases, a span from a document which was not judged relevant does match a correct answer, but this number is hard to interpret: It could be that a document does contain a correct answer, but was simply not judged during the TREC evaluations, but it could also be the case that a document contains a string matching an answer without allowing one to draw the conclusion that it is indeed an answer to the question, as the text excerpt (4.4.a) exemplifies.

In the discussion above, we evaluated to what extent the minimal matching sentential spans contain a correct answer with respect to all relevant documents. The next issue is to see for how many of the questions the spans allow an answer selection procedure to find at least one correct answer. Assuming that answer selection is perfect, i.e., if a minimal matching sentential span contains a correct answer, then the selection procedure will find it, it allows one to determine an upper bound for the usefulness of the spans for question answering. Table 4.10 gives the percentages of questions where at least one minimal matching sentential span, which was extracted from a relevant document, contains a correct answer. In addition to the percentages also the mean reciprocal rank (MRR) is given. The reciprocal rank of a question is 1 divided by the highest rank at which a span from a relevant document contained a correct answer, and the MRR is the average of the questions' individual reciprocal ranks, cf. Voorhees (2000a). Here, we impose another constraint on the spans, namely that they are not longer than 250 or 500 bytes (characters). We restrict the span lengths, because the role of a minimal matching sentential span is to function as a 'hotspot' for answer selection which requires more expansive analysis, including parsing, named entity extraction, etc. If the span size is large, the property of being a 'hotspot' is lost. The numbers in table 4.10 show that a question answering system that would be based purely on minimal matching sentential spans is far from perfect, even when, as we are assuming here, the answer selection component

Table 4.10: Limited minimal matching sentential spans containing a correct answer

Percentage of questions, where at least one minimal matching sentential spans (not longer than 250/500 bytes) stems from a relevant document and contains a correct answer measured at cut-off levels 5, 10, 20, and 50.

	TREC-9		TREC-10		TREC-11	
	250 bytes	500 bytes	250 bytes	500 bytes	250 bytes	500 bytes
@5	52.1%	60.0%	46.9%	53.1%	32.0%	38.1%
@10	60.9%	67.6%	56.6%	63.5%	40.1%	48.2%
@20	65.3%	74.1%	61.7%	68.6%	45.7%	53.6%
@50	68.2%	77.5%	64.0%	72.3%	46.9%	57.7%
MRR	0.39	0.44	0.34	0.37	0.23	0.27

is flawless. Despite this, these results are roughly in the same ballpark as most of the better performing current QA systems, see Voorhees (2000a, 2001b, 2002).

Table 4.10 also exhibits a continuous drop in performance from TREC-9 to TREC-11. This drop can be expected as it is in line with the retrieval results described in this and the previous chapter, where performance also drops from TREC-9 to TREC-11. As mentioned earlier, the decrease in performance is mainly due to the fact that the questions used for the different TREC editions were getting more complex over the years.

In order to approximate the setting of the TREC-9 250-byte question answering track, we evaluated the minimal matching sentential span extraction against the original TREC-9 data set, which includes question variants, and also questions where none of the participating systems found a correct answer. Under this setting, the span extraction receives an MRR score of 0.37, where for 48% of the questions, none of the top 5 spans contained a correct answer. This would place the span extraction in the top ten of participating systems, see Voorhees (2000a), which is remarkable because minimal matching sentential span extraction does not use any answer finding strategies, nor does it carry out any question analysis.

4.5 Conclusions

Considering proximity between query terms when retrieving documents requires the indexation of positional information, which increases the size of the inverted index and results in a slight overhead in efficiency. On the other hand, the results in section 4.3 indicate that proximity-based retrieval exhibits significant improvements in effectiveness compared to regular Lnu.ltc retrieval.

Despite the significant improvements of the minimal span weighting scheme over Lnu.ltc weighting, it does not improve for all queries. In section 4.3.1, we looked into a few aspects that could cause the differences, but we were unable to extract features of a question that might predict to what extent a particular question

can benefit from minimal span weighting.

Although the minimal span weighting scheme, which is our implementation of proximity-based retrieval, also incorporates aspects of coordination level matching, it is the combination of both, proximity and coordination level matching, which yields the highest improvements. Disregarding positional information and relying on coordination level matching only also leads to significant improvements over the baseline, see section 4.3.2, but the effectiveness is still significantly lower than the performance of a combined approach. If disk space is an issue, or positional information cannot be integrated for some other reason, retrieval methods do benefit from coordination level matching alone, but not to the same degree as from the combination of term proximity and coordination level matching.

Our minimal span weighting approach also allows the retrieval system to identify small text segments that can function as starting points for further processing steps, such as answer selection. In section 4.4, it was shown that the minimal matching spans contain a correct answer in 64.1–71.8% of the cases, where the document is known to contain a correct answer.

Summing up, proximity-based retrieval does significantly improve document retrieval as a pre-fetch to a question answering system, and it is useful for finding short text segments in a document that are likely to contain a correct answer.

In this chapter, we did not address the issue to what extent minimal span weighting has an impact on the retrieval performance in tasks other than document retrieval as a pre-fetch to a question answering system. One might suspect that it should have a positive impact on any retrieval task where the information need is rather specific, or where early high precision is required. During our experimentation we have also applied minimal span weighting to the TREC-11 named page finding task, where a retrieval system is supposed to find a unique web page, given a topic which describes it by name, cf. (Craswell and Hawking, 2002). Using minimal span weighting for this task resulted in an MRR score of 0.513, whereas the Lnu.ltc baseline MRR score was 0.359, which is an improvement of 43%, and is siginificant at a 99% confidence level. Applying minimal span weighting to other tasks and evaluating its effectiveness remains to be done.

Chapter 5

Learning Query Term Selection

The formulation of queries that are used for retrieving documents that contain answers to a question has a strong impact on the effectiveness of the retrieval component. In this chapter, we focus on the selection of terms from the original question. We use model tree machine learning techniques in order to assign weights to query terms according to their usefulness of identifying documents that contain an answer. The resulting model trees also provide futher information on the underlying regularities that determine the weights of query terms. The learning of query term weights is integrated into our computation of document similarity and evaluated against the TREC data sets.

The way queries are formulated has a severe impact on retrieval effectiveness. In particular, boolean retrieval is very sensitive to query formulation. Using certain words from the question in the actual retrieval query can steer the retrieval process in a wrong direction. For instance, consider question (5.1).

(5.1) What is the abbreviation for the London stock exchange? (topic id: 1667)

Should the word *abbreviation* be included in a retrieval query? If included, it will be the most dominant term in the query, because it is much less frequent than the other terms in the question, and will therefore receive a high term weight.[1] However, most documents that contain an answer to question (5.1), express it in the form of '...London Stock Exchange (LSE)...,' not using the term *abbreviation* or one of its morphological variants at all. Hence, in boolean retrieval, a query such as (5.2.a) might be too strict and result in an empty set of retrieved documents.

[1] The term frequencies were computed on the TREC document collections.

(5.2) a. `abbrevi AND london AND stock AND exchang`

b. `abbrevi london stock exchang`

In vector-space retrieval, a query such as (5.2.b) will rank documents containing the term *abbrevi* higher than documents that do not contain it, although, as discussed above, most documents providing an answer to question (5.1) do not contain it.

In addition to selecting query terms from the original question, in many cases, retrieval effectiveness can benefit from adding terms which are not mentioned in the question. For instance, for question (5.3)

(5.3) What is the temperature at the center of the earth? (topic id: 927)

most documents that contain an answer refer to *the core of the earth* instead of *the center of the earth*. Including the term *core* will help finding those documents. Research on expanding queries with semantically related terms has a long tradition within information retrieval. Numerous approaches have evolved ranging from using static global resources, such as thesauri (Voorhees, 1994) and co-occurrence lists (Peat and Willett, 1991), to using local strategies, such as blind relevance feedback (Buckley et al., 1994) and local context analysis (Xu and Croft, 1996). In this chapter, we want to focus on selecting terms from the question, but we will return to the issue of expanding queries—albeit in a very restricted way—in the next chapter.

In general, there are two ways to choose terms to formulate a query. One way is to select terms from the original question, and the other way is to expand the original question with terms. In this chapter, we will focus on term selection. The issue of term expansion is discussed in the next chapter.

The remainder of this chapter is organized as follows: The next section reviews some of the previous approaches to query formulation in document retrieval as a pre-fetch to a question answering system, and earlier work on learning query term weights for ad hoc document retrieval. Section 5.2 discusses the impact optimal query formulation can have on retrieval as a pre-fetch to question answering. Section 5.3 discusses how query term weights can be computed using previous TREC data sets for training. In section 5.4, we discuss the way question words can be represented by sets of features, so as to abstract from the actual words themselves. Section 5.5 briefly introduces some state-of-the-art machine learning approaches and motivates our choice for using model tree learning. Experimental results are discussed in section 5.6. Finally, section 5.7 provides a discussion of the results and general conclusions.

5.1 Related Work

Previous work on query formulation for question answering has mainly been done for web question answering. Kwok et al. (2001a) and Brill et al. (2002) focus on formulating query strings that approximate the way an answer is likely to be expressed. In particular this involves automated syntactical transformation, mapping

the syntax of an interrogative to the syntax of a declarative sentence. They did not investigate the issue of term selection. For instance, the ASKMSR system (Brill et al., 2002) generates for the question (5.4.a) the queries in (5.4.b).

(5.4) a. Where is the Louvre Museum located?

b. ''the Louvre Museum is located''
''the Louvre Museum is in''
''the Louvre Museum is near''
''the Louvre Museum is''
Louvre AND Museum AND near

Documents are required to strictly match one of the strings or the boolean query, but the issue whether, e.g., *Museum* is actually a good query term is not addressed.

Paşca (2001) does address the issue of term selection and term relevance. His work is closely related to the work presented in this chapter. For query formulation, he distinguishes between three type of terms: high-relevance, medium-relevance, and low-relevance query terms. Deciding to which class a given term belongs is based on a number of rules, some of which are also integrated in our approach.

To the best of our knowledge, machine learning techniques have not been applied before to query formulation in the context of question answering, but they have been applied in the context of ad hoc retrieval. Cooper et al. (1993) use logistic regression to assign weights to matching clues, such as the number of times a query term occurs in the query, the number of times a query term occurs in a document, the idf score of a matching term, and the number of distinct terms common to both query and document. In addition, they assigned weights to query terms in case some relevance information is available, as document routing or feedback retrieval. Chen et al. (1998) applied machine learning techniques for selecting query terms in the context of relevance feedback retrieval.

5.2 Optimal Query Term Selection

In this section we estimate the effect query formulation, in the form of term selection, can have on retrieval performance, by using the TREC data sets. In order to compute the optimal term selection for each question, we compare all possible ways of selecting terms from a question. I.e., given a question q in which the set of terms T occurs, we consider all possible subsets of T, and evaluate the respective performances. More formally, the set of term selection variants is defined as:

$$tsv(q) = POW(T) - \{\emptyset\}$$

where $POW(T)$ is the power set, i.e., the set of all subsets, of the set T. For obvious reasons, the empty subset is disregarded. Consider question (5.5.a), which contains the (stemmed) terms in (5.5.b).

(5.5) a. What is the chemical formula for sulphur dioxide? (topic id: 1442)

b. $T = \{\texttt{chemic}, \texttt{dioxid}, \texttt{formula}, \texttt{sulphur}\}$

Since $|T| = 4$, there are $2^4 - 1 = 15$ term selection variants. For each of the query variants a retrieval process is carried out, and the average precision is computed.

In the actual retrieval queries, all terms are required to be present in a document. E.g., the retrieval query corresponding to (5.5.b) is

```
chemic AND dioxid AND formula AND sulphur
```

Table 5.1, lists all possible selection variants for question (5.5.a) sorted by their respective average precision. The query variants can be evaluated with respect to

Table 5.1: Performances of term selection variants

rank	avg. prec.	query variant
1	0.0285	`dioxid, sulphur`
2	0.0196	`chemic, dioxid, sulphur`
3	0.0180	`sulphur`
4	0.0086	`chemic, dioxid`
5	0.0078	`dioxid`
6	0.0032	`chemic, sulphur`
7	0	`chemic, formula`
8	0	`chemic, formula, sulphur`
9	0	`chemic, dioxid, formula, sulphur`
10	0	`dioxid, formula`
11	0	`formula`
12	0	`formula, sulphur`
13	0	`chemic`
14	0	`dioxid, formula, sulphur`
15	0	`chemic, dioxid, formula`

a number of evaluation measures. Here, we used average precision, because it is widely used and combines precision and recall. Given a query q, its set of relevant documents REL_q and a ranking of documents ($rank_q : D \rightarrow \mathbb{N}$) resulting from the retrieval process, average precision of an individual query is defined as:

$$\text{avg\_prec}(q) = \frac{\sum_{d \in \text{REL}_q} p@rank(d)}{|\text{REL}_q|}$$

Here, $p@rank(d)$ is defined as $p@n$, where $n = rank(d)$. Note, that we do not use the $a@n$ measure to rank query variants for two reasons. First, given some instantiation of n, the $a@n$ measure is less discriminative than average precision, because for a given query variant, the $a@n$ will always either take the values 0 or 1. Second, the weight of a query variant would strongly depend on the choice of n, and it is hard to estimate which instantiation of n would be the most appropriate.

The total number of query variants for all queries of the three data sets are 7587 (TREC-9), 6975 (TREC-10), and 11957 (TREC-11). For each query in the three data sets, we determined the query variant with the highest average precision. Table 5.2 shows the performance gains that can be achieved if all retrieval queries are formulated optimally. As one could expect, query formulation has a significant impact

Table 5.2: Comparison of the a@n scores of optimal retrieval queries to baseline runs

	TREC-9			TREC-10			TREC-11		
a@*n*	Lnu.ltc	opt		Lnu.ltc	opt		Lnu.ltc	opt	
a@5	0.700	0.823	(+17.6%)▲	0.649	0.749	(+15.4%)▲	0.523	0.690	(+31.9%)▲
a@10	0.785	0.890	(+13.4%)▲	0.734	0.815	(+11.0%)▲	0.626	0.767	(+22.5%)▲
a@20	0.845	0.921	(+9.0%)▲	0.801	0.887	(+10.7%)▲	0.705	0.824	(+16.9%)▲
a@50	0.914	0.956	(+4.6%)▲	0.875	0.924	(+5.6%)▲	0.795	0.881	(+10.8%)▲

on the overall performance of a retrieval system, even if query formulation is just based on term selection without expanding the queries with semantically related terms. The figures in table 5.2 refer to results that were achieved by determining the optimal query formulation in hindsight, and the problem of identifying an optimal query without having any relevance assessments remains to be solved. In the remainder of this chapter we explore ways leading to optimal formulation.

5.3 Computing Query Term Weights

Our approach is to use the different query variants of a question to determine terms that are more helpful for retrieving relevant documents, and terms that harm the retrieval effectiveness for that particular question.

In the previous section, we considered only one single best-performing query variant, but in many cases there are several almost equally well-performing query variants. A look at the ranked queries variants, shows that some terms occur more frequently in higher-ranked query variants than other terms. Consider for instance table 5.1, where the terms *dioxid* and *sulphur* occur more often than the term *chemic* in high ranked variants, and the term *formula* only occurs in variants that did not retrieve any relevant documents.

An analysis of the distribution of query terms over the ranked query variants allows one to assign a weight to the query terms: If a term occurs mainly in query variants that have a high average precision it should receive a high weight, whereas a term that occurs mainly in query variants that have a low average precision should receive a low weight. Thus, the weight of a query term depends on two factors: The average precisions of the query variants in which the term occurs (its presence weight: $w_+(t)$), and the average precisions of the query variants in which the term does not occur (its absence weight: $w_-(t)$). Both, the presence and the absence weight are normalized by the sum of the average precisions of all query variants, so the weights will range between 0 and 1.

Definition 5.1 (Term presence and absence weights) Given a question q and all its query variants $tsv(q)$, the *presence weight* of term t ($w_+(t)$) is computed as:

$$w_+(t) = \frac{\sum_{q' \in tsv(q) \wedge t \in q'} \text{avg\_prec}(q')}{\sum_{q' \in tsv(q)} \text{avg\_prec}(q')}$$

Analogously, the *absence weight* of term t ($w_-(t)$) is computed as:

$$w_-(t) = \frac{\sum_{q' \in tsv(q) \wedge t \notin q'} \text{avg\_prec}(q')}{\sum_{q' \in tsv(q)} \text{avg\_prec}(q')}$$

■

The presence and the absence weight of a term t, can be combined into a single weight by subtracting the absence weight from the presence weight, which we call the *gain* of term t: $gain(t) = w_+(t) - w_-(t)$. If a query term has a positive gain it should be included in the query, but if its gain is negative, inclusion will hurt retrieval. Note, that the gain of term t always lies in the interval $[-1, 1]$.

Let us return to question (5.5.a) and its query variants in table 5.1. The presence and the absence weight, as well as the gain of each term, are shown in table 5.3. The gains of the query terms confirm the observation made earlier that *sulphur* and

Table 5.3: Example term weights

t	$w_+(t)$	$w_-(t)$	$gain(t)$
sulphur	0.808	0.192	0.616
dioxid	0.752	0.248	0.506
chemic	0.367	0.633	−0.266
formula	0.000	1.000	−1.000

dioxid are better query terms than *chemic* and *formula*.

This approach of computing term weights is based on the assumption that the terms in a question occur independently of each other and therefore the weight of a term can be computed without considering other terms in the question. Of course, this assumption does not hold in practice, but it allows us to keep the computation of term weights simple. The issue of term (in)dependence is a recurring issue in information retrieval, see, e.g., (Robertson, 1977; Robertson and Sparck Jones, 1976; Salton et al., 1982; Cooper, 1995).

5.4 Representing Terms by Sets of Features

In the previous subsection, the computation of the term weights was based on the distribution of the terms themselves over the query variants. This is problematic

for two reasons. First, the same term can have a high gain in one query, and a low gain in another. Second, if the learning algorithm is based on the surface terms themselves, it cannot assign weights to terms that did not occur in the training data. The first problem is a direct consequence of the term independence assumption. This problem could be solved by conditioning the weight of a term on a number of terms that also occur in the question, but then the second problem—how to assign weights to unseen data—becomes even more severe.

One way to address both problems is to represent terms and their contexts in a more abstract manner. Here, we use a set of features that represent certain characteristics of a query term and its role in a question. The list of features contains information about the term's part-of-speech, whether it semantically includes other terms in the question, the type of question it occurs in, etc. As mentioned above, some of the features capture aspects inherent to a term, such as part-of-speech, while others capture contextual aspects, such as semantic inclusion. Table 5.4 lists all features with a short specification of their respective values.

We will now discuss the features in more detail. Some of these features can also be found elsewhere in the literature, see, e.g., (Paşca, 2001). In particular, the specification of the features *question focus*, *superlative*, *quoted*, *number of leaves*, *modified noun*, and *person name* is based on (Paşca, 2001). All features are motivated by inspecting the TREC data, where we considered questions and documents that contain an answer. This does not imply that using different or more features will not be beneficial for selecting query terms.

Part-of-Speech. The part-of-speech feature can take values such as `NNP` (proper name, singular), `JJS` (superlative adjective), `VBZ` (verb in present tense, third person singular), etc. These are the standard part-of-speech texts from the Penn Treebank (Santorini, 1990).

Part-of-speech tagging is accomplished by using TREETAGGER (Schmid, 1994), a decision-tree-based tagger. The general parameter setting of TREETAGGER, which is based a newspaper training set, turned out to be inappropriate for tagging questions. This is due to the difference in word order between interrogative and declarative sentences. In order to improve the performance of TREETAGGER for questions, we trained it on 700 questions, 328 of which were taken from the Penn Treebank corpus,[2] and the remaining 482 were questions from the TREC-9 data set. Whereas the Penn Treebank questions were already manually part-of-speech tagged, we had to tag the questions from the TREC-9 data set ourselves. Although we did not evaluate the increase in performance of the tagger, inspecting randomly chosen questions indicated a clear improvement in tagging accuracy.

The actual values of the `part-of-speech` feature are a slight simplification of the Penn Treebank tags. For example, we do not make a distinction between singular and plural nouns, i.e., `NNP` and `NNPS` are mapped onto `NNP`, and `NN` and `NNS` are

[2]Distributed by the Linguistic Data Consortium: `http://www.ldc.upenn.edu/`.

Table 5.4: List of features for question words

Feature	Values
part-of-speech	A fixed list of part-of-speech tags from the Penn Treebank tag set
question focus	A value between 0 and 1 indicating whether the word is part of the question focus
superlative	A boolean value indicating whether the question contains a superlative adjective
question class	A fixed list of question classes
multiple occurrences	A boolean value indicating whether the word occurs more than one in the question
quoted	A boolean value indicating whether the word occurs between quotation marks
no. leaves	The number n ($n \geq 0$) of hypernyms of the word in the WordNet hierarchy that do not have any further hyponyms themselves
term ratio	$1/m$, where m is the number of unique terms in the question
classifying word	A boolean value indicating whether the word was used to classify the question
location	A boolean value indicating whether the word is part of a location name
abbreviation	A boolean value indicating whether the word is an abbreviation
upper case	A boolean value indicating whether the word starts with an uppercase letter
modified noun	A boolean value indicating whether the word is a noun that is preceded (modified) by another noun
person name	A fixed set of values indicating what part of a person's name the word is, if applicable
honorific	A boolean value indicating whether the word is a honorific term
no. incoming edges	A natural number indicating the number of edges pointing to a word in the dependency parse graph of the question
hypernym	A boolean value indicating whether the word is a hypernym of another word in the question
relative idf	A real value indicating the relative frequency of the word in the document collection compared to the frequencies of the other words in the question

mapped onto `NN`. Also, the different inflections of a verb are disregarded, and all verb forms are represented by a single tag `V`.

Question Focus. The focus of a question is a phrase describing a type of which the answer is an instance. For example, in question (5.6), the focus is *country*, in (5.7), it is *peninsula*, and in (5.8), it is *college*.

(5.6) In what country did the game of croquet originate? (topic id: 1394)

(5.7) What is a peninsula in the Philippines? (topic id: 1423)

(5.8) What college did Magic Johnson attend? (topic id: 1449)

The answer to question (5.6), which is *France*, is an instance of *country*, i.e., *France* is a *country*; analogously for the other two examples.

Whether a word is part of the question focus has consequences for the query formulation, because many documents containing an answer to the question do not explicate the instance relation. For instance, the fact that France is a country is taken to be common knowledge and therefore seldomly stated explicitly in a document. Hence requiring a document to contain words from the question focus can harm retrieval.

Note that the term *question focus* in the way it has been used in the literature on question answering does not necessarily coincide with its definition in the linguistic literature. In question answering, the question focus is sometimes also referred to as *answer type term*.

The question focus feature can take three values: 0, 0.5, and 1. If a word is not part of the question focus, the feature is set to 0. If a word is part of the question focus and the semantic head of a noun phrase, it is set to 1. For words that are part of the question focus, but are not the semantic head of a noun phrase, the feature is set to 0.5. For instance in question (5.9), the focus feature is set to 1 for the noun *explorer*, which is the semantic head of the noun phrase *Spanish explorer*, and the focus feature is set to 0.5 for the modifying adjective *Spanish*.

(5.9) What Spanish explorer discovered the Mississippi River? (topic id: 1411)

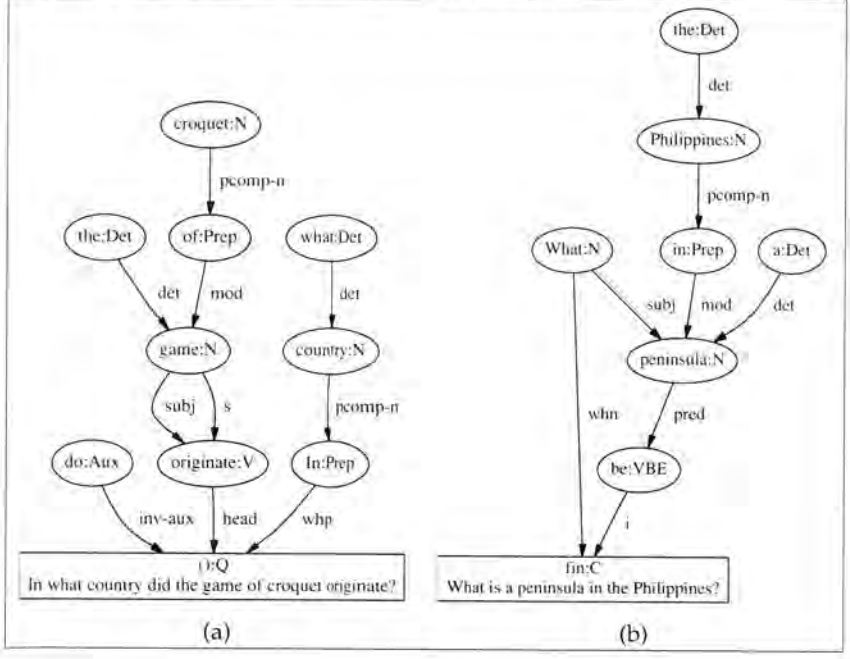
(5.10) What mythical Scottish town appears for one day every 100 years? (topic id: 1399)

For question (5.10), the focus feature is set to 0.5 for both modifying adjectives *mythical* and *Scottish*. We do not make a distinction whether a modifying word immediately precedes the head or whether there are more words between them. The semantic head is simply identified as the rightmost word of the noun phrase in the question focus, cf. (Williams, 1981).

In order to determine the focus of a question we used MINIPAR (Lin, 1998), a dependency parser. MINIPAR is a robust full parser which is able to cover about 87% of the dependency relationships in the SUSANNE evaluation corpus (Sampson, 1995) with about 89% precision, cf. (Lin and Pantel, 2001). A directed arc in a dependency graph going from node x to node y means that node x modifies node y. The arcs in

the dependency graph carry labels that indicate the type of modification. To determine the question focus, we use only a small portion of the dependency graph of a question. In particular, we focus on the outgoing arcs of nodes representing the wh-words *what* and *which*, because they modify the question focus. In figure 5.1, the dependency graphs that are generated by MINIPAR for question (5.6) and (5.7).[3] The

Figure 5.1: Examples of MINIPAR dependency graphs for questions



dependency graph in figure 5.1.a, illustrates a trivial situation, where the wh-word *what* modifies the noun *country* as a determiner. In question (5.7), focus determination is slightly more complicated. In the corresponding dependency graph, see figure 5.1.b, the wh-word *what* modifies the noun *peninsula* as a subject. Note, that in the MINIPAR representations, the `subj` (subject) arcs refer to the logical subject, and not to the syntactic subject. Two kind of modifier relations are used to identify the question focus: `det` (determiner), and `subj` (subject).

Superlative. Despite our earlier comments, under certain circumstances, words from the question focus can be relevant for query formulation. In particular, if the

[3]The graphs were constructed using AT&T's Graphviz graph displaying tool: `http://www.research.att.com/sw/tools/graphviz/`.

question contains a superlative. In question (5.11), the focus *island*, and in question (5.12), the focus *lake* are modified by a superlative adjective.

(5.11) What is the world's second largest island? (topic id: 1503)

(5.12) What is the deepest lake in America? (topic id: 1540)

In these cases, the instance relation, viz. that *New Guinea* is an island and that *Crater* is a lake, is likely to be expressed explicitly in the document containing an answer, e.g., ... *Crater Lake is America's deepest lake*. In order to be able to make this distinction the feature `superlative` indicates whether the question contains a superlative adjective.

Detecting whether a question contains a superlative adjective relies on the output of the part-of-speech tagger. If there is at least one word which is tagged as `JJS`, which is the Penn Treebank tag for superlative adjective, the `superlative` is set to 1, and 0 otherwise.

Question Class. The decision to select a term for query formulation is to some extent also based on the type of question. Question classes, or question types, specify the kind of answer the question is asking for. Question types include categories such as `date` (when did something happen?), *location* (where is something?), *agent* (who did something?), etc. For example, consider questions (5.13) and (5.14).

(5.13) Who started the Protestant reformation? (topic id: 1563)

(5.14) When did the Black Panther party start in California? (topic id: 1567)

Both questions contain the term *start*, but question (5.13) is of type `agent`, and question (5.14) is of type `date`. It turns out that including the term *start* is more important in formulating the retrieval query for question (5.13), where the gain for *start* is 1.0, than for question (5.14), where the gain for *start* is 0.094. This might be due to the fact that starting dates can be expressed in several ways.

To identify the question class, often also referred to as the *question target*, pattern matching is applied to assign one of 33 categories to the question. In total, a set of 102 patterns is used to accomplish this. Some of the patterns used are shown in Table 5.5; see also chapter 1 for some more examples. The patterns are applied in an ordered manner so that more specific patterns match first. These patterns were generated manually by inspecting a sample of the TREC questions. Also, the answer selection component described in the next subsection obeys the order in which questions were categorized to find answers for more specific targets first.

Multiple Occurrences. As mentioned above, if a word occurs in the question focus, including it in the query may harm retrieval performance. For instance, the query for question (5.15) should not contain the term *state*.

(5.15) Which U.S. state is the leading corn producer? (topic id: 1450)

Table 5.5: Types for question classification

Question target	Example patterns
name	/[Ww]hat(wa| i|\')s the name/
pers-def	/[Ww]ho(wa| i|\')s [A-Z][a-z]+/
thing-def	/[Ww]hat(wa| i|\')s an? /,/ (was|is|are|were) a kind of what/
pers-ident	/[Ww]ho(wa| i|\')s the/
thing-ident	/[Ww](hat|hich)(wa| i|\')s the /
number	/[Hh]ow (much|many) /
expand-abbr	/stand(s)? for(what)?\s*?/,/the abbreviation .+ mean\s*?/
find-abbr	/[Ww]hat(i|\')s (the|an) (acronym|abbreviation) for
agent	/[Ww]ho /,/ by whom[\.\?]/
object	/[Ww]hat (did|do|does) /
known-for	/[Ww]hy .+ famous/ /[Ww]hat made .+ famous/
aka	/[Ww]hat(i|\')s (another|different) name /
name-instance	/Name (a|one|some|an) /
location	/[Ww]here(\'s)? /,/ is near what /
date	/[Ww]hen /,/[Ww](hat|hich) year /
reason	/[Ww]hy /
what-np	-
unknown	-

(5.16) What state is the geographic center of the lower 48 states? (topic id: 1053)

On the other hand, if a word occurs in the question focus and also outside of it, excluding that term from the query may harm the results. In question (5.16), the term *state* occurs twice. Note, that although the second occurrence is the plural form of *state*, after morphological normalization, such as stemming, both occurrences are mapped to the same term. Whether a word occurs more than once is captured by the boolean feature `multiple occurrences`.

Quoted. Words that occur between quotation marks require special consideration. Quoted phrases often refer to titles of movies or theater plays, nicknames, etc. Many words that do not bear much content, and therefore are not very helpful for retrieval, are critical for retrieval if they occur in a quotation. For instance, in question (5.17), the words *gone*, *with*, and *the*, are highly frequent terms.

(5.17) What is the name of the heroine in "Gone with the Wind"? (topic id: 1478)

However, not selecting them for query formulation would only leave the word *wind* as a description of the movie title, which is certainly insufficient. In order to distinguish between words that occur in a quotation and those that do not, the boolean feature `quoted` is set to 1 if a word is quoted and 0 otherwise.

Number of Leaves. As discussed above, including words that appear in the question focus often harms retrieval. But the extent to which including question focus words harms retrieval, also depends on the generality of the word. For instance, in question (5.18), the question focus is *person*, which is a very general term, and including it into the query is likely to harm retrieval.

(5.18) What person developed COBOL? (topic id: 1595)

(5.19) What Spanish explorer discovered the Mississippi River? (topic id: 1411)

In question (5.19), the term *explorer* is rather specific and it is likely that the answer document makes the fact explicit that *Hernando de Soto* (the correct answer to the question), is an explorer. Whereas it is rather unlikely that a document containing the answer to question (5.18) explicitly states that *Grace Hopper* (the correct answer) is a person.

There are several ways to measure the generality of a term. Here, we use WordNet to count the number of concepts that are hyponyms of the question focus and that do not have any hyponyms themselves. A concept x is a hyponym of concept y, if x is a y. If the question focus is ambiguous, i.e., it belongs to several concepts, we take the sum of all hyponyms of all concepts the word belongs to. The feature `no. leaves` provides the number of hyponyms.

For instance, the term *person* occurs in three concepts in WordNet, which in total have 5765 leaves, whereas the term *explorer* occurs in one concept, which has 3 leaves. One can conclude that the term *person* is much more general than the term *explorer*, and hence less likely to occur explicitly in an answer document.

Term Ratio. A more general aspect of query formulation is the length of the original question. If a question contains many words, leaving one out in formulating the query has less of an impact on the effectiveness of the retrieval process than for questions that contain only two or three terms. The feature `term ratio` expresses the length of the original question (after removing general stop words), as its reciprocal: $1/m$, where m is the number of question words.

Classifying Word. Certain words are good indicators for classifying a question. For instance, in question (5.20), the word *abbreviation* in combination with the word *mean* indicates that the question is of type `expand-abbreviation`.

(5.20) What does the abbreviation WASP mean? (topic id: 1727)

(5.21) What is the height of the tallest redwood? (topic id: 1433)

(5.22) What province is Calgary located in? (topic id: 1845)

Similarly, the word *height* in question (5.21) indicates that the question is of type `height`, and the word *located* that question (5.22) is of type `location`. However, words that are good indicators for question classification, are infrequent in the way

answers are expressed in documents. For instance, it is very unlikely that the word *located* is used in a declarative sentence that answers question (5.22).

Whether a word is a classifying word depends also on the question category of the question at hand. If the question category is `expand-abbr`, the words *stand*, *abbreviation*, and *mean* are classifying words, but if the question category is `known-for`, the words *famous* and *made* are classifying words, see table 5.5. The classifying words are extracted from the patterns that are used for question classification.

Location. The location feature indicates whether a word is part of a location name. For many questions, it is essential to include into the query words that are part of a location name, in order to find an answer. For instance, in question (5.23), the location words *San*, *Antonio*, and *TX* are relevant terms as the question refers to the temperature of that particular location.

(5.23) What is the highest recorded temperature in San Antonio, TX?
(topic id: 1770)

(5.24) When was the Buckingham Palace built in London, England?
(topic id: 1809)

On the other hand, in question (5.24), the location words *London* and *England* seem to be superfluous as it is relatively well-known that Buckingham Palace is in London, and it is common knowledge that London is a city in England.

In order to recognize locations, we use the CLR gazetteer[4], which is a large list of locations, including cities, counties, harbors, countries, etc., containing 162,853 entries in total.

Abbreviation. If the word is an abbreviation, the value of this feature is set to 1, and 0 otherwise. If a question asks for the definition of an abbreviation, such as question (5.25), the abbreviated term obviously has to be included in the query.

(5.25) What does HTML stand for? (topic id: 1774)

(5.26) When is Fashion week in NYC? (topic id: 1756)

(5.27) What TV series did Pierce Brosnan play in? (topic id: 1768)

This is less the case for questions that do not ask for the full form of an abbreviation. In question (5.26), the word *NYC*, and in question (5.27) the word *TV* are abbreviations. Documents containing an answer do not necessarily have to contain the abbreviated word—in contrast to answer documents for question (5.25)—but they might as well contain the full form, i.e., *New York City*, and *television*, respectively.

Abbreviations are recognized by applying simple pattern matching. If a word consists of a series of capitalized letters, where each might be followed by a period,

[4]Available from `http://crl.nmsu.edu/Resources/clr.htm/`.

or a series of letters, where each is followed by a period, or occurs in a list of known abbreviations, the word is classified as an abbreviation. Maintaining a list of known abbreviations is necessary to recognize words as *mph* as an abbreviation.

Upper Case. Words starting with a capital letter are normally part of a proper name, even when the word itself is not a noun.

(5.28) Who was Woodrow Wilson's First Lady? (topic id: 1622)

(5.29) What group sang the song "Happy Together"? (topic id: 1675)

In question (5.28), the adjective *First*, is part of the proper name *First Lady*, and in question (5.29), the adjective *Happy* and the adverb *Together* are part of the proper name *Happy Together*. Proper names are particularly important query terms and and for recognizing them as such, it is not sufficient to rely on part-of-speech tags. Whether the fact that a part-of-speech tagger (TREETAGGER in our case) tags the word *First* as adjective and not as proper name has to be considered a mistake or not is difficult to say. From a syntactic point of view, *First* is clearly an adjective, and it is by convention that it is used a a proper noun in this specific context. Anyway, using the upper case feature allows us to recognize proper names that are not part-of-speech tagged as such.

Modified Noun. The information content of modified nouns is higher than the content of single nouns, because the modifier imposes additional restrictions. In question (5.30), the noun *performer* is modified by the noun *child*, and in question (5.31) the noun range is modified by the nouns *blood* and *sugar*.

(5.30) Who holds the record as the highest paid child performer? (topic id: 1602)

(5.31) What is the normal blood sugar range for people? (topic id: 1607)

In contrast, in question (5.30), the noun *record*, and in question (5.31), the noun *people* are unmodified.

A noun is marked as modified by inspecting the part-of-speech tagged question. If a noun is preceded by an adjective, noun, or possessive, the `modified noun` feature is set to `yes`, and otherwise it is set to `no`. If the word is not tagged as a noun, the feature is set to `na` (not applicable). (5.32.b) is the part-of-speech tagged output of TREETAGGER when applied to question (5.32.a). Here, *blood*, *sugar*, and *range* are all preceded by either an adjective or a noun. Hence, the `modified noun` feature is set to `yes` for the three words. *people* is preceded by a preposition, and therefore the `modified noun` feature is set to `no`. For all other words in the question, the feature is set to `na`.

(5.32)a. What is the normal blood sugar range for people? (topic id: 1421)

b.

What	is	the	normal	blood	sugar	range	for	people	?
WRB	VBZ	DT	JJ	NN	NN	NN	IN	NN	SENT

Note that our definition of modification is rather simple and does not take any internal phrase structure into account. We pursue a simple linear approach to modification, which seems sufficient as we are not interested in the exact phrase that modifies a noun, but simply have to decide whether a noun is modified or not. This also allows us to circumvent the problem of phrase structure ambiguity.

Person Name. Words that are part of a person name are a special instance of words that are part of a proper name. Person names deserve special attention, because they can be further subdivided into first, middle, and last names.

(5.33) What is Francis Scott Key best known for? (topic id: 207)

(5.34) When did George W. Bush get elected as the governor of Texas? (topic id: 1584)

In question (5.33), *Francis* is the first name, *Scott*, the middle name, and *Key*, the last name. Often, the middle name is abbreviated by using the first letter only, as in question (5.34), where the *W.* stands for *Walker*. The distinction between the different parts is important, because in many documents, the full name of a person is only used the first time the name occurs, and then later on referred to by using the last name only. Hence, last names are more important for finding an answer.

To identify person names, we use part of the U.S. Census resource,[5] which contains a list of first and last names. The list of first names contains 4,275 female and 1,219 male first names, and 101,865 last names.

Honorific. Honorific expressions include words such as *Mr.*, *Mrs.*, *Dr.*, etc. These terms do not bear much information and are therefore not essential for formulating a query.

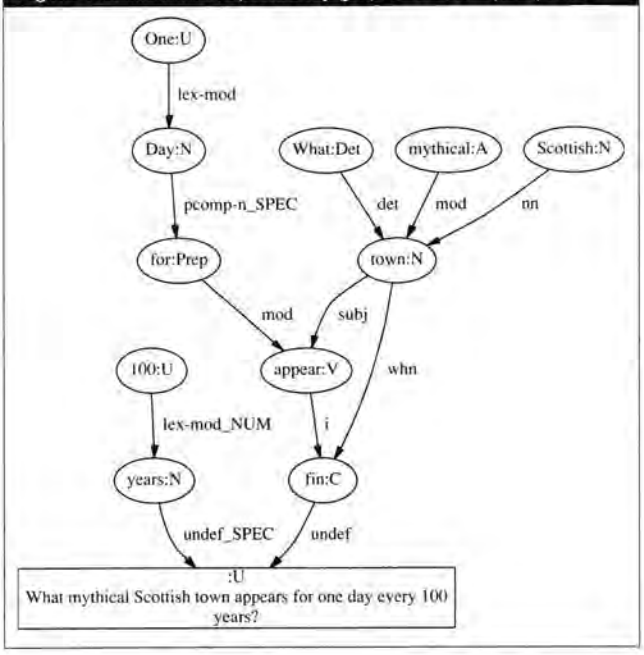
(5.35) Where did Dr. King give his speech in Washington? (topic id: 1559)

In question (5.35), insisting on the presence of the honorific word *Dr.* in a potential answer document is too restrictive, as many documents refer to Martin Luther King without using an honorific expression.

Number of incoming edges. The number of incoming edges refers to the dependency parse graph of the question which is generated by MINIPAR. If a word has a larger number of incoming edges, several words in the question are in a modifier or argument relationship with this word, and therefore it is more likely to play a central role in the question. Figure 5.2 shows the dependency graph for question (5.36).

(5.36) What mythical Scottish town appears for one day every 100 years? (topic id: 1399)

Figure 5.2: MINIPAR dependency graphs for example question



The verb *appear* has two incoming edges, and the noun *town* has three. Nodes in the graph which are not associated with a word in the question, such as the *fin* node, are not considered.

Hypernym. Sometimes questions contain words that explicitly give the type of other words in the question. For instance, in question (5.37), *croquet* is classified by the word *game*.

(5.37) In what country did the game of croquet originate? (topic id: 1394)

(5.38) What is the name of the volcano that destroyed the ancient city of Pompeii? (topic id: 1396)

Similarly, in question (5.38), the word *city* explicates that *Pompeii* is indeed a city. More technically speaking, the word *game* is a hypernym of *croquet*, and the word *city* is a hypernym of *Pompeii*. Often, this kind of information is common knowledge and not explicitly mentioned in documents containing an answer. Hence, including

[5]Available from `http://www.census.gov/genealogy/names/`.

these words in the queries might harm retrieval effectiveness. This is similar to the situation of words that appear in the question focus, as discussed above.

We use WORDNET to find words or phrases that are hypernyms of other words or phrases in the question.

Relative idf score. Another indicator of the importance of a term is its frequency in the document collection. As discussed in the previous chapters, it is common to measure importance as the inverted document frequency (idf). Here, we are more interested in the relative importance of a term with respect to the other terms in the question. The relative idf score of term t in question q is defined as

$$\text{ridf}(t,q) = \frac{\log_2(N/df_t)}{\sum_{t' \in q} \log_2(N/df_{t'})}$$

where N is the number of documents in the collection, and df_t is the number of documents in which term t occurs.

At this point, after having discussed the individual features in some detail, it might be helpful to consider some example questions. Table 5.6 provides the complete feature instantiations for a number of questions.[6] When comparing the feature representations of the words, a number of things can be noticed. For instance, three of the questions have words in the question focus: *country* in question 1394, *year* in question 1546, and *first* and *satellite* in question 1557. Although all four terms occur in the focus of the respective question, there is a clear difference with respect to their generality. The word *country* has 300 leaves as hyponyms, the word year has 21 leaves as hyponyms, *first* has 7 and *satellite* has 20 leaves as hyponyms. Therefore, *country* is less likely to be helpful for retrieving answer documents than, for instance, *satellite*, as motivated above. On the other hand, *year* has only 7 leaves as hyponyms, indicating that it is a rather specific term, which is certainly not the case. This is due to the fact that WordNet does not list all possible years as hyponyms of the concept *year*. One way to distinguish between truly specific focus words such as *satellite* and words such as *year* is the feature `classifying word`, which is set to 1 for *year*, and 0 for *satellite*.

In question 1643, the phrase *Rhode Island* was correctly identified as a location, and the `location` feature was set to 1 for both words.

In question 1470, *Herbert Hoover* was recognized as a name and for *Herbert* the `person name` feature was set to `first` and for *Hoover* it was set to last. If a noun was not recognized as part of a name the feature is set to `no`, and for part-of-speech tags other than nouns, the feature is set to `na` (not applicable).

Question 1557 contains a superlative adjective, viz. *first*, and the `superlative` feature is therefore set to 1.

[6] In table 5.6, some of the question classes had to be shortened to make the table fit on the page. The class `location` was shortened to `locat.`, `date-of-death` to `death`, and `thing-ident` to `th-id`.

Table 5.6: Example questions and their feature instantiations

word	part-of-speech tag	question focus	superlative	question class	multiple occurrences	quoted	number leaves	term ratio	classifying word	location	abbreviation	upper case	modified noun	person name	honorific	no. incoming edges	hypernym	rel. idf
	In what country did the game of croquet originate?																	(topic id: 1394)
countri	NN	1	0	locat.	0	0	300	0.25	1	0	0	0	no	no	0	1	0	0.07
game	NN	0	0	locat.	0	0	195	0.25	0	0	0	0	no	no	0	2	1	0.13
croquet	NN	0	0	locat.	0	0	0	0.25	0	0	0	0	no	no	0	0	0	0.58
origin	VB	0	0	locat.	0	0	0	0.25	0	0	0	0	na	na	0	1	0	0.20
	When did president Herbert Hoover die?																	(topic id: 1470)
presid	NN	0	0	death	0	0	2	0.25	0	0	0	0	no	no	0	1	0	0.08
herbert	NNP	0	0	death	0	0	0	0.25	0	0	0	1	no	first	0	0	0	0.34
hoover	NNP	0	0	death	0	0	0	0.25	0	0	0	1	no	last	0	1	0	0.42
die	VB	0	0	death	0	0	4	0.25	1	0	0	0	na	na	0	1	0	0.16
	What year was the movie "Ole Yeller" made?																	(topic id: 1546)
year	NN	1	0	date	0	0	21	0.20	1	0	0	0	no	no	0	1	0	0.03
movi	NN	0	0	date	0	0	15	0.20	0	0	0	0	no	no	0	0	0	0.16
ol	NNP	0	0	date	0	1	0	0.20	0	0	0	1	no	no	0	0	0	0.29
yeller	NNP	0	0	date	0	1	0	0.20	0	0	0	1	no	no	0	4	0	0.48
make	V	0	0	date	0	0	2	0.20	0	0	0	0	na	na	0	0	0	0.04
	What was the first satellite in space?																	(topic id: 1557)
first	JJS	1	1	th-id	0	0	7	0.33	0	0	0	0	na	na	0	0	0	0.12
satellit	NN	1	1	th-id	0	0	20	0.33	0	0	0	0	yes	no	0	4	0	0.49
space	NN	0	1	th-id	0	0	103	0.33	0	0	0	0	no	no	0	0	0	0.40
	Who founded Rhode Island?																	(topic id: 1643)
found	V	0	0	agent	0	0	0	0.33	0	0	0	0	na	na	0	2	0	0.31
rhode	NNP	0	0	agent	0	0	0	0.33	0	1	0	1	no	no	0	0	0	0.43
island	NNP	0	0	agent	0	0	0	0.33	0	1	0	1	no	no	0	1	0	0.26

In order to learn term weights, each of the feature vectors is adorned with its term weight, as described in section 5.3. The weighted feature vector is an instance for the machine learning algorithm. Figure 5.3 shows the actual input to the machine learning algorithm. Note that in the feature representations, there is no reference to the term itself, or the query from which the term is taken. Also, the instances (lines) in figure 5.3 are completely independent of each other.

5.5 Machine Learning Approaches

There are numerous machine learning approaches that can be used for learning query formulation, including neural networks (Hertz et al., 1991), decision trees (Quinlan, 1993), naive Bayes (Duda and Hart, 1973), and linear regression (Press et al., 1988). For the purpose of learning query formulation, the machine learning

Figure 5.3: Input data for the machine learning algorithm

Feature representation of the questions *In what country did the game of croquet originate?* (lines 1–4) and *When did president Herbert Hoover die?* (lines 5–8).

```
NN, 1, 0, location, 0, 0, 300, 0.25, 1, 0, 0, 0, yes, no, 0, 1, 0, 0.07, -1.0
NN, 0, 0, location, 0, 0, 195, 0.25, 0, 0, 0, 0, yes, no, 0, 2, 1, 0.13, 0.0
NN, 0, 0, location, 0, 0, 0, 0.25, 0, 0, 0, 0, yes, no, 0, 0, 0, 0.58, 1.0
V, 0, 0, location, 0, 0, 0, 0.25, 0, 0, 0, 0, na, na, 0, 1, 0, 0.20, -1.0
NN, 0, 0, date-death, 0, 0, 2, 0.25, 0, 0, 0, 0, yes, no, 0, 1, 0, 0.08, -0.017
NNP, 0, 0, date-death, 0, 0, 0, 0.25, 0, 0, 0, 1, yes, first, 0, 0, 0, 0.34, 0.307
NNP, 0, 0, date-death, 0, 0, 0, 0.25, 0, 0, 0, 1, yes, last, 0, 1, 0, 0.42, 0.969
V, 0, 0, date-death, 0, 0, 4, 0.25, 1, 0, 0, 0, na, na, 0, 1, 0, 0.16, 0.133
```

algorithm should satisfy two desiderata:

1. The class labels should indicate a degree of the query term's usefulness for query formulation.

2. The resulting classification rules should be interpretable.

The first desideratum is based on the intuition that simple binary nominal classification might be too strict. Given the query term weights the way they are described above, binary classification could be accomplished by distinguishing between terms with a positive and negative (or zero) weight. Ordinal classification, on the other hand, imposes an order on the classes. For instance, Paşca (2001) distinguishes between high-relevance, medium-relevance, and low-relevance query terms, which are obviously ordered. Hence, misclassifying a high-relevance query term as a medium-relevance one is less harmful than misclassifying it as low-relevance term. This kind of ordering cannot be captured by nominal classification. Assuming real-valued query term weights, as introduced above, ordinal classification requires them to be discretized. Although there are standard techniques for discretization, see e.g., (Fayyad and Irani, 1993) and (Kohavi and Sahami, 1996), it is doubtful whether these classes will correspond to an intuitive interpretation, such as high-relevance, medium-relevance or low-relevance term. Additionally, there is very limited off-the-shelf machine learning software available that supports ordinal classification, see e.g., (Frank and Hall, 2001).

This brings us to the third kind of classification: interval classification. Here, classes are real numbers, and the aim of the machine learning algorithm is to assign a real number to an unseen instance that is as close as possible to the 'actual' real value of that instance. The advantage of interval classification is that it does not involve discretization of the classes, which might be too crude a method, and several off-the-shelf machine learning programs support interval classification.

The second desideratum states that the resulting classification rules should be interpretable. This is mainly to gain some insight into question interpretation. Some

machine learning approaches, such as neural networks—although being extremely powerful—generate classification rules that are completely opaque, unless used for trivial tasks.

Decision trees, naive Bayes, and linear regression, all allow for interval classification and generate transparent classification rules. Linear regression might be too limited because it assumes that the distribution of weights over the set of features can be approximated by a linear function. Naive Bayes classification is known to be well-performing for nominal classification (Domingos and Pazzani, 1997), but it is performing rather badly for interval classification, see (Frank et al., 2000). This leaves us with decision tree learning. Probably the best-known algorithm for decision tree learning is Quinlan's C4.5 (Quinlan, 1993), but C4.5 cannot deal with continuous classes. But M5 (Quinlan, 1992), which is an extension of C4.5, does allow for continuous classification.

The M5 algorithm builds *model trees* combining conventional decision tree learning with the possibility of linear regression models at the leaves of the tree. The resulting representation is relatively transparent because the decision structure is clear and the regression models are normally easily interpretable. The idea of model trees is largely based on the concept of regression trees, which are adopted by the well-known CART system (Breimann et al., 1984). The advantage of M5 over CART is that model trees are generally much smaller than regression trees and have proved to be more accurate in a number of tasks, cf. (Quinlan, 1992).

The learning algorithm we use here, is M5′ (Wang and Witten, 1997), which is a reconstruction of Quinlan's original M5 algorithm, for which only very few details are readily available. M5′ is also reported to perform somewhat better than the original algorithm on the standard datasets for which results have been published, see (Eibe et al., 1998). M5′ is part of the WEKA machine learning software package (Witten and Frank, 1999).[7]

Figure 5.4 shows an example model tree generated by M5′, for the CPU performance dataset[8], a standard machine learning dataset from the UCI Repository. The purpose of the CPU dataset is to learn predicting the CPU performance of a computer, given a number of hardware specifications. The tree structure of the model tree in figure 5.4 is very simple, just containing one single branching (decision), which checks whether the maximum main memory (`MMAX`) is greater than 14,000 kilobytes, or not. The leaves of these two branches each hold a linear model: `LM1` and `LM2`, which are further specified in the lower part of the output. The number in front of each attribute represents its weight. For instance, in model `LM1`, the cache memory (`CACH`) is multiplied by 0.552. Because multiplication cannot be directly applied to nominal attributes, different subsets of the possible values are treated separately as binary (0/1) attributes. Both models involve one nominal attribute, called `vendor`. The expression `vendor=adviser, sperry, amdahl` is interpreted as follows: if `vendor` is either `adviser`, `sperry`, or `amdahl`, then substitute it by 1, and

[7] Freely available at `http://www.cs.waikato.ac.nz/~ml/`.

[8] Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Figure 5.4: Example output of M5

```
MMAX <= 14000 : LM1
MMAX > 14000 : LM2

Models at the leaves (smoothed):
LM1: class = 4.15
             - 2.05vendor=honeywell,ipl,ibm,cdc,ncr,basf,
                  gould,siemens,nas,adviser,sperry,amdahl
             + 5.43vendor=adviser,sperry,amdahl
             - 5.78vendor=amdahl
             + 0.00638MYCT + 0.00158MMIN + 0.00345MMAX
             + 0.552CACH + 1.14CHMIN + 0.0945CHMAX
LM2: class = -113
             - 56.1vendor=honeywell,ipl,ibm,cdc,ncr,basf,
                          gould,siemens,nas,adviser,sperry,amdahl
             + 10.2vendor=adviser,sperry,amdahl
             - 10.9vendor=amdahl
             + 0.012MYCT + 0.0145MMIN + 0.0089MMAX
             + 0.808CACH + 1.29CHMAX
```

otherwise substitute it by 0.

5.6 Experimental Results

In this section, we describe the results of applying the M5′ model tree learning algorithm to learning query term weights, as described in the previous sections. There are two kinds of results: First, the learned model tree itself and the extent to which it provides further insights into understanding questions and the roles of the words in a question. Second, the effectiveness of the learned query term weights when used for document retrieval as a pre-fetch to a question answering system.

5.6.1 Model Tree Generation

The structure of a model tree depends on the set of instances on which M5′ is trained. In order to generate the most general model tree that is possible in the current setting, we applied M5′ to all data sets together, i.e., TREC-9, TREC-10, and TREC-11. This results in 4395 instances or feature representations of question words. Figure 5.5 shows the model tree that has been generated by M5′. Since the model tree is rather complex, we will not describe it in full detail, but discuss some of its more prevalent aspects.

Figure 5.5: Model tree for questions from TREC-9, TREC-10 and TREC-11

```
personname=first,no,last <= 0.5 :
|   relidf <= 0.332 :
|   |   noleaves <= 2.5 :
|   |   |   tag=JJ,LS,CD,JJS,NN,PRP,F,NNP,NN,NNP <= 0.5 : LM1
|   |   |   tag=JJ,LS,CD,JJS,NN,PRP,F,NNP,NN,NNP >  0.5 :
|   |   |   |   qtype=number-time-period,date-of-death,number-length,
|   |   |   |   object,agent,number-much,thing-ident,name-instance,
|   |   |   |   pers-ident,location,number-time-age,number-temperature,
|   |   |   |   number-distance,number-many-people,reason,currency,
|   |   |   |   pers-def,capital,thing-def <= 0.5 :
|   |   |   |   |   uppercase=1 <= 0.5 :
|   |   |   |   |   |   tag=LS,CD,JJS,NN,PRP,F,NNP,NN,NNP <= 0.5 : LM2
|   |   |   |   |   |   tag=LS,CD,JJS,NN,PRP,F,NNP,NN,NNP >  0.5 : LM3
|   |   |   |   |   uppercase=1 >  0.5 : LM4
|   |   |   |   qtype=number-time-period,date-of-death,number-length,
|   |   |   |   object,agent,number-much,thing-ident,name-instance,
|   |   |   |   pers-ident,location,number-time-age,number-temperature,
|   |   |   |   number-distance,number-many-people,reason,currency,
|   |   |   |   pers-def,capital,thing-def >  0.5 : LM5
|   |   noleaves >  2.5 : LM6
|   relidf >  0.332 : LM7
personname=first,no,last >  0.5 :
|   relidf <= 0.289 :
|   |   noleaves <= 13.5 :
|   |   |   relidf <= 0.182 : LM8
|   |   |   relidf >  0.182 :
|   |   |   |   noleaves <= 0.5 :
|   |   |   |   |   noincomingedges <= 0.5 : LM9
|   |   |   |   |   noincomingedges >  0.5 :
|   |   |   |   |   |   wordratio <= 0.225 : LM10
|   |   |   |   |   |   wordratio >  0.225 : LM11
|   |   |   |   noleaves >  0.5 :
|   |   |   |   |   qtype=agent,number-much,thing-ident,name-instance,
|   |   |   |   |   pers-ident,location,number-time-age,number-temperature,
|   |   |   |   |   number-distance,number-many-people,reason,currency,
|   |   |   |   |   pers-def,capital,thing-def <= 0.5 : LM12
|   |   |   |   |   qtype=agent,number-much,thing-ident,name-instance,
|   |   |   |   |   pers-ident,location,number-time-age,number-temperature,
|   |   |   |   |   number-distance,number-many-people,reason,currency,
|   |   |   |   |   pers-def,capital,thing-def >  0.5 : LM13
|   |   noleaves >  13.5 : LM14
|   relidf >  0.289 :
|   |   wordratio <= 0.415 : LM15
|   |   wordratio >  0.415 :
|   |   |   relidf <= 0.739 : LM16
|   |   |   relidf >  0.739 : LM17
```

The highest branching of the model tree in figure 5.5, checks whether the word at

hand is a first name, last name or other noun. If this is not the case (i.e., `personname = first,no,last <= 0.5`), further analysis descends down the left branch—roughly the upper half of figure 5.5—and otherwise it descends down the right branch. On the next level, branching depends on the relative idf value, making a case distinction in each subtree, viz. whether the relative idf value is smaller or equal to 0.332, or whether it is smaller or equal to 0.289, respectively.

Most branchings in the tree are related to the frequency of the term, viz. `relidf`, its generality `noleaves`, and the question type (`qtype`) of the question from which the word was taken.

The model tree has 17 leaves. To each leave a linear regression model is attached (LM1–LM17). These linear models are quite complex and it is impossible to display them here in full detail, nevertheless, we want to discuss some of their aspects. In figure 5.6, an abbreviated version of model LM1 is displayed. This model confirms some of the intuitions for query term selection as discussed above. If the word occurs in the question focus, this has a negative impact on the term weight (`- 0.00816focus`). Also, if the question does not contain a superlative adjective, the query term weight is lowered (`- 0.00135superlative=0`). Words that are not used to classify the question receive a higher term weight (`+ 0.0084usedtoclassify=0`), as do question words that are not abbreviations (`+ 1.03abbreviation=0`). If a word is not a hypernym of one of the other words in the same questions, the weight is raised (`+ 0.00218hypernym=0`). Also the fact that a word is recognized as a person's last name increases the term weight (`+ 0.00131personname=last`).

Figure 5.6: An excerpt of a linear model of the model tree

Feature representation of the questions *In what country did the game of croquet originate?* (lines 1–4) and *When did president Herbert Hoover die?* (lines 5–8).

```
LM1:  class = -1.12 + 0.00243tag=R,V,JJ,LS,CD,JJS,NN,PRP,F,NNP,NN,NNP
              + 0.0169tag=JJ,LS,CD,JJS,NN,PRP,F,NNP,NN,NNP
              + 0.0055tag=LS,CD,JJS,NN,PRP,F,NNP,NN,NNP
              - 0.00132tag=JJS,NN,PRP,F,NNP,NN,NNP - 0.00172tag=NNP
              - 0.00816focus - 0.00135superlative=0
              ...
              + 0.0084usedtoclassify=0 + 1.03abbreviation=0
              + 0.03uppercase=1 + 0.0181personname=na,first,no,last
              + 0.00178personname=no,last + 0.00131personname=last
              + 0.00218hypernym=0 + 0.0112relidf
              ...
```

Before we evaluate the effectiveness of using the model tree to predict term weights for retrieval purposes, we discuss the accuracy of the learned model tree itself. Table 5.7 provides some of the figures that are generated by the WEKA machine learning package for the model tree learning algorithm. Evaluation has been

Table 5.7: Accuracy of the model tree learning algorithm	
Correlation coefficient	0.5018
Mean absolute error	0.3783
Relative absolute error	82.1%

done on the training data using ten-fold cross validation. In n-fold cross validation, the training data is arbitrarily split into n partitions. The model tree learning algorithm is applied n times to $n-1$ partitions, where each time a different partition is held out for evaluating. Overall evaluation scores are obtained by averaging over the n individual evaluation scores. The correlation coefficient indicates the degree to which the predicted value and the original vales, as provided by the test data, correlate. A value of 1 (-1) indicates perfect (inverse) correlation, and a value of 0 indicates no correlation at all. Here, a correlation coefficient of 0.5 means that the predicted and original values are weakly correlated. The mean absolute error is the mean absolute difference between the predicted value (term weight) and the original value. The relative absolute error is the mean relative difference between the predicted value and the original value expressed in percents. A relative absolute error of 100% corresponds to the error that would have been obtained by always taking the mean value of all training instances for prediction. In our experiments, the relative absolute error is 82.1%, which is rather high, but still substantially better than choosing the mean training value for prediction.

In addition to evaluating the accuracy of the whole model tree, it is also interesting to estimate the importance of a single feature or attribute for learning the query term weight. This can be done by computing the attribute's information gain, cf. (Breimann et al., 1984). Information gain measures the reduction in uncertainty, where the degree of uncertainty is measured as the entropy. The *information gain* of attribute A with respect to class C is defined as:

$$
\begin{aligned}
(5.39) \qquad \mathit{InfoGain}(A,C) &= H(C) - H(C|A) \\
&= -\sum_{c \in C} p(c) \log_2(p(c)) \\
&\quad - \left(-\sum_{c \in C} \sum_{a \in A} p(c,a) \log_2(p(c|a)) \right)
\end{aligned}
$$

Note that the information gain computes the importance of an attribute independently of other attributes.

The problem with using information gain in the current context is that a number of attributes and the learned class, the query term weight, have to be discretized. Discretization is a non-trivial process in itself, and the way discretization is carried out has an impact on the estimation of the information gain. Hence, we used a different, and less commonly used measure, viz. *regression relief*, which is a measure for estimating the importance of an attribute for learning the query term weight,

and which can easily deal with continuous attributes and classes.

Robnik-Šikonja and Kononenko (1997, 2003) introduce the regression relief algorithm (RReliefF) to estimate the weight of an attribute. The key idea of the RReliefF algorithm is to estimate the quality of an attribute according to how well it discriminates between instances (feature vectors of query terms) that are near to each other. For this purpose, an instance R is selected randomly. Then, the k nearest instances, with respect to the class value, are selected, and the difference between the value of an attribute A of R and the value of the same attribute for one of the k instances is compared with respect to the difference of their class values. This process is repeated for a number of instances, potentially all, which finally leads to a weight for each attribute. The weight can range between -1 and 1. The full details of the RReliefF algorithm can be found in (Robnik-Šikonja and Kononenko, 1997, 2003).

Table 5.8 shows the RReliefF estimates for the 18 attributes or features that were used to learn query term weights. The classes (the term weights themselves) were determined by applying the model tree that was generated from the TREC-9, TREC-10, and TREC-11 datasets. For computing the RReliefF estimates, we used the WEKA system, which provides and implementation of the RReliefF algorithm.

Table 5.8: RReliefF estimates of features

Rank	Feature	RReliefF Value
1	abbreviation	0.006088
2	qtype	0.004000
3	noleaves	0.003909
4	relidf	0.003655
5	tag	0.003272
6	focus	0.003058
7	wordratio	0.002637
8	hypernym	0.001847
9	superlative	0.001492
10	twice	0.000966
11	quotes	0.000502
12	honorific	0.000229
13	usedtoclassify	0.000163
14	uppercase	0.000109
15	noincomingedges	0.000006
16	modifiednoun	-0.000030
17	location	-0.000454
18	personname	-0.001028

The ranking of the features reveals a number of interesting aspects. First, the `personname` feature is ranked lowest according to the RReliefF estimation, but it is the highest branching feature in the model tree in figure 5.5. One explanation for this discrepancy is the fact that `personname` is apparently too general a feature to

predict query term weights by itself. The `abbreviation` feature receives the highest RReliefF estimate, although it does not appear in the model tree in figure 5.5. The high rank of the `abbreviation` feature is probably due to the fact that it occurs in all of the linear models `LM1`–`LM17` with a relatively high regression coefficient, at the leaves of the model tree. The same holds for the `qtype` feature. The `relidf` and `noleaves` features, which are also ranked high by RReliefF, also occur high in the model tree, but are apparently more helpful for predicting the term weights than the `personname` feature, because they also occur in all linear models with coefficients that are higher than the coefficients of the `personname` feature.

Unfortunately, it is hard to distillate an explanation for each of the features' RReliefF estimate from the model tree. Nevertheless the RReliefF estimate does provide some insight into the importance of a feature independent of other features, that can be used for query term selection or weighting.

5.6.2 Retrieval Effectiveness of Learned Term Weights

Above we discussed some aspects of the model tree that might shed some light on understanding a question, and the way words from the question are useful for retrieving a document that contains an answer to it.

In the retrieval approaches that were discussed in the previous chapters, the weight of a query term was dependent on two factors: The frequency of a term in a document, and the collection frequency, i.e., the number of documents containing that term. If we want to integrate the learned term weights, as described above, the computation of the retrieval status value (RSV) has to be adapted appropriately. Here, we use the learned query term weights in combination with the original retrieval status value that resulted from computing the similarity between a query q and a document d according to the Lnu.ltc weighting scheme, which results in the new retrieval status value: RSV_L, which is defined as follows:

$$RSV_L(q,d) = \sum_{t \in q \cap d} RSV(q,d) \cdot \text{weight}(fr(t,q)) \cdot \text{idf}(t)$$

Here, $fr(t,q)$ is the feature representation of term t in query q, and $\text{weight}(fr(t,q))$ is the learned weight, which results from applying the M5' model tree to that feature vector. $RSV(q,d)$ is the document similarity according to the Lnu.ltc weighting scheme, and $\text{idf}(t)$ is the idf value of term t, i.e., $\log_2(N/df_t)$.

The model tree described in the previous subsection was generated by using all three data sets. Obviously, this model tree should not be used to evaluate the effectiveness of RSV_L on the different TREC data sets, as it is completely based on seen instances. Therefore we generated three different model trees, one for each of the TREC data sets. The model tree for the TREC-9 data set used feature representations of words from TREC-10 and TREC-11 (2854 instances), the model tree for the TREC-10 data set used feature representations of words from TREC-9 and TREC-11 (3167 instances), and the model tree for the TREC-11 data set used feature representations of words from TREC-9 and TREC-10 (2769 instances).

First, we considered the performance with respect to the answer-at-n (a@n) measure. Table 5.9 shows the results of using learned query terms weights in contrast to the Lnu.ltc base line. Unfortunately, the improvements are rather modest, although

Table 5.9: Comparison of the a@n scores of learned-weights retrieval runs to baseline runs

	TREC-9			TREC-10			TREC-11		
a@n	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
a@5	0.700	0.727	(+3.7%)$^{\triangle}$	0.649	0.654	(+0.1%)	0.523	0.547	(+4.6%)$^{\triangle}$
a@10	0.785	0.806	(+2.7%)$^{\triangle}$	0.734	0.730	(-0.1%)	0.626	0.637	(+1.8%)
a@20	0.845	0.863	(+2.1%)	0.801	0.804	(±0.0%)	0.705	0.732	(+3.8%)$^{\triangle}$
a@50	0.914	0.908	(-0.1%)	0.875	0.859	(-1.8%)	0.795	0.815	(+2.5%)

still statistically significant in some cases. In a few cases, retrieval effectiveness even drops slightly. Next, we consider the performance with respect to mean average precision and the results are displayed in table 5.10. Compared to the Lnu.ltc base

Table 5.10: Comparison of MAP of learned-weights retrieval runs to baseline runs

	TREC-9			TREC-10			TREC-11		
	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
MAP	0.280	0.328	(+17.1%)$^{\blacktriangle}$	0.279	0.296	(+6.1%)$^{\blacktriangle}$	0.214	0.242	(+13.1%)$^{\blacktriangle}$

line, the improvements are clearly statistically significant. One explanation for the difference in relative improvement over the base line between measuring a@n and mean average precision could be the fact that query term weights used for training were computed with respect to the queries average precision, see section 5.3.

5.7 Conclusions

In this chapter we investigated to what extent it is possible to learn query term weights for better query formulation. As we have seen in section 5.2, keyword selection has a strong impact on the performance of the retrieval component.

In order to learn query term weights, we considered all possible ways of selecting terms from the original question for query formulation, and we used the performance results of each possible formulation in order to determine individual query term weights.

Query terms are represented as sets of features on which the M5' model tree learning algorithm is trained. The resulting model tree confirms some of the heuristics and intuitions for keyword selection than can be found in the literature, see, e.g., (Paşca, 2001). We have evaluated the retrieval with learned query weights and compared the performance to the Lnu.ltc base line. Unfortunately, the improvements are rather modest, staying far behind the potential improvements optimal query selection can yield, see section 5.2. The fact that improvements are rather modest could have two reasons: First, our approach to learning query term weights con-

tains too many errors, and, second, our way of integrating the learned weights into the computation of the retrieval status value is imperfect.

Obviously, it is hard to estimate whether the set of features we used to represent query terms is appropriate and whether it indeed captures the aspects that are indicative for predicting the importance of a query term. One important aspect that is surely missing is any representation of the answer documents in which the terms occur. In some cases the issue whether a term is helpful for retrieving answer documents simply depends on some idiosyncrasies of the documents that contain an answer. On the other hand, our training sets were of considerable size and reasonably varied, in order to abstract away from those idiosyncrasies.

Another way to improve the learning algorithm is to use an ensemble of machine learners, instead of a single one, see, e.g., (Dietterich, 1997; Breimann, 1996; Freund and Schapire, 1995, 1996). Using ensembles has been shown to be rather effective for many standard machine learning data sets, and it might also be effective in the current setting. Using an ensemble of machine learners to predict query term weights remains part of our future work.

The second question, whether the learned query weights are properly integrated into our similarity measure, is also difficult to answer, since there are many ways to define document similarity and a more comprehensive investigation of this issue remains a task to be carried out in future work.

Chapter 6

Query Expansion for Specific Question Classes

Expanding queries with terms that are semantically related to query terms, or frequently co-occur with them, has a long tradition in document retrieval. In this chapter, we investigate the expansion of queries with terms that are likely to occur in a correct answer, which is particularly applicable to questions asking for measurements, such as height, length, etc. Query expansion is accomplished by using structured queries, where the expansion terms are grouped together by a special operator. In order to work with structured queries, minimal span weighting has to be adapted appropriately. The experimental results on the TREC data sets show that query expansion yields substantial improvements over an unexpanded baseline.

When asking a question, one often has certain expectations about the answer, whether it should be a person's name, a date, a city, etc. Although the actual answer is not known to the questioner, he or she expects it to be of a certain type. In question answering systems, these expectations or type constraints are reflected by the question classification component, which controls the process of identifying elements in the document collection that are of the appropriate type.

For some types of questions, we have even more concrete information about what certain parts of an answer look like. Questions where this is typically the case are questions asking for measurements, such as the height, age, costs, or temperature of something. Here, answers have the form of a number followed by an appropriate measurement unit, such as *feet* as a unit for measuring height, and *dollars* as a unit for measuring prices. For common measures, including the aforementioned ones, the set of measurement units that are used to express the degree of a certain

property is rather small. E.g., length can be measured in *inches, feet, yards, miles, meters, kilometers*, and a number of additional units, but nevertheless, the set remains relatively small.

The fact that the number of measurement units is restricted eases the process of identifying phrases of the appropriate type. But it also has some consequences for retrieving documents that are likely to contain an answer to a question asking for a certain measure. When asking a question about a particular measure, we know that an answer document is very likely to contain one of the corresponding measurement units, and this can be exploited in the context of retrieval as a pre-fetch to a QA system. Consider question (6.1.a) which asks for the location of the Eiffel Tower.

(6.1) a. Where is the Eiffel Tower? (topic id: 1205)

b. `eiffel AND tower`

(6.2) a. How tall is the Eiffel Tower in France? (topic id: 1692)

b. `eiffel AND tower AND france AND (foot OR meter OR inch)`

The only constraint on the answer phrase is that it has to be the name of a location, such as a city or country name; Paris or France in this case. Beyond this, there are no further clues what the answer might look like. Of course, one could consider for instance words such as *in* and *near* as indicators for locations, but these words are so frequent that it is questionable whether they are discriminative enough to be useful for identifying documents that are likely to contain an answer. Including each possible location into the query used for pre-fetching is absolutely infeasible, and therefore (6.1.b) seems to be the most appropriate query for question (6.1.a). Question (6.2.b) asks for the height of the Eiffel Tower. Although at this stage, we do not know what correct answers exactly look like, we do know that they have to contain a measurement unit such as *feet* or *meters*. Given this additional information, we can formulate the more restrictive query (6.2.b).

The remainder of this chapter is organized as follows: The next section provides a brief overview of related work on query expansion in the context of question answering. Section 6.2 discusses the types of queries that eligible for expansion in the current setting, as well as terms that are used for expansion. In section 6.3 we explain how structured querying is used to deal with query expansion. Section 6.4 provides the experimental results for comparing expanded querying to unexpanded retrieval. Finally, section 6.5 gives a few concluding remarks, and an outlook on some remaining issues.

6.1 Related Work

There are a number of approaches to query formulation in the context of question answering, some of which have already been discussed in the previous chapter. Most of the work on query expansions for question answering focuses on extending the queries with semantically related terms, such as synonyms.

Magnini and Prevete (2000) describe an approach where they take the terms from the original question and add to the query morphological variants and synonyms of the original terms and the morphological variants. Their experiments were carried out for Italian questions, and synonyms were identified by consulting the ItalWordNet database (Roventini et al., 2000), which is an extension of the Italian part of EuroWordNet (Vossen, 1998). The resulting boolean queries are rather complex, and different strategies are discussed to integrate the morphological and semantic variants. Magnini and Prevete (2000) report substantial improvements when using query expansion. One shortcoming of their implementation is that word sense disambiguation is done manually, which is a significant simplification of the actual task, where this has to be done automatically. In the context of ad hoc retrieval, Sanderson (2000) shows that the quality of automatic word sense disambiguation has a very strong impact on retrieval performance.

Agichtein et al. (2001, to appear) use the World Wide Web to find answers to natural language questions. A number of keywords from the original question are used to form queries, which are expanded with phrases that are likely to occur in a declarative sentence that contains an answer. For instance, the question *What is a binturong?* is transformed into the queries: `binturong 'refers to'`, `binturong 'is a'`, and `binturong 'is usually'`, which are then posted to a web search engine such as ALTAVISTA or GOOGLE. In their experiments, they focused on four question types: person definition questions (who is/was), procedural question (how do/can I), location questions (where is, where can I), and definition questions (what is/are). By evaluating their expansion approach on a set of questions of these types, they report substantial improvements in comparison to the underlying web search engines.

Yang and Chua (2002); Yang et al. (2003) use the World Wide Web in combination with WordNet to find additional terms to expand the query. In the first step, they use the original terms from the question to pose a query to a web search engine. From the returned web pages, terms are extracted that frequently occur in the proximity of question terms. If a term occurs more frequently than could be expected by its a priori distribution, it is added to a list of expansion terms. In the second step, also terms from the WordNet glosses of the original question terms are added to the list of expansion terms. For example, the final query for question (6.3.a) looks like query (6.3.b)

(6.3) a. What Spanish explorer discovered the Mississippi River?
(topic id: 1411)

b. `Mississippi AND (French OR Spanish) AND Hernando AND Soto AND De AND 1541 AND Explorer AND (first OR European OR River)`

Note that the actual answer *Hernando De Soto* is also part of the expansion. This is a side-effect of consulting web pages and adding terms that frequently co-occur with the original question terms.

(Prager et al., 1999, 2000) introduce an approach, called *predictive annotation*, where they do not expand queries with actual terms, but with the answer type of the corresponding question. For example, the query for the question *When did the Challenger explode?* is `@SYN(DATE$, TIME$) Challenger explode`, where `@SYN` is an alternative operator, similar to the boolean `OR`.[1] Their approach requires the documents of the collection to be annotated with the respective question types. All occurrences of phrases in the document collection that belong to one of the answer types are indexed as such, along with the individual terms that comprise that phrase. Unfortunately, they do not provide any comparison between expanded and unexpanded pre-fetching that allows one to determine the effectiveness of expanding queries with the corresponding answer types. One problem of their approach could lie in the frequency of certain answer types, such as `PERSON$`, `NUMBER$` or `NAME$`, which are so frequent that they might be rather useless for boosting text segments that actually contain an answer. For some of these categories, a division into further subtypes might be helpful. For example, `NUMBER$` might be further subdivided into `SPEED$`, `TEMPERATURE$`, etc.

6.2 Query Expansion

Most approaches to query expansion for question answering either use global expansion, where knowledge resources, such as WordNet, are used to identify terms that can be added to the query, or local expansion, where additional terms are taken from documents that were retrieved by an initial query that is built from the original questions terms, much like blind feedback or local context analysis. Often both approaches also use the World Wide Web to provide enough data to make frequency based decisions feasible. Here, we use a somewhat different approach, which lies in between global expansion and predictive annotation.

As discussed above, for a number of questions, we already know some words that are likely to be part of an answer. This applies in particular to measurement questions, such as questions asking for the height, length, or age of something.

Table 6.1 lists the types of questions that are eligible for this type of expansion. Of course, there is no standard on the types of questions, and most question answering systems use different question types, see, e.g., (Hovy et al., 2001; Ferret et al., 2000), but it is fair to say that the question types used here are compatible with most classification schemes. These lists of expansion terms are based on a number of pilot experiments and they turned out to be the most appropriate ones. The different expansion terms for a given question type are added as alternatives to the query which contains the words of the original question. Note that the expansion lists are lists of words and not phrases. This limitation is due to the complications that arise in our retrieval system FLEXIR for computing similarity scores when queries contain alternatives. This issue is discussed in more detail below. For some question

[1] A $ sign is attached to the answer type name in order to distinguish it from naturally occurring terms that are identical to them.

Table 6.1: Question types and their corresponding expansion terms

question type	expansions
number-many-people	people, citizen, inhabitant, population, live
number-money	dollar, pound, $, usd, cent
number-length	meter, mile, kilometer, foot, yard
number-speed	mph, per, kmh, speed, fast, mile, kilometer
number-height	meter, inch, foot, centimeter
number-temperature	degree, fahrenheit, celsius
number-time-period	hour, day, week, month, year, decade
number-time-age	year, month, old, age
number-time-distance	anniversary, ago
number-size	square, acre, size, large
number-weight	kg, kilogram, pound, ton, lb, kiloton
number-ratio	percent, half, third, fourth, quarter, fifth
number-frequency	time, often
number-depth	meter, inch, foot, centimeter

types, such as number-speed, it seems natural to expand the query with multi-word phrases such as *miles per hour* and *kilometers per hour*, but unfortunately, this is not possible at the current stage, [2] and one has to make a decision which of the terms of such a phrase are appropriate to be added to the query.

Altough some of the expansion terms are ambiguous, e.g., the term *foot* can refer to the measurement unit or the body part, they tend to be disambiuated properly by minimal span weighting which takes into account the proximity between the different words of the query.

For some question types, a more fine-grained type of classification might be even more appropriate. E.g., the type number-height covers questions asking for the height of persons, buildings, mountains, etc. But sometimes, the height of a building is measured in floors or stories. Hence, retrieval might benefit from an additional type such as number-height-building. At the current stage, we did not further investigate the issue of a more fine-grained classification scheme, and its impact on retrieval performance.

Questions (6.4–6.6) give some examples of queries that result from query expansion.

(6.4) How high is Mount Kinabalu? (topic id: 1420)
Question type: number-height
Query: `mount kinabalu alt(meter,inch,foot,centimet)`

(6.5) How much are tickets to Disney World? (topic id: 1487)

[2] The reason for not being able to expand queries with multi-word phrases is purely caused by engineering issues.

Question type: number-money
Query: `ticket disnei world alt(dollar,pound,$,usd,cent)`

(6.6) How far away from the sun is Saturn? (topic id: 1644)
Question type: number-length
Query: `sun saturn alt(meter,mile,kilomet,foot,yard)`

Note that the queries contain the stemmed terms of the original question and the expansion list of table 6.1.

The expanded queries in examples (6.4–6.6) are not simple vectors of terms anymore, but contain an additional operator, viz. `alt`. When computing the similarity between the query and a document, only one of the terms in the scope of the `alt` operator is used for computing the retrieval status value. The terms in the scope of the `alt` operator are truly interpreted as alternatives, meaning that if more than one alternative matches it does not further contribute to the retrieval status value of the document. If a query contains an `alt`-operator, we call it a *structured query*.

Moving from simple query vectors to structured queries is necessary in order to avoid rewarding documents that contain many alternative terms. As mentioned above, alternative terms are just different ways to express the same or similar information. Note that representing an expanded query as a simple (unstructured) vector, it can happen that documents containing many measurement terms are ranked higher than documents containing many of the terms from the unexpanded query and only a few measurement terms.

6.3 Structured Querying

Many approaches to query expansion with terms conveying similar information use boolean retrieval, where these terms are connected by the boolean `OR` operator. Since the FLEXIR retrieval system, which we use throughout this thesis, is based on the vector-space model, which does not support the functionality of the boolean `OR`, a few extensions are required. The main extension is to move to structured queries, including the `alt` operator.

Using structured queries instead of simple term vector queries requires some modifications of the weighting scheme that is used to compute the similarity score between a structured query and a document. In particular, two issues have to be addressed. First, what weighting scheme is appropriate to compute the global similarity between a structured query and a document? Second, given the significant improvements in effectiveness of minimal span weighting, we would also like to use it for structured queries, but in order to do that some modifications of the minimal span weighting algorithm are required.

6.3.1 Global Document Similarity

In the experiments described in the previous chapters, we used the Lnu.ltc weighting scheme, which is commonly used in vector space retrieval. However, when using structured queries instead of query vectors, two problems arise. Two illustrate the problems, consider the Lnu.ltc weighting scheme, which was discussed in more detail in section 3.3.1:

$$sim(q,d) = \sum_{t \in q \cap d} \frac{\frac{1+\log(\text{freq}_{t,d})}{1+\log(\text{avg}_{t' \in d}\text{freq}_{t',d})} \cdot \frac{\text{freq}_{t,q}}{\max_{t' \in q}\text{freq}_{t',q}} \cdot \log\left(\frac{N}{n_t}\right)}{((1-sl)\cdot pv + sl \cdot \text{uw}_d) \cdot \sqrt{\sum_{t' \in q}\left(\frac{\text{freq}_{t',q}}{\max_{t'' \in q}\text{freq}_{t'',q}} \cdot \log\left(\frac{N}{n_{t'}}\right)\right)^2}} \tag{6.7}$$

The gray shaded factor in the nominator corresponds to the weight of the within-query frequency of term t, and the gray area in the denominator corresponds to the cosine normalization of the query term vector.

In the current implementation, query expansion is based purely on the question type and not on the terms in the original question. For instance, consider question (6.8.a) and the corresponding expanded query (6.8.b).

(6.8) a. What is the population of Maryland? (topic id: 1425)

b. `popul maryland alt(peopl,citizen,inhabit,popul,live)`

In this example, the term *popul* (the stem of *population*) occurs twice in the query. One time it comes from the original question, and the other time it is part of the list of expansion terms for questions of the type number-population. Using the within-query frequency weighting mentioned above, the weight of *popul* is considerably raised, because it occurs twice as often in the query as all the other terms. But the increase in frequency is just an artifact of the query expansion, and not a characteristic of the original question. The solution we propose is to remove terms from the query that also occur in the list of query expansion terms.

At this point, we also have to discuss the way in which alternative terms contribute to the similarity measure between a document and a query. Given a query containing an alternative operator `alt(t1,t2,t3)`, and a document d, we associate with each term a contribution weight ($cw(t,d)$):

$$cw(t,d) = \frac{1+\log(\text{freq}_{t,d})}{1+\log(\text{avg}_{t' \in d}\text{freq}_{t',d})} \cdot \log\left(\frac{N}{n_t}\right) \tag{6.9}$$

The contribution weight depends on the normalized frequency of the term t in the document, which corresponds to the L option in the Lnu.ltc weighting scheme, and the idf score of term t, which corresponds to the t option in the Lnu.ltc weighting scheme. The contribution weight allows one to order the alternative terms matching a document. For example, if a document d contains the terms t_1 and t_2, and

$cw(t_1, d) \geq cw(t_2, d)$, only term t_1 is used for computing the overall similarity between the query q and the document d. This way only the term with the highest contribution weight is used, and documents containing several alternative terms are not preferred over documents containing only one alternative term.

The other issue is the cosine normalization of the query term vector, which is indicated by the gray area in the denominator of the similarity weighting equation (6.7). By applying cosine normalization to the query vector, all weights of the terms in the vector are normalized with respect to the square root of the sum of the squared original weights. The effect is that term weights are relativized with respect to the weights of the other terms in the query. If we use structured query expansion, the question is whether the terms in the expansion list should be used for normalization, since, as mentioned above, these terms have a different status, where documents matching many expansion terms are not to be preferred over documents that match only one expansion term.

The query normalization factor in equation (6.7) remains the same for all documents. But in the case of queries containing alternative terms, the terms that are actually used can change for each document, depending on which is the term with the highest contribution weight. How this fact can be reconciled with applying query normalization is unclear to us, and we therefore decided to simply drop query cosine normalization, as it is known not to have a strong influence on document similarity, see (Salton and Buckley, 1988).

This leads us to the final definition of global similarity between a query q and a document d:

$$sim(q,d) = \sum_{t \in q' \cap d} \frac{\frac{1+\log(\text{freq}_{t,d})}{1+\log(\text{avg}_{t' \in d}\text{freq}_{t',d})} \cdot \log\left(\frac{N}{n_t}\right)}{((1-sl)\cdot pv + sl \cdot \text{uw}_d)} + \frac{\max_{t \in q_a \cap d} \frac{1+\log(\text{freq}_{t,d})}{1+\log(\text{avg}_{t' \in d}\text{freq}_{t',d})} \cdot \log\left(\frac{N}{n_t}\right)}{((1-sl)\cdot pv + sl \cdot \text{uw}_d)} \tag{6.10}$$

In equation (6.9), regular query terms and terms that occur in the scope of an alternative operator require a different treatment. We distinguish between two sets of query terms. The set q' contains the terms from the original question (after stop word removal) that do not occur in the list of expansion terms of the corresponding question type. The other set q_a contains the alternative or expansion terms. In the current context, we assume that a query contains at most one alternative operator, but the weighting scheme in (6.10), can easily be generalized to situations, where queries contain more than one alternative operator.

6.3.2 Minimal Span Weighting for Structured Queries

As we showed in chapter 4, minimal span weighting significantly improves retrieval effectiveness in the context of question answering. In order to combine minimal span weighting with structured querying, a few minor changes have to be made.

Recall that the minimal span weighting scheme consists of two factors: the global document similarity and the spanning factor, see section 4.2 for more details. When using structured queries, the global document similarity is computed as described in the previous subsection. Given a list of alternative terms, only the term with the highest contribution weights is used for computing the document similarity. However, when computing the minimal matching span of a document, we might also want to consider other occurrences of alternative terms, even if they have a lower contribution weight, but do occur in closer proximity to other matching query terms in the document.

Considering occurrences of alternative terms require a modification of definition 4.1 of matching spans. The main difference is that occurrences of different alternative terms are considered occurrences of semantically similar terms. If an alternative terms occurs in the document, it is sufficient, if the matching span contains occurrences of one of the alternative terms. Hence, we consider each alternative term separately, when determining matching spans. For instance, occurrences of the words *feet* and emphmeters are both considered as alterantive ways to measure length and including an occurrence of either one of them in the matching span is sufficient. More formally, matching spans for structured queries are defined as follows:

Definition 6.1 (Matching span for structured queries) Given a query q and a document d, $q' \subseteq q$ is the set of terms that do not occur in the scope of an `alt` operator, and $q_a \subseteq q$ is the set of alternative terms. The function $\text{term\_at\_pos}_d(p)$ returns the term occurring at position p in d. A *matching span for structured queries* (ms_s) is a set of positions that contains at least one position of each matching term from q′ and one position of a matching term from q_a, i.e. $\bigcup_{p \in \text{ms}_s} \text{term\_at\_pos}_d(p) \in \{(q' \cup \{t\}) \cap d \mid t \in q_a\}$. ■

Once the definition of a minimal span has been adapted for structured queries, the definition of a minimal matching span, see definition 4.2, can remain unchanged.

Finally, the definition of minimal span weighting, see definition 4.3, has to be slightly adapted as well. The main difference between minimal span weighting for structured queries and the original minimal span weighting scheme concerns the way the number of terms in the query and the number of matching terms are determined. As discussed above, the terms in the scope of an `alt`-operator are viewed as different ways to express the same thing. Hence, when counting the number of (matching) query terms, the set of alternative query terms (q_a) as a whole counts as one query term. The number of matching query terms is computed as $|q' \cap d| + ne(q_a \cap d)$, where $q' \cap d$ is the number of query terms that occur in the document, but are not in the scope of an `alt`-operator, and $q_a \cap d$ is the set of alternative terms that occur in the document. The $ne(\cdot)$ function checks whether the set $q_a \cap d$ is non-empty. If $|q_a \cap d| > 0$, $ne(q_a \cap d)$ returns 1, and 0 otherwise. The definition of minimal span weighting for structured queries is shown in definition 6.2.

Definition 6.2 (Minimal span weighting for structured queries)
If $|q' \cap d| + ne(q_a \cap d) > 1$, then

$$\text{RSV}'(q,d) =$$

$$\lambda\ \text{RSV}_n(q,d) + (1-\lambda)\left(\frac{|q' \cap d| + ne(q_a \cap d)}{1 + \max(mms) - \min(mms)}\right)^{\alpha}\left(\frac{|q' \cap d| + ne(q_a \cap d)}{|q'| + ne(q_a)}\right)^{\beta}$$

If $|q' \cap d| + ne(q_a \cap d) = 1$ then $\text{RSV}'(q,d) = \text{RSV}_n(q,d)$. ■

For more details on minimal span weighting see definition 4.3, where all the factors involved in minimal span weighting are discussed.

Although definition 6.1 and definition 6.2 assume the query to contain at most one `alt`-operator, it is easy to generalize the definitions to overcome this restriction.

6.4 Experimental Results

In the previous section, we showed how the minimal span weighting scheme can be adapted in order to handle structured queries, in particular to queries containing a list of alternative terms. In this section we evaluate the effectiveness of expanding queries with measurement units, see section 6.2, in combination with minimal span weighting.

As before, we used the TREC data sets for experimental evaluation. But, since query expansion is only done for a number of question types, viz. questions asking for certain measures, we will focus on questions of the appropriate type and disregard questions of a different type. Table 6.2 lists the question types and their respective frequencies (in the TREC data sets) which are used for evaluating query expansion. Although the individual question types are rather infrequent, the set of all measurement questions constitutes a fairly substantial portion of all questions in the TREC data sets. With respect to our classification scheme, only 4–6 question types, such as `date` and `location`, are more frequent than the combined set of measurement questions.

In order to evaluate the effectiveness of expanding queries for measurement questions with unit measurement terms, we focus on the subsets of measurement questions from the TREC data sets. For estimating the impact of expanding queries, we compare it to a baseline run using unexpanded queries. In the previous chapters, we used the Lnu.ltc weighting scheme as a baseline, but, since our query expansion approach also involves minimal span weighting, we use the minimal span weighting run (without any expansions) as baseline, in order to focus purely on the impact of query expansion. Table 6.3 shows the results for the different TREC data sets, using the a@n evaluation measure.

The first thing one can notice is that the minimal span weighting baseline scores are much lower for measurement questions than the average minimal span weighting scores for all queries, see table 4.1, page 76. Apparently, retrieving answer doc-

Table 6.2: Measurement questions and their frequency in the TREC data sets

question type	TREC-9		TREC-10		TREC-11	
number-many-people	11	(2.2%)	6	(1.2%)	4	(0.8%)
number-money	2	(0.4%)	0	(0.0%)	5	(1.0%)
number-length	3	(0.6%)	5	(1.0%)	3	(0.6%)
number-speed	2	(0.4%)	5	(1.0%)	2	(0.4%)
number-height	2	(0.4%)	2	(0.4%)	10	(2.0%)
number-temperature	2	(0.4%)	4	(0.8%)	2	(0.4%)
number-time-period	3	(0.6%)	2	(0.4%)	3	(0.6%)
number-time-age	1	(0.2%)	4	(0.8%)	6	(1.2%)
number-distance	1	(0.2%)	6	(1.2%)	7	(1.4%)
number-size	4	(0.8%)	0	(0.0%)	4	(0.8%)
number-time-distance	1	(0.2%)	0	(0.0%)	1	(0.2%)
number-ratio	0	(0.0%)	0	(0.0%)	5	(1.0%)
number-frequency	0	(0.0%)	1	(0.2%)	1	(0.2%)
number-depth	0	(0.0%)	0	(0.0%)	1	(0.2%)
total	32	(6.4%)	33	(6.6%)	53	(11.4%)

Table 6.3: Comparison of the a@n scores of expanded retrieval runs to baseline msw runs

	TREC-9			TREC-10			TREC-11		
a@n	msw	+exp		msw	+exp		msw	+exp	
a@5	0.400	0.500	(+25.0%)	0.517	0.724	(+40.0%)▲	0.500	0.583	(+16.6%)
a@10	0.633	0.567	(-10.4%)	0.690	0.793	(+14.9%)	0.639	0.668	(+4.5%)
a@20	0.667	0.700	(+4.9%)	0.759	0.828	(+9.1%)	0.833	0.750	(-10.0%)
a@50	0.733	0.733	(±0.0%)	0.799	0.966	(+8.4%)△	0.861	0.806	(-6.5%)

uments for measurement questions is much harder than for all question types on average, but it is hard to determine why this is the case.

In most cases, query expansion outperforms the minimal span weighting baseline, but only in a few cases the improvements are statistically significant. In some cases, even large relative improvements, such as +25.0% or +16.6%, are not statistically significant. One reason is the fact that the sample size, i.e., the number of queries, is much smaller than the sample size of the previous experiments, viz., the whole data sets, consisting of approximately 450 queries. Because getting statistically significant differences is more difficult for smaller samples, it is not too surprising that this only holds for a few of the cases in table 6.3, cf. (Siegel and Castellan, 1988).

Another observation coming from table 6.3 is that the largest improvements mainly occur at lower cut-off values, i.e., lower instantiations of n. This indicates that query expansion is particularly beneficial for question answering systems that require early high precision.

Since the results in table 6.3 are somewhat inconclusive with respect to statis-

tically significant improvements, we also consider additional evaluation measures. Table 6.4 shows the precision scores for several cut-off levels. Again, although the

Table 6.4: Comparison of the p@n scores of expanded retrieval runs to baseline msw runs

	TREC-9			TREC-10			TREC-11		
p@n	msw	+exp		msw	+exp		msw	+exp	
p@5	0.100	0.180	(+80.0%)	0.166	0.276	(+66.3%)▲	0.178	0.189	(+6.2%)
p@10	0.127	0.140	(+10.2%)	0.145	0.214	(+47.6%)▲	0.111	0.128	(+15.3%)
p@20	0.098	0.108	(+10.2%)	0.119	0.145	(+21.9%)▲	0.090	0.083	(-7.8%)
p@50	0.060	0.060	(±0.0%)	0.071	0.089	(+25.3%)▲	0.048	0.044	(-8.3%)

use of query expansion generally outperforms non-expanded minimal span weighting, most of the improvements are not statistically significant. Similar to the results in table 6.3, where we used the a@n evaluation measures, the p@n evaluation shows that the highest improvements are gained at lower cut-off levels.

Next, we compare query expansion to the baseline with respect to recall at a number of cut-off levels, as shown in table 6.5. Recall increases tremendously when

Table 6.5: Comparison of the r@n scores of expanded retrieval runs to baseline msw runs

	TREC-9			TREC-10			TREC-11		
r@n	msw	+exp		msw	+exp		msw	+exp	
r@5	0.099	0.165	(+66.7%)	0.195	0.357	(+31.8%)▲	0.239	0.275	(+15.1%)
r@10	0.214	0.281	(+31.2%)	0.273	0.481	(+76.2%)▲	0.329	0.377	(+14.6%)
r@20	0.340	0.382	(+12.6%)	0.443	0.575	(+29.8%)▲	0.515	0.483	(-6.2%)
r@50	0.465	0.478	(+2.8%)	0.602	0.770	(+27.9%)▲	0.622	0.568	(-8.7%)

using query expansion, but only the runs on the TREC-10 data set show strong statistically significant improvements. Nevertheless, query expansion appears to have a strong positive effect on retrieval effectiveness, only decreasing with respect to the baseline for the TREC-11 data set at the higher cut-off levels of 20 and 50.

Finally, we evaluate query expansion with respect to mean average precision (MAP), which combines precision and recall for all recall levels, and the results are shown in table 6.6. Mean average precision increases for all three data sets, and the

Table 6.6: Comparison of the MAP scores of expanded retrieval to msw retrieval

	TREC-9			TREC-10			TREC-11		
	msw	+exp		msw	+exp		msw	+exp	
MAP	0.135	0.196	(+45.2%)▲	0.314	0.363	(+15.6%)▲	0.215	0.242	(+12.6%)

improvements for the TREC-9 and TREC-10 data sets are both statistically significant.

Summing up, query expansion does increase the retrieval effectiveness for measurement questions. Unfortunately, the results were not statistically significant in

many cases, sometimes even if the relative improvements were quite large, exceeding 60%. This might be due to the relatively small size of the sample, which makes proving statistically significant differences more difficult. Nevertheless, when considering all results for the different evaluation measures, it seems safe to say that retrieval in the context of question answering can benefit from query expansion for measurement questions.

As a final experiment, for the a@n, p@n, and r@n evaluation measure, we put together the individual results for the TREC-9, TREC-10, and TREC-11 datasets, to see whether statistically significant differences can be observed on such a larger dataset. Note that this is purely motivated by our suspicion that the small sizes of datasets used above had a strong impact on failing to exhibit statistically significant differences. Table 6.7 shows the results for the three evaluation measures, on all three datasets put together. Using the larger dataset, we do see that expanding measurement questions leads to improvements in all but one case, where effectiveness remains unchanged. We can also see that expansion has a statistically significant impact at lower cut-offs. The trend that expansion mostly affects lower cut-offs was already observed in the discussion above, but in most cases, these improvements were not significant.

Table 6.7: Comparing expanded retrieval to msw for all TREC datasets put together

	a@n			p@n			r@n		
n	msw	+exp		msw	+exp		msw	+exp	
5	0.474	0.600	(+26.6%)▲	0.150	0.213	(+42.0%)▲	0.181	0.265	(+46.4%)▲
10	0.653	0.674	(+3.2%)	0.126	0.158	(+25.4%)▲	0.275	0.378	(+37.5%)▲
20	0.758	0.758	(±0.0%)	0.102	0.110	(+7.8%)	0.438	0.479	(+9.4%)
50	0.821	0.832	(+1.3%)	0.059	0.063	(+6.8%)	0.566	0.601	(+6.2%)

6.5 Conclusions

In this chapter, we have investigated the effects of expanding queries for certain question types, in particular for questions that ask for measurements such as height, length, age, etc. These types of questions are especially suited for simple query expansion because their answers are required to contain terms indicating the measurement unit, and the number of measurement units is rather limited in general.

In our query expansion approach we did not simply add all expansion terms to a query, but allowed queries to be structured, where all expansion terms are grouped together as alternative terms of each other. I.e., it suffices if a document contains one of the expansion terms, and matching several of them does not give an additional boost to document similarity score. Moving from unstructured to structured queries requires the minimal weighting scheme to be adapted appropriately, and section 6.3 provides the adapted definitions of the original definitions for minimal span weighting as introduced in section 4.2.

In order to evaluate the effectiveness of query expansion we focused on subsets of the TREC data sets, that contained only questions of the types that are eligible for query expansion, i.e., questions that ask for measurements. In the experiments two types of approaches are compared, minimal span weighting (as baseline) and minimal span weighting of expanded, structured queries. In general, query expansion shows large relative improvements over the baseline, especially at lower cut-off level, but, unfortunately, only a minority of the improvements are statistically significant. This might be due to the rather small sample size, getting statistically significant differences is more difficult. To address the problem of sample size, we put together the three TREC datasets, which then indeed showed statistically significant improvements at lower cut-offs for a@n, p@n, and r@n.

It is also interesting to compare our results on expanding measurement questions to our results on blind feedback expansion in chapter 3. Blind feedback expansion is simply based on co-occurrence information of terms that occur frequently in highly ranked documents of an initial retrieval run. It resulted in statistically significant decreases in retrieval effectiveness. In contrast, the results in this chapter show that query expansion, if done selectively, can lead to improvements.

In our experiments, we did not look at the performance of individual expansion terms, e.g., whether using *celcius* led to better results than *fahrenheit* for retrieving documents for a question ansking for a temperature. It might be interesting to further look into this and use this information to assign better term weights to expansion terms, as in the current system only inverse document frequency is used to weight them.

Finally, in this chapter we focused on questions asking for measurements, but it might be interesting to see how query expansion can be extended to other question types. This can be accomplished by statistically analyzing correct answers and their surrounding words to identify words that frequently co-occur with correct answers, and investigate whether certain words are corellated with a particular question category.

Chapter 7

Evaluating Retrieval within Tequesta

In this chapter, we compare three document retrieval approaches in the context of the Tequesta question answering system to see to what extent the effectiveness of the retrieval module has an impact on the overall end-to-end performance of a particular question answering system. In addition to providing us with an estimate of the impact document retrieval has on the whole question answering process, this comparison should also give us a better understanding of the strengths and weaknesses of the other modules which analyze the documents returned by the retrieval module to identify correct answers.

In the previous chapters, we focused on document retrieval as a means to select documents that are likely to contain an answer to a question. The main task of document retrieval in the context of question answering is to restrict the number of documents that have to be analyzed with more sophisticated—and therefore computationally more expensive—techniques to identify an answer from these documents.

The issue of the overall performance of a question answering system, and the impact document retrieval has on it, has been deliberately neglected in the previous chapters, in order to get a clearer picture of the different retrieval approaches themselves without having the other component of a question answering system influence the overall performance. In this chapter, we compare three retrieval approaches in the context of a specific question answering system. This allows us to investigate how the performance of a retrieval approach affects the overall performance of a question answering system. To this end, we call on our own question answering system Tequesta (Monz and de Rijke, 2001a; Monz et al., 2002), where we use three of the retrieval approaches discussed in the previous chapters and

evaluate their impact with respect to the system's ability to return a correct answer. The question answering systems described in the literature vary widely in the way in which they identify answers in the documents (or passages) returned by the retrieval component. Hence, in order to get a stable indication of the effect which different retrieval approaches have on the overall effectiveness of question answering, the retrieval approaches had to be integrated in a number of question answering systems and then compared with respect to the average changes in performance. Obviously, this is very difficult to realize, as it requires access to a number of question answering systems plus the ability to integrate different retrieval approaches into each of them.

The remainder of this chapter is organized as follows: The next section reviews some previous work on the evaluation of retrieval in the context of a specific question answering system. Section 7.2 provides a brief overview of the architecture of the Tequesta question answering system, that is used for the experiments discussed in this chapter. Section 7.3 describes the experiments that were conducted and discusses the experimental results. Finally, section 7.4 provides some conclusions and discussion on open issues.

7.1 Related Work

Up to now, there is very little work on analyzing the impact of document retrieval as a pre-fetch for question answering.

Tellex (2003); Tellex et al. (2003) compare the impact of eight passage-based and locality-based retrieval strategies that were used by TREC participants. The different approaches are compared with respect to the overall performance of a version of the MIT question answering system, see (Tellex, 2003). Tellex et al. (2003) show that the choice of the retrieval approach that is used for pre-fetching does have a significant impact on the overall performance of a question answering system. In their evaluation, algorithms that take the proximity between terms into account perform best.

Moldovan et al. (2002, 2003) provide an in-depth error analysis of their question answering system. For each component of their system they evaluate in how many cases this particular component is responsible for the system's failure to return a correct answer. One of these components is the document retrieval component. Although they evaluated in how many cases later components failed because of the retrieval component's failure, they did not compare several retrieval strategies, and their respective impact on the system's overall performance.

7.2 Architecture of the Tequesta System

In this section we describe the architecture of our TExtual QUESTion Answering system (Tequesta). Tequesta follows the general architecture as described in chapter

1, containing four main components: question analysis, document retrieval (prefetching), document analysis, and answer selection. The general functionality of each of these components has already been discussed in chapter 1, and in this section we will focus on the specific way each of the components is realized in Tequesta.

7.2.1 Question Analysis

As explained in chapter 1, the first step during question analysis is to determine the class of the question. This is accomplished by applying pattern matching, where each question class is associated with a number of patterns, see table 1.1 for some instances of the question classes that are used by Tequesta, and table 1.2 for a number of example patterns that are used to map a question to a class. Although pattern matching is a simple approach, it is rather accurate. From the 500 TREC-11 questions, only 23 (4.6%) where misclassified and 10 (2%) could not be assigned to any category, meaning that pattern-based classification classified correctly (93.4%) of the questions, with respect to the set of classes that are used by Tequesta. Of course, in some cases pattern-based question classification falls short to assign the correct class. For instance, consider questions (7.1) and (7.2).

(7.1) What is the national anthem in England? (topic id: 1507)

(7.2) What is the boiling point of water? (topic id: 1606)

Both questions are categorized as being of type `thing-ident`, which is the correct class for question (7.1), but question (7.2) should have been more appropriately classified as a question of type `number-temperature`. In order to do so, the classification procedure should know that *boiling point* is a temperature-designating expression. Knowledge of this kind is to some extent captured by machine readable dictionaries and ontologies, such as WORDNET (Miller, 1995). However, using WORDNET to assist question classification requires the phrases in the question to be disambiguated. E.g., consider question (7.3).

(7.3) What is the southwestern-most tip of England? (topic id: 1550)

In WORDNET, the word *tip* is also listed as a term referring to an amount of money, but obviously, question (7.3) does not ask for an amount of money. Given the complications that arise by using WORDNET for question classification, not dismissing its potential benefits, we decided to stick to simple pattern matching.

As discussed in section 1.1, the other role of the question analysis component is to formulate the retrieval query that is passed to the retrieval component. Query formulation is carried out in a number of steps. First, the words in the question are morphologically normalized. We use TREETAGGER (Schmid, 1994) to assign to each word its lexical root, or lemma. Then, stop words are removed from the question. A short list of 133 stop words is used to identify terms that are rather meaningless. Stop words typically include determiners, prepositions, conjunctions, and pronouns. Finally, the lexical roots of the remaining question terms are further

morphologically normalized by applying the Porter stemmer (Porter, 1980). This unordered set of stemmed terms forms the retrieval query.

7.2.2 Document Retrieval

The document retrieval module uses FlexIR (Monz and de Rijke, 2001b, 2002), which is a vector space-based retrieval system. FlexIR allows one to use a wide range of retrieval approaches, some of which were discussed in the previous chapters. In the context of question answering, the standard setting is minimal span weighting, as explained in chapter 4. The retrieval component returns a ranked list of the top 1000 document ids and their corresponding minimal matching spans. Not all of the top 1000 documents are considered by the subsequent modules. The exact number of documents that is further analyzed depends on the specifications of the subsequent modules, in particular the document analysis component, but in general, the top 20 documents are used for further analysis.

7.2.3 Document Analysis

Given the ranked list of documents delivered by the retrieval engine, and the question class, coming from the question analysis module, the document analysis component aims to identify phrases in the top documents that are of the appropriate type to answer the question. Depending on the question class, answer type phrase recognition is accomplished by applying pattern matching, consulting knowledge bases, such as WORDNET and gazetteers, or a combination of both. For a few question classes, table 7.1 shows some of the patterns that are used to identify phrases of the appropriate type. Note that the patterns are a slight simplification of the patterns that are actually used in the implementation of Tequesta, which is mainly done to retain readability. Each phrase that is matched by one of the patterns associated with the question class and that occurs in close proximity to terms from the question, is marked as a candidate answer; see chapter 1 for alternative ways of linking a phrase of the appropriate type to the question.

In the case of questions asking for locations (`location`) or persons (`pers-ident`), Tequesta consults large lists of person and location names, also know as *gazetteers*. For locations, the CLR gazetteer is used,[1] which is a large list of locations, including cities, counties, harbors, countries, etc., containing 162,853 entries in total. To identify person names, we use part of the U.S. Census resource,[2] which contains a list of first and last names. The list of first names contains 4,275 female and 1,219 male first names, and 101,865 last names. If a phrase in a top document matches one of the entries in the relevant database, it is marked as a candidate answer.

Using gazetteers has two shortcomings. Identifying locations by looking them up in a gazetteer tends to result in many false positives, i.e., phrases that match

[1] Available from `http://crl.nmsu.edu/Resources/clr.htm/`.

[2] Available from `http://www.census.gov/genealogy/names/`.

Table 7.1: Sample patters for question classification used in Tequesta

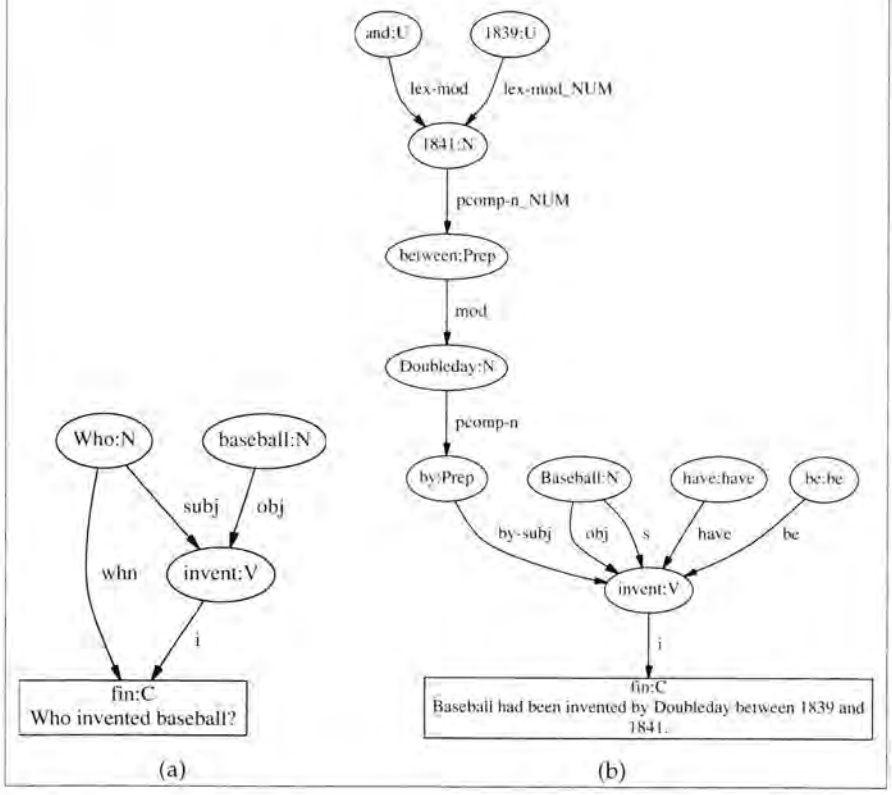
Question class	Example patterns
`date`	`/(in|early|late|during) 1[0-9]{3} /,` `/(early|late|during) (1[0-9])?[2-9]0\'?s/`
`date-birth`	`/(born|baptized|baptised|birth) in 1[0-9]{3}/,` `/\(1[0-9]{3}-1[0-9]{3}\)`
`date-death`	`/(died|buried|cremated|death) in 1[0-9]{3}/,` `/\(1[0-9]{3}-1[0-9]{3}\)/`
`number-height`	`/[0-9\.]+ (feet|foot|meter) (tall|high|height)/`
`number-length`	`/[0-9\.]+ (feet|foot|meter|kilometer) long/,` `/length of [0-9\.]+ (feet|foot|meter|kilometer)/`
`number-money`	`/[0-9\.\,]+ (dollars|pound|cents|bn)/`
`number-speed`	`/[0-9\.\,]+/ (mph|m\.p\.h\.|miles per hour)`
`number-temperature`	`/[0-9\.] degrees (Fahrenheit|Celsius|Centigrade)?/`
`number-time-age`	`/[0-9\.\,]+ (days|weeks|months|years|decades) old /,` `/age of [0-9\.\,]+ (years)? /`
`number-time-distance`	`[0-9\.\,]+ (days|weeks|months|years|decades) ago /`

an entry in the gazetteer, but do not refer to a location in the context provided by the document. To some extent this problem can be solved by the answer selection component, see below, where the frequency of a candidate answer plays a role in selecting the answers that are eventually returned to the user. Using a gazetteer to identify a person name has the disadvantage that lists of person names are inherently incomplete. For instance, neither the first name *Yasser* nor the last name *Arafat* are mentioned in the U.S. Census lists. To overcome this problem, Tequesta also uses patterns to identify names. Often, at the beginning of a document, a person is introduced by a longer description such as *Mr. Yasser Arafat* or *Palestinian leader Yasser Arafat*. These patterns check whether a phrase is preceded by a honorific phrase, such as *Mr.*, *Mrs.*, or a job title, such as *leader*, *president*, or *spokesman*. Job titles are extracted from WORDNET. If such a pattern matches, the phrase is added to the list of person names that are considered for the document at hand.

Analogous to the question classes listed in table 7.1, for questions of the type `location` or `pers-ident`, candidate answers are linked to the question by proximity.

For the question types `agent` and `object`, linking is accomplished in a more sophisticated way. Here, Tequesta compares the dependency parse of the question with the dependency parse of a sentence containing a phrase of the appropriate type (person name or organization name), and some words from the question. Consider the two dependency graphs displayed in figure 7.1. Both graphs were generated by MINIPAR (Lin, 1998), a robust dependency parser. (a) is the graph for the question *Who invented baseball?* (topic id: 244), and (b) is the graph for a sentence containing the correct answer *Doubleday*. Linking the candidate answer to the question is accomplished by partial graph matching. Both graphs contain an edge labeled `obj`, going from *baseball* to *invent*. In the question graph (a), the node of the wh-word

Figure 7.1: Examples of MINIPAR dependency graphs for questions



who designates the slot that has to be filled by the candidate answer. Although (b) does not contain a node that is connected to the *invent* node by an edge labeled `subj` (the way *who* is connected to *invent* in (a)), it does contain a node pointing to *invent* via a `by-subj` edge, which again is pointed to by the *Doubleday* node. The example displayed in figure 7.1 shows that it is also possible to match dependency graphs if they differ in voice (active vs. passive). If the dependency graph of the question and the graph of the answer sentence have the same voice, matching is even more trivial.

Finally, we consider questions of the type `what-np`. Here, documents are scanned for noun phrases that are an instance of the question focus. E.g., a candidate answer has to be an author for question (7.4), a school for question (7.5), and a group for question (7.6).

(7.4) What author wrote under the pen name "Boz"? (topic id: 1741)

(7.5) What school did Emmitt Smith go to? (topic id: 1498)

(7.6) What group sang the song "Happy Together"? (topic id: 1675)

Tequesta uses two strategies to check whether a phrase is an instance of the question focus. WORDNET lists a number of hypernym relations between senses. Optimally, the question focus and the phrases in the document would be sense-disambiguated, but here, we use a simpler approach. If there is a hypernym relationship between one of the senses of the question focus and one of the senses of the noun phrase in the document, Tequesta considers the latter to be an instance of the former. For instance, WORDNET does contain a hypernym relationship between *Dickens* (the correct answer to question (7.4)) and *author*. Note that hypernym relations are transitive, i.e., if x ISA y, and y ISA z, then x ISA z, and Tequesta also considers transitivity when checking for instancehood.

Although WORDNET contains approximately 66,000 hypernym relations between senses of noun phrases, it is inherently incomplete. For example, the correct answer to question (7.5) is *Escambia High School*, but the fact that this is an instance of the question focus *school* is not contained in WORDNET. As a fallback strategy, Tequesta considers noun phrases that have the question focus as their rightmost part, and modify it with additional nouns or adjectives. With respect to question (7.5), the noun phrase *Escambia High School* has the question focus *school* as its rightmost part (case differences are disregarded here), and modifies it with *Escambia High*.

In many cases, neither WORDNET nor the fallback can establish a hypernym relationship, even if it does actually the exist. For instance, to find the correct answer to question (7.6), one has to establish that *The Turtles* are a group, but unfortunately, the only group of musicians listed by WORDNET are *The Beatles*, disregarding instances of other meanings of the word *groups*. There are several approaches to extract some hypernym relations from corpora in an automatic fashion, see, e.g., (Hearst, 1998; Mann, 2002; Fleischman et al., 2003), but this has not yet been integrated into Tequesta.

Similar to the candidate answer selection procedures for the other question types, except `agent` and `object`, phrases that are of the appropriate type are linked to the question by the proximity within which they occur to terms from the question.

7.2.4 Answer Selection

Within Tequesta, answer selection is accomplished by considering the frequency of a candidate answer, an approach which is also known as *redundancy-based answer selection*. Most of the procedures that identify candidate answers rely on linking a candidate to the question by proximity. Hence, all candidate answers are weighted equally. But there are two exceptions. First, if the question is of type `agent` or `object`, and the candidate answer could be linked to the question by partially matching the dependency graph of the question, and the graph of the sentence

containing the candidate answer, this candidate receives a higher score. Second, if the question is of type `what-np`, candidate answers that are in a WORDNET hypernym relationship with the question focus receive a higher weight than candidate answers that are identified by means of the fallback strategy. In the second case, the weight of the candidate answer is actually not based on the confidence with which it is linked to the question, but it is based on the confidence that this phrase is indeed an instance of the question focus.

7.3 Experimental Results

In this section we compare three of the retrieval approaches that were discussed in the previous chapters with respect to their impact on the overall performance of the Tequesta question answering system. In particular, we compare Lnu.ltc weighting, minimal span weighting, and expanded minimal span weighting for measurement questions.

7.3.1 Evaluation Criteria

In the previous chapters, the performance of the retrieval component was measured with metrics based on the criterion of retrieving documents that contain a correct answer to a question. At this point, we are interested in the ability of the Tequesta question answering system to return a correct answer. In chapter 1, we have discussed the way question answering systems are evaluated in TREC's question answering track. Here, we will use the mean reciprocal rank (MRR) as the main evaluation measure, and the ranked list of returned answers is limited to five answers.

Evaluating a question answering system manually is a tedious process, and therefore we use the answer patterns provided by NIST for evaluation. Although using pattern matching to see whether an answer is correct is not as reliable as manual inspection, it still gives a reasonable approximation of the effectiveness of a question answering system, cf. (Voorhees and Tice, 2000a).

We also distinguish between two forms of evaluation. One way is to simply use patterns without checking whether the document from which the answer was extracted is a document that actually contains a correct and supported answer. We will refer to this as *lenient evaluation*. The other way is to check whether a pattern matches the answer, and whether the document is marked as a document that contains a correct and supported answer. We will refer to this as *strict evaluation*. Note that this way of carrying out strict evaluation is an approximation of strict evaluation as it is carried out by the human assessors in the TREC question answering track. This difference is mainly due to the fact that it is beyond the current state-of-the-art to assess automatically whether a document supports a certain answer.

Throughout this section, we limit ourselves to questions that were answered by at least one of the TREC participants, and therefore are provided with a pattern that allows us to identify correct answers. Also, we disregard questions that were known

not to have a correct answer in the respective TREC data set, as the Tequesta system does not try to answer those questions by saying that there is no correct answer in the document collection.

7.3.2 Minimal Span Weighting within Tequesta

As we have seen in chapter 4, minimal span weighting greatly outperforms retrieval based on the Lnu.ltc weighting scheme. Now, the question is to what extent the Tequesta system benefits from the improved retrieval component. In order to focus on the impact of the similarity weighting scheme itself, and not on the text units that are returned by the retrieval component, we had both approaches return the same unit, viz. the minimal matching sentential span, see definition 4.4. Although minimal matching spans were also computed for the Lnu.ltc weighting scheme, they were not used for computing document similarity. The Lnu.ltc weighting scheme is just like the minimal span weighting scheme, see definition 4.3, where only the global similarity is used to compute the retrieval status value, i.e., λ is set to 1.

Table 7.2 shows the percentages of questions that were correctly answered at the respective top-5 ranks, and the MRR score, for the three TREC data sets. As

Table 7.2: Lenient evaluation of Tequesta using Lnu.ltc vs. msw retrieval

	TREC-9			TREC-10			TREC-11		
rank	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
1	16.7%	21.0%	(+25.8%)	17.1%	21.5%	(+25.7%)	16.2%	18.2%	(+12.4%)
2	20.6%	26.9%	(+30.6%)	21.3%	26.3%	(+23.5%)	20.3%	23.2%	(+14.3%)
3	22.9%	29.0%	(+26.6%)	23.1%	28.6%	(+23.8%)	23.2%	26.1%	(+12.5%)
4	25.1%	30.0%	(+19.5%)	24.5%	30.0%	(+22.5%)	25.9%	28.4%	(+9.7%)
5	26.5%	31.4%	(+18.5%)	25.4%	31.0%	(+22.1%)	28.2%	30.6%	(+8.5%)
MRR	0.203	0.252	(+24.1%)▲	0.203	0.252	(+24.1%)▲	0.204	0.227	(+11.3%)△

one can see, using minimal span weighting instead of Lnu.ltc weighting also has a substantial positive effect on the overall performance of the Tequesta system. For all three data sets, the improvements are statistically significant, with a confidence of 99% for TREC-9 and TREC-10, and a confidence of 95% for TREC-11.

Next, we compare both retrieval approaches by using strict evaluation. Table 7.3 shows the results. Again, minimal span weighting clearly outperforms Lnu.ltc weighting, and the improvements are statistically significant for all three TREC data sets.

Unfortunately, the absolute evaluation scores of the Tequesta system are rather low, compared to many other systems participating in TREC over the years. This is mainly due to the fact that for most question classes candidate answers are linked to the question by simply considering proximity, which is too simplistic an approach in many cases.

To get a better understanding of the performance changes brought about by using minimal span weighting for retrieving documents, we take a closer look at the

Table 7.3: Strict evaluation of Tequesta using Lnu.ltc vs. msw retrieval

	TREC-9			TREC-10			TREC-11		
rank	Lnu.ltc	msw		Lnu.ltc	msw		Lnu.ltc	msw	
1	12.3%	15.9%	(+29.3%)	13.2%	16.2%	(+22.7%)	9.9%	11.7%	(+18.2%)
2	15.1%	21.6%	(+43.1%)	16.9%	21.0%	(+24.3%)	11.5%	15.1%	(+31.3%)
3	16.5%	23.3%	(+41.2%)	18.9%	23.3%	(+23.3%)	13.3%	17.6%	(+32.3%)
4	19.2%	24.9%	(+29.7%)	19.6%	23.8%	(+21.4%)	16.2%	19.4%	(+19.8%)
5	21.0%	26.3%	(+25.2%)	20.3%	24.9%	(+22.7%)	16.9%	20.5%	(+21.3%)
MRR	0.152	0.200	(+31.6%)▲	0.160	0.197	(+23.1%)▲	0.122	0.149	(+22.1%)▲

ten most frequent question classes—according to Tequesta's classification scheme—for each of the three TREC data sets. Table 7.4 shows the changes for the TREC-9 data set with respect to the ten most frequent question classes in that data set. Some

Table 7.4: Lnu.ltc vs. msw MRR scores for TREC-9 per question class

		lenient			strict		
question type	freq.	Lnu.ltc	msw		Lnu.ltc	msw	
location	77	0.214	0.295	(+37.9%)	0.147	0.246	(+67.5%)
agent	54	0.203	0.301	(+48.3%)	0.196	0.249	(+27.0%)
what-np	46	0.184	0.207	(+12.5%)	0.104	0.113	(+8.7%)
thing-ident	45	0.049	0.052	(+6.1%)	0.004	0.007	(+75.0%)
date	43	0.348	0.401	(+15.2%)	0.269	0.302	(+12.3%)
thing-def	28	0.286	0.286	(±0.0%)	0.214	0.214	(±0%)
pers-def	23	0.337	0.478	(+41.8%)	0.302	0.443	(+46.7%)
number-many	22	0.341	0.364	(+6.7%)	0.296	0.273	(-7.8%)
pers-ident	22	0.337	0.413	(+22.6%)	0.214	0.367	(+71.5%)
number-many-people	11	0.212	0.272	(+28.3%)	0.212	0.272	(+28.3%)

of the question classes have really low MRR scores, e.g., `thing-ident` and `what-np`, and it becomes evident that simple proximity-based linking does not work for these classes. However, when looking at the classes for which Tequesta is better performing, such as `date`, `pers-def`, and `pers-ident`, one can also see that using minimal span weighting instead of Lnu.ltc weighting results in substantial improvements in effectiveness. The only question class for which Tequesta's performance decreases by using minimal span weighting is `number-many`, and only when using strict evaluation.

Table 7.5 shows the results for the ten most frequent question classes in the TREC-10 data set. For some of the question classes the improvement are extremely high, in particular `thing-ident`, but this is due to their low absolute MRR score. But even for question classes where the absolute MRR scores are higher, e.g., `location`, and `date`, using minimal span weighting still results in a much better performance of the question answering system.

Finally, table 7.6 shows the MRR scores for the ten most frequent question classes

Table 7.5: Lnu.ltc vs. msw MRR scores for TREC-10 per question class

		lenient			strict		
question type	freq.	Lnu.ltc	msw		Lnu.ltc	msw	
thing-def	105	0.195	0.205	(+5.1%)	0.193	0.202	(+4.7%)
thing-ident	58	0.034	0.087	(+155.9%)	0.016	0.035	(+118.6%)
what-np	46	0.221	0.307	(+38.9%)	0.183	0.251	(+37.2%)
location	43	0.338	0.432	(+27.8%)	0.244	0.271	(+37.2%)
date	33	0.493	0.523	(+6.1%)	0.357	0.394	(+10.4%)
agent	20	0.287	0.433	(+50.9%)	0.177	0.358	(+102.3%)
pers-ident	20	0.200	0.300	(+50.0%)	0.150	0.200	(+33.3%)
expand-abbr	12	0.250	0.250	(±0.0%)	0.083	0.167	(+101.2%)
also-known-as	11	0.000	0.109	undef.	0.000	0.109	undef.
date-of-birth	8	0.125	0.186	(+48.8%)	0.125	0.188	(+50.4%)

in the TREC-11 data set. Similar to the other two data sets, Tequesta benefits from

Table 7.6: Lnu.ltc vs. msw MRR scores for TREC-11 per question class

		lenient			strict		
question type	freq.	Lnu.ltc	msw		Lnu.ltc	msw	
date	81	0.401	0.436	(+8.7%)	0.261	0.315	(+20.7%)
thing-ident	72	0.031	0.049	(+58.1%)	0.010	0.021	(+110.0%)
location	66	0.286	0.356	(+2.5%)	0.136	0.243	(+78.7%)
what-np	59	0.187	0.161	(-13.9%)	0.088	0.045	(-48.9%)
agent	24	0.113	0.160	(+42.0%)	0.063	0.097	(+54.0%)
name	22	0.220	0.189	(-14.1%)	0.072	0.061	(-15.3%)
pers-ident	21	0.147	0.111	(-24.5%)	0.075	0.111	(+48.0%)
also-known-as	13	0.015	0.026	(+73.3%)	0.000	0.026	undef.
date-of-birth	9	0.222	0.333	(+50.0%)	0.222	0.333	(+50.0%)
number-height	8	0.292	0.375	(+28.4%)	0.250	0.375	(+50.0%)

using minimal span weighting instead of Lnu.ltc weighting.

The experimental results discussed above confirm that the component-based evaluation of the retrieval module described in chapter 4 gives a good indication of the impact of the retrieval module on a question answering system as a whole.

7.3.3 Expanding Measurement Questions within Tequesta

In chapter 6, we proposed to expand queries for measurement questions, that is questions asking for the height, length, speed, etc. of something or somebody. Queries are expanded with units that are likely to be part of the answer, such as *foot*, *meter*, and *inch*. We have shown that expansion results in higher a@n, p@n, and r@n scores, in particular at lower cut-offs.

Here, we are interested in the impact of expansion on the effectiveness of the

overall question answering process. Table 7.7 shows the results for measurement questions for the three TREC data sets. Surprisingly, the overall performance of

Table 7.7: Lenient evaluation of Tequesta using expanded retrieval for measurement questions

	TREC-9			TREC-10			TREC-11		
rank	msw	+exp		msw	+exp		msw	+exp	
1	15.6%	12.5%	(-19.9%)	17.2%	13.8%	(-19.8%)	19.5%	16.7%	(-14.4%)
2	28.1%	25.0%	(-11.0%)	27.6%	27.6%	(±0.0%)	25.0%	19.5%	(-22.0%)
3	31.3%	25.0%	(-20.1%)	27.6%	27.6%	(±0.0%)	27.8%	22.2%	(-20.1%)
4	31.3%	25.0%	(-20.1%)	31.0%	31.0%	(±0.0%)	27.8%	27.8%	(±0.0%)
5	31.3%	25.0%	(-20.1%)	34.5%	34.5%	(±0.0%)	27.8%	27.8%	(±0.0%)
MRR	0.229	0.188	(-17.9%)	0.240	0.222	(-7.5%)	0.232	0.204	(-12.1%)

Tequesta drops when using minimal span weighting on expanded queries instead of minimal span weighting without expansion. The decrease in performance is substantial, though not statistically significant. As could be expected, Tequesta performance drops also when using strict evaluation. Table 7.8 shows the results for strict evaluation. Again, although substantial, the drop in performance is not statistically

Table 7.8: Strict evaluation of Tequesta using expanded retrieval for measurement questions

	TREC-9			TREC-10			TREC-11		
rank	msw	+exp		msw	+exp		msw	+exp	
1	15.6%	9.4%	(-39.7%)	13.8%	10.4%	(-24.6%)	16.7%	11.1%	(-33.5%)
2	25.0%	25.0%	(±0.0%)	24.1%	24.1%	(±0.0%)	22.2%	11.1%	(-50.0%)
3	28.1%	25.0%	(-11.0%)	27.6%	27.6%	(±0.0%)	27.8%	13.9%	(-50.0%)
4	28.1%	25.0%	(-11.0%)	31.0%	31.0%	(±0.0%)	27.8%	19.5%	(-29.9%)
5	28.1%	25.0%	(-11.0%)	34.5%	34.5%	(±0.0%)	27.8%	19.5%	(-29.9%)
MRR	0.214	0.172	(-19.6%)	0.217	0.199	(-8.3%)	0.213	0.134	(-37.1%)

significant. This could be due to fact that there are not so many measurements questions in the respective data sets, which makes it more difficult to show statistically significant differences. In the previous chapter, we put together all three TREC data sets and statistical significance testing on this larger data set enabled us to detect significant differences. We did the same for the measurement questions with respect to the MRR scores, but failed to show that using expansion results in a drop in performance that is statistically significant.

In the previous chapter, we have seen that retrieval with expanded queries for measurement questions outperforms retrieval with unexpanded queries; resulting in higher a@n, p@n, and r@n scores. However, when evaluating retrieval with expanded queries in the context of the Tequesta system, the overall performance of the question answering system drops. Remains the question what causes this difference in effectiveness. Looking at the top ranked documents from which the candidate answers are extracted one can see that most of the top documents stemming from expanded retrieval contain a measurement phrase, but many of them are not a cor-

rect answer to the question. Since answer selection is mainly based on the frequency of a candidate answer, and expanded retrieval causes more phrases of the appropriate type to be in the top documents, correct answers are sometimes overturned by phrases that are more frequent, but not a correct answer. How answer selection should be adapted to prevent this, remains an issue for further experimentation.

Although the improvements of expanded retrieval over unexpanded retrieval for measurement questions discussed in chapter 5 are not reflected by the overall performance of Tequesta, it sheds some light on the way the different components of the Tequesta system interact.

7.4 Conclusions

In this chapter we have compared three document retrieval approaches used as a pre-fetch for an actual question answering system. The experiments described in chapter 4 showed that minimal span weighting clearly outperforms Lnu.ltc based retrieval, with respect to the ability to identify documents that contain a correct answer. In this chapter we have also seen that the overall performance of the Tequesta question answering system benefits significantly from using minimal span weighting instead of Lnu.ltc weighting. Hence the effectiveness of a retrieval system does have a strong impact on the performance of the whole process of question answering.

However, judging a document retrieval approach by its ability to identify documents that contain a correct answer is not the only aspect that plays a role in selecting a retrieval approach in order to improve the overall performance of a question answering system. This was made explicit by comparing the effectiveness of using expanded queries versus unexpanded queries for measurement questions. Although retrieval using expanded queries is more successful in identifying documents that contain a correct answer than retrieval using unexpanded queries, see chapter 6, retrieval using expanded queries harmed the overall performance of the Tequesta system, albeit that decreases in performance are not statistically significant. The main reason for the decrease in performance seems to be the fact that retrieval using expanded queries returns more documents that contain a phrase that is of the appropriate type to be an answer, which makes it more difficult for alter modules to identify the actual correct answers. In other words, retrieval using expanded queries introduced more noise, which makes it harder to discriminate between documents that do contain an answer and those that do not.

This raises the question whether the evaluation measures that are used throughout chapter 3–6, viz. a@n, p@n, r@n, are appropriate for comparing document retrieval approaches in the context of question answering. In general, we think that they are, but of course idiosyncrasies of the document analysis module and the answer selection module also affect the way document retrieval can function in the context of a particular question answering system.

Chapter 8

Conclusions

This final chapter concludes this thesis by reflecting on the three research questions formulated in chapter 1. We discuss which parts of the thesis address these questions, and what general conclusions can be drawn from that. Finally, some open issues are discussed, and we propose ways in which they could be addressed in future work.

The main motivation for this thesis was to investigate the role of document retrieval in the context of textual question answering. More specifically, we wanted to know what kind of retrieval techniques could be used in order to increase the performance of question answering systems.

Document retrieval is one of the core components of most current textual question answering systems. The retrieval engine's task is to identify documents that are likely to contain a correct answer to a given question. The documents that are returned by the retrieval component are then further analyzed by the subsequent components of the question answering system, such as document analysis and answer selection. The impact of document retrieval becomes particularly obvious when the retrieval component fails to return any documents containing a correct answer. Obviously, in such a situation, even an optimally performing document analysis and answer selection component, will inevitably fail as well in identifying a correct answer. But the impact of the retrieval component on the overall performance of a question answering system is not restricted to situations where the retrieval component fails to return any documents that contain a correct answer. Also the number of returned documents containing correct answers and the variety of ways in which the answers are expressed can have an impact on the performance of question answering systems.

The retrieval techniques we have considered throughout this thesis included

standard document retrieval techniques, as well as novel retrieval techniques which are more tailored to the task of question answering.

8.1 Recapitulation

In the introduction to this thesis, we formulated the following three research questions:

1. Do retrieval techniques that are known to perform well for document retrieval perform equally well when searching for documents that contain an answer to a question?

2. What can be gained from tailoring a document retrieval engine to the task of finding documents that contain an answer to a question?

3. To what extent does the retrieval component affect the overall performance of a question answering system?

Let us go through these questions one by one. The first question was addressed by chapter 3, where we compared a number of retrieval techniques with respect to their ability to identify documents that contain a correct answer to a given question, on the basis of the query that was generated from this question. In the different standard retrieval approaches we looked at, the issues of morphological normalization, passage-based retrieval, and query expansion were addressed. We have found that more aggressive morphological normalization, such as rule-based Porter stemming, is more effective than using a machine readable dictionary to determine a word's syntactic root. This is a little surprising as rule-based stemming is better known for its ability to enhance recall (Kraaij and Pohlmann, 1996), rather than improving early high precision which seems to be more desirable in the context of question answering.

Blind feedback has become a standard technique in document retrieval because of its consistent and strong positive impact on retrieval effectiveness, cf. (Mitra et al., 1998; Robertson and Walker, 1999). However, we have found it to have a dramatic negative impact on the effectiveness when applied in the context of question answering. This is likely to be due to the fact that there are much less relevant documents in question answering than in ad hoc retrieval, and that the information that allows one to answer the question is expressed very locally, while our blind feedback approach used full documents to identify terms that are used for query expansion. One way to address the issue of locality is to use a local feedback approach such as local context analysis (Xu and Croft, 1996).

Passage-based retrieval has proved particularly useful for document collections that contain longer documents. It is also widely used for question answering, as it exploits the fact that answers to a question tend to be found in a sentence or two. We experimented with a large number of different passage sizes, we did not

find passage-based retrieval to outperform full document retrieval. Although these experimental results might appear somewhat counterintuitive, in recent work by Clarke and Terra (2003) similar findings are reported.

Summing up our experimental results with respect to the first research question, we can say that document retrieval behaves quite different in the context of question answering than ad hoc document retrieval:

- Morphological normalization, which is not known for having a strong impact on ad hoc retrieval effectiveness, has a statistically significant positive impact on retrieval for question answering.
- Blind relevance feedback, which is known for having a substantial positive impact on ad hoc retrieval, results in dramatic decreases in effectiveness when applied to retrieval for question answering.
- Passage-based retrieval, which is used by many question answering systems does not yield better results in identifying documents that contain a correct answer than full document retrieval.

From these observations one can conclude that retrieval for question answering is indeed substantially different from regular ad hoc retrieval and will therefore benefit from retrieval techniques that are tailored to the task of question answering.

This brings us to the second research question: *What can be gained from tailoring a document retrieval engine to the task of finding documents that contain an answer to a question?* We addressed this question in chapters 4–6. First, in chapter 4, we took another look at the assumption that answers tend to be expressed rather locally. As passage-based retrieval did not result in the expected improvements, a more flexible approach to retrieval considering locality might be more successful. To investigate this, we introduced a new proximity-based approach to retrieval, which we coined *minimal span weighting*. This approach takes into account the proximity within which matching terms occur in a document as well as the number of terms from the question occur in the document. This new weighting scheme resulted in substantial and statistically significant improvements.

Next to the choice of an appropriate retrieval approach, the issue of selecting words from the original question has a strong impact on retrieval effectiveness. In chapter 5, we have seen that much can be gained by optimal query term selection. In order to approximate optimal term selection, we used machine learning techniques assigning weights to the words in the question, where the weight represents how useful this term is for retrieving documents containing an answer. Unfortunately, using the learned query term weights for query formulation did not result in significant improvements. Nevertheless we believe that the machine learning framework we introduced in chapter 5 offers a good starting point to further investigate this issue in future work.

Chapter 5 covers one aspect of query formulation, namely selecting terms from the original question. In chapter 6, we focus on another aspect, namely adding

terms to retrieval queries. Query expansion is done for measurement questions, i.e., questions asking for the height, length, age, etc., of something or somebody. The experimental results show that expanding queries results in statistically significant improvements, in particular at lower cut-offs.

Summing up our findings from chapters 4–6, we can say that tailoring document retrieval to the task of question answering leads to significant improvements. Further, document retrieval for question answering can benefit from retrieval strategies that are tailored to the type of question that is asked, which is illustrated here by the improvements for measurement questions.

The experiments in chapters 3–6 focus on the potential impact that document retrieval can have on the overall performance of a question answering system. While conducting these experiments, we deliberately did not integrate the different retrieval approaches into a specific question answering system and we did not evaluate the changes in effectiveness of the complete system. When evaluating a complete question answering system the quality of the other components plays an important role as well, as they may distort our view on the potential impact of different retrieval approaches. However, at the end of the day we were obviously interested in answering our third main research question: *To what extent does the retrieval component affect the overall performance of a question answering system?*

Chapter 7 addresses this question by integrating three different document retrieval approaches into our Tequesta question answering system. Experimental results show that using minimal span weighting results in a significantly better performance of Tequesta than using Lnu.ltc based retrieval, which is in line with our findings in chapter 4. On the other hand, using expanded queries for measurement questions results in a decrease of Tequesta's performance when compared to using unexpanded queries. This is contrary to our findings described in chapter 6, where retrieval with expanded queries outperformed retrieval with unexpanded queries for measurement questions, if only the retrieval component itself is evaluated. When using expanded queries, many of the documents returned by the retrieval engine contain a phrase that counts as a candidate answer, which makes it more difficult for the answer selection component to choose a correct answer. The results on query expansion illustrate that QA components other than retrieval (such as document analysis and answer selection) may be sensitive to the kinds of documents that are returned by the retrieval component.

The results in chapter 7 illustrate two things: First, evaluating the retrieval component by itself is a reliable indicator of the general overall performance of the Tequesta question answering system, provided that the same kind of information needs, viz. questions, are used. Second, for particular question classes, the impact of the retrieval component on the overall performance depends on the way the other components of the question answering system are realized. Therefore, optimizing the retrieval component with respect to its stand-alone effectiveness is beneficial for question answering in general, but idiosyncrasies of the other components still play an important role.

Note that we did not integrate into Tequesta the results of learning term weighting, described in chapter 5, in order to assess their effect on the overall performance of Tequiesta, as they only led to minor improvements of the retrieval component.

8.2 Future Directions

Some of the issues that were addressed in the previous chapters raise follow-up questions that are beyond the scope of this thesis. In chapter 3, we saw that blind relevance feedback results in a dramatic decrease in retrieval performance. One of the explanations for this decrease seems to be the fact that terms from the full document are eligible for query expansion. But as we saw in chapter 4, locality is an important aspect for retrieving documents containing an answer and seems worthwhile to consider more sophisticated feedback techniques, such as local context analysis (Xu and Croft, 1996), that consider for expansion only terms in the proximity of matching terms from the original query. This could be nicely integrated with minimal matching spans, as described in chapter 4.

Slightly orthogonal to this, but still related to the issue of locality, is the potential use of our minimal matching spans for presenting question answering systems' output to users. Recently, Lin et al. (2003) conducted a user study that shows that a large majority of users prefers returned answers to be surrounded by some context instead of isolated exact answers. We have high hopes that our minimal spans provide a good and highly focused notion of context in which to present systems' output to users. Further research, based on experiments with end-users, are required to confirm this believe.

Next, in the question answering research community one can observe an increased usage of machine learning methods, for all components in the QA pipeline. For instance, in the past 18 months various groups have explored the use of machine learning techniques for question classification, see e.g., (Li and Roth, 2002; Suzuki et al., 2003; Zhang and Lee, 2003). Obviously, training data is essential if there is to be progress in the application of machine learning methods for QA (in the case of question classification, Li and Roth (2002) built and made available a set of 5,500 questions classified by hand). Now, returning to our own machine learning experiments in chapter 5, one particularly interesting question that we would like to investigate further is: What is the impact of having more training data available when using machine learning to assign weights to query terms and use that information to select words from the original question to formulate a retrieval query?

Another topic for future research is to further investigate the interaction between the different components in a question answering system. For instance, in chapter 6, we saw that query expansion does have a substantial positive impact on the retrieval effectiveness, when evaluated in isolation. On the other hand, the overall performance of Tequesta dropped for measurement questions, which requires better answer selection criteria to exploit the potential that is offered by retrieval with expanded queries.

More generally, we saw that one of the main outcomes of the thesis is that document retrieval for question answering has to respond to very different information needs than regular document retrieval, and this causes retrieval strategies to behave differently, depending on which of the two task they are applied to. Some of the techniques that have been introduced for retrieval over the years and that have failed to show an increase in effectiveness in, say, mean average precision, for ad hoc retrieval are worth a second look in the context of question answering. In particular, it would be interesting to re-consider techniques that are known to boost early precision.

In addition, it seems interesting to investigate whether some of the techniques that have been developed in this thesis can be applied successfully to tasks other than ad hoc retrieval or question answering. Very recently, we have conducted some pilot experiments on using minimal span weighting for identifying web pages in the context of the named page task of TREC's web track (Craswell and Hawking, 2002), and the results indicate substantial improvements over our baseline. This indicates that some of the retrieval techniques used for question answering are also applicable to other retrieval tasks with more specific information needs.

Bibliography

E. Agichtein, S. Lawrence, and L. Gravano. Learning search engine specific query transformations for question answering. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pages 169–178, 2001. 121

E. Agichtein, S. Lawrence, and L. Gravano. Learning to find answers to questions on the web. *ACM Transactions on Internet Technology*, to appear. 121

AltaVista. `http://www.altavista.com/`. 1

I. Androutsopoulos, G. Ritchie, and P. Thanisch. Natural language interfaces to databases—An introduction. *Natural Language Engineering*, 1(1):29–81, 1995. 2

Ask Jeeves. `http://www.askjeeves.com/`. 12

G. Attardi, A. Cisternino, F. Formica, M. Simi, and A. Tommasi. PiQASso: Pisa question answering system. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 633–641. NIST Special Publication 500-250, 2001. 33

G. Attardi, A. Cisternino, F. Formica, M. Simi, and A. Tommasi. Web suggestions and robust validation for QA. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 714–723. NIST Publication, 2002. 32

R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999. 36

N. Belnap and T. Steel. *The Logic of Questions and Answers*. Yale University Press, 1976. 20

D. Bikel, R. Schwartz, and R. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 1(3):211–231, 1999. 31

F. Black. *A Deductive Question-Answering System*. PhD thesis, Division of Engineering and Applied Physics, Harvard University, 1964. 36

E. Breck, J. Burger, L. Ferro, L. Hirschman, D. House, M. Light, and I. Mani. How to evaluate your question answering system every day and still get real work done. In *Proceedings of the 2nd Conference on Language Resources and Evaluation (LREC-2000)*, 2000. 84

L. Breimann. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996. 117

L. Breimann, J. Friedman, R. Ohlsen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984. 109, 113

E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. In *Proceedings of Emperical Methods in Natural Language Processing (EMNLP 2002)*, pages 257–264, 2002. 90, 91

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In E. Voorhees and D. Harman, editors, *Proceedings of the 10th Text REtrieval Conference (TREC 2001)*, pages 393–400. NIST Special Publication 500-250, 2001. 32

W. Bronnenberg, H. Bunt, J. Landsbergen, R. Scha, W. Schoenmakers, and E. van Utteren. The question answering system PHLIQA1. In L. Bolc, editor, *Natural Language Question Answering Systems*, pages 217–305. MacMillan, 1980. 2, 29

S. Buchholz and W. Daelemans. Complex answers: a case study using a WWW question answering sustem. *Natural Language Engineering*, 7(4):301–323, 2001. 32

C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In D. Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 69–80. NIST Special Publication 500-215, 1994. 90

C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using SMART: TREC 4. In D. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. NIST Special Publication 500-236, 1995. 57, 58, 68, 71

C. Buckley and E. Voorhees. Evaluating evaluation measure stability. In N. Belkin, P. Ingwersen, and M. Leong, editors, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40, 2000. 48, 49

C. Buckley and J. Walz. SMART in TREC 8. In E. Voorhees and D. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 577–582. NIST Special Publication 500-246, 1999. 47

J. Burger, F. Ferro, W. Greiff, J. Henderson, S. Mardis, A. Morgan, and M. Light. MITRE's Qanda at TREC-11. In E. Voorhees and L. Buckland, editors, *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, pages 457–466. NIST Special Publication 500-251, 2002. 43

R. Burke, K. Hammond, V. Kulyukin, S. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently-asked question files: Experiences with the FAQ Finder system. *AI Magazine*, 18(2):57–66, 1997. 12

J. Callan. Passage-retrieval evidence in document retrieval. In B. Croft and C. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310, 1994. 51, 63

H. Chen, G. Shankaranarayanan, L. She, and A. Iyer. A machine learning approach to inductive query by examples: An experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing. *Journal of the American Society for Information Science*, 49(8):693–705, 1998. 91

J. Chu-Carroll, J. Prager, C. Welty, K. Czuba, and D. Ferrucci. A multi-strategy and multi-source approach to question answering. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 124–133. NIST Publication, 2002. 50

C. Clarke and G. Cormack. Shortest substring ranking and retrieval. *ACM Transactions on Information Systems*, 18(1):44–78, 2000. 70

C. Clarke, G. Cormack, G. Kemkes, M. Laszlo, T. Lynam, E. Terra, and P. Tilke. Statistical selection of exact answers (MultiText experiments for TREC 2002). In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 162–170. NIST Publication, 2002a. 10, 50, 70

C. Clarke, G. Cormack, D. Kisman, and T. Lynam. Question answering by passage selection (MultiText experiments for TREC-9). In E. Voorhees and D. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 673–683. NIST Special Publication 500-249, 2000a. 50, 70

C. Clarke, G. Cormack, M. Laszlo, T. Lynam, and E. Terra. The impact of corpus size on question answering performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 369–370, 2002b. 10

C. Clarke, G. Cormack, and E. Tudhope. Relevance ranking for one to three term queries. *Information Processing and Management*, 36(2):291–311, 2000b. 69

C. Clarke and E. Terra. Passage retrieval vs. document retrieval for factoid question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 427–428, 2003. 45, 149

P. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, 1995. 55

A. Colmerauer. Metamorphosis grammars. In *Natural Language Communication with Computers*, pages 33–189. Springer, 1978. 37

W. Conover. *Practical Nonparametric Statistics*. John Wiley and Sons, 2nd edition, 1980. 54

W. Cooper. Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval. *ACM Transactions on Information Systems*, 13(1):100–111, 1995. 94

W. Cooper, A. Chen, and F. Gey. Full text retrieval based on probalistic equations with coefficients fitted by logistic regression. In D. Harman, editor, *Proceedings of the 2nd Text REtrieval Conference (TREC-2)*, pages 57–66. NIST Special Publication 500-215, 1993. 91

G. Cormack, C. Clarke, C. Palmer, and D. Kisman. Fast automatic passage ranking. In E. Voorhees and D. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 735–742. NIST Special Publication 500-246, 1999. 70

N. Craswell and D. Hawking. Overview of the TREC 2002 web track. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 248–257. NIST Publication, 2002. 87, 152

A. Davison and D. Hinkley. *Bootstrap Methods and Their Application*. Cambridge University Press, 1997. 55

A. Davison and D. Kuonen. An introduction to the bootstrap with applications in R. *Statistical Computing and Graphics Newsletter*, 13(1):6–11, 2002. 55

O. de Kretser and A. Moffat. Effective document presentation with a locality-based similarity heuristic. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 113–120, 1999a. 69, 70

O. de Kretser and A. Moffat. Locality-based information retrieval. In *Proceedings of the 10th Australasian Database Conference*, pages 177–188, 1999b. 69, 70

T. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18 (4):97–136, 1997. 117

P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2–3):103–130, 1997. 109

R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. John Wiley and Sons, 1973. 107

S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. Web question answering: is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 291–298, 2002. 10

B. Efron. Bootstrap methods: Another look at the jackknife. *Annals of Statistics*, 7(1): 1–26, 1979. 54

B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993. 54

F. Eibe, Y. Wang, S. Inglis, G. Holmes, and I. Witten. Using model trees for classification. *Machine Learning*, 32(1):63–76, 1998. 109

Encarta. Encarta encyclopedia. `http://encarta.msn.com/`. 12

Excite. Excite search engine. `http://www.excite.com/`. 12

R. Fano. *Transmissions of Information: A Statistical Theory of Communications.* MIT Press, 1961. 33

U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the Internation Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 1022–1029, 1993. 108

O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, C. Jacquemin, N. Masson, and P. Lecuyer. QALC—The question-answering system of LIMSI-CNRS. In E. Voorhees and D. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 235–244. NIST Special Publication 500-249, 2000. 122

C. Fillmore. *The Case for Case.* Holt, Rinehart and Winston, 1968. 34, 42

M. Fleischman, E. Hovy, and A. Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-2003)*, pages 1–7, 2003. 139

E. Frank and M. Hall. A simple approach to ordinal classification. In L. De Raedt and P. Flach, editors, *Proceedings of the 12th European Conference on Machine Learning (EMCL 2001)*, Lecture Notes in Artificial Intelligence 2167, pages 145–156. Springer, 2001. 108

E. Frank, L. Trigg, G. Holmes, and I. Witten. Naive bayes for regression. *Machine Learning*, 41(1):5–25, 2000. 109

Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory*, pages 23–37. Springer, 1995. 117

Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In L. Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156. Morgan Kaufmann, 1996. 117

J. Ginzburg. Interrogatives: Questions, facts and dialogue. In S. Lappin, editor, *Handbook of Contemporary Semantic Theory*, pages 385–422. Blackwell, 1995. 18

Google. `http://www.google.com/`. 1

A. Graesser and T. Murachver. Symbolic procedures of question answering. In A. Graesser and J. Black, editors, *The Psychology of Questions*, pages 15–88. Erlbaum, 1985. 22, 23, 24, 26, 41

N. Graesser, A. Person and J. Huber. Mechanisms that generate questions. In T. Lauer, E. Peacock, and A. Graesser, editors, *Questions and Information Systems*, pages 167–187. Lawrence Erlbaum Associates, 1992. 26

B. Green, A. Wolf, C. Chomksy, and K. Laughery. Baseball: An automatic question answerer. In E. Figenbaum and J. Fledman, editors, *Computers and Thought*, pages 207–216. McGraw-Hill, 1963. 2, 27

M. Greenwood, I. Roberts, and R. Gaizauskas. The University of Sheffield TREC 2002 Q&A system. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 724–733. NIST Publication, 2002. 50

J. Groenendijk and M. Stokhof. Questions. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, chapter 19, pages 1055–1124. Elsevier/MIT Pess, 1997. 18

Grolier. *The Academic American Encyclopedia*. Grolier Electronic Publishing, 1990. 35

C. Hamblin. Questions. *Australasian Journal of Philosophy*, 36:159–168, 1958. 18

S. Harabagiu and D. Moldovan. TextNet—a text-based intelligent system. *Natural Language Engineering*, 3(2):171–190, 1997. 27

S. Harabagiu, D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, R. V., and P. Morarescu. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, pages 274–281, 2001. 9, 10, 33, 36

D. Harman. Ranking algorithms. In R. Baeza-Yates and W. Frakes, editors, *Information Retrieval: Data Structures & Algorithms*, chapter 14, pages 363–392. Prentice Hall, 1992. 75

D. Harrah. The logic of questions. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume II, chapter 12, pages 715–764. Kluwer Academic Publishers, 1984. 18

D. Hawking and P. Thistlewaite. Proximity operators—So near and yet so far. In D. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 131–143. NIST Special Publication 500-236, 1995. 69

D. Hawking and P. Thistlewaite. Relevance weighting using distance between term occurrences. Technical Report TR-CS-96-08, Department of Computer Science, Australian National University, 1996. 69, 70

D. Hays. Automatic language data processing. In H. Borko, editor, *Computer Applications in the Behavioral Sciences*, pages 394–423. Prentice-Hall, 1962. 32

M. Hearst. Automated discovery of wordnet relations. In C. Fellbaum, editor, *WordNet: An Electronical Lexical Database*, chapter 5, pages 131–151. MIT Press, 1998. 139

U. Hermjakob. Parsing and question classification for question answering. In *Proceedings of the Workshop on Open-Domain Question Answering at ACL-2001.*, 2001. 6

J. Hertz, R. Palmer, and A. Krogh. *Introduction to the Theory of Neural Computation.* Addison-Wesley, 1991. 107

M. Hollander and D. Wolfe. *Nonparametric Statistical Methods.* John Wiley and Sons, 1973. 54

E. Hovy, L. Gerber, U. Hermjakob, M. Jink, and C. Lin. Question answering in Webclopedia. In E. Voorhees and D. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 655–664. NIST Special Publication 500-249, 2000. 43

E. Hovy, L. Gerber, U. Hermjakob, C. Lin, and D. Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the DARPA Human Language Technology conference (HLT-2001)*, pages 339–345, 2001. 122

D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–38, 1993. 54

D. Hull. Stemming algorithms—a case study for detailed evaluation. *Journal of the American Society for Information Science*, 47(1):70–84, 1996. 49

A. Ittycheriah, M. Franz, and S. Roukos. IBM's statistical question answering system—TREC-10. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 258–264. NIST Special Publication 500-250, 2001. 65

B. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: A study and analysis of user queries on the web. *Information Processing and Management*, 36(2):207–227, 2000. 69

L. Karttunen. Syntax and semantics of questions. *Linguistics and Philosophy*, 1(1): 3–44, 1977. 19

M. Kaszkiel and J. Zobel. Passage retrieval revisited. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, 1997. 50

M. Kaszkiel and J. Zobel. Term-ordered query evaluation versus document-ordered query evaluation for large document databases. In B. Crof, A. Moffat, C. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 343–344, 1998. 75

M. Kaszkiel and J. Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52(4):344–364, 2001. 50

B. Katz, J. Lin, and S. Felshin. Gathering knowledge for a question answering system from heterogeneous information sources. In *Proceedings of the ACL 2001 Workshop on Human Language Technology and Knowledge Management*, 2001. 33

E. Keen. Term position ranking: Some new test results. In N. Belkin, P. Ingwersen, A. Pejtersen, and E. Fox, editors, *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 66–76, 1992. 69, 70

S. Keenan, A. Smeaton, and G. Keogh. The effect of pool depth on system evaluation in TREC. *Journal of the American Society for Information Science and Technology*, 52 (7):570–574, 2001. 47

M. Kendall. A new measure of rank correlation. *Biometrika*, 30(1–2):81–93, 1938. 52

L. Kitchens. *Exploring Statistics: A Modern Introduction to Data Analysis and Inference.* Brooks/Cole Publishing Company, 2nd edition, 1998. 54

R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 114–119, 1996. 108

W. Kraaij and R. Pohlmann. Viewing stemming as recall enhancement. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 40–48, 1996. 59, 148

J. Kupiec. MURAX: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 181–190, 1993. 35

C. Kwok, O. Etzioni, and D. Weld. Scaling question answering to the web. In *Proceedings of the 10th Wordl Wide Web Conference (WWW'10)*, pages 150–161, 2001a. 90

K. Kwok, L. Grunfeld, N. Dinsl, and M. Chan. TREC-9 cross language, web and question answering track experiments using PIRCS. In E. Voorhees and D. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 419–429. NIST Special Publication 500-249, 2000. 70

K. Kwok, L. Grunfeld, N. Dinsl, and M. Chan. TREC2001 question-answer, web and cross language experiments using PIRCS. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 452–465. NIST Special Publication 500-250, 2001b. 43

W. Lehnert. *The Process of Question Answering: A Computer Simulation of Cognition.* Lawrence Erlbaum Associates, 1978. 24, 26, 38, 39, 41

W. Lehnert. A computational theory of human question answering. In A. Joshi, B. Webber, and I. Sag, editors, *Elements of Discourse Understanding*, pages 145–176. Cambridge University Press, 1981. 38

W. Lehnert. Cognition, computers and car bombs: How Yale prepared me for the 90's. In R. Schank and E. Langer, editors, *Beliefs, Reasoning, and Decision Making: Psycho-logic in Honor of Bob Abelson*, pages 143–173. Lawrence Erlbaum Associates, 1994. 41

S. Levinson. *Pragmatics.* Cambridge University Press, 1983. 22

W. Li. Question classification using language modeling. Technical Report IR-259, Center for Intelligent Information Retrieval, University of Massachusetts, 2002. 6

X. Li and D. Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 556–562, 2002. 151

D. Lin. Dependency-based evaluation of minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, 1998. 97, 137

D. Lin and P. Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001. 97

J. Lin, A. Fernandes, B. Katz, G. Marton, and S. Tellex. Extracting answers from the web using data annotation and data mining techniques. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 474–482. NIST Publication, 2002. 32

J. Lin, D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz, and D. Karger. What makes a good answer? The role of context in question answering. In *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction (INTERACT-2003)*, 2003. 151

F. Llopis, A. Ferrández, and J. Vicedo. Passage selection to improve question answering. In *Proceedings of the COLING 2002 Workshop on Multilingual Summarization and Question Answering*, 2002. 45, 52, 63, 68

H. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165, 1958. 68, 70

B. Magnini, M. Negri, R. Prevete, and H. Tanev. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 425–432, 2002. 10

B. Magnini and R. Prevete. Exploiting lexical expansions and boolean compositions for web querying. In *ACL Workshop on Recent Advances in Natural Language Processing and Information Retrieval*, 2000. 120, 121

G. Mann. Fine-grained proper noun ontologies for question answering. In *Proceedings of SemaNet'02: Building and Using Semantic Networks*, 2002. 139

M. Maybury. Toward a question answering roadmap. Technical report, MITRE, 2002. 84

G. Miller. WORDNET: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. 9, 27, 38, 135

M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–214, 1998. 50, 148

D. Moldovan, M. Paşa, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems (ToIS)*, 21(2):133–154, 2003. 45, 134

D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 33–40, 2002. 45, 134

C. Monz. Document retrieval in the context of question answering. In F. Sebastiani, editor, *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03)*, Lecture Notes in Computer Science 2633, pages 571–579. Springer, 2003. 64

C. Monz and M. de Rijke. Tequesta: The University of Amsterdam's textual question answering system. In *Proceedings of Tenth Text Retrieval Conference (TREC-10)*, pages 513–522, 2001a. 5, 133

C. Monz and M. de Rijke. The University of Amsterdam at CLEF 2001. In *Proceedings of the Cross Language Evaluation Forum Workshop (CLEF 2001)*, pages 165–169, 2001b. 49, 136

C. Monz and M. de Rijke. Shallow morphological analysis in monolingual information retrieval for Dutch, German and Italian. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Proceedings of the 2nd Workshop of the Cross-Language Evaluation Forum (CLEF 2001)*, LNCS 2406, pages 262–277. Springer Verlag, 2002. 49, 136

C. Monz, J. Kamps, and M. de Rijke. The University of Amsterdam at TREC 2002. In E. Voorhees and L. Buckland, editors, *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, pages 603–614. NIST Special Publication 500-251, 2002. 5, 133

C. Mooney and R. Duval. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Sage Quantitative Applications in the Social Science Series No. 95. Sage Publications, 1993. 55

MSN Search. `http://search.msn.com/`. 12

D. Musser and A. Saini. *STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library*. Addison Wesley, 1996. 75

S. Na, I. Kang, S. Lee, and J. Lee. Question answering approach using a wordnet-based answer type taxonomy. In E. Voorhees and L. Buckland, editors, *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, pages 512–519. NIST Special Publication 500-251, 2002. 43

M. Paşca. *High-Performance Open-Domain Question Answering from Large Text Collections*. PhD thesis, Southern Methodist University, 2001. 45, 91, 95, 108, 116

D. Palmer and M. Hearst. Adaptive sentence boundary disambiguation. In *Proceedings of the Fourth Conference on Applied Natural Language Processing*, pages 78–83, 1994. 82

H. Peat and P. Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science (JASIS)*, 42(5):378–383, 1991. 90

C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors. *Proceedings of the 2nd Workshop of the Cross-Language Evaluation Forum (CLEF 2001)*, Lecture Notes in Computer Science 2406, 2002. Springer Verlag. 48

A. Phillips. A question-answering routine. Memo. 16, Artificial Intelligence Project, 1960. 4, 31

M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980. 7, 49, 136

J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184–191, 2000. 35, 122

J. Prager, D. Radev, E. Brown, A. Coden, and V. Samn. The use of predictive annotation for question answering in TREC8. In E. Voorhees and D. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 399–410. NIST Special Publication 500-246, 1999. 122

W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes In C*. Cambridge University Press, 1988. 107

Prise. Z39.50/Prise 2.0. `www.itl.nist.gov/iaui/894.02/works/papers/zp2/zp2.html`, Accessed in February 2003. 44, 47

J. Quinlan. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence (AI'92)*, pages 343–348, 1992. 109

J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. 107, 109

B. Raphael. *SIR: A Computer Program for Semantic Information Retrieval*. PhD thesis, MIT, Mathematics Department, 1964. 37

Y. Rasolofo and J. Savoy. Term proximity scoring for keyword-bases retrieval systems. In F. Sebastiani, editor, *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03)*, Lecture Notes in Computer Science 2633, pages 207–218. Springer, 2003. 70

D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002)*, pages 41–47, 2002. 9

D. Ravichandran, A. Ittycheriah, and S. Roukos. Automatic derivation of surface text patterns for a maximum entropy based question answering system. In *Proceedings of the 3rd Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 85–87, 2003. 9

J. Reynar and A. Ratnaparkhi. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, 1997. 82

T. Rietveld and R. van Hout. *Statistical Techniques for the Study of Language and Language Behaviour*. Mouton de Gryter, 1993. 55

I. Roberts. Information retrieval for question answering. Master's thesis, University of Sheffield, 2002. 45, 52

S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33 (4):294–304, 1977. 94

S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976. 94

S. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In E. Voorhees and D. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 151–162. NIST Special Publication 500-246, 1999. 45, 50, 148

S. Robertson, S. Walker, and M. Beaulieu. Okapi at TREC-7: Automatic ad hoc, filtering, VLC and interactive. In *The 7th Text REtrieval Conference (TREC 7)*, pages 253–264. NIST Special Publication 500-242, 1998. 45, 70

M. Robnik-Šikonja and I. Kononenko. An adaptation of relief for attribute estimation on regression. In D. Fisher, editor, *Proceedings of 14th International Conference on Machine Learning ICML'97*, 1997. 114

M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning*, to appear, 2003. 114

P. Roget. *Roget's International Thesaurus*. Thomas Y. Crowell Publisher, 1946. 33

A. Roventini, A. Alonge, F. Bertagna, N. Calzolari, B. Magnini, and R. Martinelli. Italwordnet, a large semantic database for Italian. In *Proceedings of the 2nd Conference on Language Resources and Evaluation (LREC-2000)*, pages 783–790, 2000. 121

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988. 8, 57, 58, 68, 126

G. Salton, C. Buckley, and C. Yu. An evaluation of term dependence models in information retrieval. In *Proceedings of the 5th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 51–173, 1982. 94

G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983. 80

G. Sampson. *English for the Computer—The SUSANNE Corpus and Analytic Scheme*. Clarendon Press, 1995. 97

M. Sanderson. Retrieving with good sense. *Information Retrieval*, 2(1):49–69, 2000. 121

B. Santorini. *Part-of-speech tagging guidelines for the Penn Treebank*. Deptartement of Computer Science, University of Pennsylvania, 3rd revision, 2nd printing edition, 1990. 95

J. Savoy. Statistical inference in retrieval effectiveness evaluation. *Information Processing and Management*, 33(4):495–512, 1997. 54

R. Scha. *Logical Foundations for Question Answering*. PhD thesis, University of Groningen, 1983. 2, 21, 29

R. Schank. *Conceptual Information Processing*. Elsevier Science Publishers, 1975. 39

H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, 1994. 49, 82, 95, 135

S. Scott and R. Gaizauskas. University of Sheffield TREC-9 Q&A system. In E. Voorhees and D. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 635–XX. NIST Special Publication 500-249, 2000. 45

J. Searle. *Speech Acts*. Cambridge University Press, 1969. 22

S. Siegel and N. Castellan. *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, 2nd edition, 1988. 54, 129

R. Simmons. Answering english questions by computer: A survey. *Communications of the ACM*, 8(1):53–70, 1965. 17, 27

R. Simmons. Natural language question answering systems: 1969. *Communications of the ACM*, 13(1):15–30, 1969. 27

R. Simmons, S. Klein, and K. McConlogue. Indexing and dependency logic for answering english questions. *American Documentation*, 15(3):196–204, 1963. 4, 32

A. Singhal, G. Salton, M. Mitra, and C. Buckley. Document length normalization. *Information Processing & Management*, 32(5):619–633, 1996. 57, 58, 69

M. Soubbotin and S. Soubbotin. Patterns of potential answer expressions as clues to the right answer. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 134–143. NIST Special Publication 500-250, 2001. 9

M. Soubbotin and S. Soubbotin. Use of patterns for detection of likely answer strings: A systematic approach answer. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 175–182. NIST Special Publication, 2002. 9

K. Sparck Jones. Automatic indexing. *Journal of Documentation*, 30(4):393–432, 1974. 54

K. Sparck Jones. Automatic language and information processing: Rethinking evaluation. *Natural Language Engineering*, 7(1):29–46, 2001. 51

K. Sparck Jones and C. van Rijsbergen. Report on the need for and provision of an ideal information retrieval test collection. Technical Report British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975. 46

R. Srihari and W. Li. Information extraction supported question answering. In E. Voorhees and D. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 185–196. NIST Special Publication 500-246, 1999. 3

J. Suzuki, H. Taira, Y. Sasaki, and E. Maeda. Question classification using HDAG kernel. In *Proceedings of the ACL 2003 Workshop on Multilingual Summarization and Question Answering*, 2003. 6, 151

S. Tellex. Pauchok: A modular framework for question answering. Master's thesis, Massachusetts Institute of Technology, 2003. 45, 134

S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–47, 2003. 45, 134

J. Thorne. Automatic language analysis. ASTIA 297381, Final Technical Report, Arlington, Va., 1962. 4, 33

C. van Rijsbergen. *Information Retrieval*. Butterworths, 2nd edition, 1979. 54

J. Vicedo, L. Llopis, and A. Ferrández. University of Alicante experiments at TREC 2002. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 557–564. NIST Special Publication, 2002. 50

E. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 61–69, 1994. 90

E. Voorhees. Overview of the TREC-9 question answering track. In E. Voorhees and D. Harman, editors, *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 71–80. NIST Special Publication 500-249, 2000a. 12, 20, 84, 85, 86

E. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing and Management*, 36(5):697–716, 2000b. 48

E. Voorhees. Evaluation by highly relevant documents. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–82, 2001a. 48

E. Voorhees. Overview of the TREC 2001 question answering track. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 42–51. NIST Special Publication 500-250, 2001b. 12, 86

E. Voorhees. The TREC question answering track. *Natural Language Engineering*, 7 (4):361–378, 2001c. 4, 71

E. Voorhees. Overview of the TREC 2002 question answering track. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 115–123. NIST, 2002. 12, 21, 40, 86

E. Voorhees. Evaluating the evaluation: A case study using the TREC 2002 question answering track. In *Proceedings of the 3rd Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2003)*, pages 260–267, 2003. 49

E. Voorhees and C. Buckley. The effect of topic set size on retrieval experiment error. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 316–323, 2002. 49

E. Voorhees and D. Harman. Overview of the sixth text retrieval conference (trec-6). In E. Voorhees and D. Harman, editors, *Proceedings of the Sixth Text REtrieval Conference (TREC 6)*, pages 1-24. NIST Special Publication 500-240, 1997. 79

E. Voorhees and D. Harman. Overview of the seventh text retrieval conference (TREC-7). In E. Voorhees and D. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, pages 1–23. NIST Special Publication 500-242, 1998. 48, 52

E. Voorhees and D. Harman. Overview of the eigth text retrieval conference (TREC-7). In E. Voorhees and D. Harman, editors, *Proceedings of the Eigth Text REtrieval Conference (TREC-7)*, pages 1-24. NIST Special Publication 500-246, 1999. 48

E. Voorhees and D. Tice. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 200–207, 2000a. 12, 84, 140

E. Voorhees and D. Tice. The TREC-8 question answering track evaluation. In E. Voorhees and D. Harman, editors, *Proceedings the Eighth Text REtrieval Conference (TREC-8)*, pages 83–105. NIST Special Publication 500-246, 2000b. 12

P. Vossen, editor. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, 1998. 121

Y. Wang and I. Witten. Induction of model trees for predicting continuous classes. In *Proceedings of the Poster Papers of the European Conference on Machine Learning (ECML)*, pages 128–137, 1997. 109

E. Wendlandt and J. Driscoll. Incorporating a semantic analysis into a document retrieval strategy. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 270–279, 1991. 34, 35

J. Wilbur. Non-parametric significance tests of retrieval performance comparisons. *Journal of Information Science*, 20(4):270–284, 1994. 54

R. Wilkinson, J. Zobel, and R. Sacks-Davis. Similarity measures for short queries. In D. Harman, editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 277–286. NIST Special Publication 500-236, 1995. 81

E. Williams. On the notions 'lexically related' and 'head of a word'. *Linguistic Inquiry*, 12:245–274, 1981. 97

I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999. 109

I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann Publishing, 2nd edition, 1999. 47, 75

W. Woods. Transition network grammars for natural language analysis. *Communications of the ACM*, 13(10):591–606, 1970. 28

W. Woods. Lunar rocks in natural English: Explorations in natural language question answering. In A. Zampoli, editor, *Linguistic Structures Processing*, pages 521–569. Elsevier North-Holland, 1977. 2, 28

H. Xu and H. Zhang. ICT experiments in TREC 11 QA main task. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 296–298. NIST Publication, 2002. 50

J. Xu and B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, 1996. 65, 90, 148, 151

J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. TREC 2002 qa at BBN: Answer selection and confidence estimation. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 290–295. NIST Publication, 2002. 32

H. Yang and T. Chua. The integration of lexical knowledge and external resources for question answering. In *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, pages 155–161. NIST Publication, 2002. 121

H. Yang, T. Chua, S. Wang, and C. Koh. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40, 2003. 121

D. Zhang and W. Lee. Question classification using support vector machines. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 26–32, 2003. 6, 151

J. Zobel. How reliable are the results of large-scale information retrieval experiments? In B. Crof, A. Moffat, C. van Rijsbergen, R. Wilkinson, and J. Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, 1998. 47

J. Zobel and A. Moffat. Exploring the similarity space. *ACM SIGIR Forum*, 32(1): 18–34, 1998. 8, 58

Index

Summary in Dutch

Informatie is één van de meest waardevolle goederen van de moderne maatschapij. Met de opkomst en brede verspreiding van de computer is het opslaan van enorme hoeveelheden gegevens zeer efficiënt en goedkoop geworden. We hebben nu ongekende hoevelheden informatie tot onze beschiking. Tegen deze achtergrond komt de vraag op hoe we toegang verkrijgen tot de informatie waarin we uiteindelijk geïnteresseerd zijn. Het vraagstuk van de ontwikkeling van methoden en programmatuur die ons, op een automatische manier, helpen bij het vinden van relevante informatie wordt in onderzocht in het onderzoeksgebied van de *information retrieval*. Gedurende de laatse decennia zijn zeer geavenceerde *document retrieval* systemen ontwikkeld. Eén van de onderzoekstakken binnen de information retrieval houdt zich bezig met vraag-antwoord systemen. Vraag-antwoord systemen maken het mogelijk dat een gebruiker een natuurlijke taal-vraag stelt, en niet zoals gebruikelijk is bij de meeste document retrieval systemen, zijn of haar informatiebehoefte formuleert met behulp van een lijst van sleutelwoorden. In het zeer recente verleden hebben vraag-antwoord systemen een ware renaissance beleefd die met name is toe te schrijven aan het gebruik van grote corpora.

Moderne vraag-antwoord systemen zijn sterk afhankelijk van document retrieval systemen als een middel om documenten te identificeren die met hoge waarschijnlijkheid een antwoord op een gegeven vraag bevatten. Dit proefschrift onderzoekt in hoeverre verschillende document retrieval benaderingen—zowel standaard als nieuwe—gebruikt kunnen worden in de context van vraag-antwoord systemen. Dit proefschrift vergelijkt verscheidene document retrieval benaderingen met het oog op hun vermogen om documenten te identificeren die inderdaad een correct antwoord bevatten. Daarnaast onderzoeken wij in hoeverre de kwaliteit van een bepaalde document retrieval benadering invloed heeft op de algehele kwaliteit van een specifiek vraag-antwoord systeem.

De uitkomsten van het onderzoek naar deze vragen zijn verschillend. Bijvoorbeeld, sommige standaard technieken waarvan we weten dat zij de kwaliteit van

een regulier document retrieval systeem verbeteren, hebben juist een negatief effect op de kwaliteit van een document retrieval system dat bedoeld is om documenten te vinden die een antwoord op een vraag bevatten. Aan de andere kant leiden sommige document retrieval technieken die nauwelijks succesvol zijn gebleken voor standaard document retrieval, zoals retrieval gebaseerd op de proximiteit van woorden, tot statistisch significante verbeteringen in de context van vraag-antwoord systemen.

Om het effect te kunnen meten dat verschillende retrieval technieken op een specifiek vraag-antwoord system hebben, hebben wij ons eigen vraag-antwoord systeem Tequesta gebruikt. Ook hier waren de uitkomsten van het onderzoek niet éénduidig. In één geval leidde een beter presterend retrieval systeem inderdaad tot betere resultaten van het gehele vraag-antwoord systeem, maar in een ander geval leidde het juist tot slechtere resultaten. De reden hiervoor ligt in de interactie tussen de verschillende componenten van een vraag-antwoord systeem, en de gevoeligheid van sommige componenten voor "ruis" geïntroduceerd door andere componenten. Desondanks kunnen wij op grond van dit onderzoek concluderen dat de kwaliteit van de document retrieval component van een vraag-antwoord systeem een duidelijk effect heeft op de kwaliteit van het gehele systeem.