Using Language Models in Question Answering

Dissertation

zur Erlangung des Grades

des Doktors der Ingenieurwissenschaften

der Naturwissenschaftlich-Technischen Fakultät II

-Physik und Mechatronik-

der Universität des Saarlandes

von

Andreas Merkel

Saarbrücken

2008

Tag des Kolloquiums:	14.07.2008
Dekan:	Univ.–Prof. Dr.rer.nat. A.Schütze
Mitglieder des Prüfungsausschusses:	
Vorsitzender:	Univ.–Prof. Dr.–Ing. Ch. Xu
Gutachter:	Univ.–Prof. Dr.rer.nat. D. Klakow
	Univ.–Prof. Dr.–Ing. G. Weikum (MPII)
Akad. Mitarbeiter:	Dr.–Ing. B. Martin

Für meine Frau Ina,

die mir tagtäglich die Kraft gegeben hat

diese Aufgabe zu bewältigen.

Acknowledgements

First of all I would like to thank Dietrich Klakow, who gave me the opportunity of doing a PhD thesis at his chair. I have learned very much from him and he gave me a splendid scientific supervision. I would also like to thank him for his inspiring ideas and conversations.

When we started research in the field of question answering at the Spoken Language System (LSV) chair, we were just a small number of people. Over the years, this group has steadily grown and we have done a lot of projects together. The discussions with many of the researchers in the LSV group have been very fruitful for my personal research. Here, I would like to thank in particular Dan Shen, Irene Cramer, Barbara Rauch, Jochen Leidner, Michael Wiegand, Stefan Kazalski, Christoph Lauer, Munawar Hussain, Yasir Iqbal, Andreas Beschorner, and Akira Kakinohana.

During the last years I have published some papers together with colleagues. I would like to thank Dietrich Klakow, Jochen Leidner, Michael Wiegand, Dan Shen, Irene Cramer, Stefan Schacht, Stefan Kazalski, and Michaela Hunsicker for interesting discussions and really good cooperation. I also want to mention Ute Stroh, Michael Wiegand, Jochen Leidner, and John McDonough for proof–reading the papers and also for fruitful discussions.

Furthermore I would like to thank the people that provided administrative support during my PhD studies. In particular I thank Karin Staab for proof–reading the thesis and Simon Hirth for general administrative support and mental encouragement.

I also want to thank my parents–in–law, Anita and Dieter Stroh, who accompanied me on my way.

Finally, I would like to acknowledge the financial support, that I received from the Federal Ministry of Education and Research (BMBF) for the SmartWeb project under the contract number 01 IMD01 M. Without BMBF's support the research for this thesis would not have been possible.

Short Summary in German

Mit dem Erscheinen des “World Wide Web” (WWW) in den 1990er Jahren versuchten Web–Suchmaschinen wie Altavista oder Yahoo Zugang zu dem schnell wachsenden Internet zu bieten. Da sie als öffentliche Dienste konzipiert waren, wurde die Anfragesprache so vereinfacht, dass die Mehrheit der Benutzer diese Suchmöglichkeiten auch nutzen konnte. Die meisten der aktuellen Suchmaschinen benutzen eine einfache stichwortbasierte Suche, um relevante Informationen wiederzufinden. Normalerweise enthalten diese gefundenen Dokumente dann lediglich ein oder mehrere Wörter der eigentlichen Suchanfrage. Daher kann man diese Art der Informationssuche nur als grobe Suche nach relevanten Textdokumenten bezeichnen. Wenn es das Ziel einer Suche ist, relevante Daten zu einem gegebenen Thema zu finden, so ist diese Suchmethode ausreichend. Allerdings suchen immer mehr Menschen spezifische Informationen zu einer Frage. Sie möchten eine genaue Antwort zu einer individuellen Suchanfrage, die sie in natürlicher Sprache gestellt haben und dabei nicht sämtliche Dokumente nach möglichen Antworten durchsuchen.

Dieser Problematik wird mit Hilfe eines standardisierten Fragebeantwortungssystems nachgegangen. Hierbei ist es möglich, eine präzise Antwort zu einer natürlich–sprachlichen Suchanfrage zu erhalten. Bis heute wurden unterschiedlichste Ansätze zur Lösung dieser komplexen Aufgabe verfolgt. Die meisten dieser Methoden nutzen tiefe linguistische Analysen, um die Bedeutung der Anfrage zu “verstehen” und mögliche Antwort–Kandidaten oder relevante Textabschnitte innerhalb sehr großer Textsammlungen, wie z.B. dem Internet, zu finden.

In dieser Arbeit wird ein sprachmodellbasierter Ansatz für verschiedene Bestandteile eines kompletten Fragebeantwortungssystems beschrieben. Dies beinhaltet die Verarbeitung der natürlichsprachlichen Suchanfrage sowie die Suche nach relevanten Dokumenten, Textabschnitten und Sätzen.

Das Modul der Frageverarbeitung versucht die natürlichsprachliche Nutzeranfrage zu “verstehen”. Dies ist notwendig, um weitere Entscheidungen im Verlauf des Gesamtsystems, z.B. bei der Wahl der möglichen Antworten, zu treffen. In dieser Arbeit wird daher zur Analyse der Frage ein statistischer Ansatz mit Sprachmodellen beschrieben. Der Hauptvorteil hierbei liegt, bezogen auf andere linguistische Methoden, im deutlichen Zeitgewinn.

Die Dokumentsuche dient dabei als eine Art Filter, indem die Anzahl der Dokumente, die

von den nachfolgenden Modulen bearbeitet werden müssen, reduziert wird. Obwohl für die Standarddokumentsuche traditionelle Algorithmen, wie *tf–idf* oder *Okapi*, optimal funktionieren, zeigt die vorliegende Arbeit, dass sprachmodellbasierte Methoden besser für die vielfältigen Aufgaben von Fragebeantwortungssystemen geeignet sind.

Eine ähnliche Aufgabe wie die Dokumentsuche übernimmt die “Suche nach Textabschnitten”. Hier werden alle relevanten Dokumente in kleinere Sinnabschnitte aufgespalten. Danach werden diese Abschnitte neu sortiert, um bestmögliche Resultate zu erhalten. Ein Sonderfall dieser Methode ist die so genannte “Satz–Suche”, bei der als Textabschnitt lediglich ein Satz benutzt wird. Auch auf diesen Teil eines Fragebeantwortungssystems wird in dieser Arbeit eingegangen.

Die Ergebnisse der Arbeit zeigen, dass sprachmodellbasierte Methoden in einem eigens implementierten Fragebeantwortungssystem genauso gut oder sogar noch besser funktionieren, als derzeitige, moderne Systeme. Ein wesentlicher Vorteil des beschriebenen Systems liegt in der Nutzung schneller, statistischer Algorithmen gegenüber den vergleichsweise langsamen, tiefen linguistischen Analysen anderer Ansätze.

Summary in German

In den letzten Jahrzehnten wuchs die Notwendigkeit wichtige Informationen wiederzufinden stetig an. Mit dem Bekanntwerden des Internets in den 1970er Jahren wurden zudem immer mehr Informationen von einer immer größer werdenden Bevölkerungsgruppe benutzt. Heutzutage existiert eine fast endlos erscheinende Anzahl von nutzbaren Informationen im Internet. Zusätzlich besitzen viele Firmen noch eigene Intranets mit z.T. wichtigen, firmeninternen Informationen.

In Anbetracht der Menge an Informationen haben die Menschen sehr schnell realisiert, dass sie Hilfe brauchen, um spezifische Daten wiederzufinden. Zu diesem Zweck wurden die ersten Suchmaschinen entwickelt. Allerdings waren diese Suchmaschinen sehr schwierig zu handhaben, da sie eine spezifische und komplexe Anfragesprache besaßen. Es gab nur wenige Personen, die wussten, wie diese Programme zu benutzen waren. So konnten nur diese so genannten “Information Broker” bei der Suche nach Informationen behilflich sein.

Der nächste logische Schritt kam mit dem Erscheinen des “World Wide Web” (WWW) in den 1990er Jahren. Web–Suchmaschinen wie Altavista oder Yahoo versuchten Zugang zu dem schnell wachsenden Internet zu bieten. Da sie jedoch als öffentliche Dienste konzipiert waren, wurde auch die Anfragesprache vereinfacht, so dass die Mehrheit der Benutzer diese Möglichkeiten auch nutzen konnte. Die meisten der aktuellen Suchmaschinen benutzen eine einfache stichwortbasierte Suche, um relevante Informationen wiederzufinden. Normalerweise enthalten diese gefundenen Dokumente dann lediglich ein oder mehrere Wörter der eigentlichen Suchanfrage. Daher kann man diese Art der Informationssuche nur als grobe Suche nach relevanten Textdokumenten bezeichnen. Wenn es das Ziel einer Suche ist, relevante Daten zu einem gegebenen Thema zu finden, so ist diese Suchmethode ausreichend. Allerdings suchen immer mehr Menschen spezifische Informationen zu einer Frage. Sie möchten eine genaue Antwort zu einer individuellen Suchanfrage, die sie in natürlicher Sprache gestellt haben und dabei nicht sämtliche Dokumente nach möglichen Antworten durchsuchen.

Dieser Problematik wird mit Hilfe eines standardisierten Fragebeantwortungssystems nachgegangen. Hierbei ist es möglich, eine präzise Antwort zu einer natürlichsprachlichen Suchanfrage zu erhalten. Bis heute wurden unterschiedlichste Ansätze zur Lösung dieser komplexen Aufgabe verfolgt. Die meisten dieser Methoden benutzen tiefe linguistische Analysen, um die Bedeutung der Anfrage zu “verstehen” und mögliche Antworten oder relevante

Textabschnitte innerhalb sehr großer Textsammlungen, wie z.B. dem Internet, zu finden.

In dieser Arbeit wird daher zur Anayse der Frage ein statistischer Ansatz mit Sprachmodellen beschrieben. Dies beinhaltet die Verarbeitung der natürlichsprachlichen Suchanfrage sowie die Suche nach relevanten Dokumenten, Textabschnitten und Sätzen.

Das Modul der Frageverarbeitung versucht die natürlichsprachliche Nutzeranfrage zu "verstehen". Dies ist notwendig, um weitere Entscheidungen im Verlauf des Gesamtsystems zu treffen.
Zu diesem Zweck wird der genaue Typ der Anfrage bestimmt, z.B. *Person* bei der Frage *Wer gründete die Guiness Brauerei?* Abhängig von der benutzten Klassifizierung sind auch andere Typen, wie *Ort, Beschreibung* oder *Entitäten* möglich. Dieses Modul ist sehr wichtig für das komplette System, da der Fragetyp auch in vielen anderen Bereichen eines Fragebeantwortungssystems benutzt wird. Z.B. korrespondiert er oft mit dem erwarteten Antworttyp und wird daher auch benutzt, um mögliche Antwortkandidaten zu bestimmen. In dieser Arbeit wird daher ein statistischer Ansatz mit Realisierung durch Sprachmodelle beschrieben, um die Benutzeranfrage genau zu anaysieren. Der Hauptvorteil hierbei liegt, bezogen auf andere linguistische Methoden, im deutlichen Zeitgewinn.

Die Dokumentsuche dient dabei als eine Art Filter, in dem die Anzahl der Dokumente, die von den nachfolgenden Modulen bearbeitet werden müssen, reduziert wird. Durch diese Einschränkung des Suchraums können die nachfolgenden, z.T. linguistischen, Algorithmen robuster und schneller arbeiten. Obwohl für die Standard–Dokumentsuche traditionelle Algorithmen, wie *tf–idf* oder *Okapi* optimal funktionieren, zeigt die vorliegende Arbeit, dass sprachmodellbasierte Methoden besser für die vielfältigen Aufgaben von Fragebeantwortungssystemen geeignet sind.

Nun kann es sein, dass die gefundenen Dokumente immer noch zu groß sind, oder mehrere Themen behandeln. Deshalb werden alle relevanten Dokumente in kleinere Sinnabschnitte aufgespalten. Danach werden diese Abschnitte neu sortiert, um bestmögliche Resultate zu erhalten. Diese Art der Dokumentverarbeitung ist derzeit Standard in Fragebeantwortungssystemen. In der vorliegenden Arbeit wird zugleich ein sprachmodellbasierter Ansatz zur Neusortierung der Textabschnitte beschrieben.

Ein Spezialfall dieses Ansatzes ist die so genannte "Satz–Suche", bei der als Textabschnitt lediglich ein Satz benutzt wird. Auch auf diesen Teil eines Fragebeantwortungssystems wird in dieser Arbeit eingegangen.

Die Ergebnisse der Arbeit zeigen, dass sprachmodellbasierte Methoden in einem eigens implementierten Fragebeantwortungssystem genauso gut oder sogar noch besser funktionieren, als derzeitige, moderne Systeme. Ein wesentlicher Vorteil des beschriebenen Systems liegt in der Nutzung schneller, statistischer Verfahren gegenüber den vergleichsweise langsamen, tiefen linguistischen Analysen anderer Ansätze.

Abstract

In the last decades, the need for finding information grew continuously. With the upcoming of the Internet in the seventies, more and more information was accessed by more and more people. Today, the amount of accessible information on the Internet seems to be endless and numerous companies have additional intranets with additional secret information.

In consideration of the amount of information, people realized very early that they will need help in finding specific data. As a result, the first search engines were developed. But those retrieval engines were very difficult to handle, because of their specific and complex query language. There were just few people who knew how to use such systems, so, merely the so–called *information brokers* were able to help if there was an information need.

The next, logical step came with the upcoming of the *World Wide Web – WWW* in the nineties. Web search engines like Altavista or Yahoo tried to give access to the quickly expanding Internet. But because these are public services, the query languages are much easier to use than the earlier search engines. Most of today's information retrieval systems simply use a keyword–based search to find relevant documents. Normally, these documents just contain one or more words given by the user's query. Therefore it is a simplistic search for relevant text documents. If the goal of a search was to find data for a given topic, this might be sufficient. But more and more people have a specific information need. They want to have an exact answer to a given natural language question, so, they do not want to look at possible document candidates to find the answer.

This is what is done in a standard Question Answering (QA) system. It is possible to find a precise answer string to a given natural language question. Today, there are many approaches to this very complex task. Most of them use deep linguistic methods to understand the meaning of questions and possible answer candidates or to find relevant text snippets in a very large collection like the Internet.

In this thesis, we describe a language model approach to parts of a complete QA system. It includes the processing of the natural language query as well as the retrieval of relevant documents, passages and sentences.

In the query processing module the system tries to understand the user's natural language questions. This is necessary to make decisions in later parts of the system. The document retrieval module acts as a filter. It reduces the amount of documents that the following components have to handle. Similarly, in the passage retrieval step, all relevant documents

from the previous component are split up into text passages. Then, a re–ranking is done to find the best matching snippets. A special case of the passage retrieval module is selecting just one sentence as text passage. The thesis also covers a statistical approach to this part of QA system.
The results show that the language model based modules in our QA system perform equally well or even better than current state–of–the–art systems. Due to the heavy use of fast statistical algorithms the main advantage of our system is an efficiency gain compared to the slower deep analysis linguistic methods used in other approaches. A second benefit of using language models is the ability to train them for new languages. So, such a QA system is also very flexible for using in a multi–lingual environment.

Contents

List of Figures

List of Tables

Chapter I

Introduction

In this chapter, we want to introduce the topic and present a motivation of the thesis. In particular, we discuss the task of information retrieval and how to extend such systems to provide users with more specific pieces of information. This will directly result in a description of question answering systems. After an overview of the thesis, our contributions in the field of language model based question answering are given.

1 Motivation

The idea of searching and finding relevant information is as old as storing them. For thousands of years people have already written down their knowledge and other people have tried to find them.

With the introduction of the Internet this these problems have got worse, because nearly everyone is now able to provide any kind of data to the rest of the world.

There is a growing amount of information today. People have a nearly endless repertoire of sources they can use to satisfy their information need. In particular, there are lots of newswire pages, like *The New York Times*[1], *Reuters*[2], or *BBC*[3], online encyclopedias, like *Wikipedia*[4], and also other unstructured sources of information, like personal Web pages.

“To 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”

[1]http://www.nytimes.com
[2]http://www.reuters.com
[3]http://www.bbc.co.uk
[4]http://www.wikipedia.org

(Monz 2003).

That is why *Information Retrieval* (IR) systems become an important part of our daily life. Worldwide, millions of people use Internet based search engines like *Google*[5] or *Altavista*[6] to find various information like news, pictures, music or other text documents. Satisfying this information need has become a real challenge, for developers and users of such systems. Users normally want to receive some relevant data out of millions of documents and developers have to implement algorithms which fulfill this need with an additional time constraint of just some few seconds per query.

In today's Internet IR systems, users can submit a set of keywords, which represent their information need. These *queries* are then processed by the system and a sorted list of relevant documents is returned. Hence, such systems are also referred to as *Document Retrieval* systems.

For example, if a user wants to know the date of birth of *John Lennon*, he/she probably searches the Web using *John, Lennon*, and *Birthday* as keyword query. Afterwards, the resulting documents have to be manually scanned to find the desired answer.

For such queries, the use of online encyclopedias is much more simple. By submitting the query about *John Lennon*, a system like *Wikipedia* provides a biography about the person. So, the date of birth and other interesting information can be found by reading just the one document.

But, it would make much more sense, if it was possible to directly ask a system *When was John Lennon born?* In this case, the answer of the system to this natural language question should not be a set of documents but the concise answer *October 9, 1940*.

Such systems have been developed since the 1950s and are called *Question Answering* (Q&A) systems, because they try to provide the user with a concise answer to his specific information need.

2 From Information Retrieval to Question Answering

The question answering systems which have been developed since 1950 are different to the Q&A system we describe in this thesis. The function of these early engines are mainly restricted to specific domains, like Baseball or moon rocks. The underlying data structures

[5]`http://www.google.com`
[6]`http://www.altavista.com`

were mostly database driven and therefore provided the system with structured data to process.

With the beginning of the annual Text REtrieval Conference (TREC) in 1990 and in particular the start of the question answering track in 1999 (Voorhees and Harman 1999), the task of Q&A systems switched from closed–domain to open–domain systems. This means, that those systems can provide a user with a concise answer to a natural language question by analyzing unstructured data, like text documents, newswire articles or the Internet–based Web documents.

As already mentioned, standard IR engines use document retrieval algorithms to satisfy a user's information need. For the task of question answering this is not sufficient, although it is a necessary step to find the correct answer to a given natural language question.

Normally, the first step of a Q&A system is to analyze the natural language question. In this part, the specific type of a question is determined. For example, if the question is *When was John Lennon born?*, the corresponding question type is *Date*. But questions about other classes, like *Person*, *Entity* or *Definition* are also possible. This information can be used in various other modules of the complete Q&A system.

The following steps try to extract the accurate answer out of the given corpus, for example the complete Internet. Because it is not possible to process each document in this corpus, a filtering, or pre–fetching step (Monz 2003) has to be performed. Normally, this is done using standard document retrieval approaches. So, the idea is to reduce the search space in which a correct answer has to be found. It is necessary to reduce the search space because the following components may use long–lasting deep analysis algorithms which strongly depend on the size of the processed corpus. Therefore, it is important to process just the documents which seem relevant to a query to get answers within an appropriate period of time.

But the document collection might still be too large, or some documents contain a variety of different topics. If this is the case, again, the following components, like *Sentence Retrieval* or *Answer Selection*, will have to analyze more text than is necessary in order to find the correct answer. To overcome this problem, it is essential to further reduce the size of the collection. This can be done by splitting up the text segments into smaller passages.
After dividing the documents, a second retrieval step is necessary in order to re–rank the new collection. By doing so, the corpus size and thus the search space is reduced again. If those pieces of text just contain one sentence, this step is also called *Sentence Retrieval*.

Finally, relevant pieces of information are selected out of the text segments. This is done by analyzing the passages using part of speech tagging, named entity tagging, and other linguistic algorithms.
After re–ranking those answer candidates the best possible answer is returned to the user.

3 Standard Tasks in Question Answering

Now, as we know how open–domain question answering is performed, we want to talk about the various tasks for such systems.
Primarily, the task of today's Q&A systems is to provide a user with a concise answer to a natural language question. This can be done by using text documents, like newswire articles, as corpus, but also the Internet might be an adequate collection for answering a non–specific question.

In 1999, this "main" task became part of the Text REtrieval Conference (TREC) as a textual question answering task (Voorhees and Harman 1999). "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 if question answering techniques and the rapid interchange of ideas to further the research in that area" (Monz 2003).
The weakness of Q&A at TREC is that it mainly focuses on English systems where both, questions and answers, have to be in English.

This disadvantage was compensated by the *Cross Language Evaluation Forum* (CLEF) where both, monolingual in different languages and special multilingual Q&A tasks were introduced in 2003.
The CLEF Web site describes the task as follows (CLEF): "The NIST TREC Q&A tracks (this year is the seventh round) have stimulated progress in Q&A state–of–the–art, establishing widely accepted and standardized evaluation measures and requirements. Nevertheless, multilinguality has always been outside of the scope of the TREC Q&A evaluation exercises, which, up to now, have focused on systems for English.
Within the framework of the Cross Language Evaluation Forum (CLEF), a pilot track for non–English monolingual and cross–language QA systems was successfully set up for the first time in Europe in 2003. Since then, three other campaigns have been carried out, recording a costant increase in the number of participants and in the results achieved."

Provided languages for this year's CLEF Q&A task are Bulgarian, German, Spanish, French, Italian, Dutch, Portuguese, and Romanian.

Another task which is based on question answering is the search for geographic information. For this purpose, there is a special task at CLEF which is called *Cross–Language Geographical Information Retrieval* (GeoCLEF). It is described as follows:

"Geographical Information Retrieval (GIR) concerns the retrieval of information involving some kind of spatial awareness. Given that many documents contain some kind of spatial reference, there are examples where geographical references (georeferences) may be important for IR. For example, to retrieve, re–rank and visualize search results based on a spatial dimension (e.g. "find me news stories about riots near Dublin City"). In addition to this, many documents contain geo–references expressed in multiple languages which may or may not be the same as the query language. This would require an additional translation step to enable successful retrieval.

Existing evaluation campaigns such as TREC and CLEF do not explicitly evaluate geographical IR relevance. The aim of GeoCLEF is to provide the necessary framework in which to evaluate GIR systems for search tasks involving both spatial and multilingual aspects." (GeoClef)

4 Outline of the Thesis

Chapter 2 describes the background on question answering systems. Here, a short history of Q&A is given which also includes some of the earlier, closed–domain approaches. We further describe approaches to how to solve the task of Q&A using state–of–the–art techniques, like natural language processing algorithms or external knowledge bases. The chapter also contains a description of a standard open–domain question answering system as well as evaluation methods of measuring the success of such a system.

Chapter 3 In this chapter, we provide background information about statistical language modeling. This includes general information on language modeling, how to perform smoothing and other important applications like Maximum Entropy models. Then, the task for language model based information retrieval is described. Finally, we propose a theoretical model of taking the notion of language modeling to the task of question answering.

Chapter 4 This chapter describes an application of language model based question answering we used for the SmartWeb project. In particular, we introduce our approach to performing passage retrieval by using statistical language models.

Chapter 5 presents related work in the field of applications for statistical language modeling and question answering.

Chapter 6 Our studies in the area of language model based question classification are introduced in this chapter. We present some standard and improved models which include count based absolute discounting or log–linear interpolation using bigram statistics. In the result section we show that this approach even performs better than using SVM. We also introduce in this chapter the notion of confidence measures for question types. These values can be used in later modules for increasing sentence retrieval performance or for the task of answer extraction.

Chapter 7 We explain our experiments on the task of document retrieval using language models in this chapter. It introduces the different steps we used to improve the results of this task. This includes fixing the number of returned documents as well as tuning the smoothing parameters of the language models.

Chapter 8 This chapter presents results for passage retrieval and re–ranking using a language model based approach. While using the same methodology as described for document retrieval in order to increase the passage retrieval performance, we experimented with different background collections to improve the re–ranking of text fragments.

Chapter 9 Sentence retrieval is a special case of passage retrieval where just one sentence is included in the passage. In this chapter, we introduce some optimization steps, like a weighted expansion of queries and sentences, based on Baysian smoothing using Dirichlet priors.

Chapter 10 Finally, we want to summarize the results of our experiments and give some hints for further research work in the field of language model based question answering.

5 Contributions

Finally, we want to give a short overview over our contributions in the field of using language models in question answering.
As mentioned so far, parts of a complete Q&A system were implemented using statistical language models. In particular, this includes:

- Language model based question classification. For this task, some improved language models were developed and the notion of confidence measures were introduced.
- By using language models for the task of document retrieval, we also could improve the effectiveness of a Q&A system.
- The same approach was done to improve passage retrieval. In addition, we used different background collections to improve the re–ranking of the text passages.
- A further focus was the improvement of the sentence retrieval module in a Q&A system. Here, we used some specific optimization steps to obtain a better set of results for this task.

Chapter II

Background on Question Answering

The chapter *Background on Question Answering* covers the main features of question answering (Q&A) systems. It presents a short history of such systems and shows some possible methods of solving this task. Furthermore, we describe a complete open–domain question answering system on the basis of a standard model.

The rest of the chapter is organized as follows. Section II.1 describes existing systems for both, closed and open–domain Q&A. In section II.2 we provide some possible strategies for approaching the task of finding answers. The section II.3 gives a complete overview of a state–of–the–art Q&A system by means of the most important modules of such systems. Here, we will also discuss variants of other state–of–the–art systems, most of them used for the Text REtrieval Conference. The last section(II.4) covers current evaluation metrics for both, standard retrieval methods and Q&A systems. It also introduces the Text REtrieval Conference (TREC) organized by the National Institute of Standards and Technology (NIST) as a benchmark for text retrieval systems and in particular for the task of Q&A.

1 A Short History of Question Answering Systems

As mentioned in chapter I.3, classic Q&A systems can be distinguished by their ability in answering questions for just a specific domain (closed–domain) or domain independent (open–domain). Former systems were the first ones which worked in a productive manner because they just had to rely on databases or knowledge bases.

One of the earliest domain–specific Q&A systems is *BASEBALL* (Green et al. 1963). In this database–driven approach, a user can ask specific questions about locations, dates and results of baseball matches. The *LUNAR* system (Woods 1977) also tried to answer natural language questions for the domain of moon rocks and soil of the Apollo 11 mission (1969). To help high–school people to solve problems in the area of algebraic exercises, the *STUDENT* (Winograd 1977) system was established. Another database–oriented approach was *PHLIQA* (Bronnenberg et al. 1980, Scha 1983). In this system, users could ask short queries about fictitious information concerning European computer systems and companies using them.

At that time, the first systems began to use methods from the area of artificial intelligence. For example *QUALM* (Lehnert 1978, Lehnert 1994), another natural language processing approach, made use of the so called *conceptual dependency approach* and *scripts to represent expected behavior in a situation*. It's main effort was to answer questions in the context of story understanding.

All the systems mentioned so far use structured data in the form of hand–coded knowledge bases or databases. The *START* system (Katz 1997) uses just a kind of knowledge base to answer simple natural language questions. But it makes use of the unstructured information from the Internet instead of structured data. That follows the general configuration of the system. In a first step, Web documents are used to fill up the knowledge base, whereas in a second step, sentences are generated from this knowledge base.

The *FAQFINDER* (Burke et al. 1997) uses various methods from information retrieval and natural language processing to "provide answers to user questions through the retrieval of previously asked questions residing in the Internet Frequently Asked Questions (FAQ) files, primarily USENET FAQ file" (Mlynarczyk and Lytinen 2005). Therefore, the system calculates the similarity of a user query with the queries in the FAQ. If there is a similar query, the *FAQFINDER* will return the adequate answer. To calculate a similarity score, it uses vector–space inspired model with word frequencies. Other linguistic methods like parsing or part–of–speech tagging have been used as well.

The disadvantage of all Q&A systems described so far is the need for a specific domain. Normally, this also includes the need for hand–crafted knowledge bases or databases. This is not only expensive, but makes it also nearly impossible to use those systems in another application than the original. All these are reasons why those systems have never played a decisive role for commercial systems in that area of research. So, the logical conclusion was to build up more flexible Q&A systems which do not rely on specific domain knowledge.

As already mentioned, an advantage of such open–domain Q&A systems is that there is no need for knowledge bases. But this means that they have to find their information within other sources, in this case using plain–text documents. These document collections can contain newspaper articles, books, stories, encyclopedias as well as web documents, such as forums or web logs (*Blogs*). That is why such systems are also called text–based Q&A systems (Monz 2003). Users can ask natural language questions and the system will search the answer within the given text collection. To do so, the system has to analyze the question, then selects documents which are in some sense similar to the question and tries to extract the specific information out of the text. This means that a Q&A system contains elements from various research fields, like natural language processing, information retrieval and information extraction.

2 Approaches to the Question Answering Task

In this section we describe some possible techniques which are useful when implementing an open–domain Q&A system. This includes the use of natural language processing in combination with information retrieval and extraction as well as some explanation about external components like part of speech tagger or WordNet.
The main goal of a Q&A system is the use of natural language questions to provide a user with a specific piece of information. To do so, the system has to "understand" the semantics of a given question. In most systems, this is done with the help of natural language processing (NLP) techniques. These methods can include deep linguistic techniques like dependency representations build from parse trees (Paşca 2003) or simple statistical language models to classify the question into a taxonomy, like we did in this thesis.

Information Retrieval

Another interest of research in Q&A is the use of information retrieval methods. When the user question has been analyzed and a system query has been build, relevant documents should be searched out of an existing text collection. To find those text documents, the similarity between the query and a document is calculated[1].

[1]This chapter just covers the basic notions of the classic IR models. Please read (Baeza–Yates and Ribero–Neto 1999) for more details.

This can be done in various ways. But most of them use keyword–based queries instead of natural language questions. And normally, those methods also assume that the terms within the query are independent. A representative for those “classic” information retrieval approaches is the standard and extended *Boolean retrieval* (van Rijsbergen 1979, Salton and McGill 1983), which simply uses set theory and Boolean algebra to rank documents according to a query. The main disadvantage of this model is that it is based on a binary decision criterion, which just can predict if a document is relevant or not. A second flaw is that it is not so easy to “translate” a user’s information need to the specific semantic of that model (Baeza–Yates and Ribero–Neto 1999). Therefore, it is not that easy for users to express their needs within the Boolean algebra and so they just use some simpler expressions which directly leads to worse results.

A second, very important information retrieval approach is the *Vector Space Model* (Salton, Yang, and Yu 1975). In this approach, each term in a query and a document is assigned with a non–binary weight (term weight). Each value for query and document can be regarded as a vector and so, a degree of similarity can be calculated. This is done for each document to get a ranking of the collection.
Normally, the similarity between the query and a document is calculated by the *cosine similarity*, which computes the cosine of the angle between the two vectors.

The next step is to find an adequate method for calculating the weight of the index terms. There are many possible approaches (Salton and McGill 1983), but *tf–idf* is the most popular method. It combines the term frequency (tf) with the inverse document frequency (idf).
The *tf factor* is calculated as the raw frequency of a term in a given document. Therefore, it is a measure of how well the term describes the document (Baeza–Yates and Ribero–Neto 1999). This is referred to *intra–clustering similarity*.
On the other hand, the *idf factor (inverse document frequency)* calculates the inverse raw frequency of a term in the complete document collection. This factor models the fact, that a term which very often occurs among the document collection cannot help to distinguish between a relevant or non–relevant document. This part is referred to as *inter–cluster similarity* (Baeza–Yates and Ribero–Neto 1999).

Another “classical” model, which is in some sense similar to the approach we used in this thesis (Lavrenko and Croft 2003), is the *Probabilistic Information Retrieval Model* (Robertson 1977, Robertson and Walker 1994). It also tries to use a probabilistic approach

to find a set of relevant documents to a user query.
The idea of this model is as follows (Baeza–Yates and Ribero–Neto 1999, Manning, Raghavan, and Schütze 2007). First, the *information need* (or query) and the document have to be translated into adequate representations. Then, based on those models, the system computes how well the document representation satisfies the information need. More specific, the model tries to calculate the probability that a document is of *relevance* for the user.
One fundamental assumption is that there is an initial set of documents which are relevant. So, if it is possible to specify the properties of this *ideal* set, it should be easy to find relevant documents. But, the problem is, that these properties of relevance are not known. There should be an initial guess followed by a user interaction (*user feedback*) to refine the properties of the set of relevant documents.

Other, more sophisticated models, like the *Okapi BM25* (Robertson et al. 1994) or the *Latent Semantic Indexing* approach (Deerwester et al. 1990, Berry, Dumais, and O'Brien 1995) mostly base on the "classical" models described above and are not further covered in this thesis.

Web pages are in some sense a little bit more special than "normal" text documents. On those pages, it is possible to link to other pages which are of interest for the author of the page. On the basis of these *Hyperlinks*, there is a variety of possible approaches to rank web pages. The two methods best–known are the *HITS* algorithm (Kleinberg 1998) and PageRank (Page et al. 1998, Brin and Page 1998).

The HITS (Hypertext Induced Topic Search) algorithm bases on *authorities*, pages with many links pointing to it, and *hubs*, pages with a high number of outgoing links. The idea is now to find even better hubs and authorities. Better authorities will be found, if they have good hubs as predecessors whereas better hubs are specified with good authorities as successors. A good explanation of the iterative implementation of the HITS algorithm can be seen in Weikum (2005). A major drawback of this approach is that it needs an initial set of "root pages", which have to be assigned via relevance ranking.

The PageRank algorithm, which is a part of the searching strategies used by Google, also includes the structure of the Web to find relevant pages. The key idea behind this approach is that a random user starts a random walk on the Web. It follows outgoing hyperlinks with a given probability and never takes an already used link backwards. He can also randomly jump (or teleport) to another web page with another probability. During this random walk,

the user visits some pages more often than others. Obviously, these pages have a higher authority and better hubs than others.
The probability of being on a node during the walk can also be computed with the help of Markov chains (Manning, Raghavan, and Schütze 2007).

NLP tools for Information Extraction

The ultimate goal of research on Natural Language Processing is to parse and understand language (Manning and Schütze 1999). This goal also holds for the task of Q&A, where it is essential to understand user questions as well as the given answers. The NLP community provides us with a variety of useful tools to reach this target, like *Part–of–Speech tagger* or *Named Entity Recognition* systems.

Part–of–Speech Tagger

The field on application of Part–of–Speech (POS) tagger, or simply *tagger*, is to assign an adequate part of speech to each word in a text. This includes proper nouns, adjectives, verbs, adverbs and so on. There is a variety of possible tags, but the most widely used tag set in NLP is the *Penn Treebank tag set*[2] (Marcus, Santorini, and Marcinkiewicz 1993).
Regarding a regular Q&A system, there are many modules which can make use of a POS tagger, like question processing or answer extraction. For example, if a user searches for facts, normally the answer is a noun. Thus, a sentence representation like
John–NNP Lennon–NNP was–VBD a–DT member–NN of–IN the–DT Beatles–NNP
may help to extract the answer to the question “John Lennon was a member of which group?” because the only proper noun which does not occur in the question is the word “Beatles”.
For experimenting with the *Alyssa* system we used Brill’s tagger (Brill 1992, Brill 1994).

Named Entity Recognition

In general, the task of recognizing named entities (NE), like persons, locations or organizations in a text is just a classification task. Typical classes for given words are names of persons or organizations, locations, dates, monetary amounts or percentages. For example, if the question asks for the word “Masouleh”, then the NE tagged sentence

[2]This set is also used in the *Alyssa* Q&A system, which is partly described in this thesis.

*Some 30 people have been killed and 45 others been injured in floods and landslide caused by torrential rains in the historical city of ⟨ENAMEX id=1 type="**LOCATION**"⟩ **Masouleh** ⟨/ENAMEX⟩*

can give the answer that "Masouleh" is a location.

There is an approach which uses statistical, learned methods to tag entities in text (Bikel et al. 1997). It is a flexible approach, easy to implement, which is mainly based on hidden Markov models.

Parser

The idea of parsing (or syntactic analysis) is to be able to generate a syntactic, phrase data structure (parse tree) out of a given sentence (Manning and Schütze 1999). Most commonly used parsers are statistically motivated and use manually annotated training data to get count statistics. There are also parsers which use the output of Part–of–Speech taggers to build up their own parse tree.

The information gained from those data structures can be useful for example to map the user question to an answer (Shen, Kruijff, and Klakow 2005).

External Resources

In this section, we want to present an external knowledge resource which is helpful for the task of NLP in general or for Q&A in special. There are many toolkits available, but in this section we exemplarily want to introduce *WordNet*[3]. Normally, these packages are freely available and represent semantical connections between word entities.

WordNet

WordNet is a dictionary for English language. It links English nouns, verbs, adjectives, and adverbs to set of synonyms that are in turn linked through semantic relations that determine word definitions (Miller 1995). *WordNet* is not a traditional lexical database but it was designed for computational use. Each word in the collection is linked to other words with the same meaning (set of synonyms or *synset*). There is also a general definition for each *synset*. The *semantic relations* between words or word senses are for example synonyms,

[3]`http://wordnet.princeton.edu/`

antonyms, hyponyms, and much more.

WordNet also covers the problem of word ambiguity. For example, the synset *car* has five different meanings, and is therefore a member of five different synsets. It can be: *car, auto, automobile, machine, motorcar (a motor vehicle with four wheels; usually propelled by an internal combustion engine)*. But also the synset *car, elevator car (where passengers ride up and down)* is valid for the word *car*.

Totally, *WordNet 3.0* contains 117,659 synsets and a total number of 147,278 unique noun, verb, adjective, and adverb strings.

3 An Open Domain Question Answering System

In today's world, the effective finding of information becomes a crucial part of our lives. But the searching and finding of relevant information is not that easy. Most information, we are looking for, can theoretically be found on the Internet. If a user has a specific information need, he visits a common search engine and *translates* his need into a keyword query, which can be processed by such Internet search systems. As results, he will get a list with potentially relevant documents and perhaps some text snippets from the corresponding web page. Then, the user can look at those documents and can search the needed information by hand.

The task of an open–domain question answering (Q&A) system is to provide a user's information need, normally expressed by a natural language question, with a short answer. Whereas today's search engines return lists of documents containing the answer as described, a Q&A system returns just the fact a user is interested in. So, most of the queries a Q&A system has to answer are *factoid* questions. Those factoid questions are fact–based and can be answered with a simple statement. An example for a question of that type is *In what year was Susan Butcher born?*, which can simply be answered with the year *1954*. But, in an *open–domain* Q&A system, questions to all possible topics can be asked. This also includes questions for persons, locations, organizations, and so on. This topic will be picked up in the next section, when describing how to classify a user question in order to better find correct answers.

Other possible questions for a Q&A system are *list* and *definition* questions. A list question expects a list of different instances of a factoid question as returning values, like *List names*

of characters in ”Harry Potter and the Goblet of Fire”, which asks for a list of person names in the book “Harry Potter and the Goblet of Fire”.
On the other hand, a definition question asks for more detailed information about persons or things. In this case, the answer is not a short fact, but a longer text snippet which gives more detailed information about the demanded facts. Examples for this kind of questions are *Who is Angela Merkel?* or *What is a Question Answering System?*.

The question classes we named as possible types for Q&A systems are also the types which are used at the Text REtrieval Conference (TREC) to evaluate and compare different Q&A approaches from different research groups all over the world. At TREC 2006, 27 institutes attended the competition with their systems. So it is easy to see that it is not possible to represent all different approaches in a system overview.

Therefore, a more general architecture of a textual open–domain question answering system is presented in Figure II.1. It contains all modules, which all state–of–the–art open–domain question answering system have in common. Now a short description of all modules follows. Longer descriptions with examples and citations can be read in the following sections.

The system begins on top with the *information need* a user has in mind. This “need” has to be translated into a *natural language (NL) question*. The style of the NL–question is exactly the same as when asking other persons. So, the user doesn’t have to build a complicated keyword–based query to search for information.
In a next step, the question is processed and a query is build. The processing step includes the definition of the *question type* as well as other linguistic processings, like parsing, whereas the *query construction* mainly builds a query which is best for the following information retrieval modules.

After a query is constructed, the information retrieval part has to find relevant documents out of a possibly large corpus. This might be a text collection of newswire data or even the Internet. In this step, the size of the corpus is reduced to make it easier for following modules to find an answer.
But because most documents are still very large or cover more than one topic, they are split up to smaller text fragments and are re–ranked according to the query (*passage retrieval*).
Normally, these passages just contain a few sentences or at an extreme case, just one sentence. In that case, this step is called *sentence retrieval*. This procedure ensures, that the following modules simply get the amount of text they can process.

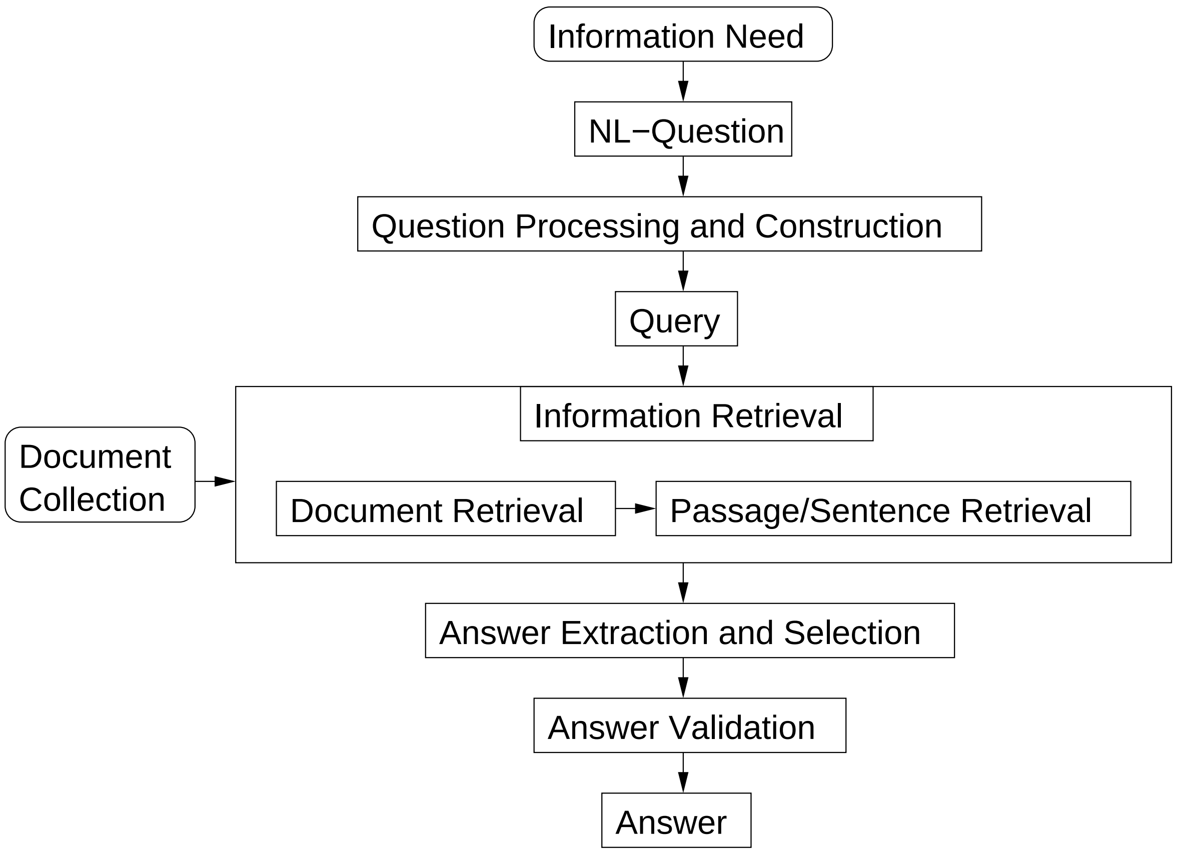


Figure II.1: Information flow in a Q&A system.

The newly ranked text passages (or sentences) are then transmitted to the *answer extraction and selection* module. Here, possible answer candidates are extracted out of the text snippets. When having a list of possible answers, further linguistic methods are used to re–rank the set of answers. Normally, also the question type, gained in the *question processing* step, is used.

The last module denoted in Figure II.1 is an optional step. In this module, each answer from the set of answer candidates is validated. There are many possible ways of checking an answer, but the most popular and easiest way is to search the Internet for that given answer and parts of the question.

After finishing the validation, the system has just to decide how many answers to return. Normally, in case of a factoid question, the Q&A engine returns the top answer. This is not sufficient for list questions; here, the top–N answers are returned, where N has to be defined.

In this thesis, we don't explain a complete Q&A system. We have just investigated statistical

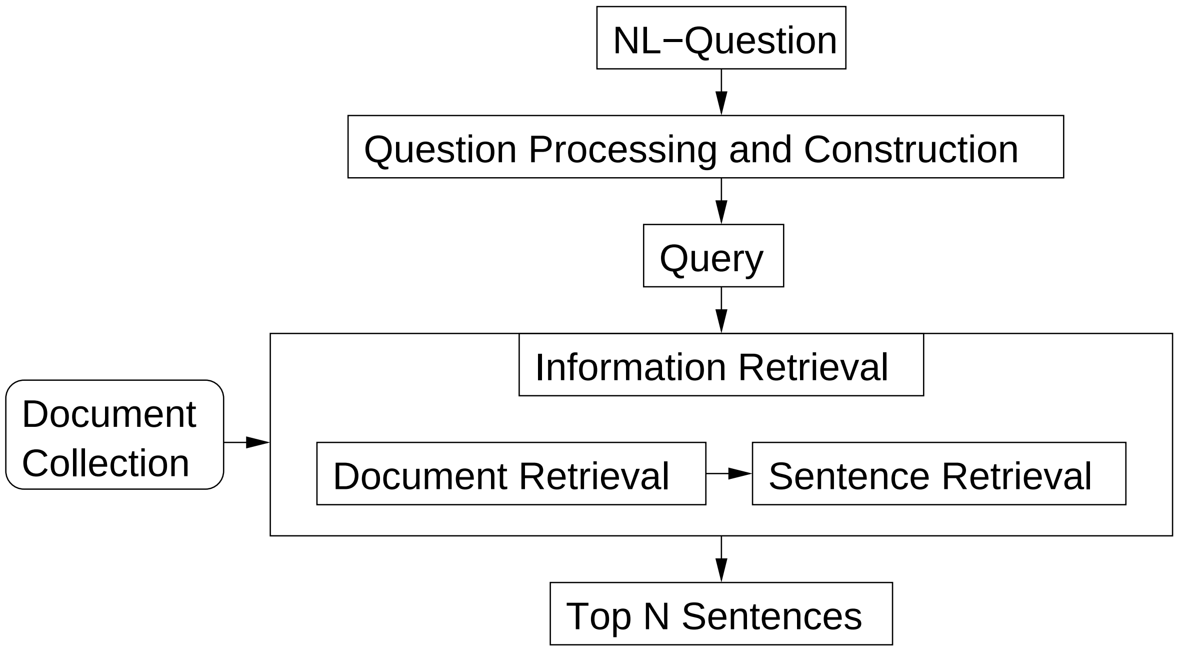


Figure II.2: Information flow of Q&A part used in this thesis.

methods for the modules shown in Figure II.2. It can be seen, that this describes the upper part of a complete system. There is still the *information need* and the *question processing and query construction* with the query to do the retrieval.

The *information retrieval* modules are also present with document retrieval, but instead of doing passage retrieval, we decided to directly take just sentences as default. Nevertheless, we also did experiments with different passage retrieval approaches. Chapter VIII presents more information.

After doing a sophisticated sentence retrieval, the system returns a specified number of sentences. In this thesis, we disregard the answer extraction and selection as well as the answer validation. But as we used these modules for our TREC system, we discuss them in the following sections.

3.1 Question Processing

The first step in a Q&A system is the processing of the question. This is done, because the system has to “understand” the meaning of the information need a user has, i.e. what piece of information should be returned by the system to answer the question. The processing can be done in many ways, depending on the theoretical approach of the system. But normally, the specific class of a question (*question type*) is determined and a query is produced for later retrieval modules, also depending on the query language the retrieval modules provide. Further processing steps are optional, but we will describe some approaches we used in our Q&A system *Alyssa*.

Question Typing

The classification of the user's question into different types is one of the main parts of this thesis. Please refer to chapter VI for more detailed information.

Coarse Class	**Fine Class**	**Description**
ENTITY		entities
	animal	animals
	body	organs of body
	currency	currency names
	dis.med.	diseases and medicine
HUMAN		human beings
	group	a group or organization of persons
	ind	an individual
LOCATION		locations
	city	city names
	country	country names
NUMERIC		numeric values
	count	number of sth.
	date	dates
	percent	fractions

Table II.1: Examples of question types proposed by Li and Roth.

The approach, described in this thesis, uses the classification taxonomy proposed by Li and Roth (2002). It consists of 6 coarse and 50 fine grained classes. For example, all numeric values, like dates, prices or counts, are summarized within on coarse class *NUMERIC*. Some examples of the classes are shown in Table II.1. As already mentioned, different approaches use different classes. But most current Q&A systems use at least similar classes like the coarse ones described in this approach.

The theoretical approach we used in this thesis is language model based. It uses a Bayes classifier to determine the best possible question type. We applied this model because it is known to produce the minimum number of misclassifications if the correct probabilities are known.

As mentioned, language models are utilized to calculate the correct probabilities. More specific, the probabilities are smoothed applying a backing–off approach called Kneser–Ney–Smoothing[4]

To train the statistical models, the 5500 questions provided by the Cognitive Computing Group at University of Illinois at Urbana–Champaign[5] were adopted.

Other approaches, we will discuss later in this thesis, make use of machine learning approaches to classify questions, i.e. the system described by Zhang and Lee (2003) uses support vector machines. A sophisticated approach applying pattern matching as classification paradigm is presented by Suzuki et al. (2003). The system described by Zhang and Lee (2004) also uses a statistical language modeling approach which is not as sophisticated as the approach presented in this thesis.

Question Preprocessing

The *Alyssa* Q&A system uses a simple three–step–approach to do the question preprocessing. It is mainly based on an anaphora resolution.

In a first step, all pronouns in the question are substituted by the main target noun phrase. The second step consists of the replacements of noun phrases in the question, which have the same head word as the target, but a shorter length. If both steps fail to include the target into the question, a noun phrase is selected by rules and replaced with the target noun phrase.

Question Pattern Matching

In the TREC Q&A question sets, there are a number of specific "types" of questions which occur more often than others. For example, question for a specific person X, like *In what year was person X born?* or *In what country was person X born?*, are much more frequent than other questions.

So, we decided to manually build question patterns to map such a question to different question classes. Whereas Kaisser and Becker (2004) uses the syntactic structure of a question to classify it, the *Alyssa* approach uses classes defined in terms of meaning.

In addition to the question patterns, further *answer patterns* were created. If a question

[4]Please read section VI.1.3 for a more detailed discussion of the used language models.
[5]`http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`

matches the question patterns described above, the corresponding answer pattern can be used in later modules to extract the possible answer out of a sentence.

Dependency Relation Path Extraction

If a question cannot be matched by the question patterns, a statistical model is used to determine the relations between a question and possible answer candidates. Following Shen and Klakow (2006), the dependency path between the question and an answer sentence is compared and possible answers are extracted. To parse the question and extract dependency relations in this stage, we use the *Minipar* (Lin 1994) parser.

Query Construction

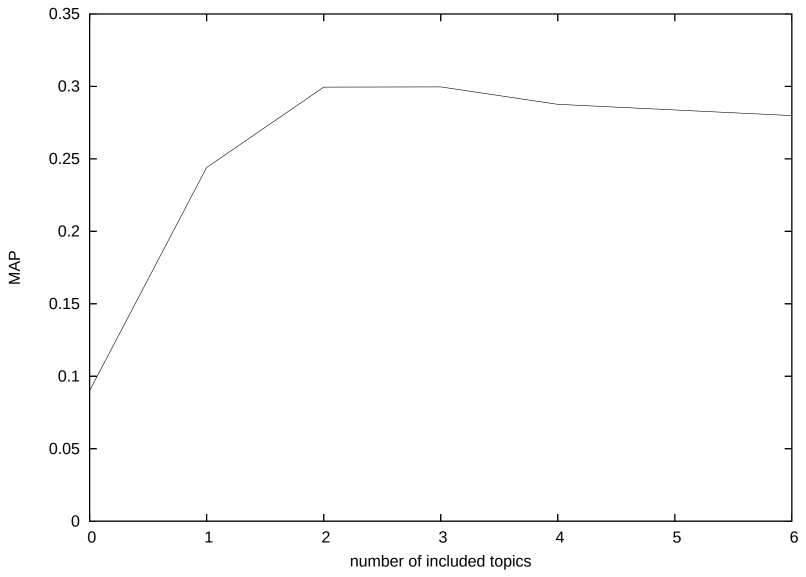


Figure II.3: Number of included topics vs MAP for TREC 2004 data.

Finally, a query for the following information retrieval modules is created. As explained in Miller, Leak, and Schwartz (1999) each term in the query can be weighted with a score. This means that repeating a specific term multiple times would give the latter term more importance in form of a higher score. Because this also holds for our language model approach, we expanded the query in this step with the topic of the question. Figure II.3 shows the effects on our document retrieval systems when we expand the query with the topic multiple times on TREC 2004 data. On the x–axis the number of included topics is shown whereas on the y–axis the *Mean Average Precision (MAP)* is plotted. The performance in-

# included topics	Mean Average Precision
0	0.0900
1	0.2440
2	0.2995
3	0.2997
4	0.2876
6	0.2799

Table II.2: Number of included topics and corresponding MAP for TREC 2004 dataset

creases until including the topic three times. So, if the topic is added too often it gets too much weight and other possible important keywords are scored too lowly and, therefore, the retrieval system performs worse.

Table II.2 also shows the impact of including the topic on the document retrieval. The performance increases until the topic was added three times. This means, if we add the topic too often, it gets too much weight and other possibly relevant keywords are scored too lowly. This would result in a worse retrieval performance. Due to the fact that there is just a very small performance gain between adding the topic twice or three times, we decided to include the topic in our experiments only twice.

3.2 Information Retrieval

Normally, the search for the correct answer begins with the search for relevant pieces of text. In modern information retrieval (IR), there are many ways of doing efficient retrieval as already described in section II.2. In this thesis, we will focus on doing a statistical approach called language model based information retrieval[6] because for our experiments it works best in efficiency and performance.

In open domain question answering, there are two possible text sources. One is a collection of newswire articles, mainly used for the Text REtrieval Conferences Q&A task to compare different systems and the other is the Internet. But regardless of the collection type, there are a hundred of thousand up to millions of documents, which can contain the answer. This makes it impossible to analyze all of them with linguistic methods. Therefore, all Q&A systems have at least a document retrieval component to fetch a set of relevant documents. This *pre–selection* (Monz 2003) helps further components to find possible answers more

[6]A detailed introduction of language model based IR can be found in chapter III.

efficiently.

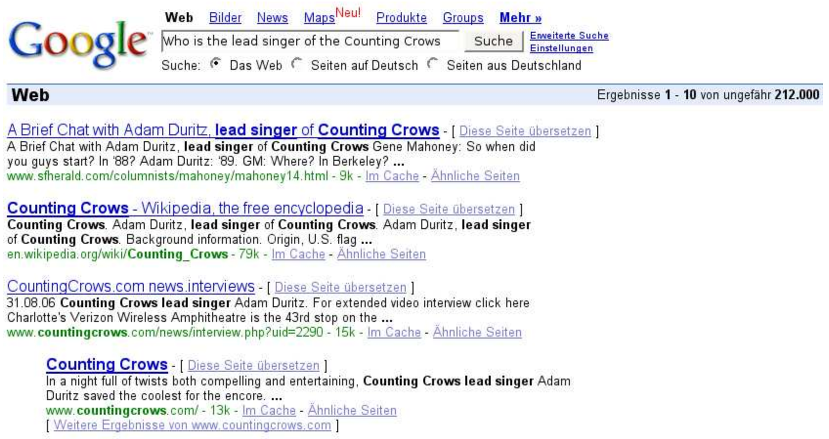
Retrieval strategies strongly depend on the tasks. Whereas most systems of the TREC task use their own retrieval engines, or out–of–the–box IR toolkits like Lemur[7], standard Internet Q&A systems rely on existing web search engines like Google or Altavista. Such systems often offer not only a set of links to relevant pages but also short text snippets with relevant parts of the web site. These information can also be used to find possible answer candidates.

But in both cases, for most instances a simple document retrieval step is not sufficient to find a relevant answer. Relevant full text documents, retrieved from web pages or from news articles, are still too long for the successing modules to do deep linguistic analysis.

Therefore, further, more sophisticated, retrieval approaches are necessary to achieve better results. Normally, these methods split the documents into smaller pieces of text.

In the following subsections, we will describe the single retrieval methods in more detail. This includes document retrieval, passage retrieval as well as the special case of retrieving sentences.

3.2.1 Document Retrieval


Google
Web Bilder News Maps Neu! Produkte Groups Mehr »
Suche
Erweiterte Suche
Einstellungen
Suche: Das Web Seiten auf Deutsch Seiten aus Deutschland

Who is the lead singer of the Counting Crows

Web Ergebnisse **1** - **10** von ungefähr **212.000**

A Brief Chat with Adam Duritz, **lead singer** of **Counting Crows** - [Diese Seite übersetzen]
A Brief Chat with Adam Duritz, **lead singer** of **Counting Crows** Gene Mahoney: So when did you guys start? In '88? Adam Duritz: '89. GM: Where? In Berkeley? **...**
www.sfherald.com/columnists/mahoney/mahoney14.html - 9k - Im Cache - Ähnliche Seiten

Counting Crows - Wikipedia, the free encyclopedia - [Diese Seite übersetzen]
Counting Crows. Adam Duritz, **lead singer** of **Counting Crows**. Adam Duritz, **lead singer** of **Counting Crows**. Background information. Origin, U.S. flag **...**
en.wikipedia.org/wiki/**Counting_Crows** - 79k - Im Cache - Ähnliche Seiten

CountingCrows.com news.interviews - [Diese Seite übersetzen]
31.08.06 **Counting Crows lead singer** Adam Duritz. For extended video interview click here Charlotte's Verizon Wireless Amphitheatre is the 43rd stop on the **...**
www.**countingcrows**.com/news/interview.php?uid=2290 - 15k - Im Cache - Ähnliche Seiten

Counting Crows - [Diese Seite übersetzen]
In a night full of twists both compelling and entertaining, **Counting Crows lead singer** Adam Duritz saved the coolest for the encore. **...**
www.**countingcrows**.com/ - 13k - Im Cache - Ähnliche Seiten
[Weitere Ergebnisse von www.countingcrows.com]

Figure II.4: Example of document retrieval results using Google.

Document retrieval is the most common step for all Q&A systems. The main aim is to translate the user's information need into a set of possible relevant documents which can contain the answer to the user's question. Because this is typically not the last retrieval

[7]http://www.lemurproject.org

step, the *pre–fetching* of relevant documents is also often called *document pre–fetching* (Monz 2003). This also acts as a kind of filter. Special filtering methods (IR algorithms) are used to get relevant documents out of a large collection. As mentioned above, this collection is usually too large to process every document, so, most, or even all state–of–the–art Q&A systems use a document retrieval module for pre–selection.

For all document retrieval experiments, we used some kind of pre–processing. To remove morphological information, a stemmer is used for indexing and question processing. There is a variety of different stemmers available, like the Porter stemmer (Porter 1980) or the Krovetz stemmer (Krovetz and Croft 1992). For our experiments, we choose the most commonly used Porter stemmer implemented in Lemur or versions downloadable at Porter's web page[8].

97.1	Q0	NYT20000120.0096	1	-3.13419	Exp
97.1	Q0	NYT19980710.0072	2	-3.40417	Exp
97.1	Q0	NYT19991207.0201	3	-3.44968	Exp
97.1	Q0	NYT20000727.0115	4	-3.64661	Exp
97.1	Q0	NYT20000811.0090	5	-3.65305	Exp

Table II.3: Example of document retrieval results using the Lemur toolkit.

For example, Table II.3 shows the top–5 document retrieval results using the Lemur toolkit, whereas Figure II.4 shows the results for the Google web search engine, both for the TREC 2005 question "Who is the lead singer of the Counting Crows?". In both cases the results just contain "links" to the original documents. In case of Lemur, they are news articles, i.e. *NYT20000120.0096* from the *New York Times* news paper. Consistently, Google returns hyperlinks to other web pages.

The only difference is, that Lemur also returns scores for the searches, i.e. *-3.13419* for the best document, whereas Google supports the user with relevant text snippets from the appropriate web page.

The reduction of the document collection size also reduces the search space for following components. For linguistic approaches, i.e. in the answer extraction module, it is easier to select possible answer candidates out of a smaller amount of text. So, there is a high interdependence between document retrieval and further answer selection modules. This means by selecting the correct number of retrieved documents, the efficiency of the complete

[8] `http://tartarus.org/~martin/PorterStemmer/`

system will be raised.

This leads to the next problem, namely the number of retrieved documents. When selecting too many documents, the following extraction modules need too much computational time to find possible candidates. It can also introduce much noise to the collection, which might result in worse performance in extracting answer candidates (Monz 2003). If it comes to the worst, this will hurt the system's usefulness. And if this should be a real world application the system might be completely futile.

On the other hand, if the set of retrieved documents is too small, it might be the case that no document containing the answer is retrieved. So, succeeding modules cannot extract any correct answer candidates and the overall performance of the system will degrade.

One scientific goal of this thesis is to find more sophisticated methods to do document retrieval which will increase the overall performance of a Q&A system. This also includes the task of parameter optimizing for retrieving documents using language models. Results can be found in chapter VII.

Because document retrieval is such a critical part of Q&A systems, we also want to present some other important work in that direction.

For example, Salton and Buckley (1988) show some sophisticated term–weighting methods for queries in the vector space model. Their recommended methods strongly depend on the type of the query and the characteristics of the document collection (Zhai and Lafferty 2002).

Llopis, Ferrández, and Vicedo (2002) present in their work a comparison of passage retrieval and document retrieval as a retrieval strategy. This does not mean a chaining of document and passage retrieval but a comparison of both as first level retrieval.

Nearly the same idea was described by Roberts (2002). They also compared the impact of document retrieval vs. passage retrieval for question answering systems. But, they additionally do some research about the length of the used passages for the complete system. They showed, that using two paragraphs as one passage is more efficient than using a complete document.

Finally, we want to mention Clarke and Terra (2003). The content of their research paper is also a comparison of their own passage retrieval module, based on *Okapi*, with a standard document retrieval strategy. Their findings show, that using document retrieval only returns much more results with correct answers, but in the case of question answering, passage retrieval might still be relevant, because of the smaller fraction of text, to find possible answer candidates.

3.2.2 Passage Retrieval

For most state–of–the–art question answering systems, a pure document retrieval is not sufficient. First, a complete document might still be very large. For example, newswire texts can have up to 2000 words and more (referred to the AQUAINT corpus[9]) and therefore it is still very difficult to find the adequate piece of information in it.

A second reason is that there might still be some topic shifts in a document. Regarding weblog (BLOG) pages, there could be a switch in the topic of discussion. It may start with a discussion about a new movie which will end in a discussion about what TV standard is best suited to view this movie.

Therefore, most Q&A systems use a more fine grained text retrieval, often called *Passage Retrieval*. In such implementations, a complete document is split up into smaller text fragment using a variety of different methods. So, normally a text passage is a document fragment containing several sentences. Figure II.5 shows a part of an AQUAINT news text. In this case, the *SGML* framework provides a good opportunity to split the document. After splitting the text using the $<p>$ tags, each passage contains just about 2 sentences.

Another advantage of using passages instead of complete documents is that it is easier for the following modules to extract possible answer candidates. When using deep linguistic algorithms like parsing, it is much more efficient to work on a smaller piece of text.

Important work in the field of passage retrieval, was done i.e. by Allan (1996). Some *Passage Retrieval* methods are also described in Wade and Allan (2005), like *tf–idf*, *query likelihood* or *relevance modeling approaches*. They also show how to evaluate such passage retrieval systems.

Approaches by using text passages (Salton, Allan, and Buckley 1993) or answer passages (O'Connor 1980) also show an improvement in the effectiveness of their retrieval systems.

Cui et al. (2005) present in their work a way of using term density ranking to retrieve possible text fragments. They match the dependency relations between the questions and answer passages. The scores are computed using fuzzy relation matching based on statistical models.

As mentioned above, it is also useful to do passage retrieval for web pages. Cai et al.

[9]Please see chapter II.4.2 for more information about the AQUAINT corpus.

```
<P>
    Camilla Parker Bowles & LR ; slept over recently at Sandringham , the Queen 's estate in Norfolk ,
    says W , which was a first for Prince Charles & LR ; longtime pal .
    Parker Bowles often spends the night at the Prince 's London digs , but an overnight at the Queen
    's place is " a step forward toward bringing the British people around " to accept her special
    friendship with the Prince , says W .
</P>
<P>
   A Malibu mansion bought by Dodi Fayed & LR ; two months before he and Princess Di & LR ; were
   killed has gone on the market for $7.5 million .
   The house is featured in the July issue of Architectural Digest ; the managers of Fayed 's estate
   may have timed the sale to capitalize on the publicity .
</P>
<P>
   Richard Gere & LR ; , proponent of all things calm and serene and Buddhist , says he 's "
   encouraged " by India 's testing of nuclear weapons .
   According to the Times of London , Gere said , " I think other nations in the region have been
   kowtowing to China , the U.N. and other superpowers for too long now . "
</P>
```

Figure II.5: Example of AQUAINT document.

(2004) show how to segment web pages into smaller *blocks*. They compare four different ways which are especially useful for web pages, including fixed–length passages, *Document Object Model* (DOM) based segmentation, a vision–based approach and a combination of all models. They also show the impact of these methods on web search.

Other important work in the field of passage retrieval for question answering systems was done by Clarke et al. (2000). In TREC 9, they introduced their Q&A system *MultiText*, which is mainly based on the length of the passage and weights of the terms. In this system, a passage can start and stop at any position in the text. Beside the passage retrieval, a passage post–processing is used to select the five top–ranked text fragments.

A comparison of various passage retrieval systems for the task of question answering is given in Tellex et al. (2003). They showed the impact of different document and passage retrieval approaches in a Q&A system. For document retrieval, *Lucene*[10], *PRISE*[11] and an oracle, which just return relevant documents, is used.

For the task of passage retrieval, they investigated state–of–the–art systems, like *Okapi BM25* with sliding windows, *MITRE* using the number of terms a passage has in common with the query, an approach by IBM, which uses a series of distance measures, a combination of all methods, called *Voting*, and many other. Their three most important findings were

[10]`http://lucene.apache.org`

[11]`http://www-nlpir.nist.gov/works/papers/zp2/zp2.html`

that boolean querying works well for the task of question answering, the choice of the document retrieval system is very important and finally, the best algorithm in their experiments was density based.

As we focus on statistical language models for the task of Q&A in this thesis, we also want to mention some statistical approaches to passage retrieval.

A language model based system for the general task of passage retrieval is introduced by Liu and Croft (2002). In their work, they compare the language model and a relevance model with full–text document retrieval using the *INQUERY* (Callan, Croft, and Harding 1992) system.

There are also some language model based approaches for passage retrieval for the task of question answering. Corrada–Emmanuel, Croft, and Murdock (2003) for example, present in their work a language model based system for the TREC 9 competition. In particular, they take tagged entities into account and build answer models for each corresponding answer type. As models, they also discuss the query likelihood and a relevance model. Passages are build by splitting the top 20 documents into sentences and forming them into passages of 250 byte length at most[12].

Zhang and Lee (2003) present in their work a statistical language model based passage retrieval using relevance models. To do so, they get an initial set of relevant passages and build a language model for them. In a second step, they get possible relevant data from the Internet and construct a *Web language model* in an analogous manner. After mixing the two models, they include further constraints, like the occurrence of the answer–type in the passage and correct answer context.

Finally, they rank the passages using the *Kullback–Leibler–Divergence*.

3.2.3 Sentence Retrieval

Actually, *sentence retrieval* is just a special case of *passage retrieval*. While in passage retrieval the size of a text fragment might be several sentences, in this special case a passage contains just one sentence. Therefore, this is the smallest entity for following modules to find relevant information.

But because one sentence alone contains just little or no context information at all, there is a need to combine this model with adequate other models to overcome the problem of data

[12]This was the maximum passage size for the TREC 9 Q&A task.

```
 <S rank="1">
   Many of the 800 to 900 films produced each year by the Indian movie industry , which is sometimes
   called Bollywood , feature spectacularly scenic backgrounds that are filmed in faraway locations ,
   typically Switzerland .
 </S>
 <S rank="2">
   The center of the film industry is in Bombay , from which the name Bollywood is derived .
 </S>
 <S rank="3">
   Mukta Arts , his movie production company , has just closed the first-ever public offering of shares by
   a big firm from Bollywood , the name for the Hindi movie industry located around Mumbai
   ( Bombay )
</S>
<S rank="4">
   And Andrew Lloyd Webber , the superstar British musical machine with a keen sense of the popular , is
   preparing to stage Bombay Dreams , a new musical set in Bollywood .
</S>
```

Figure II.6: Example of sentence retrieval output.

sparseness. In the statistical language model based approach, we introduce in this thesis, we solve the problem by using adequate smoothing models. For example, a sentence model is smoothed with the document, the sentence is derived from, and this model is further smoothed with the document collection. A more detailed discussion of this topic is found in chapter IX.

Another problem of using just one sentence as a paragraph is that successing modules, like deep linguistic approaches to select answer candidates, might possibly fail because of the lack of context. But, as described in section II.3.5, this does not hold for the *Alyssa* system, partially described in this thesis. Figure II.6 shows the top four example results for *Alyssa's* sentence retrieval for the question *Where is Bollywood located?* As can be seen, possible answer candidates can be found in the top three answer sentences.

Other applications for sentence retrieval are presented i.e. in Otterbacher, Erkan, and Radev (2005). They show in their work how to solve “the problem of question–focused sentence retrieval from complex news articles describing multi–event stories published over time”. This means, that most of the questions are time–sensitive. To do sentence retrieval in this context, they use a stochastic, graph–based approach.

Another application is the TREC task of *Novelty Detection* described in Larkey et al. (2002) and Allan, Wade, and Bolivar (2003). This task mainly consists of two parts

1. find relevant sentences to a given query and

2. find novel sentences out of the result list found in the first step.

As described in Allan, Wade, and Bolivar (2003), the difficult part is the finding of relevant sentences. In contrast to the Q&A task, this challenge starts with a set of relevant documents.
In their work, sentence retrieval is done using a stopword list and a stemmer. For ranking sentences, they compared the standard vector–space model using tf–idf, a language model based approach using the Kullback–Leibler divergence and a *Two–stage smoothing* as described in Zhai and Lafferty (2002).

A comparison between language models based on multinomial and multiple Bernoulli is shown in Losada (2005). Multinomial models are de facto standard in today's language model based retrieval algorithms, whereas multiple Bernoulli was the original proposal by Ponte and Croft in 1998 (Ponte and Croft 1998).
Multiple Bernoulli is based on the assumption that texts are a *bag of words*, and so cannot handle non–binary notion of term frequencies. But this is generally no problem for small vocabularies, like sentence retrieval. On the Novelty track, they showed an improvement when using the Multiple Bernoulli approach for sentence retrieval.

A translation model for sentence retrieval for the same Novelty task is described in Murdock and Croft (2005). To do a monolingual retrieval, they create a parallel corpus to use their translation models. Because they lack of a proper parallel corpus, they estimated the translation model using *Mutual Information*, lexicons, WordNet and an Arabic–English corpus. To smooth the sentences of the language model based translation model, they use the surrounding 5–11 context sentences.

Murdock and Croft (2004) also propose the translation model for the task of question answering. The idea behind the translation of the user question into a more or less complex answer is that sentences are possibly too short to compute a multinomial distribution. Therefore, a translation model is computed to overcome this problem. This model, which also makes use of synonymy and word relations, is mainly based on the *IBM Model 1* (Brown et al. 1993). They also showed that query expansion will not work on short documents, like sentences.

For the task of definitional question answering, Cui, Kan, and Chua (2004) proposed a promising approach. Their system uses external knowledge resources like WordNet and the Internet as well as a soft pattern matching method to find relevant sentences. As result, they showed that using Web knowledge improves the efficiency of definitional Q&A.

Another approach for sentence retrieval in context of question answering is described in Wu, Zhao, and Xu (2006). Here, cluster–based language models for Chinese question answering is presented. To group the sentences into clusters, two approaches are proposed, the *One–Sentence–Multi–Topics* and *One–Sentence–One–Topic* method.

3.3 Answer Extraction

After the retrieval steps, a Q&A system still has to find possible answers out the text fragments. As result of this module, one or more answers are returned. If just the top answer is needed, the system typically returns the best one from the set of possible answer candidates. Since this is a very time–consuming step, a good retrieval is indispensable. Normally, the answer extraction starts with question–based sentence ranking on the set of retrieved sentences (Paşca 2003).

Inspired by *Information Extraction* methods, there are many possible ways to identify answers. For example, an answer extraction module can use named entity recognition, parse trees, patterns and other approaches to identify possible candidates. Likewise the frequency of specific answers can be used to make a decision for the best one. To take the number of occurrences of an answer is also called *redundancy–based answer selection* (Clarke et al. 2002).

In the *Alyssa* system, a relation path correlation–based method was used to ranking answer candidates. This means, that using the correlation measure, the dependency relations of candidate answers and mapped question phrases are compared with the corresponding relations in the question (cf. Shen and Klakow (2006)).

3.4 Web–based Answer Validation

When finding one or more answers in the answer extraction part, a possible validation can improve the performance and the quality of the complete system.

There are many ways to validate an answer candidate. In this section, we want to discuss the possibility to use the Web as knowledge resource for answer validation. For example, the TREC Q&A task normally expects newswire articles as answers. This kind of answer can easily be validated using web search engines (Shen et al. 2006, Kaisser, Scheible, and Webber 2006).

Not only web search engines like *Google* can be used to verify an answer, but also other knowledge bases, like online encyclopedias, i.e. *Wikipedia*, can help to find the correct answer.

One possible option to use the Web for answer validation is to send the question as well as answer candidates to a search engine and count the number of resulting pages. These numbers are then used to re–rank the set of answer candidates (Neumann and Sacaleanu 2005, Magnini et al. 2001). Alternatively, the answer with the most hits is taken as the correct one and returned to the user.

But there are also systems going the other way around. Kaisser, Scheible, and Webber (2006) describe in their work the experiments with their system at TREC 2006. They start with a Web search to find possible answer candidates in the Internet. After fetching a list of candidates, those are validated with the AQUAINT newswire corpus. This approach got very good results at TREC 2006 Q&A competition.

Magnini et al. (2001) exploit in their work the web redundancy to quantify the connection between an answer and the user question. After modeling patterns for question and answers (“validation patterns”), they use a Web search engine to retrieve possible answers and compute scores including the number of retrieved documents. Then, the degree of relevance of the Web search is estimated using different approaches, i.e. a pointwise mutual information (MI), the maximum likelihood (ML) ratio and the corrected conditional probability.

A similar approach is used by Neumann and Sacaleanu (2005). Here, the question–answer pair is sent to a Web search engine and “the resulting total frequency (TFC) is used to to sort the set of answer candidates according to the individual values of TFC”. The answer with the highest TFC is then selected and returned to the user.

(Shen et al. 2006, Shen et al. 2007) also use a similar methodology to validate possible answer candidates. In this approach, the questions are transformed into different forms using patterns. These forms include *Bag–of–Words*, *Noun–Phrase–Chunks* or *Declarative Forms*. The declarative form leads to the best results, but it is not guaranteed that the search engine will return adequate text snippets. So, all forms are weighted and a score is computed to rank the answer candidates.

3.5 Current Approaches

This section introduces some state–of–the–art question answering systems to show differences as well as similarities of today’s approaches. Most of the systems presented in this chapter actively participated at TREC 2006, amongst others the *Alyssa* system, which was developed at our chair and is partially described in this thesis.

The first system we want to present is not really a Q&A engine concerning our definition of Q&A. The *AnswerBus* question answering system developed by Zheng (2002) answers open–domain questions by returning a set of relevant sentences. This is contradictory to our definition, where a short answer has to be provided, but it is related in a broader sense, because it accepts natural language questions and provides the user with answer sentences. Another advantage of the AnswerBus Q&A system is the multilingual approach. It accepts questions in six different languages, English, German, French, Spanish, Italian and Portuguese. After a translation of the questions, it provides the user with answer sentences in English.

The information source used for this approach is the Web and therefore, a working version can be found on the AnswerBus homepage[13]. Overall, five different search engines[14] are used to retrieve a set of ten possible answer sentences. To obtain the best results, the three most reasonable search engines are selected, depending on the task of the question. For example, if the question asks for newswire texts, *Yahoo News* should be preferred to other engines, like *Google*.

After generating a specific query for each search engine used in a run, the top documents returned are split into sentences and a term matching algorithm is used to find relevant sentences. Finally, some natural language processing methods are used to re–rank the set of relevant sentences. For example, the question type is determined and used to judge sentences as well as named entities and a kind of co–reference resolution.

The work presented by Kaisser, Scheible, and Webber (2006) describes the system they used for the TREC 2006 Q&A task. It is mainly based on lexical resources, or more specific, on frame semantics. For this purpose, they implemented an algorithm based on *FrameNet* (Baker, Fillmore, and Lowe 1998), *PropBank* (Palmer, Gildea, and Kingsbury 2005) and *VerbNet* (Schuler 2005). Because of the novelty of those resources compared to *WordNet*, their main goal was to prove the usefulness for the task of Q&A.

The system mainly consists of two different algorithms to find relevant information. The first one builds answer patterns out of the questions, search the Web for possible answer candidates and validates the answer. In particular, it uses the *MiniPar* (Lin 1998) parser and

[13] `http://www.answerbus.com`

[14] Google, Yahoo, WiseNut, AltaVista, and Yahoo News.

the lexical resources *FrameNet*, *PropBank* and *VerbNet* to construct abstract structures of possible answer sentences. Out of these structures, exact queries are generated and sent to Web search engine. So, the exact answers can directly be read from the resulting Web documents using the abstract structures.
The second algorithm builds keyword–based queries for the Web and tries to find an adequate answer. More specific, the keywords from the query are sent to a Web search engine and the resulting documents are split into sentences. Then, a rule–based and weighted algorithm is used to compare the dependency structure of *PropBank* examples with the candidate sentences to extract possible answers.

After using the two different algorithms, the (Web) answer candidates are mapped to the AQUAINT corpus. To do so, queries are created out of the question and possible answers. Then, the *Lucene* (Hatcher and Gospodnetić 2004) retrieval engine is used to find relevant documents. After locating sentences containing the answer, the sentences are scored according to the presence or absence of key terms. This method is also used to process definitional questions.

Whittaker et al. (2006) also present in their work an approach to the TREC 2006 Q&A task. The underlying theoretical model for answering factoid questions is a data–driven, completely non–linguistic approach. In this approach, an answer only depends on the given question. So, they use the Bayes classifier on the product of the probability of the answer given, some information bearing set of features from the question (*retrieval model*) and the probability of the set of features describing the question type given the answer (*filter model*). The *retrieval model* is “essentially a language model which models the probability of an answer sequence [. . .] given a set of information–bearing features [. . .]” (Whittaker et al. 2006). It models the proximity of an answer to information–bearing feature set.
On the other hand, the *filter model* “matches an answer with features in the question–type set. [. . .] This model relates ways of asking a question with classes of valid answers” (Whittaker et al. 2006). This means, the filter maps some features of the answer, i.e. dates, to different question types, like *when* questions.

The factoid part of the system also uses the Web to generate possible answer candidates. For final results, they use a combination of their English, Spanish and French system with different weights.
To answer the list question task, the same algorithms are used, but instead of returning the

top answer, this module returns the top 10 answers from the factoid system.

Definitional (“other”) questions are answered in another way. Here, they treat the task as a variant of text summarization and so, an algorithm from speech summarization is used to detect a possible answer. The data is only extracted from AQUAINT, in contrast to the factoid stream.

They also used a kind of pre–processing for questions and target documents. In this system, the question pre–processing just consists of removing the punctuation and mapping of each word to upper–case. The target document preparation is mentioned as generating a query for a Web search engine, downloading the top 500 documents and removing of the HTML markup.

Finally, a backprojection to the AQUAINT corpus is done using an external program, called *Aranea* system (Lin and Katz 2003).

An online system of this work can be found on the *Asked* homepage[15].

The *La Sapienza* system presented by Bos (2006) is the complete opposite to the last approach. Whereas the former system was data–driven and non–linguistic, this Q&A engine is completely linguistically inspired. A “Combinatory Categorical Grammar is used to generate syntactic analyzes of questions and potential answer snippets, and Discourse Representation Theory is employed as formalism to match the meanings of questions and answers” (Bos 2006).

In the first analysis step, the question is tokenized and parsed. The output is then used to receive further information like answer type, answer cardinality and tense as well as some background knowledge from lexical resources, i.e WordNet. Finally, a query for doing document retrieval is created.

For doing document retrieval, the complete AQUAINT corpus is pre–processed. This comprises the removing of the *SGML* syntax, a sentence boundary detection and the tokenization of the sentences. Then, *mini–documents* are created by merging two sentences using a sliding window.

The document retrieval itself uses two kinds of different queries with different degrees of difficulty. Then, the *Indri* (Metzler and Croft 2004) search engine is used to retrieve 1.500 of the created *mini–documents*.

For answer extraction and selection, the *Discourse Representation Structure* of the question

[15]`http://asked.jp`

and answer documents is compared to find potential answers. Finally, the answer candidates are re–ranked.

Schlaefer, Gieselmann, and Sautter (2006) decribe in their work the *Ephyra* Q&A system for TREC 2006. It consists of a flexible framework to join several question analysis and answer extraction techniques, combined with a variety of knowledge bases.

For doing factoid question answering, they use two different approaches, an answer type analysis, and an approach using textual patterns to classify, interpret and extract answers. Both methods use the Web to find answer candidates and then do a backprojection to the AQUAINT corpus.

Before doing the retrieval step, a co–reference resolution using the question target and a question normalization, which removes punctuation and unnecessary phrases, is done. Then, a keyword–based query is constructed and a retrieval using *Yahoo* and *Indri* is performed. The results of this module are text paragraphs.

The answer extraction by answer types uses about 70 different named entities, from which adequate patterns (regular expressions) are specified to extract possible answers in later modules. Depending on the type of the named entity (NE), different NE taggers are used.

When using the answer extraction by pattern matching, textual patterns, which are automatically learned using question–answer pairs, are produced to classify and interpret questions and to extract answers from text snippets.

To backproject the Web answer candidates to the AQUAINT corpus, the same extraction algorithms are used as described above. If this produces no results at all, answer candidates are extracted from the passage output produced using Indri.

The *Language Computer Corporation* (LCC) also presented their work at TREC 2006.

Harabagiu et al. (2006) show with the CHAUCER system for "combining several strategies for modeling the target of a series of questions" and for "optimizing the extraction of answers" (Harabagiu et al. 2006).

For target processing, a *Maximum Entropy* classifier is used to determine the type of the target. Then, a keyword–based query from the target is constructed to find relevant passages for this specific target. Using a subset of top 50 passages, LCC's PropBank–based parser is used to "generate natural language questions from each predicate found in the top–ranked passages" (Harabagiu et al. 2006).

The question processing used in CHAUCER consists of three components; a keyword expansion, a question coreference, and an answer–type detection.

The keyword expansion uses a set of heuristics to include synonyms and a variety of other keywords to the query. A heuristic–based name aliasing and nominal co–reference resolution is used in the question co–reference module. The last step, the answer type detection uses a two–stage Maximum Entropy–based approach to classify the expected answer type of a question. The system has a base of more than 300 different named entity types.

There are also many different methods to pre–process the AQUAINT corpus. First, they do full syntactic parses of each document. Then, three semantic parsers are used to elaborate semantic dependencies. After doing a named entity recognition and a mapping of temporal expressions, a nominal and pronominal coreference resolution is done. Finally, *Lucene* is used to perform document and passage retrieval.

The optimized answer extraction (AE) and selection module in CHAUCER uses five different approaches to identify possible answers from passages. This includes i.e. an entity–based AE, which makes use of the more than 300 entity types, a pattern–based AE, which uses hand–crafted patterns, a soft pattern–based AE, which uses automatically generated patterns and much more. Then, the top 5 answer candidates from each approach are re–ranked using a *Maximum Entropy* model.

After extracting possible answer candidates, the top 25 answers are sent to the selection module. This method uses a textual entailment system developed at LCC to return the most reasonable answer.

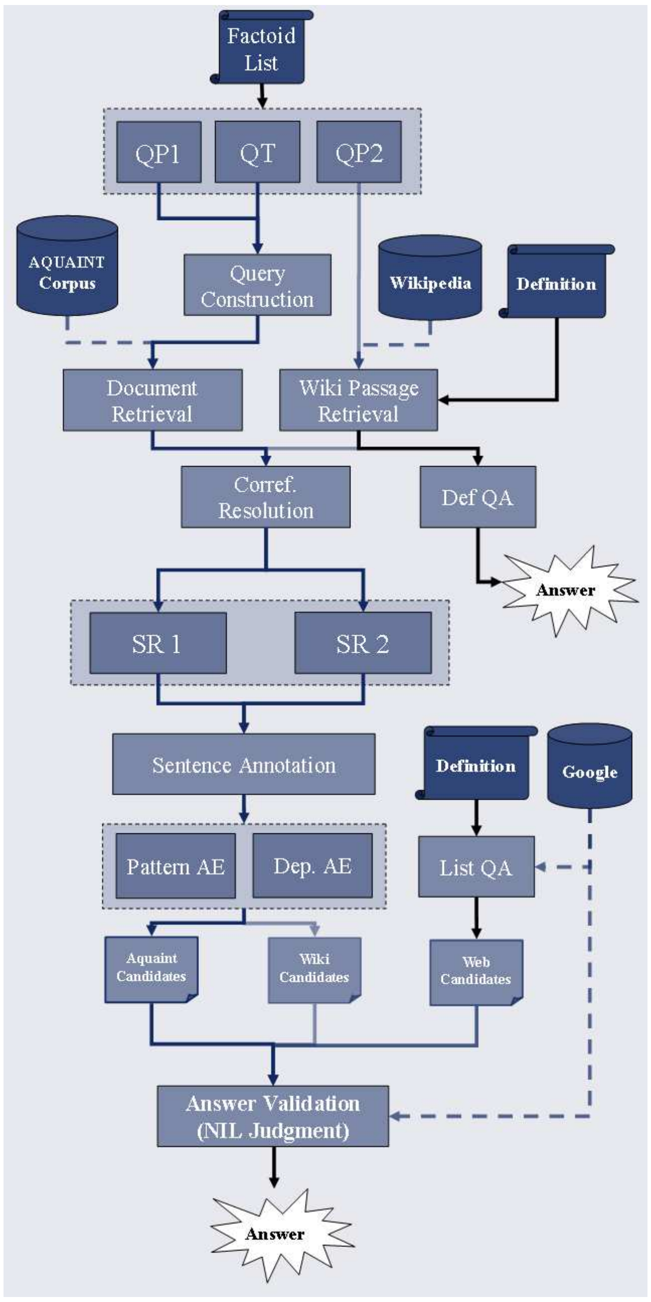


Figure II.7: Module overview of the *Alyssa* system.

The last state–of–the–art system, we want to present in this section, is the *Alyssa* system (Shen et al. 2006), our group developed for the Q&A task at TREC 2006. This system is a statistically–inspired and very flexible approach to perform open–domain question answering. It uses a “cascade of language model (LM)–based document retrieval, LM–based sentence extraction, Maximum Entropy–based answer extraction over a dependency relation representation followed by a fusion process that uses linear interpolation to integrate evidence from various data streams [. . .]" (Shen et al. 2006).

Figure II.7 shows the architecture of the *Alyssa* system with the used components.

For factoid questions, the first step is a question analysis. Here, a question pre–processing, using a simple anaphora resolution approach, is performed. Then, the expected answer type is extracted by chunking questions with the *Abney chunker* (Abney 1989). Using the noun phrases of the chunks, the expected answer types are located. Because there are questions at TREC occurring with a high frequency, hand–made patterns are defined to map these kinds of questions into different classes. For each of these classes, an adequate answer pattern is created. Finally, if a question could not be classified by the methods mentioned above, the dependency relation between a question and an answer sentence is computed. In this step, the question is parsed by using the *Minipar* parser.

A second module in this step is question classification and typing, which is discussed in detail in chapter VI. It uses a Bayes classifier with language models to compute the best possible answer candidate for a given question.

After constructing a query by including the question target for multiple times, a language model–based document retrieval is performed using the *Lemur* toolkit for information retrieval. Then, the top 60 documents are fetched and split up into sentences. The answer sentences are re–ranked using a sophisticated language model–based approach which also takes the expected answer types from preceding steps into account. More detailed information about document and sentence retrieval can be found in chapter VII and IX.

For answer extraction, the resulting sentences are first processed by using linguistic tools like *LingPipe*[16] for named entity recognition, Abney's chunker and MiniPar. Then, two answer extraction strategies are applied to the processed sentences. The first one uses answer patterns which indicate expected answer positions in surface sentences (*Surface Text Pattern Matching*).

The second answer extraction approach calculates a score to compare the dependency relations between answer candidates and "mapped question chunks in sentences with corresponding relations in questions" (Shen and Klakow 2006). A Maximum Entropy ranking model is then used to determine the top answer candidates for a given question.

To get even better answer candidates, we also used further sources, like Web search engines and the *Wikipedia* online encyclopedia. Therefore, a final selection of the best answer

[16] `http://www.alias-i.com/lingpipe/`

has to be done. For this purpose, a fusion module takes the different answer candidates as input and uses a weighted linear interpolation to select the best possible answer.

4 Evaluation of Question Answering Systems

For each implementation of a software–based system, an evaluation is necessary to measure the success of this system. Following Baeza–Yates and Ribero–Neto (1999), the first step of an evaluation is the proof of functionality, this means a test, if the system works as specified. Normally, this step also includes a kind of error analysis.

A second measure is the evaluation of the systems performance in time and space consumption. The fewer time and space a software approach needs to perform a task, the better is the system. For most implementations, there is a strong connection between time and space complexity which "allows trading one for the other" (Baeza–Yates and Ribero–Neto 1999).

And finally, an evaluation of software systems can involve a comparison with other approaches for the same task. For the task of Q&A this benchmarking is done by the *Text REtrieval Conference*, presented in section II.4.2.

4.1 Evaluation Metrics

For retrieval systems, there are some evaluation metrics which cover the quality of the returned answers to a user's information need. They include for example how precise a set of answers is. Normally, such type of evaluation is called *retrieval performance*. In this section, this *retrieval performance* evaluation for information retrieval and question answering systems is discussed. To do so, further test collections and evaluation measures are needed to describe the performance.

To make the problem more explicitly, we want to introduce a retrieval example. This example guides through the different evaluation measures and shows how the approaches work in practice. Table II.4 shows the results of an imaginary search engine. The results of Table II.5 present the ranking of human experts for the same question. The complete corpus contains 1000 documents, labeled $D1$ to $D1000$ (Weikum 2000).

1.	D877
2.	D432
3.	D558
4.	D121
5.	D47
6.	D932
7.	D111
8.	D865
9.	D99

Table II.4: Example results of an imaginary information retrieval system.

1.	D558
2.	D633
3.	D47
4.	D955
5.	D877
6.	D111

Table II.5: Example results of a human expert.

Recall and Precision

The most common metrics for information retrieval systems are *Recall* and *Precision*. They describe the ability of a system to extract relevant documents (Baeza–Yates and Ribero–Neto 1999). In particular, the *Recall* measures the completeness of an IR engine, which means the capability of the system to find all relevant documents.
The *Precision* measures the system's ability to retrieve just relevant documents. They are defined as follows.

- "**Recall** is the fraction of relevant documents which has been retrieved" (Baeza–Yates and Ribero–Neto 1999):

$$\text{Recall} = \frac{\#\text{relevant documents in answer set}}{\#\text{relevant documents in collection}} = \frac{|Ra|}{|R|} \quad \text{(II.1)}$$

  where $|Ra|$ is the number of relevant documents in the answer set and $|R|$ is the set of relevant documents.

- "**Precision** is the fraction of the retrieved documents which is relevant" (Baeza–Yates and Ribero–Neto 1999):

$$\text{Precision} = \frac{\#\text{relevant documents in answer set}}{\#\text{documents in answer set}} = \frac{|Ra|}{|A|} \tag{II.2}$$

where $|Ra|$ is again the number of relevant documents in the answer set and $|A|$ is the number of all documents in the answer set.

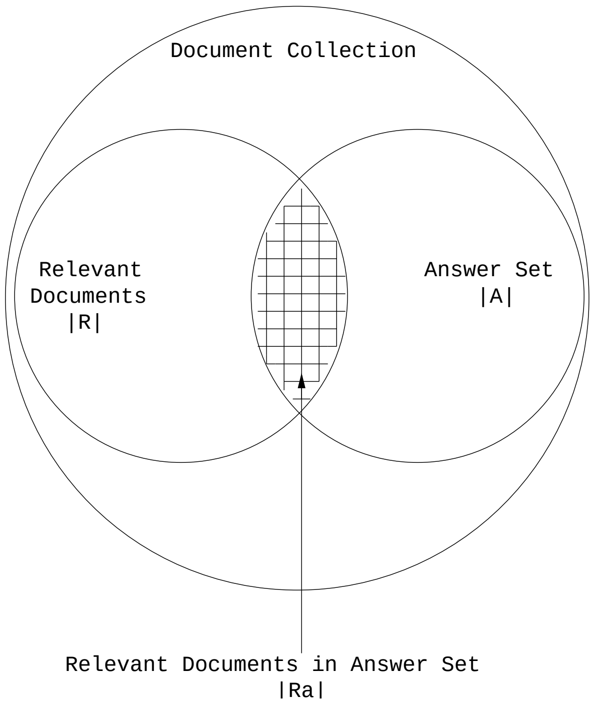


Figure II.8: Precision and recall example for set theory.

All sets, i.e. the collection, the set of relevant documents, the answer set, and the intersection of both, are illustrated in Figure II.8.

The need for getting high precision or high recall strongly correlates to the task of the information retrieval engine. Tasks, like finding the day of birth or the place of birth of a person do not necessarily need a high precision, because every relevant document contains the same information.

However, if the system should find more general information, like the opinions about a movie, a high recall is needed, because in this case, relevant documents will contain different information, like different opinions about the movie. For the task of question answering, a higher precision is preferable because we theoretically need just one relevant document containing the answer. This also implies that the answer extraction can find the answer in this document.

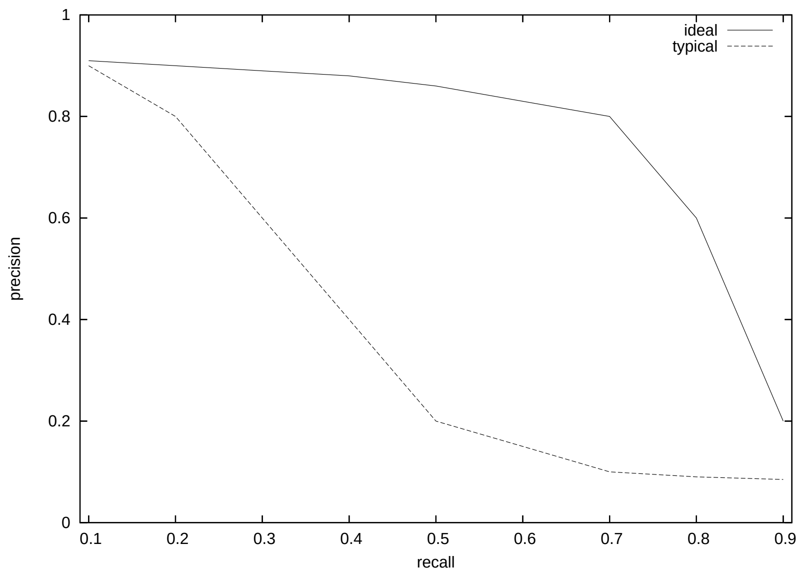


Figure II.9: Ideal and typical precision–recall graph.

Also precision and recall strongly correlate to themselves. Ideally, all documents found are relevant, which results in $Precision = Recall = 1$. But because of inexact questions and the limitations of search engines, this is almost impossible. Figure II.9 shows an ideal and a typical precision–recall curve. On the x–axis, recall is plotted, whereas on the y–axis, the precision of the system can be found.

As illustrated, the ideal precision remains very high when increasing the recall. But the typical curve shows that, in practice, the precision drops very fast for increasing values of recall. This means, when raising the recall, more noise is introduced and therefore the precision decreases.

Concerning the example from page 42, precision and recall are computed as follows:

- Precision:

$$\text{Precision} = \frac{\#\text{relevant documents in answer set}}{\#\text{documents in answer set}} = \frac{4}{9} \tag{II.3}$$

- Recall:

$$\text{Recall} = \frac{\#\text{relevant documents in answer set}}{\#\text{relevant documents in collection}} = \frac{4}{6} = \frac{2}{3} \tag{II.4}$$

Average Precision

The *Average Precision* is “the average of the precision value obtained for the top set of n documents existing after each relevant document is retrieved [. . .]” (Manning, Raghavan, and Schütze 2007). The following formula describes this fact

$$\text{avgPrec}(r) = \frac{1}{n}\sum_{i=1}^{n}\text{Prec}_i(r) \tag{II.5}$$

where n is the number of the set of relevant documents and $Prec_i(r)$ is the precision at recall level r.

Precision at n

It is also very common to cut the number of returned documents and calculate the precision with this set. When using the top n documents, this is also called *precision at* n or simply $P@n$.

Mean Average Precision

The *Mean Average Precision* (MAP) is currently the most important evaluation metric for TREC competitions. It "provides a single–figure measure of quality across recall levels" (Manning, Raghavan, and Schütze 2007). This means, the average precision described above is averaged over all given queries. It is defined by

$$\text{MAP}(Q) = \frac{1}{q}\sum_{j=1}^{q}\frac{1}{n}\sum_{i=1}^{n}\text{Prec}_i \tag{II.6}$$

where q is the number of given queries and n is the number of relevant documents.

"When a relevant document is not retrieved at all, the precision value in the above equation is taken to be 0. For one question, the average precision approximates the area under the uninterpolated precision–recall curve, and so the MAP is roughly the average area under the precision–recall curve for a set of queries" (Manning, Raghavan, and Schütze 2007).

R–Precision

The *R–Precision* describes the precision after r documents. It defines the mean of average precision at document ranks 5, 10, 15, 20, 30, 50, 100, 200, 500, and 1000.

F–Measure

An evaluation metric which combines the precision and recall for a query is given by the *F–Measure*. It is defined by the weighted harmonic mean of both, precision and recall:

$$F = \frac{1}{\alpha\frac{1}{\text{Prec}} + (1-\alpha)\frac{1}{\text{Rec}}} \tag{II.7}$$

As described above, a special case of this formula is the *Harmonic Mean* ($F1$). In this case, α is set to $\frac{1}{2}$. The harmonic mean is 0, if no relevant document is retrieved, and 1, if all retrieved documents are relevant. "Further, the harmonic mean F assumes a high value only when both recall and precision are high. Therefore, determination of the maximum value for F can be interpreted as an attempt to find the best possible compromise between recall and precision" (Baeza–Yates and Ribero–Neto 1999).
Considering our example again, the *F–Measure* is computed as follows

$$F = \frac{1}{\alpha\frac{9}{4} + (1-\alpha)\frac{3}{2}} = \frac{1}{\frac{3}{4}\alpha + \frac{3}{2}} \tag{II.8}$$

For the special case of harmonic mean, the equation is solved to $\frac{8}{15}$.

E–Measure

The *E–Measure* also combines the precision and recall values for a query. It was introduced by van Rijsbergen (1979) to allow a user to model his interest in recall or precision on his own. It is defined by

$$E = 1 - \frac{1+b^2}{\frac{1}{Prec} + \frac{b^2}{Rec}} \tag{II.9}$$

where $Prec$ and Rec are precision and recall and the parameter b specifies the users interest in recall and precision. The larger b is chosen, the more is a user interested in precision than in recall and vice versa. For the special case of $b = 1$, the *E–Measure* is the complement of the harmonic mean, as defined above. Regarding our example from page 42 for this case the *E–Measure* is computed as follows

$$E = 1 - \frac{1+b^2}{\frac{9}{4} + \frac{3}{2b^2}} = 1 - \frac{2}{\frac{9}{4} + \frac{3}{2}} = 1 - \frac{8}{15} = \frac{7}{15} \tag{II.10}$$

Fall–Out

The *Fall–Out* is an evaluation measure which describes the capability of a IR system to retrieve as few irrelevant documents as possible. The formula of this measure is as follows

$$\text{Fall–Out} = \frac{\#\text{non–relevant, retrieved documents}}{\#\text{non–relevant documents}} \tag{II.11}$$

This means for our example of an imaginary search engine that the *Fall–Out* is defined as

$$\text{Fall–Out} = \frac{5}{994} \tag{II.12}$$

BPref

When the judgements for a specific query are not complete, the *BPref* measure is still able to compute the performance of the system. In particular, it computes the fraction of known relevant and irrelevant documents:

$$\text{BPref} = \frac{1}{R}\sum\frac{|n\text{higher ranked than}r|}{\min(n,r)} \tag{II.13}$$

Accuracy

An evaluation measure for the performance of a complete question answering system or parts of it, the *Accuracy* is defined as

$$\text{accuracy} = \frac{\#\text{correct answers}}{\#\text{returned answers}}. \tag{II.14}$$

Hence, it measures the percentage of answerable questions using a specific number of returned answers.

Mean Reciprocal Rank

The *Mean Reciprocal Rank* (MRR) is a precision–oriented evaluation measure for information retrieval and question answering systems. It calculates the reciprocal rank r_q for a given query q where the first correct answer occurs. If there is no correct answer at all, the MRR is 0. The following formula defines the MRR

$$\text{MRR} = \frac{1}{n}\sum_{q=1}^{n}\frac{1}{r_q} \tag{II.15}$$

where n is the number of queries in the system.
For our small example, the *MRR* is 1 because there is just one query and a valid answer is on the first rank.

Misclassification Error Rate

The *Misclassification Error Rate* (MER) defines the percentage of locations with errors in a classified map. It is the probability that a given system incorrectly classifies an instance from a sample obtained in a stage later than a training sample.

4.2 Question Answering at the Text REtrieval Conference (TREC)

There is a long tradition in open–domain question answering. There are publications for more than four decades, like the *ORACLE* (Phillips 1960) or the *ALA* (Thorne 1962) system. But there have been few other publications about this topic when the Text REtrieval Conferences started an own Q&A track in 1999 (Voorhees and Harman 1999). Since that date, the TREC evaluation for Q&A has been the de facto standard for comparing such systems. At this track, participating user groups normally have one week to run their systems on a set of questions provided by NIST. After an official evaluation, there is an annual conference, where people can present their systems and can exchange their experiences. Nearly all of the experiments we did in this thesis rely on TREC data or evaluation metrics provided by NIST.
To understand the idea behind the *Text REtrieval Conference* (TREC), we have to look back to about 25 years before the first conference.
In the late 1960s, Cleverdon (1967) created the first test collection for their different indexing approaches. This collection contained about 1.400 documents, 225 queries about aerodynamics, and a list of relevant documents according to the collection. And it was also used by other researchers in the field of information retrieval. In the following years, many other test collections appeared and were used to determine the performance of IR systems.
Following Voorhees and Harman (2007), there are mainly two major disadvantages of this proceeding. First of all, the different groups neither were bound to a specific test collection

nor had to use the same evaluation measures or compare their system results.

The second flaw was the size of existing collections. No test corpus had a realistic size to do proper information retrieval.

So, in 1990 the *National Institute of Science and Technology* (NIST) was asked to create an adequate test collection with standardized evaluation metrics and a forum to discuss the results of the different system approaches. This project was then developed as part of the *Defense Advanced Research Projects Agency* (DARPA) Tipster project (Merchant 1993). The four main goals of TREC can be found in Voorhees and Harman (2007)

- To encourage research in text retrieval based on large test collections
- To increase communication among industry, academia, and government by creating an open forum for the exchange of research ideas
- To speed the transfer of technology from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real–world problems
- To increase the availability of appropriate evaluation techniques for use by industry and academia, including the development of new evaluation techniques more applicable to current systems

Since the first official conference meeting in November 1992, TREC is the de facto standard for text retrieval benchmarks. When the first experiments started, the provided test collection was about 100 times larger than the largest existing corpus. “Over the years, information retrieval systems such as SMART (Buckley, Salton, and Allan 1993), SPIDER (Schauble 1993), OKAPI (Robertson et al. 1994) and INQUERY (Allan 1996) were often improved based on their performance results on large test collections within TREC” (Paşca 2003).

Related sub–tasks to text retrieval were organized into separate task experiments, also called *tracks*. Since 1992, there were 21 different related tracks, like *Ad hoc*, *filtering*, *NLP*, *Speech*, *Cross–language*, *Novelty* or the *Question Answering* track.

In the following, we want to shortly describe the different tasks from TREC 2006. These definitions can be found on the TREC homepage[17].

[17]`http://trec.nist.gov`

Blog track The purpose of the blog track is to explore information seeking behavior in the blogosphere.

Enterprise track The purpose of the enterprise track is to study enterprise search: satisfying a user who is searching the data of an organization to complete some task.

Genomics track The purpose of the track is to study retrieval tasks in a specific domain, where the domain of interest is genomics data (broadly construed to include not just gene sequences but also supporting documentation such as research papers, lab reports, etc.).

Legal track The goal of the legal track is to develop search technology that meets the needs of lawyers to engage in effective discovery in digital document collections.

Question Answering track A track designed to take a step closer to information retrieval rather than document retrieval.

SPAM track The goal of the SPAM track is to provide a standard evaluation of current and proposed Spam filtering approaches, thereby laying the foundation for the evaluation of more general email filtering and retrieval tasks.

The *Question Answering* track has been part of TREC since 1999 (TREC–8). Over the years, the task has often changed, mostly concerning the question set, the test collection or the kind of answers.
So, when starting the track in 1999, the collection contained about 528.000 newspaper and newswire articles. The answers were complete strings limited to a length of 50 or 250 bytes. The biggest change in this paradigm was at TREC 2002 (Voorhees 2002), when a new document collection was introduced. Now, instead of (up to five) answer strings of a specific size, the system's answer had to be one exact answer.
The new test collection, the *AQUAINT* corpus[18] (Graff 2002) contains approximately 1.033.000 newswire and newspaper articles with roughly 375 million words from three different sources: the Xinhua News Service (People's Republic of China), the New York Times News Service, and the Associated Press Worldstream News Service. The *AQUAINT* collection is mainly used for the Q&A track.

[18]`http://www.ldc.upenn.edu/Catalog/docs/LDC2002T31/`

```
<target id = "154" text = "Christopher Reeve">
    <qa>
        <q id = "154.1" type="FACTOID">
            What year was Christopher Reeve paralyzed?
        </q>
    </qa>

    <qa>
        <q id = "154.2" type="FACTOID">
            How many "Superman" movies did he make?
        </q>
    </qa>

    <qa>
        <q id = "154.3" type="FACTOID">
            During what years were these "Superman" movies made?
        </q>
    </qa>

    <qa>
        <q id = "154.4" type="FACTOID">
            Which actress co-starred in the most "Superman" movies with Reeve?
        </q>
    </qa>

    <qa>
        <q id = "154.5" type="FACTOID">
            What year did Reeve commence his theatrical career?
        </q>
    </qa>

    <qa>
        <q id = "154.6" type="LIST">
            List titles of movies, other than "Superman" movies, that Christopher Reeve acted in.
        </q>
    </qa>

    <qa>
        <q id = "154.7" type="OTHER">
            Other
        </q>
    </qa>
</target>
```

Figure II.10: Example of TREC Q&A questions for target 154.

The TREC–2002 question set contained 567 questions divided into 75 different targets. Each target corresponds to a specific topic and contains several questions asking for factoid, list or definitional questions about this topic. For example, Figure II.10 shows the set of questions for the target 154, *Christopher Reeve*.

After a question set is released by TREC, each participating group has a specific amount of time (normally one week) to fully automatic answer the questions and return them to NIST. Then, the answers are manually evaluated by NIST employed assessors. They judge an answer whether it is

globally correct , which means that the answer is completely correct

locally correct , which means that the answer is correct, but the collection contains a contradictory answer that the assessor believes is better suited

non–exact , which means that the answer is included in the returned string, but it contains also more information

unsupported , which means that the answer is correct, but the document, where the answer is extracted from is not correlated with the question and

incorrect , which is simply the wrong answer.

Chapter III

Background to Statistical Language Modeling

This thesis studies the impact of *Statistical Language Models* (SLM) in the context of *Question Answering* (Q&A). Hence, we want to give an introduction of these models in this chapter. SLM are a well–known mechanism originally developed for the task of speech recognition (Jelinek 1997). In 1998, Ponte and Croft (1998) suggested to use SLM also for information retrieval (IR) and so constituted a new, highly accepted paradigm to search information in a mathematically well founded way.

The rest of this chapter is organized as follows. First, we want to give a short introduction of general language models, including advantages and disadvantages. In section III.2, the approach for using SLM in IR is discussed as well as some further applications for the task are shown. The last section III.3 shortly presents how we introduced the language modeling approach to the task of question answering.

1 Statistical Language Models

As mentioned in the introduction, this section wants to introduce statistical language models. In principle, a SLM is a probability distribution over a text sequence. This means, it can calculate the likelihood of a given string in a given context, like a specific language. For example, consider the following probabilities for different sequences of words

- $P_1($“This is a thesis”$)$

- P_2(“Thesis this a is”)
- P_3(“This is a bottle”)
- P_4(“Dies ist eine Doktorarbeit”)

This example shows, that the selection of the ”correct“ probability strongly correlates to the language or the topic of the task we are modeling. For most of the applications described in this thesis, a *Bag–Of–Word* approach is used where $P_1 = P_2$.
But, such a model of ”language of the kind familiar from formal language theory can be used either to recognize or to generate strings“ (Manning, Raghavan, and Schütze 2007). Thus, this kind of probabilistic mechanism is also called *Generative Model*.
But SLM are also generative. For example, using a language model decision boundaries can be determined. So, it is possible to discriminate if a specific document is relevant to an information need or if a question belongs to a specific class. In later chapters, we will utilize this characteristic to model different tasks for our Q&A approach.
Originally, the idea of using SLM comes from the task of speech recognition. The Bayes classifier

$$\hat{W} = arg \max_{W}(P(A|W)P(W)) \tag{III.1}$$

describes the approach of speech recognition, where W is a possible text string, $\hat{W}$ is the optimal recognized piece of text, $P(A|W)$ is the acoustic model and $P(W)$ is the language model. The role of SLM in this context is to act as a kind of grammar. This includes the differentiation of similar–sounding phrases, like

- A woman, without her man, is nothing.
- A woman: without her, man is nothing.

This means, the language model provides the likelihood of a word given a specific history, i.e. the likelihood of the word ”house“ given the string sequence ”The white“.
For the better understanding of SLM Figure III.1 illustrates a general ”Source–Channel Framework“, which can easily describe the different applications of language models. This framework was originally presented by Shannon (1948) as a *Model of Communication Systems*.

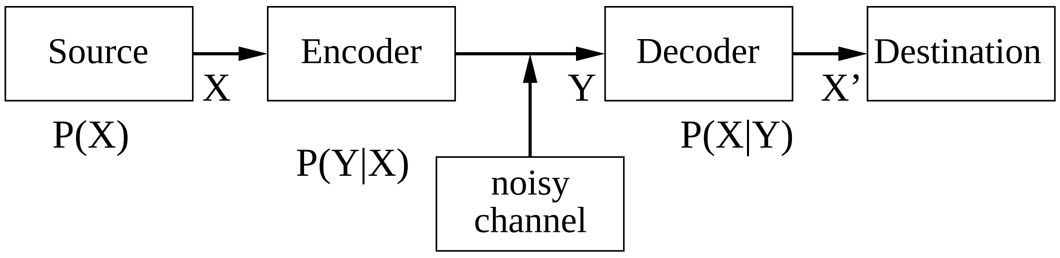


Figure III.1: The Model of Communication System.

Following Zhai (2005) a source produces X with a specific probability $P(X)$. Then, X is moved though an encoder and a noisy channel, which results in Y with the probability $P(Y|X)$. Finally, a decoder should reproduce X' with the likelihood $P(X|Y)$.
Hence, the best possible X' should reach the destination. This can be modeled by the Bayes classifier

$$X' = \text{argmax}_X P(X|Y) \tag{III.2}$$

This classifier can be reformulated using *Bayes rule* (cf. Manning and Schütze (1999)).

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{III.3}$$

Applying the rule to formula III.2, it results in

$$X' = \text{argmax}_X P(Y|X)P(X) \tag{III.4}$$

where $P(Y)$ can be disregarded, because it is independent to X.
Now, if X is specified as text, $P(X)$ is called a language model.

X	Y	**Example Application**
word sequence	speech signal	speech recognition
English sentence	Chinese sentence	machine translation
summary	document	summarization
document	query	information retrieval

Table III.1: Examples of applications for Source–Channel Framework.

Zhai (2005) also gives some examples for different applications. Table III.1 shows four different interpretations of X and Y along with the adequate application. Hence, the tasks of speech recognition, machine translation, text summarization and information retrieval can be explained.

Using SLM in real applications like information retrieval, speech recognition or machine translation has advantages and disadvantages. Lavrenko (2003) give a good overview of the problems. According to them, advantages of using language models, especially for the task of information retrieval, are

- a formal mathematical model
- a simple, well–understood framework
- the integration of both indexing and retrieval models
- a natural use of collection statistics (no heuristics)
- avoiding tricky issues ”relevance“, ”aboutness“, etc.

On the other hand, using SLM also brings some disadvantages, like

- difficulty to incorporate notions of ”relevance“ or user preferences
- the concept of relevance feedback or query expansion are not straightforward
- the accommodation of phrases, passages and Boolean operators is not easy

The task of extending, or building more sophisticated SLM tries to solve some of the problems described above. We will also describe some extended language models for information retrieval in the next chapter, i.e. find a solution for the problem of having no adequate concepts for relevance feedback.

The next important question is how to calculate the probability of a complete sequence of n words $P(w_1 \ldots w_n)$. Normally, there is not enough training data to estimate such probabilities. Hence, the solution is to decompose the sequence into smaller parts using the *chain rule*. This results in

$$P(w_1 \ldots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2w_1)\ldots . \tag{III.5}$$

But, also some of the probabilities in formula III.5 are still to complicated to train. Therefore, the *Markov Assumption*

$$P(w_i|w_1 \dots w_{i-1}) = P(w_i|w_{i-M+1} \dots w_{i-1}) \tag{III.6}$$

is made to overcome the problem. This means the history $w_1 \dots w_{i-1}$ is shortened to the $M-1$–word history $w_{i-M+1} \dots w_{i-1}$. Hereafter, this history is called h_M or simply h if we don't want to define an M. After shortening the history, we call such models *M–gram Model* (Klakow 2003).

The simplest way of estimating such a language model is to simply ignore the history. These models are called *unigram language models*:

$$P_{\text{unigram}}(w_1 \dots w_n) = P(w_1)P(w_2) \dots P(w_n) \tag{III.7}$$

Zerogram	$P(w_1\|w_i \dots w_{i-1}) = \frac{1}{\|V\|}$
Unigram	$P(w_1\|w_i \dots w_{i-1}) = P(w_i)$
Bigram	$P(w_1\|w_i \dots w_{i-1}) = P(w_i\|w_{i-1})$
Trigram	$P(w_1\|w_i \dots w_{i-1}) = P(w_i\|w_{i-2}w_{i-1})$

Table III.2: Examples of popular language models.

So, the words are independently generated from each other. This fact is also known as *Independence Assumption*. Although words are normally not independently generated in documents or questions, most existing approaches for information retrieval use this fact to model their systems, i.e. Zhai and Lafferty (2001).

Because the order of words in this model is irrelevant, it is also called a *Bag of Words* model (Manning, Raghavan, and Schütze 2007). In particular, it is a multinomial distribution over words, where a piece of text can be regarded as a sample drawn according to this word distribution (Zhai 2005).

But there are also some more difficult types of SLM. For example, the *Bigram language model* uses a one–word history, and therefore is defined by

$$P_{\text{bigram}}(w_1 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2) \dots P(w_n|w_{n-1}) \tag{III.8}$$

Table III.2 gives a short overview of popular language models used for different applications. A special case is the *Zerogram*. Here, the uniform distribution $\frac{1}{|V|}$ is used to estimate the language model, where $|V|$ is the size of the vocabulary.

Further, more sophisticated approaches, like *Remote–dependency language models* , i.e. the *Maximum Entropy model*, or *Structured language models* are discussed later or can be found in Jelinek (1997) or Manning and Schütze (1999).

A measure of quality for SLM is the *Perplexity*. It is defined by

$$\mathrm{PP} = P(w_1 \ldots w_n)^{-\frac{1}{n}} = \exp(-\sum_{wh} f(wh) \log P(w|h)) \tag{III.9}$$

where n is the number of words in the corpus, h is the history, $f(wh)$ is the relative frequency of the word w after history h, and $P(w|h)$ is the language model.
When using a Zerogram with V–word vocabulary to calculate the perplexity, then $PP = V$. This means the perplexity defines the number of possible words. Considering an *M–gram model* it can be interpreted as the median number of possible completions after a given history. As a consequence, a lower perplexity results in a smaller error rate (Klakow 2003).

But normally, instead of minimizing the perplexity to obtain the most suitable language model, it is also possible to maximize the *Log–Likelihood*

$$F = -\log \mathrm{PP} = \sum_{wh} f(wh) \log P(w|h) \tag{III.10}$$

Maximizing the Log–Likelihood is done by using *Lagrange Multiplier* with the marginal constraint that probabilities are normalized, i.e. $\sum_w P(w|h) = 1 \forall h$. Solving this equation results in a very good estimator for conditional probabilities, the *Maximum Likelihood estimator*, defined by

$$P_{\mathrm{ML}}(w|h) = \frac{N(wh)}{N(h)} \tag{III.11}$$

where $N(wh)$ is the absolute number of word w with history h in the training collection.

But, the Maximum Likelihood estimator has one big disadvantage. The count of the word and a history $N(wh)$ may vanish. This will inevitably result in zero probabilities for this event, and for special applications, like information retrieval, in a zero probability for the complete ranking.
To show that the problem of unseen words cannot be compensated by using a larger vocabulary, we want to introduce the *Out–Of–Vocabulary rate* (OOV–rate)

$$\text{OOV–rate} = \frac{\#\text{OOV words in test corpus}}{\#\text{words in test corpus}} \tag{III.12}$$

An OOV word is defined as a word in the test corpus which is not covered by the training collection. For example, if the test corpus contains 3000 words and 108 unknown words, it results in: OOV–rate $= \frac{108}{3000} = 0.036 = 3.6\%$.
Following Klakow (2003), the OOV–rate decreases approximately by $1/$size of vocabulary. Hence, the problem of unseen events is a problem in principle and cannot be resolved by using more data.

Absolute Discounting

To overcome the problem of *Zero Probabilities*, smoothing has to be done to allocate a non–vanishing probability to those events.
In principle, the idea is very easy. There is a need to find a further probability distribution $P_{\text{BG}}(w|h)$ with non–vanishing values for w and h. This normalized function can simply be added to the original distribution, with a given weight $\alpha(h)$. The resulting *Backing–Off* distribution is called *Absolute discounting*. It is defined by

$$P(w|h) = \begin{cases} \frac{\mathrm{N}(wh) - d}{\mathrm{N(h)}} + \alpha(h) \cdot P_{\text{BG}}(w|h) & \text{if N(wh)} > 0 \\ \\ \alpha(h) \cdot P_{\text{BG}}(w|h) & \text{else} \end{cases} \tag{III.13}$$

where d is the *discounting parameter* and $P_{\text{BG}}(w|h)$ is the backing–off distribution, with $\sum_w P_{\text{BG}}(w|h) = 1$.
A very popular choice for a Bigram language model $P(w_i|w_{i-1})$ is the Unigram $P_{\text{BG}}(w|h) = P(w_i)$. But, this choice is not necessarily optimal. Please read Kneser and Ney (1995) to see, how optimal values for $\alpha(h)$, d, and $P_{\text{BG}}(w|h)$ can be computed.

Add Epsilon

There are many other smoothing techniques than *Absolute Discounting* for various different tasks. The most simple one is the *Floor Discounting*, or *Add Epsilon*. In this approach, a small value ε is added to each count. After normalization, we get

$$P(w|h) = \frac{N(wh) + \varepsilon}{N(h) + V \cdot \varepsilon} \tag{III.14}$$

For large vocabulary V, the denominator term $V \cdot \varepsilon$ can be problematic and introduce noise to the probability. But due to the simplicity and the non–powerful performance of the model, it is mostly used for taskes where quality is not important (Klakow 2003).

Linear Interpolation

The next smoothing method we want to introduce is *Linear Interpolation*, which is still very popular for the task of information retrieval. This smoothing technique was first introduced by Jelinek and Mercer (Ney, Essen,and Kneser 1994), and is therefore also referred to *Jelinek–Mercer* smoothing. It is defined by

$$P(w|h) = (1 - \varepsilon)P_{ML}(w|h) + \varepsilon P_{BG}(w|h) \tag{III.15}$$

There is also a variant presented by IBM, where the ε depends on the history h. But this model is not so popular, because there are many variables, which are difficult to estimate. In IR, ε is normally determined by searching the complete parameter space. This model works very good for longer queries (Zhai and Lafferty 2001).

Dirichlet Smoothing

A multinomial distribution, in particular a Bayesian smoothing, with a Dirichlet distribution as the conjugate prior for estimating language models results in the *Dirichlet Smoothing*

$$P(w|h) = \frac{N(wh) + \mu P_{BG}(w|h)}{\sum_w N(wh) + \mu} \tag{III.16}$$

This variant is also very popular for the task of information retrieval when using shorter queries (Zhai and Lafferty 2001). As shown in the next chapters, best results are obtained by using this smoothing technique for question answering.

Maximum Entropy Models

”*Maximum Entropy* (ME) modeling is a framework for integrating information from many heterogeneous information sources for classification. The data for a classification problem are described as a (potentially large) number of features. These features can be quite complex and allow the experimenter to make use of prior knowledge about what types of information are expected to be important [. . .]“ (Manning and Schütze 1999). This means that first the training collection is analyzed and then an adequate model is created by maximizing the entropy using the features as constraints which have to be satisfied by the model. The idea behind maximizing the entropy of a model is ”motivated by the desire to preserve as much uncertainty as possible“ (Manning and Schütze 1999).

In this subsection, we want to show this idea on the basis of an example, before presenting the general definition of maximum entropy.

The example is taken from Klakow (2003) and shows the creation of a Bigram language model. To simplify matters, compound probabilities are used as notation, but the transfer to conditional probabilities should be very easy.

So, the probability $P(w_1w_2)$ should be calculated. The training corpus should be relatively small, so Unigrams can be properly estimated. This results in:

$$\sum_{w_1} P(w_1w_2) = P(w_2) \text{ and } \sum_{w_2} P(w_1w_2) = P(w_1) \tag{III.17}$$

Now, the problem is that there are $2|V|$ conditions given by the two formulas. But there are $|V|^2$ free parameters for estimation. So, additional postulations have to be made. By maximizing the entropy

$$H = -\sum_{w_1w_2} P(w_1w_2)\log P(w_1w_2) \tag{III.18}$$

the hope is to include as little bias as possible.

Now, both constraints are included using Lagrange multipliers. So, the equation

$$\begin{aligned} H' &= -\sum_{w_1w_2} P(w_1w_2)\log P(w_1w_2) \\ &\quad + \sum_{w_2} \lambda(w_2)(\sum_{w_1} P(w_1w_2) - P(w_2)) \\ &\quad + \sum_{w_1} \lambda(w_1)(\sum_{w_2} P(w_1w_2) - P(w_1)) \end{aligned}$$

has to be derived by the Bigram probability.
After solving all following equations, the result of the maximum entropy problem is

$$P(w_1 w_2) = P(w_1)P(w_2) \tag{III.19}$$

Hence, as long as there is no given correlation for the constraints, maximizing the entropy means a postulation of independence.

This was just an example, but now we want to generalize the notion of linear constraints. A common equation of general constraints is given by

$$\sum_{wh} f_S(wh)P(wh) = K_S \tag{III.20}$$

where K_S is the required constraint and $f_S(wh)$ is the important feature function

$$f_S(wh) = \begin{cases} 1 & \text{if (wh)} \in S \\ 0 & \text{else} \end{cases} \tag{III.21}$$

which defines the special kind of a constraint.
Common applications of this model are i.e. text classification (Manning and Schütze 1999), or in particular for the task of Q&A, the answer extraction. The answer extraction module in the *Alyssa* system also uses a maximum entropy model to select and extract possible answer candidates (Shen and Klakow 2006).

The Kullback–Leibler Divergence

Cover and Thomas (2001) writes, that it is also possible to minimize the *Kullback–Leibler Divergence* (KL–divergence) instead of maximizing the entropy. The formula of the KL–divergence, which should be minimized to a given function $\pi(wh)$, is given by

$$D(P||\pi) = \sum_{wh} P(wh) \log \frac{P(wh)}{\pi(wh)} \tag{III.22}$$

This equation results in the general ME approach, if the function $\pi(wh)$ is equally distributed. This approach is a real generalization and the solution of this kind of functions is called *Log–Linear Models*.

Log–Linear Models

As mentioned above *Log–Linear models* are the solution of the generalized form of maximum entropy models. One can show that this solution is unique and that there exists an algorithm which converges to it. This *Generalized Iterative Scaling* algorithm is explained in detail in Manning and Schütze (1999).
Now, if we minimize the KL–divergence $D(P||\pi)$ with the linear constraints

$$\sum_{wh} f_S(wh)P(wh) = K_S \tag{III.23}$$

it results in

$$P(wh) = \pi(wh)\mu \prod_S \mu_S^{f_S(wh)} \tag{III.24}$$

When taking the logarithm of this solution, the formula

$$\log P(wh) = \log(\pi(wh)\mu) \sum_S f_S(wh) \log \mu_S \tag{III.25}$$

is linear. That is why this approach is called log–linear model. For proving the existence of the solution, please read Darroch and Rattcliff (1972). The use of language models for log–linear interpolation was first introduced by Klakow (1998).
We also used this kind of model for doing question answering, in particular for classifying question types.

Class–Based Language Models

The last models we want to introduce are the *Class–Based Language Models*. Occasionally it might be that ”normal“ language models are too detailed. For example, a language model for planes of different colors should be modeled, i.e. *The * plane*, where ”*“ is *red, blue, yellow*, etc. Then, the color could be modeled by a class.
Hence, given a set of classes $C = c_1, \ldots, c_n$ and a unique mapping

$$V \mapsto C \tag{III.26}$$

$$w_i \mapsto c(w_i) \tag{III.27}$$

a class–based language model is for a word–M–gram is defined by

$$\begin{aligned}
P(w_i|w_{i-1}\dots w_{i-(n+1)}) &= P(w_i c(w_i)|w_{i-1}\dots w_{i-(n+1)}) \\
&= P(w_i|c(w_i)w_{i-1}\dots w_{i-(n+1)}) \cdot P(c(w_i)|w_{i-1}\dots w_{i-(n+1)}) \\
&\approx P(w_i|c(w_i)) \cdot P(c(w_i)|c(w_{i-1})\dots c(w_{i-(n+1)}))
\end{aligned}$$

where $P(w_i|c(w_i))$ is called *Emission Probability* and $P(c(w_i)|c(w_{i-1})\dots c(w_{i-(n+1)}))$ is called *Transitional Probability*.

The advantages of using class–based language models are that there are fewer free variables to estimate and hence, are easier to train. For example, if a M–gram has $S_W = |V|^M - 1$ free parameters, a class–based language model just has $S_C = (|C|^M - 1) + (|V| - |C|)$ free parameters to estimate. The first term is derived from the transition probabilities and the second term comes from the emission probability.
This means for given values $V = 64000$, $C = 1000$, and $M = 3$, it results in $S_W = 2.6 \cdot 10^{14}$ and $S_C = 10^9$.

To obtain an estimation of the emission probability $P(w|c(w))$, we maximize the log–likelihood, defined by

$$F = \sum_{wh} f(wh) \log P(w|c(w))P(c(w)|c(h)) \tag{III.28}$$

with the normalization as constraints

$$\sum_{w_i \in c(w)} P(w_i|c(w)) = 1 \qquad \forall w \tag{III.29}$$

which are included using Lagrange multipliers. As a final result, the maximum likelihood estimation defines the emission probability

$$P(w|c(w)) = \frac{f(w)}{\sum_{w_i \in c(w)} f(w_i)} \qquad \forall w \tag{III.30}$$

where $f(w)$ is the relative frequency of word w in the corpus.

An analogous calculation is done to obtain the transition probabilities.

2 Language Modeling for Information Retrieval

In 1998, Ponte and Croft (1998) proposed a new technique for doing information retrieval using statistical language models (SLM). This approach combines all advantages and disadvantages we presented in the last section and is still a very popular alternative to the standard retrieval mechanism we discussed in chapter II.

Section III.1 shows in detail how language models are estimated and the problems when using them. Now, we want to apply these models for the task of information retrieval, in particular how we can use SLM for ranking. There are two popular approaches, the *Document Likelihood* and the *Query Likelihood*.

Document Likelihood

In the document likelihood, a language model for each question Q is estimated. The ranking itself is to calculate the likelihood of a given document D of being a random sample from this model. Hence, the SLM ”predicts“ a typical relevant document (Lavrenko 2003). The formula

$$P(D|Q) = \prod_{w \in D} P(w|Q) \tag{III.31}$$

shows this fact. Additionally, the term independence assumption is used to calculate the product of each word in the document.

Following Lavrenko (2003), there are some problems in using document likelihood for calculating the rank of a document. For example, the probabilities are comparable when using documents of different length. The model also prefers documents with frequent, perhaps short function words. This fact becomes much more apparent when regarding the document with the highest rank

$$\max_D \prod_{w \in D} P(w|Q) = \max_{w \in D} P(w|Q) \tag{III.32}$$

There are many approaches to fix the document likelihood, but because it is not standard for doing information retrieval or question answering, we don't want to go into detail. Please read Zhai and Lafferty (2001b) or Lavrenko (2003) for further information about this kind of model.

Query Likelihood

On the other hand, the standard approach for ranking documents using language models is the query likelihood. It is the opposite of the document likelihood and defined by the probability of

$$P(Q|D) \tag{III.33}$$

So, a language model for every document in the collection is calculated. It is derived from the document likelihood by using Bayes rule (formula III.3).

$$P(D|Q) \propto P(Q|D)P(D) \tag{III.34}$$

where the term $P(Q)$ in the denominator is neglected because it is independent from the document D. Normally, the prior probability $P(D)$ is regarded as a uniform distribution and therefore it is also omitted.

When applying the independence assumption, we explained in the last section, it results in

$$P(Q|D) = \prod_{q \in Q} P(q|D) \tag{III.35}$$

This means, the documents are ranked by their ”ability“ to generate a query.

Lavrenko (2003) also describe some drawbacks of this approach. For example, they explain, that there is no notion of relevance in the model and that everything is just randomly sampled. Other flaws are that user feedback and query expansions are not part of this model, and that it does not directly allow weighted or structured queries.

These disadvantages are correct for the simple approach. However, there are more sophisticated models which try to exactly overcome these problems. Some of these approaches are described in chapter V.

Multinomial and Multiple–Bernoulli Models

”Although the seminal approach to introduce language modeling in information retrieval was based on a multiple–Bernoulli distribution (Ponte and Croft 1998), the predominant modeling assumption is now centred on multinomial models. Scoring is simpler in multinomial models, and, basically, there is no much evidence giving good reasons to choose multiple–Bernoulli over multinomial in general“ (Losada 2005).

The multinomial approach is defined by

$$P(q_1 \dots q_n|D) = \prod_{i=1}^{n} P(q_i|D) \qquad \text{(III.36)}$$

In this model, the fundamental event is the identity of the i’th query token. Hence, the observation is a sequence of events, one for each query token (Lavrenko 2003). Most of current state–of–the–art approaches use this model.

A multiple–Bernoulli model, as the model defined by Ponte and Croft (1998) for the task of information retrieval is described by

$$P(q_1 \dots q_n|D) = \prod_{w \in q_1 \dots q_n} P(w|D) \cdot \prod_{w \notin q_1 \dots q_n} (1 - P(w|D)) \qquad \text{(III.37)}$$

Compared to the multinomial model, here, the fundamental event is the occurrence of the word w in the query. This means, that the observation is a vector of binary events, one for each possible word (Lavrenko 2003). This model is just used to introduce language models for information retrieval (Ponte and Croft 1998, Ponte 1998). Losada (2005) compares both models for the task of sentence retrieval and concludes that for this task, multiple–Bernoulli models are better suited because of the short length of sentences.

Although both models are similar, they are fundamentally different. Following Lavrenko (2003), both models assume word independence, but the meaning of this fact is different. Also, both use smoothed probability estimations, but in a completely different event space.

So, the advantages of multiple–Bernoulli models are that they are arguably better suited to information retrieval. This is mainly due to the check for presence or absence of a query

term. Also, there are no issues with observation length.
Lavrenko (2003) describe the advantages of multinomial models as the account for multiple word occurrences in a query. These models are also well understood because of the research which is done in this area. This enables the integration of such models with *Automated Speech Recognition, Machine Translation* and *Natural Language Processing*, as well.

Smoothing

To calculate the probabilities of a language model, the maximum likelihood estimator is used

$$P_{\text{ML}}(q|D) = \frac{N(q,D)}{\sum_q N(q,D)} \tag{III.38}$$

A detailed description of this estimator is given in section III.1. Again, there is a problem when calculating the likelihood for unseen events. To overcome this problem of ”zero probabilities“, smoothing is done as described in the last section. Here, mainly *Jelinek–Mercer*, *Dirichlet*, and *Absolute Discounting* is used to smooth these probabilities.

Smoothing and the TF–IDF weighting

Hence, smoothing plays also a very important role for the task of information retrieval or question answering. To make this fact more clear, Zhai and Lafferty (2001) describe the connection between smoothing and the tf–idf weighting scheme.
The assumption for understanding the connection is that smoothing methods use two distributions. The first model is used for ”seen“ events ($P_s(w|d)$), and the second one models the distribution of ”unseen“ words ($P_u(w|d)$).
Now, the probability of a query q given a document d can be reformulated as

$$\begin{aligned}
\log P(q|d) &= \sum_i \log P(q_i|d) \\
&= \sum_{i:N(q_i,d)>0} \log P_s(q_i|d) + \sum_{i:N(q_i,d)=0} \log P_u(q_i|d) \\
&= \sum_{i:N(q_i,d)>0} \log \frac{P_s(q_i|d)}{P_u(q_i|d)} + \sum_i \log P_u(q_i|d)
\end{aligned}$$

where $N(w, d)$ means the count of word w in document d.

"The probability of an unseen word is typically taken as being proportional to the general frequency of the word, e.g., as computed using the document collection. So, let us assume that $P_u(q_i|d) = \alpha_d P(q_i|C)$, where α_d is a document–dependent constant and $P(q_i|C)$ is the collection language model. Now we have

$$\log P(q|d) = \sum_{i:N(q_i,d)>0} \log \frac{P_s(q_i|d)}{\alpha_d P(q_i|C)} + n \log \alpha_d + \sum_i \log P(q_i|C) \qquad \text{(III.39)}$$

where n is the length of the query. Note that the last term on the righthand side is independent of the document d, and thus can be ignored in ranking" (Zhai and Lafferty 2001).

The rest of equation III.39 can be split up into two parts. The first one describes a weight for each matched term in document and query, and the second term introduces document–dependent constant, which measures how much probability mass will go into the collection model.

The numerator of the first term, $P_s(q_i|d)$, can be regarded as the tf–weighting.

"Thus, the use of $P(q_i|C)$ as a reference smoothing distribution has turned out to play a role very similar to the well–known idf. The other component in the formula [...][is] playing the role of document length normalization, which is another important technique to improve performance in traditional models.

The connection just derived shows that the use of the collection language model as a reference model for smoothing document language models implies a retrieval formula that implements tf–idf weighting heuristics and document length normalization. This suggests that smoothing plays a key role in the language modeling approaches to retrieval" (Zhai and Lafferty 2001).

Improved Smoothing Techniques

Finally, we want to introduce two improved smoothing techniques we used for our experiments in language model based question classification (cf. section VI.1).

The first model presented in this section is called *Improved Absolute Discounting*, or simply *UniDisc*. It was introduced by Klakow (2006b) for the task of smoothing very small adaptation corpora. It is mainly based on absolute discounting smoothing

$$P(w|c) = \begin{cases} \frac{\mathrm{N}(w,c) - d}{\mathrm{N}} + \alpha P_{BG}(w|c) & \text{if } \mathrm{N}(w,c) > 0 \\ \\ \alpha \cdot P_{BG}(w|c) & \text{else} \end{cases} \tag{III.40}$$

where $N(w,c)$ is the frequency of term w in combination with a class c, d is the discounting parameter, and $P_{\text{BG}}(w|c)$ is the collection language model, whereas α denotes the backing–off weight. Equation III.40 itself describes a unigram backing–of model as defined in section III.1.

In a ”normal“ absolute discounting model, the discounting parameter d is independent of the term frequency. But, for our improved absolute discounting, we want to present a d which depends on term counts. It is defined by the rational function

$$d(N) = \frac{d_0 + s(N-1)}{1 + g(N-1)} \tag{III.41}$$

where d_0 is the original, count independent discounting parameter, whereas s and g are additional parameters.

The second smoothing method we improved for our experiments is based on the log–linear language models, presented in section III.1. The model is defined by

$$P(w_i|w_{i-1}c) = \tfrac{1}{Z_\lambda(w_{i-1}c)} \quad P_{UniDisc}(w_i|c)^{1.0} \\ P_{AbsDisc}(w_i|w_{i-1}c)^{\lambda} \tag{III.42}$$

where $\frac{1}{Z_\lambda(w_{i-1}c)}$ is a normalization weight depending on w_{i-1}, c and the parameter λ. The model $P_{UniDisc}(w_i|c)$ is the improved absolute discounting, we described above and $P_{AbsDisc}(w_i|w_{i-1}c)^{\lambda}$ is a ”normal“ absolute discounting model using bigram statistics. The parameter λ denotes the interpolation weight.

Further sophisticated smoothing methods, like the *Translation Model*, are described in chapter V.

3 Language Modeling for Question Answering

In this section we want to provide a theoretical background on language model based question answering for TREC question sets. Therefore, the task is considered as a classification problem. In particular, for a given question Q_i of a series of TREC questions $Q_1 \dots Q_N$ on

a topic T, the answer A_i is sought–after. Because a valid answer at TREC should also be supported, a sentence S_i and a document D_i has to be provided.
Hence, the classification task is defined by

$$(\hat{A}_i, \hat{S}_i, \hat{D}_i) = \mathrm{argmax}_{A_i, S_i, D_i} P(A_i, S_i, D_i | Q_i, T, Q_1 ... Q_{i-1} A_1 ... A_{i-1}) \tag{III.43}$$

where $A_1 \dots A_{i-1}$ are the answers to previous questions. In the Q&A system described in this thesis, the dependency on previous questions is ignored, but future work should model this fact to avoid duplicate answers and to increase the performance.
So, a simplified classification problem is

$$(\hat{A}_i, \hat{S}_i, \hat{D}_i) = \mathrm{argmax}_{A_i, S_i, D_i} P(A_i, S_i, D_i | Q_i, T) \qquad . \tag{III.44}$$

Now we can decompose the probability

$$P(A_i, S_i, D_i | Q_i, T) = P(A_i | S_i, D_i Q_i, T) P(S_i | D_i Q_i, T) P(D_i | Q_i, T) \tag{III.45}$$

Here, $P(D_i|Q_i, T)$ is the statistical language model for document retrieval, $P(S_i|D_i Q_i, T)$ is a model for sentence retrieval and $P(A_i|S_i, D_i Q_i, T)$ models the part of answer extraction.

In speech recognition, a similar decomposition is done. But for this task, only two terms are necessary: the acoustic model and the language model (Jelinek 1997). As the probabilities of the two models in speech recognition have different dynamic ranges, it is helpful to introduce some exponents to compensate this flaw. The same approach can be used for the task of Q&A, although one more term is concerned. But this just results in one more exponent, as shown in the following equation:

$$P(A_i, S_i, D_i | Q_i, T) \approx P(A_i | S_i, D_i Q_i, T)^{\alpha} P(S_i | D_i Q_i, T)^{\beta} P(D_i | Q_i, T)^{\gamma} \tag{III.46}$$

One of the three parameters is redundant and so, one of them can be set to one or all can satisfy that $\alpha + \beta + \gamma = 1$.
Instead of optimizing the last formula, it is also possible to optimize

$$\log P(A_i, S_i, D_i|Q_i, T) \approx \alpha \log P(A_i|S_i, D_iQ_i, T) + \beta \log P(S_i|D_iQ_i, T) + \gamma \log P(D_i|Q_i, T) \tag{III.47}$$

because of the monotony of the logarithm function. The result is a linear interpolation of the scores of the different components of a Q&A system. This equation also provides a strategy to fuse the scores of the different components.

The Document Retrieval Model

The document retrieval model $P(D_i|Q_i, T)$ is further decomposed using Bayes rule. The result is defined by

$$\text{argmax}_{D_i} P(D_i|Q_i, T) = \text{argmax}_{D_i} P(Q_i, T|D_i)P(D_i) \tag{III.48}$$

In addition, we assume a uniform prior $P(D_i)$ as explained in section III.1. The resulting term can be further converted to

$$P(Q_i, T|D_i) = P(Q_i|T, D_i)P(T|D_i) \approx P(Q_i|D_i)P(T|D_i) \approx P(Q_i|D_i)P(T|D_i)^\delta \tag{III.49}$$

where the first approximation assumes an independence of the question from the topic which may not be correct. The second approximation introduces an exponent to model the different dynamic range as already described above.

The Sentence Retrieval Model

For the sentence retrieval model, the question type is introduced

$$P(S_i|D_i, Q_i, T) = \sum_t P(S_it|D_i, Q_i, T) \approx \max_t P(S_i, t|D_i, Q_i, T) \tag{III.50}$$

The last approximation assumes that only a single question type is dominating the sum. This is equivalent to the assumption, that the probabilistic models always assign a question type with very little ambiguity.

Additionally, the sentence S_i is further decomposed into a tuple consisting of the set of named entity types which are present in the sentence. Also, the key words of the sentence are introduced, so $S_i = (NE_i, W_i)$. Therefore, the model $P(W_i, NE_i, t_i|D_i, Q_i, T)$ has to be estimated. This results in

$$\begin{aligned}
P(S_i|D_i, Q_i, T) &= \max_t P(W_i, NE_i, t|D_i, Q_i, T) && \text{(III.51)}\\
&= \max_t P(t_i|NE_i, W_i, D_i, Q_i, T) && \text{(III.52)}\\
&\quad \cdot P(NE_i|W_i, D_i, Q_i, T) && \text{(III.53)}\\
&\quad \cdot P(W_i|D_i, Q_i, T) && \text{(III.54)}
\end{aligned}$$

where the first term is a question type classifier depending on the named entities in a sentence. The second term is the result of a named entity recognizer and the last term is the score of the sentence retrieval module, i.e. the result of the module described in chapter IX.

The Answer Extraction Model

This model is in a sense the most complex one, because it incorporates many complex features derived from various sources like parse trees of the question and the answer candidate sentence. This problem is explained in detail in chapter II, and, for our Q&A system, we decided to use a maximum entropy approach to estimate the model parameters. As an implication, no further decomposition has to be done.

Chapter IV

Application

1 SmartWeb – Multi Language Question Answering

The *SmartWeb* project (Wahlster 2007) was funded by the Federal Ministry of Education and Research (BMBF[1]). The main goals are described in the SmartWeb project proposal (Smartweb 2007):
The World Wide Web (WWW) has dramatically simplified and accelerated the worldwide access to digitally saved information. But, there currently are two major barriers to access the information:

- The access to the contents is mainly optimized for PC's with large displays. Instead of a simple, intuitive access using natural language over ubiquitary mobile phones, information retrieval systems currently use textual content for searching. It is not possible to customize this content to a user using each possible modality.
- So far, the content of the Internet was just "readable" by machines, but not "understandable". Because Internet–based information is mostly presented in natural language, all retrieved documents are currently understandable by human users. Furthermore, despite of advanced retrieval and ranking techniques, results do often not match the user's information need.

The SmartWeb project tries to overcome these problems. It provides a user with a multimodal access to the semantic web. This includes written language as well as speech

[1]Bundesministerium für Bildung und Forschung.

recognition, optionally with touch screen applications. An example of such an application is asking a question like *What is the name of this player?* and parallely touching the specific player on the screen.

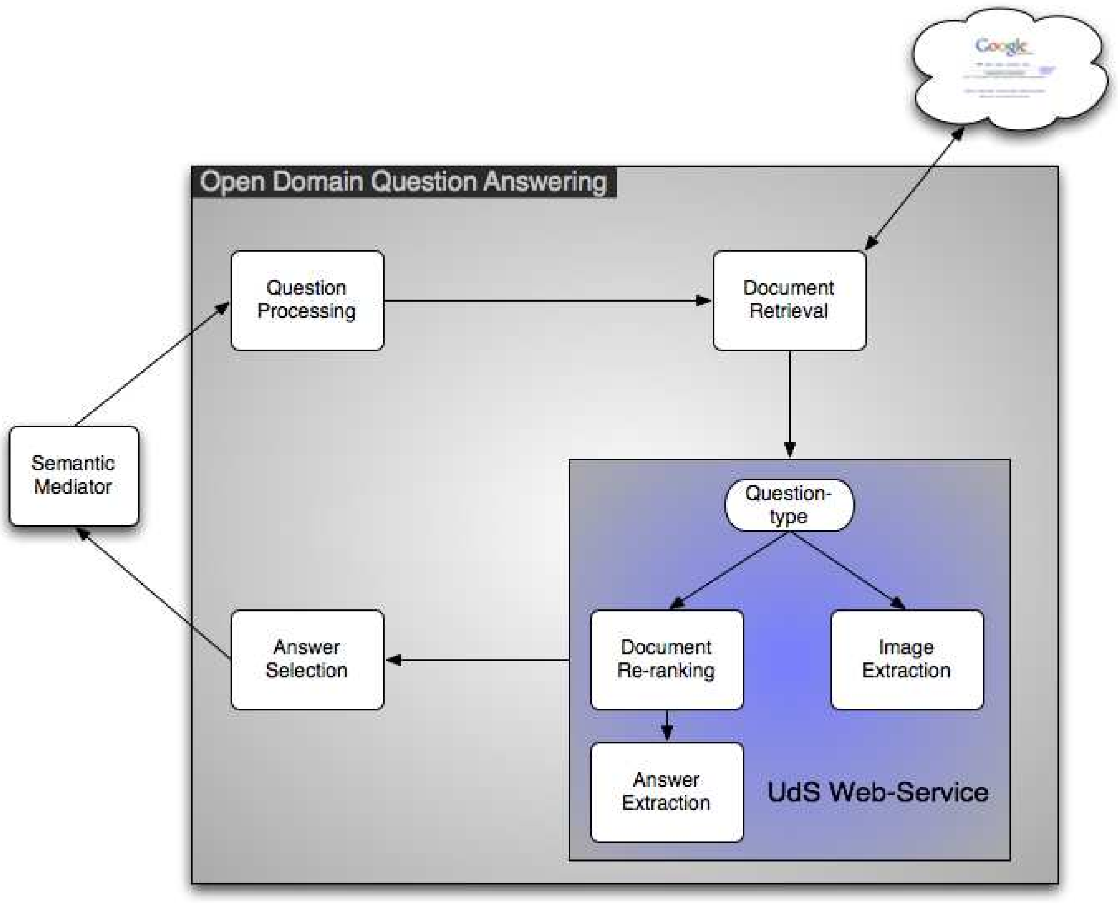


Figure IV.1: Overview of the Q&A system in SmartWeb

The main application of the SmartWeb project was the World Cup 2006 in Germany. So, most knowledge bases are optimized for this task. But, there is also a component included, which implements an open domain question answering system to cover all kinds of possible questions. This Q&A systems is a corporate work from the German Research Center for Artificial Intelligence (DFKI[2]), Siemens and Saarland University. It implements a multilingual question answering system for German and English language.

Figure IV.1 presents the overall architecture of the question answering system in SmartWeb. All included modules are realized as *Web Services* and for communication between them, a pre–defined XML form is used. Finally, this XML file will contain all information and results

[2]Deutsches Forschungszentrum für Künstliche Intelligenz.

for later use.

On the left–hand side, the *Semantic Mediator* is shown. This component gets the question from the speech recognizer and distributes it to various adequate modules, depending on the type of question. So, open domain questions are always routed to the Q&A system.

The next stage is the *Question Analysis*. In this step, DFKI implement algorithms to determine question– and answer type. For example, question types are *factoid*, *Definition* or *Abbreviation*, whereas the answer type specifies *Person*, *Location*, etc.

Besides this, the analysis of the best word chain, the query focus, all keywords from the natural language question, a list of important named entities and query optimized for Web search engines are provided and put to the XML file (Neumann and Sacaleanu 2005).

The *Document Retrieval*, which is also implemented by DFKI, uses the results from the question answering to find a set of relevant documents. Here, a standard Web search engine is used and a list of the top–N documents is added to the XML file, where top–N can be set offline.

The bottom right part of Figure IV.1 shows the Web Service developed at Saarland University. Here, first, the question type is checked because questions about images and figures are answered by an *Image Extraction* module developed at Siemens.

If the question asks for factoid information, the *Passage Retrieval* module downloads the documents retrieved in the last step, uses discourse algorithms to make passages out of the Web sites, and re–ranks them.

Finally, a language model based approach is used to find and extract relevant information (*Information Extraction*). When finishing this step, all necessary information is put on the XML file and sent to the *Answer Selection*.

The last part is again implemented by DFKI and selects the best suited answers from the list of possible answer candidates. The ranked list is then added to the XML file and sent back to the Semantic Mediator. From there, the answers are used by the dialog machine to produce an adequate answer to the user question.

Because the multilingual passage retrieval is a real application for the theoretical approaches described in this thesis, we want to explain this part a bit more in detail.

Figure IV.2 presents the data flow off the passage retrieval implemented in SmartWeb. As already explained, the Web Service communicates with the Semantic Mediator via a XML

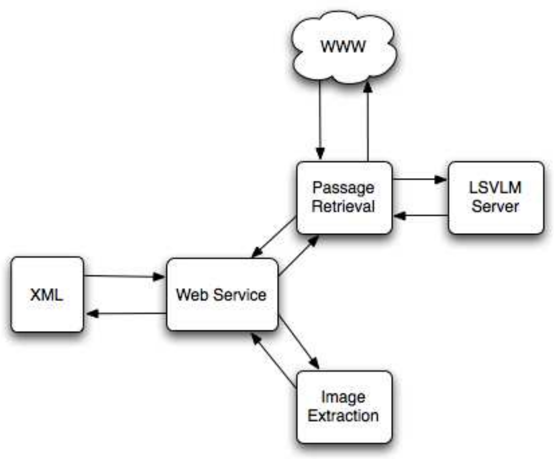


Figure IV.2: Overview of the passage retrieval component in SmartWeb

form. At this stage, the XML file contains the question type, which was added by the question analysis module. So, if the question asks for images, it is routed to the *Image Extraction* module and images with adequate captions are extracted using Web–based *PDF* files and Googles image search function.

If the question is of any other type, like *Location, Person, etc.*, it is routed to the passage retrieval core module. There, the Web documents are downloaded in parallel. At the same time, an internal timeout sees to it that the complete process finishes in an adequate time span which can be configured.

Then, passages are created out of the downloaded Web pages. Our first approach used a *Discourse Page Segmentation* algorithm for this task. This means, passages are created using the HTML syntax. The disadvantage of using such an implementation is that the size of the passages dramatically varies.

This leads us to our second approach. Now, the text segments are not specified by the HTML syntax, but each Web document is split up into single sentences. So, the number of sentences defining a passage can be declared in a configuration file. Using this approach, it is also possible to define *Sliding Windows*, which use a specific amount of sentence overlap, which can also be declared within the configuration file.

One big problem was the different encoding of Web pages. Some documents specify their encoding using Meta tags, but for most pages, the encoding is unknown. So, we had to implement a module which guesses the used encoding and transforms the complete text of a document to *UTF–8*.

Figure IV.2 also shows the connection of the Web Service with a separate, socket–based server for re–ranking the passages. We do this to obtain an optimal performance of the system, because this server uses the language modeling toolkit we developed at the department (LSVLM)[3].

Currently, the LSVLM server implements a *Jelinek–Mercer* linear interpolation using unigram statistics. We also implemented all optimization features described in chapter IX, like the simple query construction, stemming, using the Porter stemmer, dynamic stopword reduction, and the weighted expansion of the query and the passage.

Additionally, the weight of the document corresponding to the passage is included in the complete score of the retrieval engine.

Another advantage of using a server for doing passage re–ranking is that it can be easily replaced if a new server is available, which possibly implements better algorithms. It can also be reused by other applications by simply implementing the used socket connection.

Apllying these features, we also experimented with long–range models, like bigrams (Song and Croft 1999), and log–linear models (Klakow 1998). But non of them resulted in an increase in performance for this task.

To support the following *Answer Extraction* module, we did some preparation work when processing the text passage. So, the minimum and maximum distance between two query keywords in a text segment is computed and added to the outgoing XML form.

This means for example, if the query is *When did Napoleon die?*, the minimum and maximum distance in the passage between *When* and *Napoleon*, *When* and *die*, and *Napoleon* and *die* is computed.

This information is then later used to extract the best suited answer out of the set of possible answer candidates.

Finally, a special algorithm was implemented to process Web pages from the *Wikipedia* online encyclopedia. These documents turned out to be a good knowledge source for answering factoid questions. That is also the reason why many state–of–the–art question answering systems for research also use this source to answer questions or to verify that a given answer is correct (Shen et al. 2006).

[3]See chapter IX for more information about the LSVLM.

Chapter V

Other Related Work

As described in chapter II, a question answering (Q&A) system consists of many modules. Hence, there is also a lot of work done in this area. This chapter focuses on applications and approaches we did not mention in the single sections dealing with our experiments.

The first application presented in this chapter is a sophisticated smoothing technique for information retrieval known as *Two–Stage Smoothing* (Zhai and Lafferty 2002). Based on the problems, that the optimal parameter set of a smoothing method depends on the document collection and the query, and that these parameters are normally estimated using a brute–force search, Zhai and Lafferty (2002) proposed a new smoothing technique to overcome these problems. They recommended a general framework based on risk minimization (Zhai and Lafferty 2006) and the two–stage model as a special case (Zhai and Lafferty 2002).

In the first step of the model, Bayesian smoothing using Dirichlet priors is used to smooth the document collection, and in the second stage, this document language model is further interpolated with a query background model.

To automatically set the retrieval parameters, they also suggest a leave–one–out method to estimate the Dirichlet parameters and a mixture model to obtain the interpolation weights.

The *Risk Minimization Framework* mentioned above is presented in Lafferty and Zhai (2001) and Zhai (2002). Here, a framework is introduced that is able to combine both, document and query models. Hence, the ranking is done using a probabilistic function based on Bayesian decision theory. In particular, a language model for each document and a model for each query is estimated. The retrieval problem is then solved using risk minimization.

The advantage of this model is its generality. This means by choosing specific models and loss functions, other frameworks can be rebuild.

A very early work from Song and Croft in 1999 (Song and Croft 1999) proposes a more general language model for the task of information retrieval compared to the Ponte and Croft approach in 1998. Their models are based on a range of data smoothing techniques, which include curve–fitting functions, model combinations and the Good–Turing estimate (Manning and Schütze 1999).
The more interesting thing is that they move away from the term "independence assumption" and use N–grams to estimate language models instead of unigrams. In particular, experiments using bigrams and trigrams are presented, where word pairs are shown to work best for this task.

Berger and Lafferty (1999) as well as Jin, Hauptmann, and Zhai (2002) suggest the use of statistical machine translation techniques to solve the problem of information retrieval. The idea behind this proposal is simply to "translate" a document into a query. This means, the user's information need, as a fragment of an "ideal" document, is translated into a query. It can be regarded as generation process: the probability should be estimated that a query was generated by a translation of the document. So, the "translation" relationship between words in the query and words in a document can be directly modeled (Zhai 2005).

The training of these models is easy when relevance judgements are available. Then, the query and the documents are used to train the translation model.
But, when lacking this information, Berger and Lafferty (1999) use synthetic queries for a large document collection as training data, whereas Jin, Hauptmann, and Zhai (2002) use the title a pseudo query and document body to train their translation model.

Another sophisticated approach is to use document clusters to perform retrieval. *Cluster–based Retrieval* is based on the hypothesis that similar documents will match the same information needs (van Rijsbergen 1979). In a document–based retrieval system, documents are ranked given a specific query and a set of relevant documents is returned. By contrast, a cluster–based retrieval system groups documents by their similarity and returns a list of documents based on the clusters they come from.

Liu and Croft (2004) and Kurland and Lee (2004) both present language models for cluster–based retrieval. Whereas Liu and Croft propose models for ranking or retrieving clusters and for using clusters to smooth documents, Kurland and Lee show a variety of algorithms for integrating corpus similarity structure, modeled via clusters, and document specific information (Kurland and Lee 2004).

Language models with a smaller index, faster performance and better results than standard models are introduced by Hiemstra, Robertson, and Zaragoza (2004) with their work about *Parsimonious Language Models* for information retrieval. The idea is to create models that have fewer non–zero probabilities to describe the data. “A smaller model means that less storage overhead and less CPU time is needed” (Hiemstra, Robertson, and Zaragoza 2004). An application for these models is for example the use in Smartphones, where storage and CPU are limited.
Hiemstra et al. applied their model at three stages of the retrieval process, i.e. at indexing time, at search time and, finally, at feedback time.

The notion of *Relevance* for the language modeling approach is described by Lafferty and Zhai (2003), Lavrenko and Croft (2003) and Spärck Jones et al. (2003). They all introduce the classical probabilistic approach to information retrieval (Robertson and Spärck Jones 1976) to the field of language modeling.
Lafferty and Zhai show that both models are equivalent, because they are based on different factorizations of the same generative relevance model (Lafferty and Zhai 2003).
The focus of Lavrenko’s and Croft’s work is the estimation of relevance models when no training data are available. They also use their resulting model to experiment in the field of cross–language retrieval.
Spärck Jones et al. also describe in their work how to bring the notion of relevance to the language modeling approach. In particular, they examine the use of multiple relevant documents and applying relevance feedback. Their approach also uses parsimonious language models.

Beside information retrieval, *Topic Detection and Tracking* (TDT) is also an application area for language models. The task of TDT is similar to IR, but instead of searching relevant documents for a query, here, the topical structure of multilingual news streams should be determined. In contrast to ad–hoc retrieval, topic tracking requires matching scores to be comparable across topics (Kraaij and Spitters 2003).
Spitters and Kraaij (Spitters and Kraaij 2000, Kraaij and Spitters 2003) describe approaches of how to use unigram language models for the task of TDT. They tested the document likelihood and query likelihood ratio, and variants of both models, based on different length normalization techniques.
Another focus of their work is the use of clustering for the grouping of stories. Here, a

language model based approach is used to calculate the similarity between a cluster and a story.

We have already mentioned the task of *Cross Language Information Retrieval* (CLIR) above, where documents in languages different from the query language are searched. The idea of language model based CLIR is to build a language model which can be used to translate a query to documents.
Lavrenko, Choquette, and Croft (2002) introduce a relevance model for this task. In their work, a general, content–independent model for CLIR is developed. Another relevance model is proposed by Lavrenko and Croft (2001) which refers to the probability distribution specifying the expectation frequency of any word in the document relevant to the query.

Compression–based language models for the task of *Text Categorization* are presented by Teahan and Harper (2003). They apply the information theoretically founded approach of text compression for categorization of texts. Therefore, two different models are suggested. The first one bases on the ranking by document cross entropy with respect to a category model, and the second one bases on document cross entropy difference between category and the complement of the category models (Teahan and Harper 2003).

Text Summarization is used to reduce or shrink a large amount of information to what is absolutely necessary to understand a text. So, summaries help saving time for a reader.
Mittal and Witbrock (2003) present an automatic, non–extractive, language model based text summarization for Web pages. In this context, “non–extractive” means, the the information is not extracted from the original text, but a generative model is used to produce the summarization. Their work shows a machine translation approach to the subject of summarization. First, language models are used to overcome the problem of unsorted output, and second, the output is created on the basis of trigrams.

If it is possible to match the document scores of different search engines for a given query to normalized probabilities, a lot of new applications may be conceivable. Manmatha (2003) presents in his work empirically “that the score distributions on a per query basis may be fitted using an exponential distribution for the set of non–relevant documents and a normal distribution for the set relevant documents” (Manmatha 2003). He also shows how to estimate a score distribution if no relevance information is available by using a mixture model consisting of an exponential and a normal distribution. Finally, some applications, like combining different search engines, are presented.

Mittendorf and Schäuble (1994) and Miller, Leak, and Schwartz (1999) introduce *Hidden Markov Models* (HMM) for the task of information retrieval, in particular for document and passage retrieval.

For example, Miller, Leak, and Schwartz (1999) experimented with a framework including blind feedback, bigram modeling, query weighting, and document–feature dependent priors. Their experiments show that this approach also performs very good for ad hoc information retrieval.

An extension of the language modeling approach to information retrieval by explicitly modeling the importance of a term is given by Hiemstra (2002). In this complex model it is possible to decide if words are stop words, mandatory words, or some pairs of words (bigrams) should be considered as phrases, etc.

“Decisions on stop words and non stop phrases are typically taken by the system, based on a trade–off between search quality and search speed. Mandatory terms and phrases are typically selected by the user” (Hiemstra 2002). Finally, they stated that statistical ranking algorithms motivated by the language modeling approach perform quite well in such an empirical setting.

A new *Dependency Language Model* for information retrieval is presented by Gao et al. (2004). In their work, they extend the unigram language modeling approach by relaxing the independence assumption. Therefore, the linkage of a query is introduced as a hidden variable. This linkage tries to explain the interdependency of query terms.

Then, a query is generated from a document in two stages. In the first step, the linkage is generated and in the second step, each term is generated depending on other related terms according to the linkage. In the end, some applications for the task of information retrieval, like smoothing with different techniques, are presented.

Another dependency language modeling approach for information retrieval which extends the existing unigram model by relaxing the independence assumption is introduced by Cao, Nie, and Bai (2005). But in this approach various word relationships can be integrated. Their work presents two different kinds of relationships, word co–occurences and relationships extracted from WordNet. Their main conclusion is, that including many different kinds of resources into a language model for information retrieval provides better results.

Chapter VI

Query Construction

This chapter covers our approaches to use language models for the task of question classification. Normally, a user question can be assigned to a specific type, like a person or a location, etc. This is a very important step, because in this module, the decision is made what piece of information should be extracted in later stages. A language model based approach is described in section VI.1.
The best possible class is determined by calculating a score for each question type and returning the class with the highest score. These confidence measures can also be used to decide if a returned question class should be used or not. We discuss them in section VI.2.

1 Question Typing

Here, we propose a language model based approach to classify user questions in the context of question answering systems. As categorization paradigm, a Bayes classifier is used to determine a corresponding semantic class. We present experiments with state–of–the–art smoothing methods as well as with some improved language models. Our results indicate that the techniques proposed here provide performance superior to the standard methods, including support vector machines.

This section is mainly based on Merkel and Klakow (2007c).

1.1 Introduction

In a spoken question answering system a speech recognizer is used on top of a question answering (QA) framework. For example, such a system was developed as a part of the Smartweb project (Wahlster 2004). In this system, a speaker asks a natural language question and the system provides an answer. In contrast to a classical document retrieval framework, which just returns relevant documents to a user query, a QA system answers with accurate responses to a question posed in natural language (Shen et al. 2006). Thus, the task is very complex and document retrieval is only a small part of the entire system. In order to provide the user with the correct answer, the QA system has to “understand” the meaning of a question. For example, if a user asks the question *When was James Dean born?*, the answer of the system should be a date and not a country. This means that the QA system has to analyze the question before taking further steps. Normally, this is done in the *query construction*. In this part, the user question is classified into several semantic categories. This categorization helps the system to reduce the search space and, hence, is very useful in finding and verifying resulting answers. It may also help to determine the search strategies for further retrieval modules (Li and Roth 2002).

Most existing QA systems do not use more than 20 semantic categories to classify questions. In the approach we describe in this chapter, a more fine–grained classification taxonomy is applied. The original hierarchy was introduced in Li and Roth (2002) and describes 6 coarse– and 50 fine–grained classes. As we will show, it is sufficient to optimize only the fine–grained classes; thus, in our experiments, we used this taxonomy for classification. Based on this taxonomy, a Bayes classifier with language models as categorization paradigm was employed. With this framework, we show that our approach is at least as good as systems discussed in current literature (Zhang and Lee 2003).

A similar experimental system for retrieving video data was presented in Klakow (2006a).

Normally, most QA systems just use no more than 20 coarse classes for classification, but in this chapter we decided to use the taxonomy proposed by Li and Roth (2002).

We applied the fine–grained classification in our experiments because they proved that they are more useful to locate and verify answers. Based on this categorization we used a Bayes classifier with language models as classification paradigm. We will demonstrate that this approach outperforms systems in current literature.

1.2 Methodology

The framework we used to classify the user questions is described in this section. For the task of information retrieval, a language model based approach was introduced by Ponte and Croft (1998). They showed that this method performs better than traditional state–of–the–art retrieval systems. In this chapter, we intend to propose the same techniques for the task of classifying user questions. The main advantage of using a language model based approach is the large supply of known techniques to calculate and smooth probabilities. This is necessary, because there is so little training data available. On average, there are only about 100 training questions per class.

As described in section VI.1.1, a Bayes classifier was used to perform the categorization task. In this case, it is defined by

$$\hat{c} = \mathrm{argmax}_c P(Q|c)P(c) \qquad . \tag{VI.1}$$

The advantage of applying such a classifier is the certainty of obtaining a minimum error rate provided that all probabilities are exactly known. The term $P(Q|c)$ denotes the conditional probability of the user question Q given semantic class c. $P(c)$ is the prior probability of this class. If we consider the problem of data sparsity for training, $P(Q|c)$ has to be calculated as the product of all occurring query terms:

$$P(Q|c) = \prod_{i=1}^{n} P(w_i|w_{i-1}c) \tag{VI.2}$$

where $Q = \{w_1 \ldots w_n\}$. The prior probability $P(c)$ is calculated as a unigram language model on the specific class c. This is contrary to most of the current literature (Zhai and Lafferty 2001), where the prior information is considered uniform, and therefore can be neglected. Smoothing is not required, because all of the semantic classes occur more than four times in the training data, which is often sufficient to calculate the maximum likelihood probabilities for language model estimation.

The next section shows how to calculate $P(w_i|w_{i-1}c)$. To avoid the problem of zero probabilities, which would result in excluding specific terms from the classification, smoothing methods are introduced.

1.3 Smoothing Methods

In this section, we will illustrate how to smooth zero probabilities, which can occur when query terms are not seen in combination with a specific semantic class. For that purpose, we introduce unigram as well as bigram language models.

1.3.1 Standard Smoothing Methods

Zhai and Lafferty (2001) presented three different smoothing methods based on unigram statistics for the task of information retrieval. In the following sections, we also introduce these standard methods for the task of question classification.

1.3.1.1 Jelinek–Mercer

A smoothing technique based on linear interpolation was first introduced by Jelinek and Mercer (Zhai and Lafferty 2001). This technique was based on the probability estimate

$$P_\lambda(w_i|c) = (1-\lambda)\frac{N(w_i,c)}{\sum_{w_i} N(w_i,c)} + \lambda P_{BG}(w_i|c) \tag{VI.3}$$

where $N(w_i,c)$ is the count of the word w_i in combination with the class c and $P_{\text{BG}}(w_i|c)$ is the “background” probability for unseen events. Possible distributions are introduced in section VI.1.4.

The interpolation weight is defined by λ; higher values of λ induce more smoothing.

1.3.1.2 Bayesian Smoothing with Dirichlet Priors

If a multinomial distribution for estimating a language model is considered and a Dirichlet distribution is used as the conjugate prior, it results in the smoothed probability estimate

$$P_\mu(w_i|c) = \frac{N(w_i,c) + \mu P_{BG}(w_i|c)}{\sum_{w_i} N(w_i,c) + \mu} \tag{VI.4}$$

where $N(w_i,c)$ is the frequency of word w_i and class c, $P_{\text{BG}}(w_i|c)$ is the collection model and μ is the smoothing parameter.

1.3.1.3 Absolute Discounting

Absolute discounting is the most common and popular smoothing method in speech recognition. It is defined by

$$P_\delta(w_i|c) = \frac{\max\left(N(w_i,c)-\delta,0\right)}{\sum_w N(w_i,c)} + \frac{\delta B}{\sum_{w_i} N(w_i,c)} P_{BG}(w_i|c) \tag{VI.5}$$

where $N(w_i, c)$ are the observation frequencies determined on the training data. The term $P_{\text{BG}}(w_i|c)$ denotes the backing–off model trained on background data and B specifies how often $N(w_i, c)$ is larger than the smoothing parameter δ.

1.3.2 Improved Smoothing Methods

In addition to the smoothing algorithms defined in section VI.1.3.1, we here consider improved methods to estimate smoothed probabilities.

1.3.2.1 Improved Absolute Discounting (UniDisc)

The improved absolute discounting smoothing method introduced in Klakow (2006b) for smoothing very small adaptation corpora is known as *UniDisc*. This technique was based on the probability estimate

$$P_d(w|c) = \begin{cases} \frac{\mathrm{N}(w_i,c)-d}{\mathrm{N}} + \alpha P_{BG}(w_i|c) & \text{if } \mathrm{N}(w_i,c) > 0 \\ \\ \alpha \cdot P_{BG}(w_i|c) & \text{else} \end{cases} \tag{VI.6}$$

Equation (VI.6) shows a unigram back–off model as described in section VI.1.3.1.3, where $N(w_i, c)$ is the frequency of the term w_i in combination with class c, the discounting parameter is defined by d, and $P_{BG}(w_i|c)$ is the back–off language model. The term α denotes the back–off weight. In this case, the discounting parameter is independent of the term frequency. By contrast, the rational function

$$d(N) = \frac{d_0 + s(N-1)}{1+g(N-1)} \tag{VI.7}$$

describes a discounting parameter which depends on term counts, where d_0 is the absolute discounting parameter introduced in section VI.1.3.1.3, whereas s and g are additional parameters.

1.3.2.2 Log–Linear Interpolation

The use of language models for log–linear interpolation was first proposed in Klakow (1998). The exact model we will use in our experiments is defined by

$$\begin{aligned} P(w_i|w_{i-1}c) = \tfrac{1}{Z_\lambda(w_{i-1}c)} \quad & P_{UniDisc}(w_i|c)^{1.0} \\ & P_{AbsDisc}(w_i|w_{i-1}c)^{\lambda} \end{aligned} \qquad \text{(VI.8)}$$

where $Z_\lambda(w_{i-1}c)$ is a normalization weight depending on w_{i-1}, c and the parameter λ. The term $P_{\text{UniDisc}}(w_i|c)$ is the improved absolute discounting method described in the previous section and $P_{\text{AbsDisc}}(w_i|w_{i-1}c)$ is an absolute discounting approach using bigram statistics. The parameter λ denotes the interpolation weight.

1.4 Background Models

In this section, possible background models we used for the smoothing methods described in section VI.1.3 are shown.

1.4.1 Zerogram

The simplest background model, which uses no information about words, is the zerogram defined by

$$P_{BG}^{Zero}(w_i|c) = \frac{1}{|V|} \qquad \text{(VI.9)}$$

where $|V|$ is the size of the vocabulary.

1.4.2 Unigram

Another commonly used distribution is the unigram model. It can be computed with the well–known maximum likelihood estimator:

$$P_{BG}^{Uni}(w_i|c) = \frac{N(w_i)}{\sum_{w_i} N(w_i)} \qquad . \qquad \text{(VI.10)}$$

$N(w_i)$ means the frequency of the word w_i in the training corpus. Note that both variants are independent of the class c.

1.4.3 Bigram

For the task of information retrieval, bigram language models are not very popular. Nevertheless, we want to introduce them for classifying user questions. They are described by the formula

$$P_{BG}^{Bi}(w|vc) = \frac{N(wv,c)}{\sum_v N(v,c)} \quad . \tag{VI.11}$$

Here, $N(wv,c)$ denotes the frequencies of the word–pair wv and the class c.

1.5 Experiments

This section describes the dataset we used for our experiments as well as the results of the different smoothing approaches.

1.5.1 Dataset

We used the 5,500 questions provided by the Cognitive Computing Group at University of Illinois at Urbana Champaign[1] as training data. The evaluation was done with the TREC10[2] dataset, which contains 500 test questions. The vocabulary we used in our experimental setup consists of about 10,000 words extracted from the training data.
As evaluation metric, the misclassification error rate (MER), which specifies the percentage of misclassified semantic classes for the evaluation test data, was chosen.
For our experimental setup, we used the classification taxonomy defined in Li and Roth (2002). It consists of 6 coarse– and 50 fine–grained semantic classes. Figure VI.1 shows the correlation between the number of errors for the coarse– and fine–grained classes as a scatter plot. It proves that both types of classification correlate very well. Hence, for the balance of this work, we concentrate exclusively on improving the performance of the fine–grained classes.
For our experiments at TREC 2007, we extended the proposed taxonomy by classes described in Table VI.1. This was done because last year's Q&A tracks at TREC showed an increasing number of questions of those types.

[1] `http://l2r.cs.uiuc.edu/∼cogcomp/Data/QA/QC/`
[2] `http://trec.nist.gov`

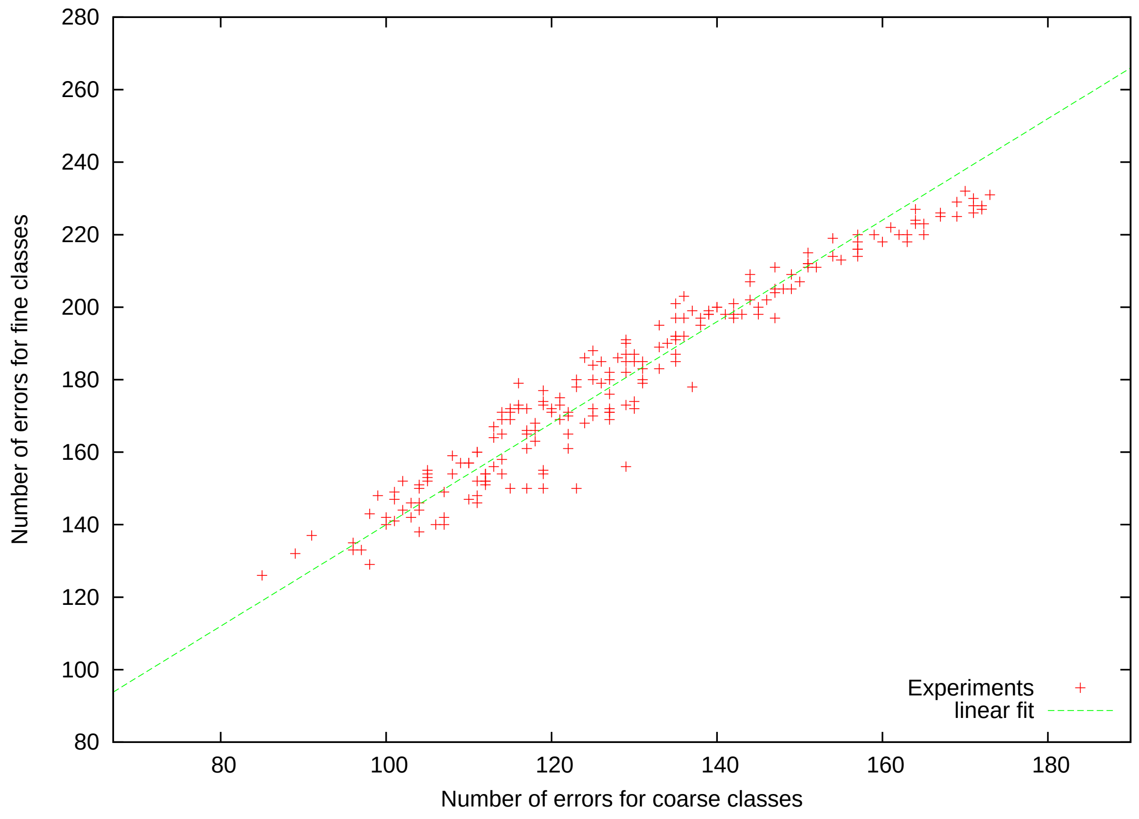


Figure VI.1: Correlation between number of errors for coarse and fine classes.

Because, so far, there is no training data for the types introduced in Table VI.1, we had to extend the original 5,500 questions provided by the Cognitive Computing Group at UIUC by the new classes. But to get a more reasonable number of training data for those classes, we additionally annotated the TREC 2001–2006 questions. So, the current training set contains about 10% of questions labeled with the new classes (Shen et al. 2007).

New Class	**Description**
ENTY:cremat:movie	all moving images
ENTY:cremat:books	any printed matter
ENTY:cremat:song	any piece of music
HUM:ind:actor	actor of movies and TV programmes
HUM:ind:author	author of printed matter
HUM:ind:composer_performer	musical performer and/or composer

Table VI.1: List of the new question types used for TREC 2007 experiments.

1.5.2 Results

The results of our experiments are discussed in this section. As experimental methods, we used the standard and improved smoothing algorithms defined in section VI.1.3.

1.5.3 Jelinek–Mercer Interpolation

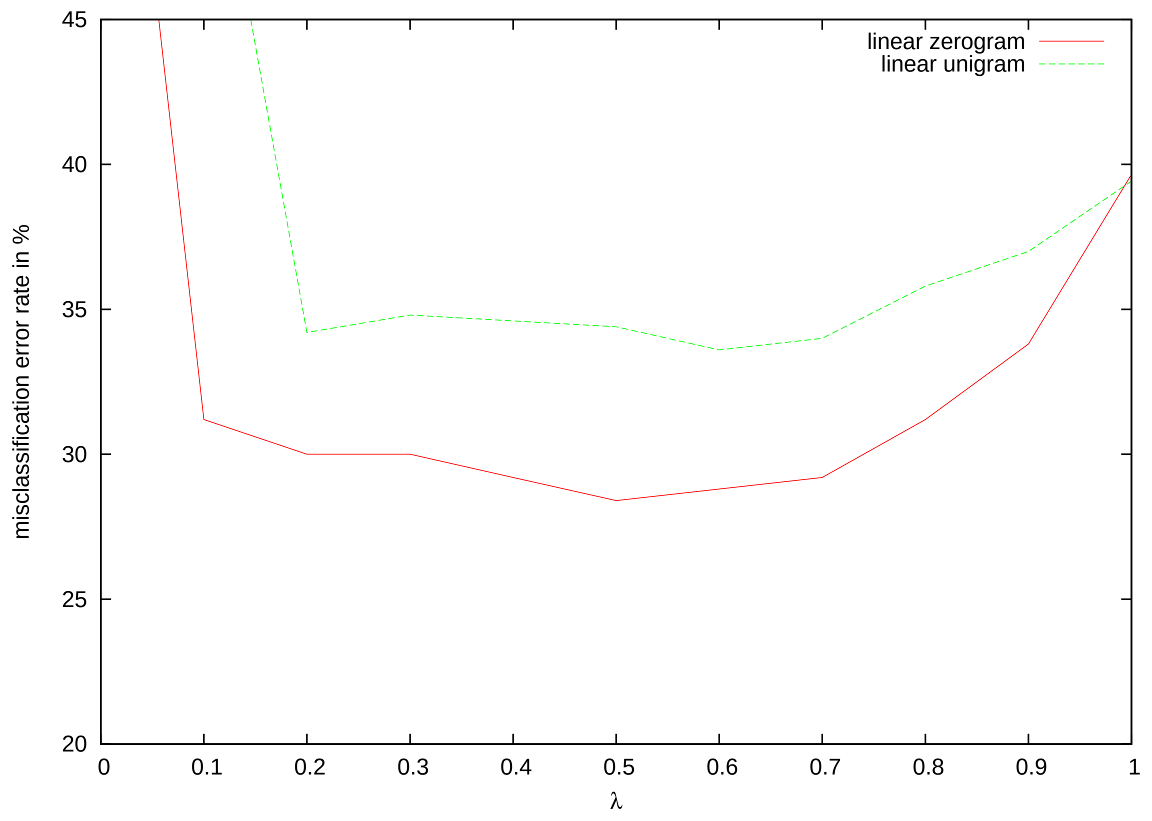


Figure VI.2: Misclassification error rate for different linear interpolation smoothing parameters.

Figure VI.2 shows the results of the Jelinek–Mercer interpolation for different smoothing weights. It proves that using a zerogram background distribution performs slightly better than unigram information. Both curves are relatively independent of the interpolation weight and reach their minimum MER at approximately 0.5, which means that smoothing is necessary. For a high smoothing parameter ($\lambda \approx 1$), the unigram language model is slightly better than the zerogram distribution.

1.5.4 Dirichlet Priors

The experiments with Dirichlet priors are presented in Figure VI.3. In this case, the zerogram model performs significantly better than the unigram distribution. A smoothing value

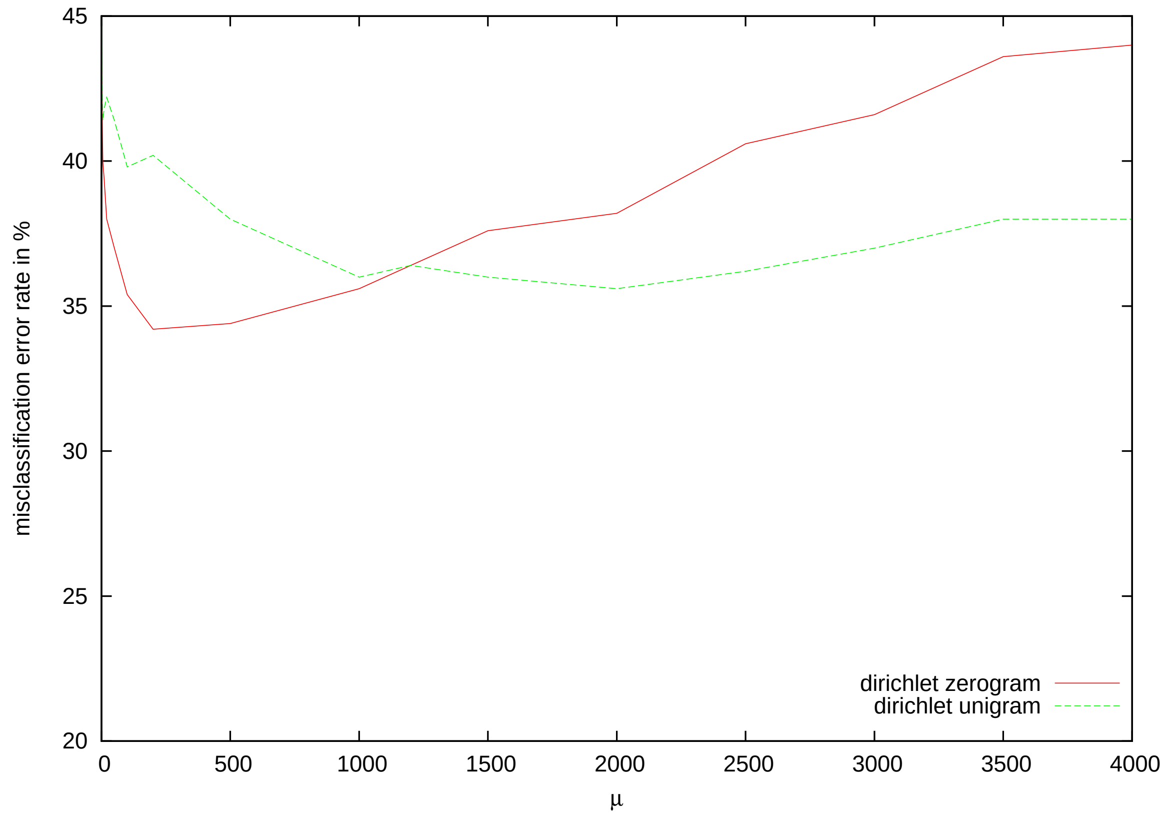


Figure VI.3: Misclassification error rate for different Dirichlet prior smoothing parameters.

of $\mu = 200$ is yielded the best performance, but this time, both methods depend more on the smoothing parameter. Surprisingly, the behavior of the unigram background model in comparison with the zerogram distribution is completely different. These results indicate that the unigram model requires much more smoothing to enhance the performance. When using a parameter $\mu \geq 1200$, the unigram performs better than the zerogram language model.

1.5.5 Absolute Discounting

For absolute discounting (Figure VI.4), both used background distributions strongly depend on the smoothing parameter. In contrast to the other standard methods, best results are obtained when using a large discounting value. Again, the unigram model performs better than the zerogram distribution for a high smoothing value. But in this case, the unigram

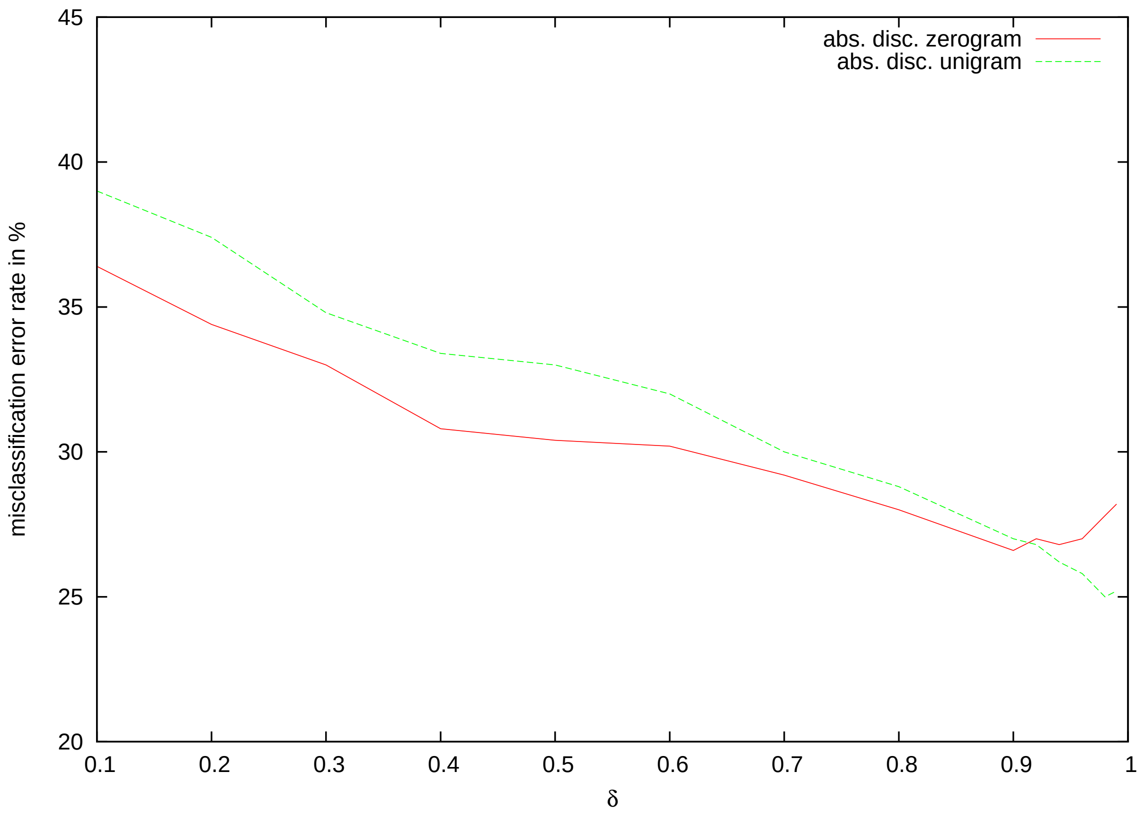


Figure VI.4: Misclassification error rate for different discounting parameters.

statistics exceeds the zerogram background model.

1.5.6 UniDisc

The experimental results for the first improved language model are shown in Figure VI.5. For this setting, the best performance was gained when using the discounting parameter $d_0 = 1$. From this follows that events, which appear just once, are discarded. Thus, only terms with multiple occurrences are used for classification. As singletons are mostly nouns, this is a reasonable result, because such events contain no useful information for the task of question categorization.

The combination of both additional discounting parameters is presented as a contour plot. It shows the area of best performance at $s = 0.8$ and $g = 0.007$ with a MER of 20.6%.

1.5.7 Log–Linear Interpolation

Figure VI.6 demonstrates the experiments for the log–linear interpolation approach. It proves

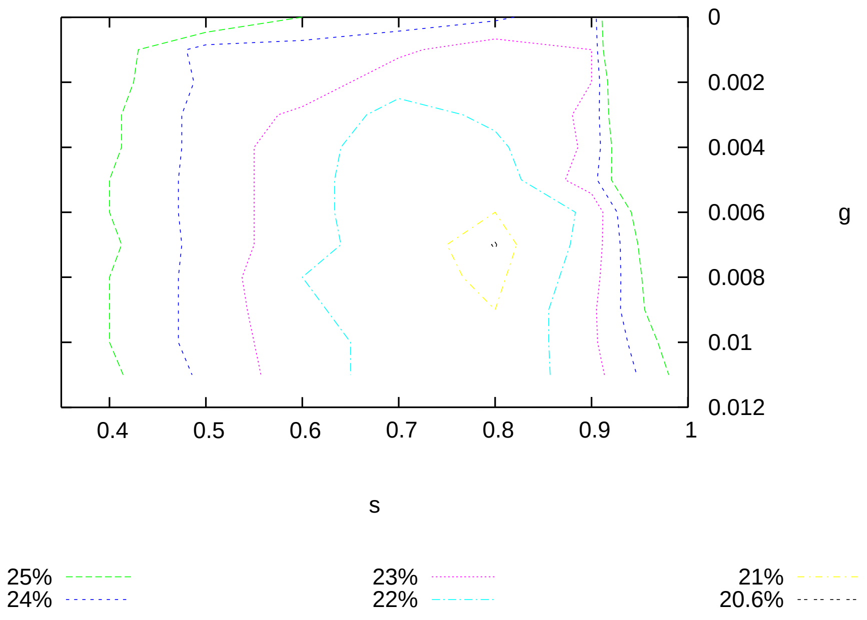


Figure VI.5: Contour plot for misclassification error rate for different discounting parameters for UniDisc experiments.

that using absolute discounting with bigram statistics for interpolation results in an additional performance gain. The best MER is achieved at approximately $\lambda = 0.1$. For $\lambda \geq 0$, the distribution remains relatively independent to the smoothing parameter.

1.6 Conclusion

In this chapter, we have presented a language model based approach to question classification in the context of spoken question answering. Our methodology is based on a Bayes classifier and uses several state–of–the–art and improved methods to smooth unseen events. We also showed the effects of using different background models, such as zerogram and unigram models.

Table VI.2 gives an overview of all smoothing methods used in our experiments. It shows the number of misclassified categories as well as the misclassification error rate (MER).

In general, our improved approaches perform better than the standard smoothing methods. Here, the absolute discounting method is best, whereas Dirichlet priors performs best for

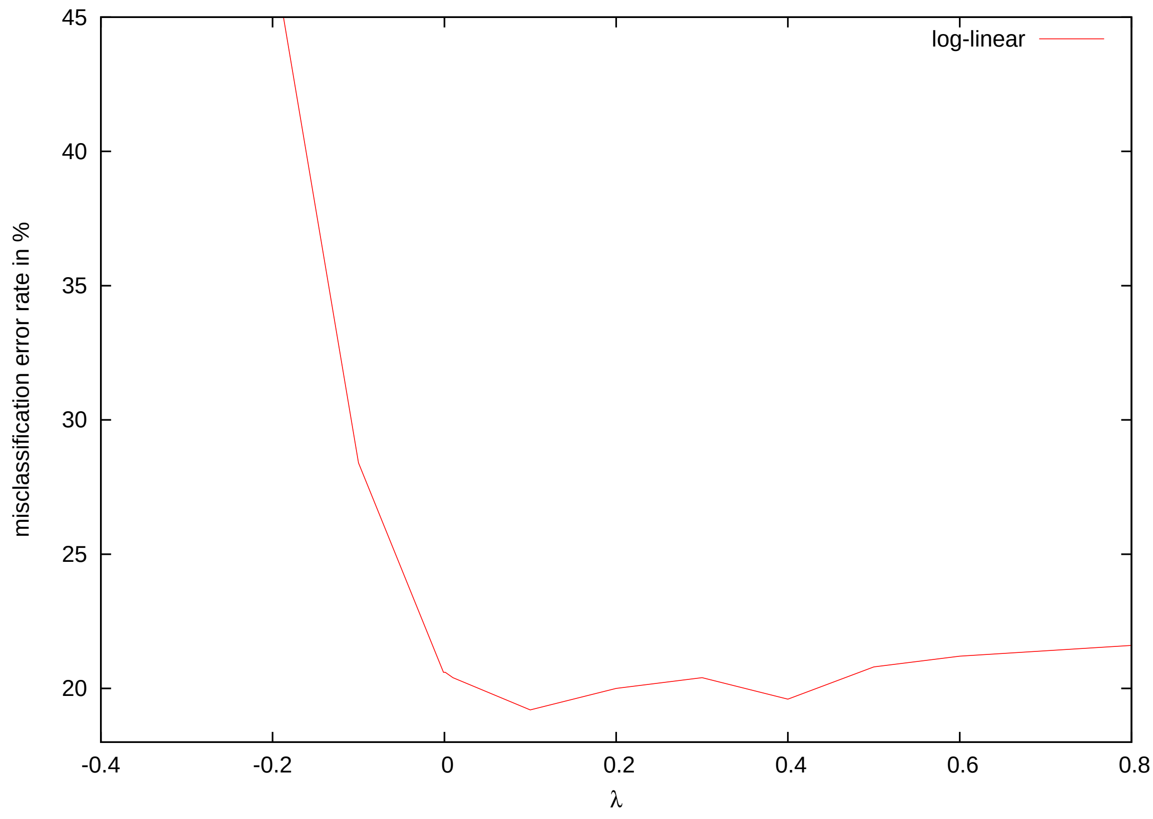


Figure VI.6: Misclassification error rate for different log–linear smoothing parameter.

the task of sentence retrieval in question answering (Shen et al. 2006). With regard to the improved methods, the enhanced absolute discounting experiments performed better than the standard back–off algorithm. The approach with the best results is the log–linear interpolation method, which achieved a MER of 19.2%. These results are comparable to other state–of–the–art methods, like SVMs. But, in terms of training, this approach can be computed in linear time complexity and therefore is much faster compared to most existing implementations which have to be computed in polynomial time.

Method	MER
Jelinek–Mercer	28.4%
Dirichlet Prior	34.2%
Absolute Discounting	25.5%
UniDisc	20.6%
Log–Linear	19.2%

Table VI.2: Comparison of proposed language model based approaches for query classification. In contrast, the best traditional method in literature is the Support Vector Machine (SVM) with a MER of 19.8%.

2 Confidence Measures

In the last section, we introduced a language model based approach for the task of question classification. For a more efficient use of the resulting question classes, a measure is presented in this section, which specifies the confidence in the module, that the returned class is correct.

2.1 Introduction

The idea of calculating confidence measures for the task of question answering is originally derived from formula III.51. Here, the confidence measures as we want to introduce them in this section, are a part of the language model based approach for question answering. In particular, it is part of the first term in the formula, where a question type classifier depending on the named entities in a sentence has to be calculated.

The use of confidence measures is not new. For example, for the task of information retrieval, it is a very common approach to rank documents. The confidence measure of a retrieval system is computed for each document to show how it matches the information need of a user.

In this section we want to introduce the notion of confidence measures for the task of question classification.

We described in the last section, that a question class is computed by using a Bayes classifier as classification paradigm (chapter VI.1). This algorithm calculates a score for each possible question class and returns the class with the best score. As we will show in this section, these scores can be regarded as confidence values for a specific class.

When using this definition of confidence measures, it is not only possible to return the best class but also to return a specific number (top$N$) of possible classes. These classes with the corresponding confidence values can then be made available to various other modules of the system.

For example, the confidence measure of a class can be used to decide whether it is applicable to specify the question type or not. We will show this discrimination characteristic in section VI.2.4.

And finally, it can also be used in succeeding modules, i.e. in the answer extraction module, to give further scores for finding the best answer candidate.

2.2 Dataset

Because calculating the confidence measures of question classes is nearly the same task as described in the last section (VI.1), we also used the same datasets as specified there. To simplify matters, we shortly recapitulate the used datasets.

As training data, we used the 5,500 questions provided by the Cognitive Computing Group at University of Illinois at Urbana Champaign. For evaluating the experimental results, the TREC10 dataset was used. For our experimental setup, we used the classification taxonomy defined in Li and Roth (2002).

For further experimenting, mainly for the task of TREC 2007, we extended the existing taxonomy by classes such as movie, books and songs. Table VI.1 provide more information about this facts. This was done because last year's Q&A tracks for TREC showed an increasing number of questions of those types.

Because there are no training data for those types, we had to extend the original 5,500 questions described above by the new classes. But to get a more reasonable number of training data for those classes, we additionally annotated the TREC 2001–2005 questions. So, the current training set contains about 10% of questions labeled with the new classes.

As testing set for this new kind of training data, we further annotated the TREC 2006 question set. Finally, we calculated histograms for both datasets, each containing all classes and the four most frequent classes in an experiment.

2.3 Used Methods

As explained in the introduction, the idea of calculating confidence measures originally comes from formula III.51. Here, the approach of using language models for the task of question answering is split up into different parts. One part is to determine the question type classifier depending on the named entities in a sentence, i.e. $P(c|NE_i, W_i, D_i, Q_i, T)$, where c is the question class, NE_i is a named entity, W_i is a query word, D_i a specific document, Q_i the query, and T the topic of a query.

As a simplification, we can also calculate

$$P(c_1|Q)$$

where this equation split the confidence measures into the single question types. It also describes that it is possible to return more than just one class with a confidence value, as it is standard for the task of question classification. So, the top$N$ classes together with their confidence values can be used by the Q&A system, if needed.

Following this, estimating the probability of a question type given a query can be expressed by the following equation

$$
\begin{aligned}
P(c|Q) &\propto P(Q|c)^{w(c)} \cdot P(c) && \text{(VI.12)}\\
&= P(Q|c) \cdot P(c)^{\frac{1}{w(c)}} && \text{(VI.13)}\\
\Rightarrow P(c|Q) &= \frac{P(Q|c) \cdot P(c)^{\frac{1}{w(c)}}}{\sum_{c'} P(Q|c') \cdot P(c')^{\frac{1}{w(c')}}} && \text{(VI.14)}
\end{aligned}
$$

Here, the first approximation (VI.12) is done using the Bayes rule. The only difference is that the class dependent weight $w(c)$ is introduced to further weight the probability of the question given the class c. The idea of using such a weight is derived from the task of speech recognition, where weights are introduced to compensate the flaw of having different dynamic ranges for two or more different distributions (Jelinek 1997).

The final equation for calculating confidence values for a specific class is shown in formula VI.14. The denominator is introduced to obtain a well–formed probability distribution for $P(c|Q)$. Now, if this probability is summarized over all possible classes, it results in one, which means that it is normalized.

Because this likelihood is very similar to the task of question classification, it is very easy to extend the implementation to also provide the system with confidence values for the specific classes.

2.4 Experiments

In this section, we want to present some of our results concerning the confidence measure of a question class. The plots just show the distribution of confidence values for the best possible question type. Values for the top$N$ classes are also available, but as they seemed to deliver no further insight, we concentrate on the best possible class.

The resulting graphs are presented as histogram plots for correct and incorrect classified questions. The axis of abscissae always shows the confidence measure whereas on the axis of ordinates the no–normed number of correct or incorrect documents is printed.

2.4.1 TREC 10

The first set of experiments is made by just using the 5,500 questions provided by UIUC as training set and the TREC 10 question set as testing data. The exact dataset is explained in section VI.2.2.

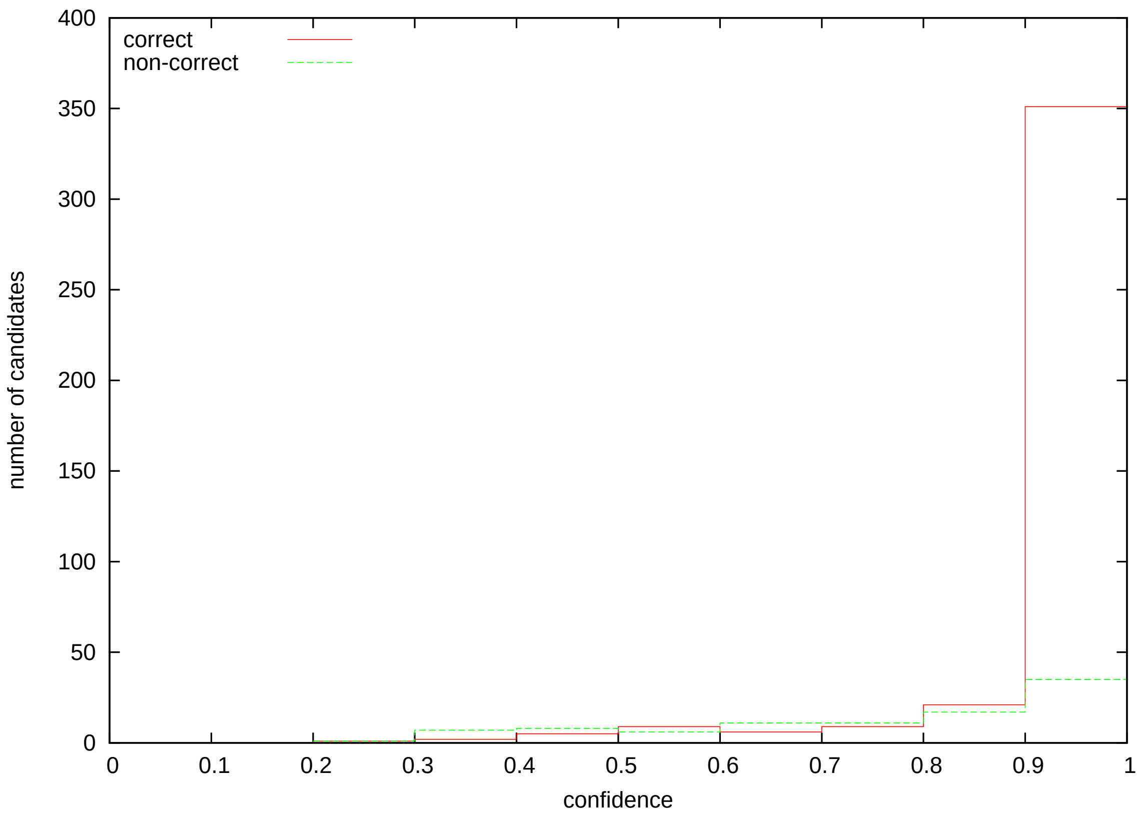


Figure VI.7: Histogram of confidence values for correct/incorrectly classified questions for the TREC 10 dataset.

Figure VI.7 shows the histogram graph for the combination of all question types used. It is very obvious that for confidence values below 0.9 the number of correct and incorrect classified questions is rather equal. This means that it is not possible to make a decision for these values whether a returned class is correct or not.

But, if the confidence measure is higher than 0.9 the number of correctly classified questions is much higher than the number of incorrect ones. So, if a value in this range is returned by the question typing module, the system can highly trust this classification and take the question type as correct.

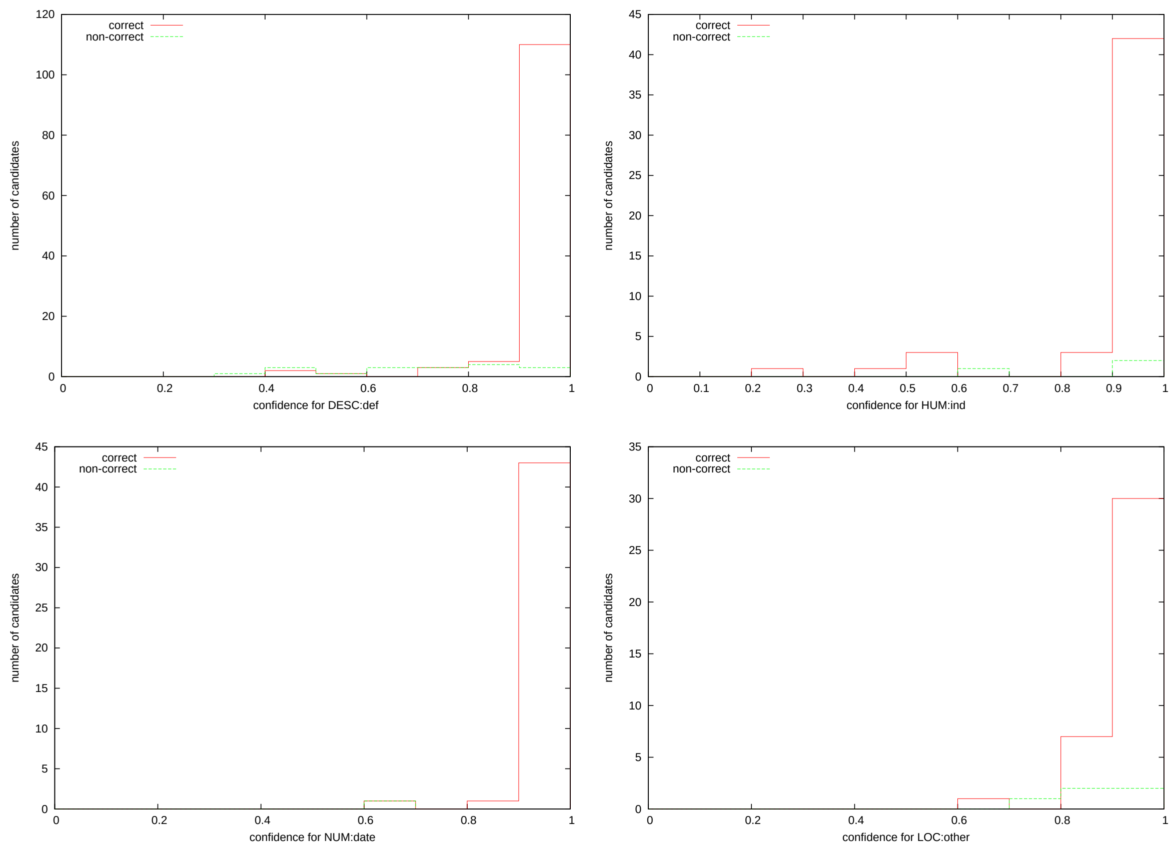


Figure VI.8: Histograms of confidence values for correct/incorrectly classified questions of the four most frequent question types for the TREC 10 dataset.

The result graphs of Figure VI.8 show the histograms of the four most frequent question types for the used dataset.

Hence, on the top left–hand side the results for the class *Description:definition* (DESC:def) is presented. It shows nearly the same characteristics as the results using all classes. This means that a class with a confidence measure of 0.9 or higher is taken as correct.

The rest of the plots show the same behavior for the question types *Human:individual* (HUM:ind), *Number:date* (NUM:date), and *Location:other* (LOC:other). For all figures it is obvious to take a class with confidence values above 0.9 as correct and neglect all types with a lower measure.

2.4.2 TREC 2006

The same experiments were also done using the extended dataset described in section VI.2.2. Here, we added some additional question types to the original taxonomy, like movies or songs. Therefore, the complete TREC 2001– 2005 question sets are additionally annotated using the new types. The results presented in this section are then evaluated with the TREC 2006 question set, we also annotated for this task.

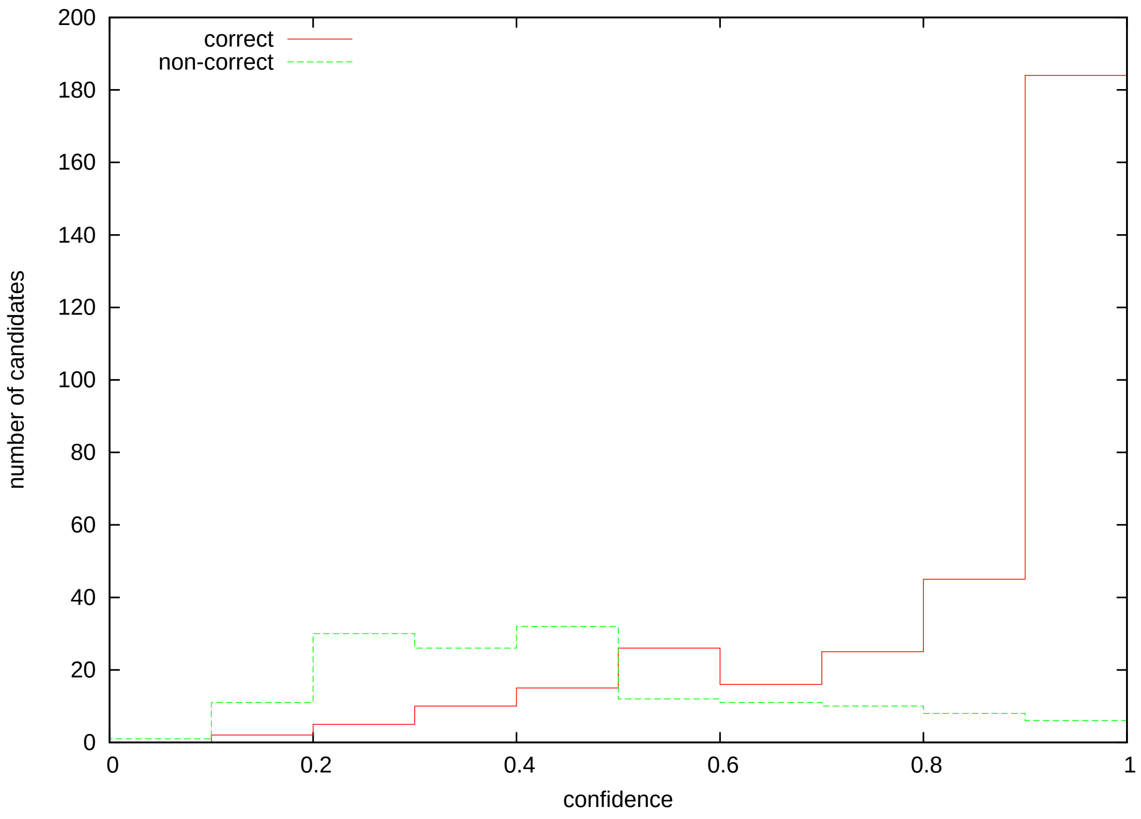


Figure VI.9: Histogram of confidence values for correct/incorrect classified questions for the TREC 2006 dataset.

Again, Figure VI.9 displays our results using all possible question types. Compared with the graph in Figure VI.7 the results presented for these experiments are not that straightforward. Nevertheless, it can be shown that for a confidence value above 0.9, the number of correctly classified answers is still much higher than the number of misclassifications. So, for these experiments, we can also argue that a confidence value of 0.9 or higher can be taken as correct.

In a strict sense, this fact also holds for values above 0.5. But as the difference between the number of correctly and incorrectly classified questions is very small, we take the confidence measure of 0.9 as threshold.

The histograms in Figure VI.10 again show the experimental results for the four most frequent question classes. In particular, results for *Human:individual* (HUM:ind), *Location:other* (LOC:other), *Number:count* (NUM:count), and *Number:date* (NUM:date) are presented.

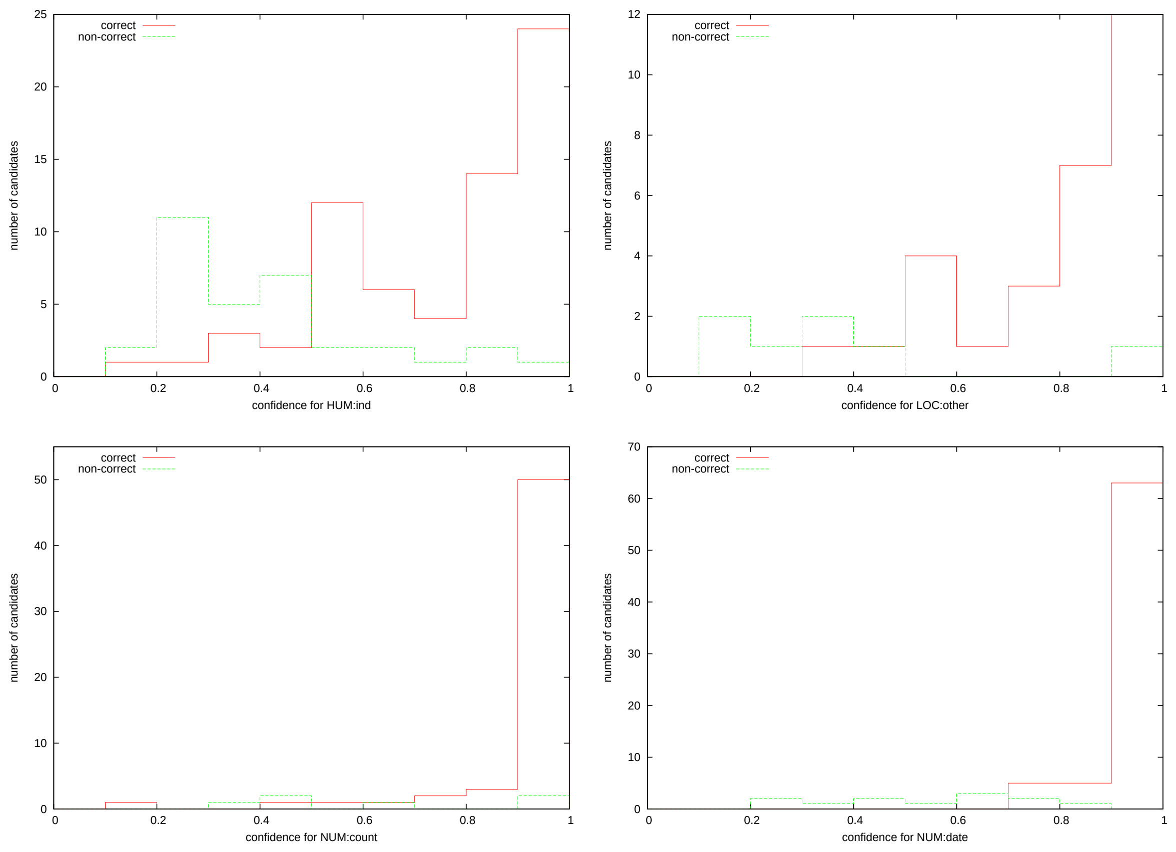


Figure VI.10: Histograms of confidence values for correct/incorrect classified questions of the four most frequent question types for the TREC 2006 dataset.

Although the number of correct and incorrect questions varies very much for specific question types, it can be seen that using a value of 0.9 or higher is sufficient to distinguish between correctly and incorrectly classified question types. As in our previous experiments, a value above 0.9 means that the returned class can be taken as correct. For some of the question types, i.e. HUM:ind and LOC:other, a lower confidence value might also be useful to distinguish between correct and incorrect classes.

2.5 Conclusion

In this section we presented an approach to calculate confidence measures for question types using the question classification algorithm described in section VI.1.

We derived the idea of generating confidence values from the theoretic approach to a language model based question answering system (III.3), where a question type classifier depending on named entities in a sentence has to be calculated.

As results of our experiments regarding the best class returned we showed that a confidence measure of 0.9 or higher is sufficient to take a classified question type as correct.
Experiments using the extended classification taxonomy described in section VI.2.2 and TREC 2006 as testing set also showed that for special question classes it might be useful to take a lower confidence value to differentiate between correctly and incorrectly classified questions. This fact has to be researched in future work.

Chapter VII

Document Retrieval

Document retrieval is the most common step for all Q&A systems. The main goal is to translate the user's query into a set of possibly relevant documents which can contain the answer to the user's information need. Because this is typically not the last retrieval step, the *pre–fetching* of relevant documents is also often called *document pre–fetching* (Monz 2003). This also acts as a kind of filter. Special filtering methods (IR algorithms) are used to get relevant documents out of a large collection. As mentioned above, this collection is typically too large to process every document, so, most, or even all state–of–the–art Q&A systems use a document retrieval module for pre–selection.
This chapter is mainly described in Hussain, Merkel, and Klakow (2006).

1 Introduction

In this chapter, we explain our methodology to the task of document pre–fetching. As we used a statistical approach to question answering, we also applied language models to find relevant documents to a user query. These methods were first introduced by Ponte and Croft (1998) and are now a common alternative to traditional algorithms, like tf–idf or Okapi because of their formal mathematical model. Other reasons of the popularity of statistical language models for this task are their simple framework, their use for indexing and retrieval models and so on. A discussion about further advantages and disadvantages can be found in chapter III.
For the experiments we present in this section, standard smoothing techniques to calculate the statistical language models are used. The results are then optimized to return as few

documents as possible to the following Q&A modules. This is necessary because following algorithms, like answer extraction, use computational complex techniques and work much faster and robust when processing a smaller amount of documents.
So, document retrieval is done in two steps. First, the number of returned documents is fixed and, afterwards, the adequate smoothing technique is optimized to obtain the best performance. We also talk about the experimental setup we used for the TREC 2007 challenge. Here, a different approach was chosen, because two different corpora had to be processed. The rest of this chapter is organized as follows. The used methodology and datasets are described in section VII.2. The baseline experiments and the corresponding results are discussed in section VII.3. Our new experiments concerning document retrieval and fetching for TREC 2007 are mentioned in section VII.4. And finally, section VII.5 concludes the chapter about document retrieval.

2 Methodology

Before optimizing the smoothing algorithms, the dataset and methods used for the experiments are explained. This also includes the pre–processing steps we did before performing the document retrieval.

2.1 Dataset

The training document set or corpus for evaluation is the AQUAINT collection that consists of 1,033,461 documents taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. We selected AQUAINT as some well established standard task, which is helpful to compare our work with the state of the art. Our question set for evaluation contains 50 factoid questions, from TREC topic 1394 to 1443, that look for short, fact–based answers. These topics cover a range of different question type classifications, such as:

- Location: ”In what country did the game of croquet originate?”
- People: ”Who was the first person to run the mile in less than four minutes?”
- Date: ”When did the shootings at Columbine happen?”
- Amount: ”How old do you have to be to get married in South Carolina?”

While conducting document retrieval we do not use this classification information. Some of these questions do not have answers within the corpus. In all our experiments, stemming is applied. No stop word removal is performed, since we do not want to be biased by any artificial choice of stop words and we believe that the effects of stop word removal should be better achieved by exploiting language–modeling techniques. Relevance judgments are obtained from the judged pool of top retrieved documents by various TREC participating retrieval systems. Table VII.1 shows the different datasets and the label used for them in this paper.

For the task of TREC 2007 two new datasets were introduced. The AQUAINT 2 is a newswire corpus similar to AQUAINT. The documents contained in this corpus are proofread, so we can expect the texts to be written in proper English. BLOG 06 is a newly created corpus comprising common blog pages from the web. Therefore, these web sites are not proofread and can contain misspellings, ungrammatical formulations and incorrect punctuation marks. The documents are unprocessed web pages extracted from the web and, thus, include HTML tags and script language fragments (cf. Shen et al. (2007)).

2.2 Used Methods

The input to the system are the corpus and a set of questions. The output is a ranked list of documents for each question. Bellow is an explanation of each of the system components.

KeyFileIndexer & Stemmer: This component builds a key file index of AQUAINT corpus. Stemming is done along with indexing, using the Krovetz stemmer. The generated index is used by each retrieval method.

Question Stemmer: It is responsible for converting questions into queries by stemming

Dataset	**Label**
AQUAINT collection	C
Retrieved document collection	dc
Retrieved document for a given query	$dc(q_i)$
Single document	d
Query collection	Q
Single query	q_i
Single passage	p
Passage collection	pc

Table VII.1: Labels used for different datasets

them. As for KeyFileIndexing, the Krovetz stemmer is used for question stemming. These queries are used by the retrieval methods to perform document retrieval.

Retriever: This component is responsible for the actual retrieval of documents. There are number of retrieval methods that we have tested. They are explained in following section.

3 Experiments

A number of popular retrieval techniques exist, which include both traditional and language modeling techniques. We evaluate the performance of some of these techniques on our test data. The retrieval methods evaluated in this section are standard *tf–idf*, *Okapi*, and the language modeling framework. The *Dirichlet Prior, Jelinek–Mercer*, and *Absolute Discounting* smoothing methods are the three methods that we have tested. They belong, in a general sense, to the category of interpolation–based methods, in which we discount the counts of the seen words and the extra counts are shared by both the seen words and unseen words. The *Lemur toolkit* is used to run the experiments, because it is efficient and optimized for fast retrieval. The *Lemur toolkit* is specially designed for research work. It provides both traditional and language modeling based retrieval algorithms and has been used by many research groups in the IR community. The basic idea behind the language modeling approach is to estimate a language model for each document and rank documents by the likelihood of the query according to the language model.

3.1 Evaluation Methodology

Our goal is to study the behavior of individual retrieval methods and smoothing techniques as well as to compare different methods. Unlike traditional retrieval techniques, in case of language modeling retrieval techniques, we experiment with a wide range of parameter values for each smoothing method. In each run, the smoothing parameter is set to the same value across all queries and documents. While it is certainly possible to set the parameters differently for individual queries and documents through some kind of training procedure, it is beyond the scope of our work. In order to study the behavior of a particular smoothing method, we examine the sensitivity of non–interpolated average precision to variations in a set of selected parameter values. Along with finding the optimal value of smoothing parameters, we also need to find the optimal number of retrieved documents N. Therefore we first fix the number of retrieved documents by comparing the non–interpolated average precision for a varying number of documents retrieved, using each retrieval method. For the purpose of comparing smoothing methods, we first optimize the performance of each method using the non–interpolated average precision as the optimization criterion, and then compare the best runs of each method. The optimal parameter is determined by searching the entire parameter space.

3.2 Results

This section explains results obtained from different retrieval methods. We first derive the expected influence of the number of documents retrieved by plotting the non–interpolated average precision against the document number for each retrieval method. We examine the sensitivity of retrieval performance by plotting the non–interpolated average precision at N documents against the different values of the smoothing parameter. The following section explains the reason for retrieving a finite number of N documents per query.

3.2.1 Document Size Tuning

In this section, we study the behavior of each retrieval technique for different numbers of documents retrieved. We examine the sensitivity of retrieval performance by plotting the non–interpolated average precision, with fixed smoothing parameters for this experiment where required, against a different number of documents retrieved. The smoothing parameter values are taken from previous work (Zhai and Lafferty 2001). For the Dirichlet Prior the value of prior is set to $2,000$, for Jelinek–Mercer the value of λ is fixed at 0.8, and similarly for Absolute Discounting the value of δ is preset to 0.8. The plot in Figure VII.1 displays the non–interpolated average precision for a different number of documents retrieved. It can be seen that with the increase in the number of document, the performance also rises. It can also be seen that the increase in performance after 500 documents is relatively marginal. All retrieval methods show this trend. For the number of retrieved documents N greater than 500 the cost of computing is significantly larger compared to the gain in performance. Therefore N is fixed at 500. Overall the Dirichlet Prior performed best by far. One reason for this could be that on average our queries are not verbose. Our experiments support the claim that language modeling techniques perform better than traditional ones, e.g. tf–idf and Okapi. Another noticeable fact is that performance ordering of retrieval methods is independent from the number of retrieved documents.

3.2.2 Parameter Tuning for Language Modeling Techniques

In this section, we study the behavior of individual smoothing methods. We examine the sensitivity of retrieval performance by plotting the non–interpolated average precision at 500 retrieved documents against the different values of the smoothing parameters. The analysis of our results follows.

For Jelinek–Mercer, the value to λ is varied between zero and one. The plot in Figure VII.2 shows the non–interpolated average precision for different settings of λ. As depicted in the plot, an optimal value of λ is near 0.5, which indicates that our queries are of mixed length. According to Zhai and Lafferty (2001), the optimal point for short queries is around 0.1 and for long queries it is generally around 0.7. This is because long queries need more smoothing and less emphasis is placed on the relative weighting of terms.

For Dirichlet Prior, the value of prior μ is varied between 500 and $5,000$ with intervals of 500. The plot in Figure VII.3 illustrates the non–interpolated average precision for different settings of the prior sample size. As mentioned in Zhai and Lafferty (2001), the optimal prior μ varies from collection to collection and depends on query lengths. For our dataset and questions it is around $1,000$.

For Absolute Discounting, the value to δ is varied between zero and one. The plot in Figure VII.2 shows the non–interpolated average precision for different settings of *d*. The optimal value of δ is near 0.8, which fortifies the claim by Zhai and Lafferty (2001) that the optimal value for δ tends to be around 0.7.

Overall the Dirichlet Prior performed best using prior of $1,000$ and 500 retrieved documents. Then follows Absolute Discounting, which is better than Jelinek–Mercer. The good performance of Dirichlet Prior is relatively insensitive to the choice of μ. Indeed, many non–optimal Dirichlet runs are also significantly better than the optimal runs for Jelinek–Mercer and Absolute Discounting. This is because our queries are not long. Jelinek–Mercer is supposed to perform the best for long queries. According to Zhai and Lafferty (2001), Jelinek–Mercer is much more effective when queries are more verbose. As displayed by Table VII.2, tf–idf

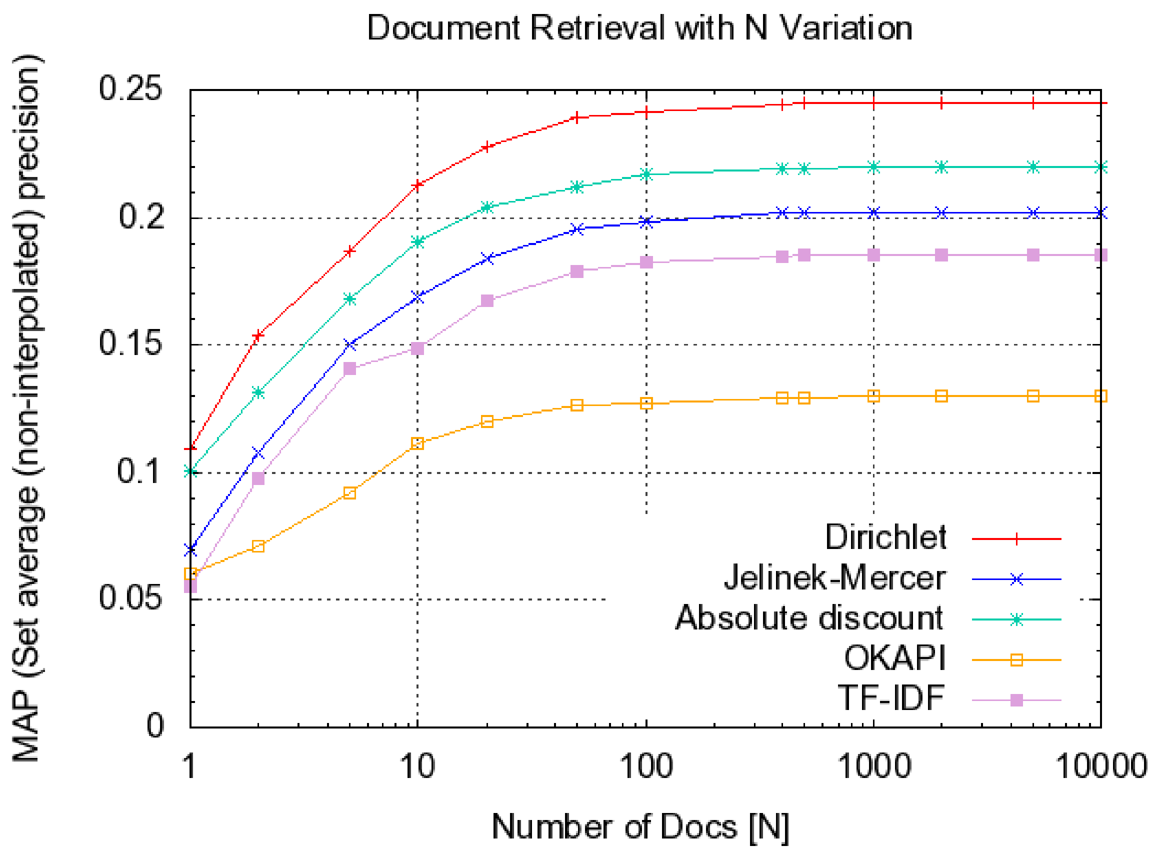


Figure VII.1: Document Retrieval with varying number of documents retrieved. For Dirichlet Prior the value of prior is set to 2000, for Jelinek–Mercer the value of λ is set to 0.8 and for Absolute Discounting the value of δ is set also to 0.8.

performed slightly worse than Jelinek–Mercer, while Okapi performed even worse.

4 Experiments for TREC 2007

The results of the experiments we made for TREC 2007 are mainly taken from Shen et al. (2007).

4.1 Two–stage Document Retrieval

As mentioned in the introduction, for TREC 2007 experiments two new text corpora were used to perform question answering. We carried out three steps to prepare both corpora for document retrieval. In the first step, we extracted the plain text from the BLOG 06 documents by removing HTML tags and script language fragments. For this purpose, we used

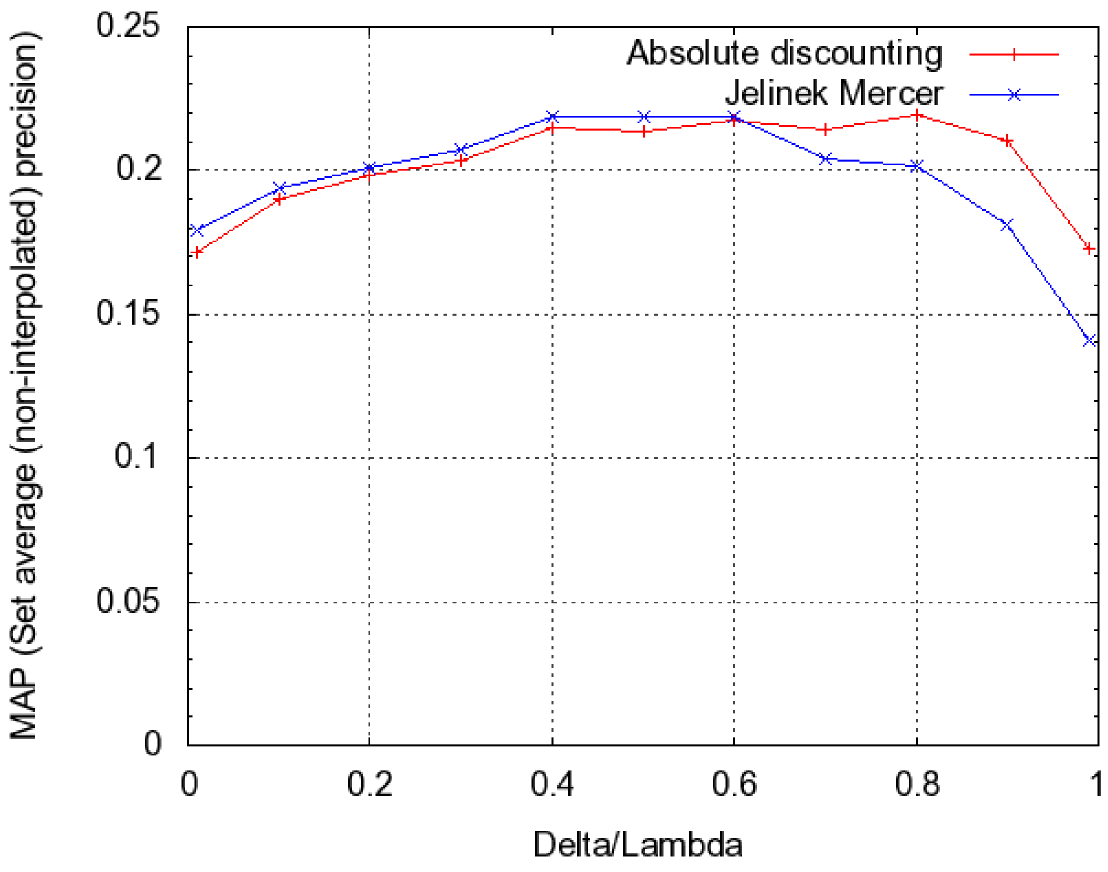


Figure VII.2: Plot of non–interpolated average precision against smoothing parameter, with smoothing parameter varying from 0.01 to 0.99. Number of retrieved documents fixed at 500.

CyberNeko Tools for XNI[1]. We preserved tables of contents and other meta–information that are not part of the actual document. Fortunately, the language model based sentence extraction consistently ignores these text fragments. In the second pre–processing step, sentence boundaries are detected in BLOG 06 and AQUAINT 2 documents using an HMM–based approach. Finally, the documents with annotated sentence boundaries are indexed by the Lemur Toolkit for Language Modeling and Information Retrieval. A single corpus comprising all documents from BLOG 06 and AQUAINT 2 would result in a huge corpus containing 3,215,171 + 906,777 documents. Unfortunately, we were not able to use the language modeling module of Lemur for these data sizes. The other document retrieval methods, such as tf–idf or Okapi, were working on these data but previous experiments on AQUAINT show that the document extraction using language modeling performs much better than tf–idf or Okapi (Merkel and Klakow 2007a). So, we decided to split the document retrieval in two

[1]`http://people.apache.org/~andyc/neko`

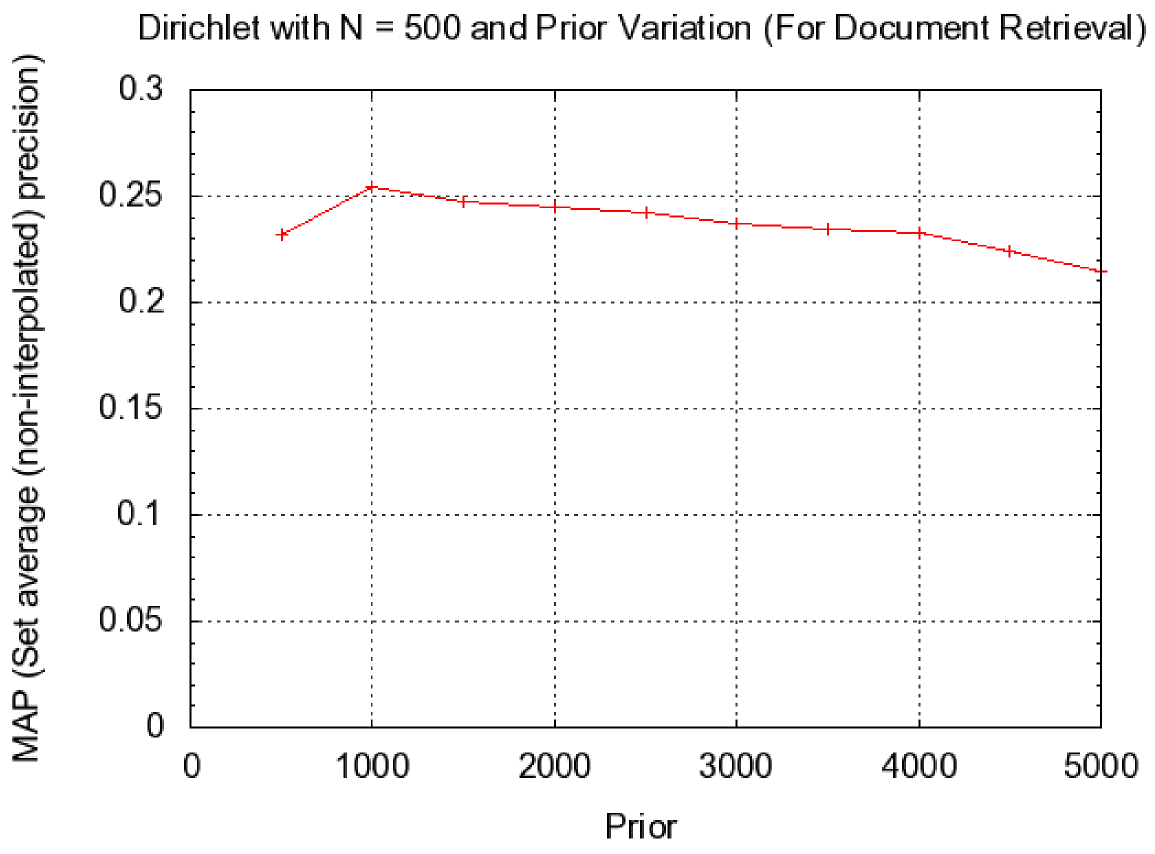


Figure VII.3: Plot of non–interpolated average precision against prior (μ). Dirichlet Prior with prior varying from 500 to 5000. Number of retrieved documents fixed to 500.

stages. In both stages we used language modeling to extract the documents.

1. In the first stage, documents are extracted separately from BLOG 06 and AQUAINT 2. We extract the same amount of documents from both corpora. The size of documents to be extracted is larger in the second stage than in the first. The more documents are extracted during the first stage the better a one–stage document retrieval is approximated.

2. After extracting documents from both corpora we create a new corpus from the extracted documents. This corpus is much smaller than a merged corpus of both corpora.

3. After building the merged corpus the relevant documents are extracted for further processing.

Method	**Parameter**	**MAP**
Dirichlet Prior	μ = 1000	0.254
Jelinek–Mercer	λ = 0.5	0.219
Absolute Discounting	δ = 0.8	0.219
tf–idf	-	0.185
Okapi	-	0.130

Table VII.2: Non–interpolated average precision for best run of each retrieval methods. With μ of 1000, δ of 0.8, and λ of 0.5

Table VII.3 shows the results obtained by different retrieval methods. In our experiments we used the question set from TREC 2006 and extracted the documents from AQUAINT. 30 documents were extracted in each run, because our QA System performs best with this number of extracted documents. We obtained the parameter settings for each model from previous experiments (Hussain, Merkel, and Klakow 2006). We observe that tf–idf and Okapi perform worse than language modeling in a one–stage document retrieval. Our two–stage document retrieval also performs better on a single corpus than the one–stage document retrieval. Furthermore, the amount of irrelevant documents that are ranked higher than relevant documents decreases in our two–stage document retrieval. This can be explained by the filtering out irrelevant documents in the first stage resulting in a less noisy language model in the second retrieval stage. Overall, the language model based document extraction performs better than traditional document extraction methods like tf–idf or Okapi. The results in these experiments confirm our previous experiments.

4.2 Dynamic Document Fetching

In our experiments we discovered that slight variations in the number of extracted documents affect the overall performance of our QA System. Using a question type independent document retrieval, we obtain best results when 30 documents are extracted per question. Additionally, we observed that the global change in performance induced by varying the

Method	**MAP**	**R-prec**	**P@10**	**bPref**
Two–Stage	0.2646	0.335	0.1353	0.6549
One–Stage	0.2539	0.2328	0.1342	0.5514
Okapi	0.2249	0.2088	0.1300	0.5106
tf–idf	0.2052	0.188	0.1069	0.4917

Table VII.3: Results of Document Retrieval: TREC 2006 question set on AQUAINT.

	One–Stage	**Two–Stage**
MAP	0.2661	0.2646
R–Prec	0.2328	0.335
P@10	0.1342	0.1353
bPref	0.8116	0.6549

Table VII.4: Results of Document Retrieval: Comparison between One–Stage and Two–Stage Retrieval using BLOG06 and Aquaint2 corpus.

number of retrieved documents does not always correlate with the changes in performance observed on the individual question types. Figure VII.4 illustrates the dependency of extracted documents, question type and number of correct answers. We can observe, for example, that for the coarse question type HUM, extracting 10 documents leads to worse performance than extracting 30 documents. The question type LOC, on the other hand, behaves in the opposite way.

The fine grained question types show similar changes in performance. With the optimal

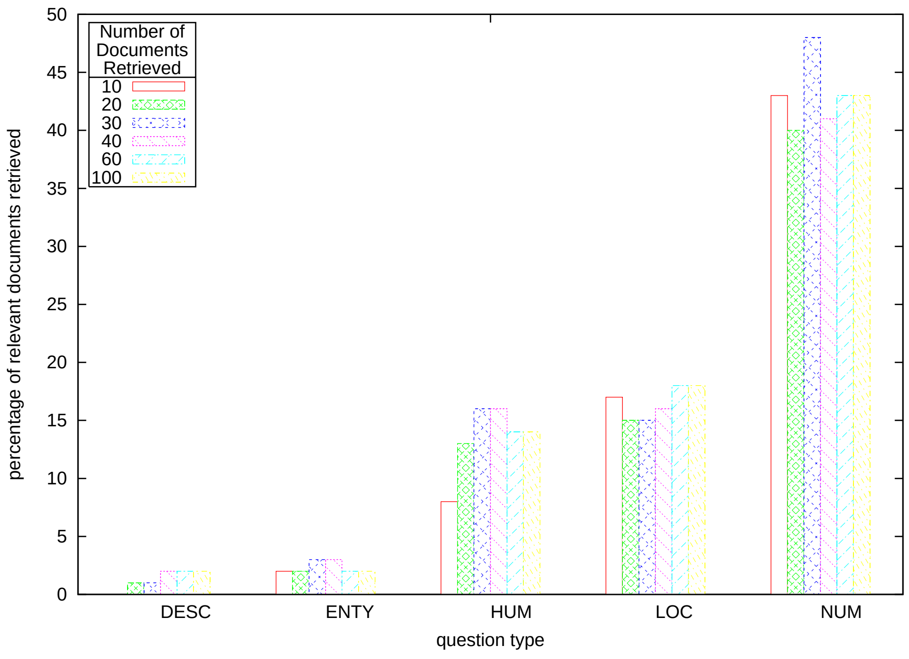


Figure VII.4: Correct Answers per Question Type and Extracted Documents.

number of documents for each question type we can improve the performance of our system for the TREC 2006 question set on AQUAINT from 83 correct factoid answers to 93 correct answers. This is an increase of 12%. But this optimization would lead to overfitting, so we decided to change only the number of documents for question types with clear

maxima and leave other question types that fluctuate unchanged. Thus, we get a document retrieval which improves the performance by 6 questions out of 567, an increase of 7%.
The dynamic document extraction could be further improved by including artificial question type dependent terms. For example, a special document extraction for the question type LOC using the artificial term *location* for the presence of an entity of that kind in a candidate answer document increases the performance on these questions by 4 correct answers. Such improvements need, however, further investigation to avoid overfitting.

5 Conclusion

In this chapter, we presented our experiments for the task of document retrieval for question answering systems. We showed that it makes sense to downsize the number of returned documents for doing question answering. So, a similar performance was achieved by processing a smaller number of documents. After fixing the number of returned documents, the used document retrieval methods were optimized. For our experiments, we used standard language model based smoothing techniques and compared them to some traditional algorithms, like tf–idf or Okapi.
Our experiments showed that the Dirichlet Prior performed best with prior of $1,000$ and that keeping the number of documents retrieved to 500 is both efficient and sufficient. The retieved document sets in this chapter are then further used as a baseline for doing passage retrieval in chapter VIII.

Chapter VIII

Passage Retrieval

Passage retrieval is an essential part of question answering systems. In this chapter, we use statistical language models to perform this task. Previous work has shown that language modeling techniques provide better results for both, document and passage retrieval.
The motivation behind this section is to define new smoothing methods for passage retrieval in question answering systems. The final objective is to improve the quality of question answering systems in order to isolate the correct answer by choosing and evaluating the appropriate section of a document.
In this work we use a three–step approach. The first two steps are standard document and passage retrieval using the Lemur toolkit. As a novel contribution we propose as the third step a re–ranking using dedicated backing–off distributions. In particular backing–off from the passage–based language model to a language model trained on the document from which the passage is taken shows a significant improvement. For a TREC question answering task we can increase the mean average precision from 0.127 to 0.176.
The content of this chapter is mainly taken from Hussain, Merkel, and Klakow (2006).

1 Introduction

Recently lot of work has been carried out on open–domain Question Answering Systems. These Q&A Systems include an initial document and/or passage retrieval step. Retrieved passages are then further processed using a variety of techniques to extract the final answers. The passage retrieval method strongly influences the performance of QA System. This is especially true for real systems where computational resources are limited. A good passage retrieval system means that only a small number of top–ranked passages need to be analyzed to find the answer. In this paper we compare the existing retrieval methods, both traditional and language modeling based ones, for document and passage retrieval. We used the AQUAINT document collection as training and test corpus. Out of all methods tested, by choosing the best passage retrieval method as our baseline, we define and test new language models to improve retrieval performance. These language models are defined by different data collections (passage collection, document collection, corpus) and are interpolation based unigram language models.

The rest of this chapter is organized as follows. Related work is discussed in section VIII.2. Section VIII.3 presents the passaging of documents and passage retrieval performed. Our experiments are shown in section VIII.4, whereas section VIII.5 explains the process of re–ranking. We conclude the paper by discussing our results and future work in section VIII.6.

2 Related Work

This section discusses the state of the art in the field of passage retrieval.

Passage retrieval is an important component of QA Systems and it directly influences overall performance.

For example, Clarke et al. (2000) introduces a passage retrieval system for the TREC *Question Answering track* (*MultiText*). They use a question pre–processing, a passage retrieval and a passage post–processing step to select the top five text sections out of a set of documents. In the pre–processing step, the question is parsed and "selection rules" (patterns) are defined. Each text block for the passage retrieval algorithm can start and end with any query term. The score of such a passage is calculated by the text size and the number of occurring query terms. Then a new passage with a required length around the center point of the original passage is produced. Finally, the patterns are used to post–process the passage retrieval results.

Another interesting approach is presented by Corrada–Emmanuel, Croft, and Murdock (2003). In their paper, three methods to score relevant passages are shown. They compare the famous query likelihood, relevance modeling and a bigram answer model. In this model, the expected answer type is taken into account. Therefore, text selections are replaced by their named entity tag and an answer model is trained. Then three different methods for backing–off the bigram are used. They show that the bigram method provides a performance superior to the other approaches.

Zhang and Lee (2003) also developed a language modeling approach to passage question answering. Their system consists of a question classification component and a passage retrieval component.

Tellex et al. (2003) carried out a Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. They evaluated MITRE's passage retrieval algorithm presented by Light, the Okapi BM25 weighting scheme, the MultiText algorithm, IBM's passage retrieval algorithm, SiteQ's passage retrieval algorithm, Alicante's passage retrieval algorithm, ISI's passage retrieval algorithm, and one new algorithm of their own called Voting. They implemented a voting scheme that scored each passage based on its initial rank and also based on the number of answers the other algorithms returned from the same document. The most important findings of their work is that boolean querying performs well for question answering, that the choice of the document retriever is very important and, that the best algorithms use density based scoring.

Some work has been done to improve the document retrieval by performing passage retrieval.

Callan (1994) examined passage level evidence in document retrieval. Three different approaches were taken to determine passage size and location: paragraphs, bounded paragraphs, and fixed–length windows.

The use of language modeling for passage retrieval and comparison with document–based retrieval was done by Liu and Croft (2002). They also made a comparison with results from the INQUERY search engine.

There is also some further literature in the field of language model based passage retrieval for QA. In Zhang and Lee (2004) an LM based question classification and an LM based passage retrieval approach is shown. To optimize their text selection, they first look at an initial set of relevant passages and construct a language model. Then, relevant web data is used to built a second language model. Finally, they mix the two models and include some further constraints like answer type and answer context information.

Cai et al. (2004) explored the use of page segmentation algorithms to partition web pages into blocks and investigated how to take advantage of block–level evidence to improve retrieval performance in the web context.

3 Methodology

Passage retrieval is mainly used for three purposes. First, passage retrieval techniques have been extensively used in standard IR settings, and have proven effective for document retrieval when documents are long or when there are topic changes within a document. Second, from an IR system user’s point, it may be more desirable that the relevant section of a document is presented to the user than the entire document. Third, passage retrieval is an integral part of many question answering systems. We perform passage retrieval for question answering systems. This section explains our methodology to establish a baseline using existing techniques developed for passage retrieval, which include both traditional and language modeling based retrieval methods. For our experiments, we first retrieve documents (cf. chapter VII), then split these documents into passages. We call these passages *passage documents*, and use the collection of these passage documents as a corpus for retrieval of passages relevant to each query.

3.1 Passage Making

Passages are created using the following procedure. The top 500 retrieved documents are selected (early tests showed that increasing this number had no significant effect on system performance), see chapter VII for details of the document retrieval. The selected documents are then split into passages by a ”passage maker”. Our passage making technique is based on document structure (cf. Berger and Lafferty (1999), Agichtein and Gravano (2000), and Clarke et al. (2000)). This entails using author–provided marking (e.g. period, indentation, empty line, etc.) as passage boundaries. Examples of such passages include paragraphs, sections, or sentences. Since our corpus is well structured (SGML form), we used paragraphs as passages.

3.2 Dataset

The query topics are the same as used for document retrieval (cf. chapter VII). For each query we have a distinct corpus consisting of passages created from the top 500 retrieved documents. See chapter VII for more on document retrieval.

3.3 Used Methods

For passage retrieval we used the same set of retrieval methods as for document retrieval explained in chapter VII. Likewise, the evaluation methodology is also the same as for document retrieval (cf. chapter VII).

4 Experiments

We first derive the expected influence of the number of passages retrieved by plotting the non–interpolated average precision against the size of the retrieved passage set for each retrieval function. Then, we examine the sensitivity of retrieval performance by plotting the non–interpolated average precision at N passages against the different values of the smoothing parameter.

4.1 Results

In this section, we study the behavior of each retrieval technique for different number of retrieved passages, which is similar to what we did for document retrieval in chapter VII. We examine the sensitivity of retrieval performance by plotting the non–interpolated average precision, with fixed smoothing parameters where required, against the different number of retrieved passages. For Dirichlet Prior the value of the prior μ is set to $1,000$, for Jelinek–Mercer the value of λ is set to 0.4, and for Absolute Discounting the value of δ is set also to 0.4. The plot in Figure VIII.1 shows the non–interpolated average precision for different number of retrieved passages. It can be seen that with the increasing number of retrieved passage documents, the performance also increases. But the increase in performance after 500 passages is relatively marginal. This trend is approved by all retrieval methods. For passage document number N larger than 500 the cost of computing is significantly large compared to the gain in performance. Therefore the passage document number N is fixed at 500. Overall Dirichlet Prior performed best. Our experiments also show that there is no significant performance difference between retrieval methods, i.e. the curves are pretty close to each other. The performance of Okapi is slightly worse than language modeling techniques. Tf–idf showed worse performance. Another noticeable fact is that Dirichlet Prior performance improves significantly for N between 1 and 10.

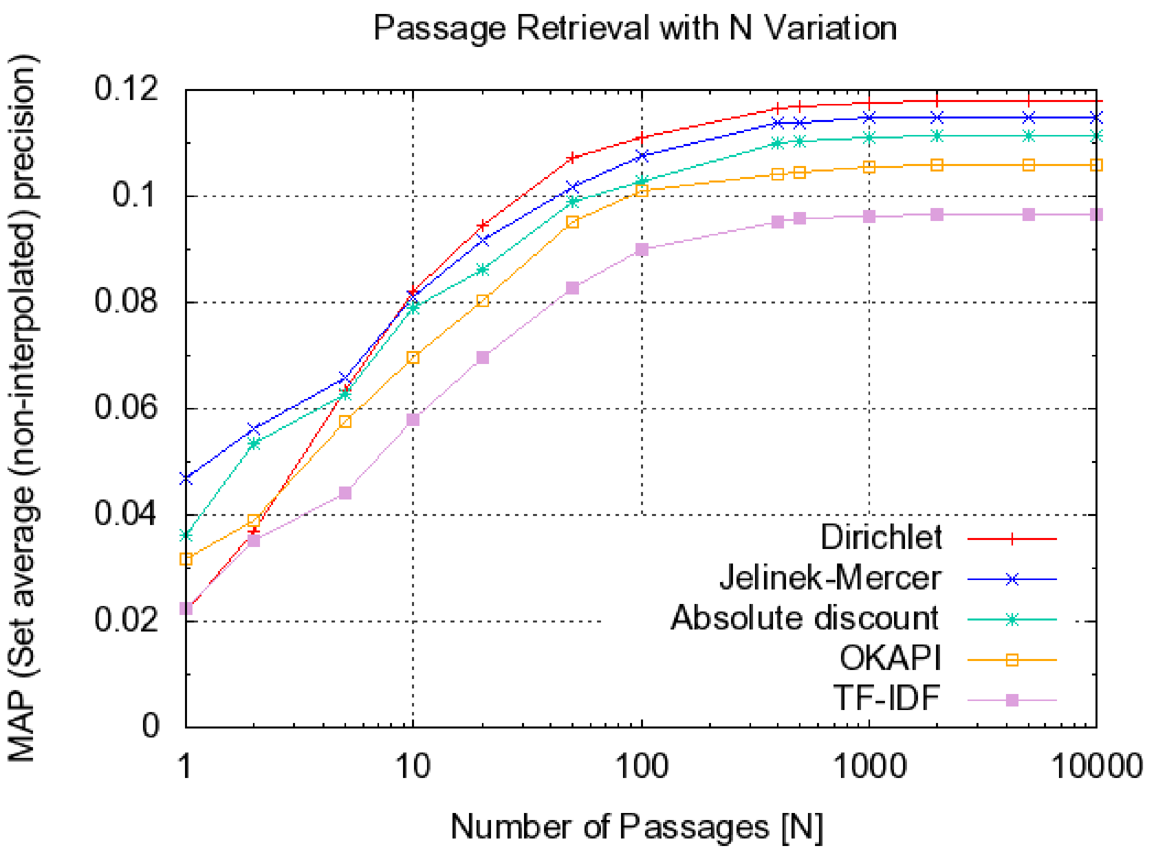


Figure VIII.1: Passage Retrieval with varying number of retrieved passages. For Dirichlet Prior the value of prior is set to 1000, for Jelinek–Mercer the value of λ is set to 0.4 and for Absolute Discounting the value of δ is set also to 0.4.

4.1.1 Parameter Tuning for Language Modeling Techniques

This section covers the behavior of individual smoothing methods, as we did for document retrieval in chapter VII. We examine the sensitivity of retrieval performance by plotting the non–interpolated average precision at 500 passages against different values of the smoothing parameter. Below, there is an analysis of our results.

Jelinek–Mercer smoothing: For Jelinek–Mercer, the value of λ is varied between zero and one. The plot in Figure VIII.2 shows non–interpolated average precision for different settings of λ. As depicted in the plot, optimal value of λ is near 0.4, which indicates that our queries are of mixed length. According to Zhai and Lafferty (2001), for short queries the optimal point is around 0.1 and for long queries the optimal point is generally around 0.7, as long queries need more smoothing and less emphasis is placed on the relative weighting of terms.

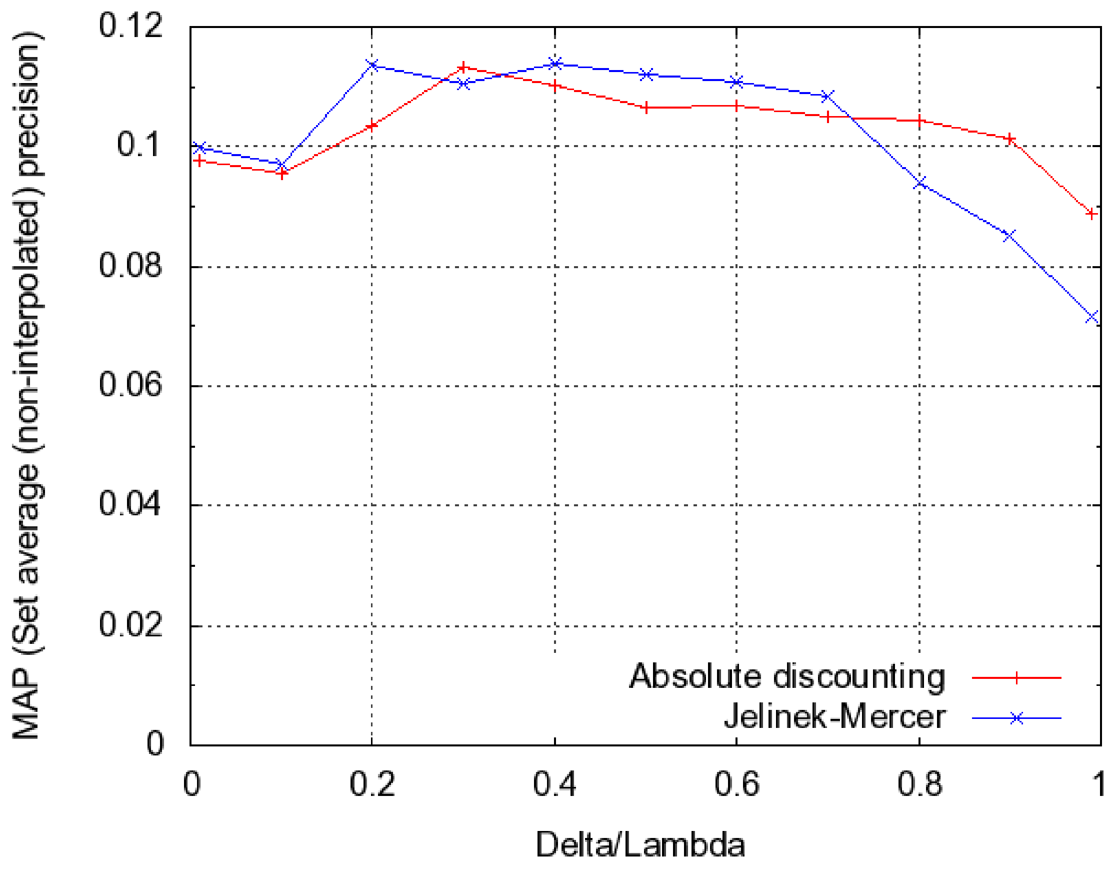


Figure VIII.2: Plot of non–interpolated average precision against smoothing parameter, with smoothing parameter varying from 0.01 to 0.99. Number of retrieved passages fixed at 500.

Dirichlet Prior: The value of Dirichlet Prior μ is varied between 1 and $5,000$ with intervals of 500. The plot in Figure VIII.3 illustrates the non–interpolated average precision for different settings of the prior sample size. As mentioned in Zhai and Lafferty (2001), the optimal prior μ varies from collection to collection and depends on query lengths. For our dataset and questions it is around 500.

Absolute Discounting: The value of δ is varied between zero and one. The plot in Figure VIII.2 shows the non–interpolated average precision for the different settings of *d*. The optimal value of δ is near 0.3.

Overall the Dirichlet Prior performed best using μ of 500 and 500 retrieved passage documents. Then follows Jelinek–Mercer, which is slightly better than Absolute Discounting. But the difference of performance is not very significant. Okapi performed slightly worse than Absolute Discounting while tf–idf performed worst.

Table VIII.1 gives a comparison of the best runs by each technique.

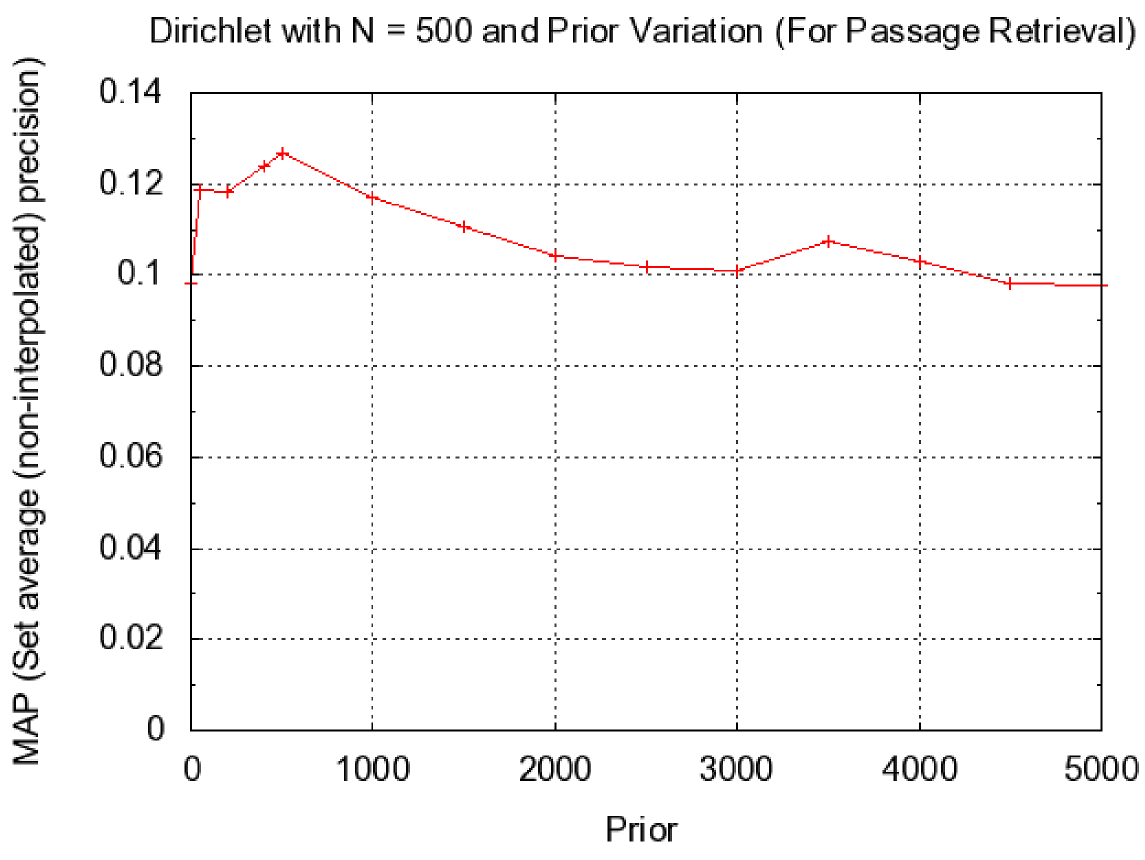


Figure VIII.3: Plot of non–interpolated average precision against prior (μ). Dirichlet Prior with prior varying from 500 to 5000. Number of retrieved passages fixed at 500.

5 Passage Re–Ranking

This section explains our statistical language model approach, which is based on an interpolation smoothing scheme. Since Lemur is not flexible enough to implement such custom models, we shifted to our own language modeling toolkit. This toolkit is very flexible in generating custom language models. It uses perplexity to rank the documents. To check the similarity between the two toolkits, an experiment was carried out using Jelinek–Mercer

Method	**Parameter**	**MAP**
Dirichlet Prior	μ = 500	0.127
Jelinek–Mercer	λ = 0.4	0.114
Absolute Discounting	δ = 0.3	0.113
tf–idf	-	0.105
Okapi	-	0.096

Table VIII.1: Non–interpolated average precision for best run of each retrieval methods

smoothing technique to regenerate the results produced by Lemur. These results confirmed the validity of results generated by our toolkit.

5.1 Experimental Setup

Our experimental setup consists of document collections generated by experiments explained in previous sections. Following sections explain our datasets, experimental methods, and the system architecture.

5.1.1 Dataset

The query topics are the same as used for document retrieval (chapter VII). The corpus C for evaluation is the AQUAINT collection that consists of documents taken from the New York Times, the Associated Press, and the Xinhua News Agency newswires. Also, we have the document collection dc and the passage collection pc obtained from our previous experiments. All these collections are stemmed and no stop word removal is performed.

5.1.2 Evaluation Methodology

Our toolkit uses perplexity to rank the documents. For the purpose of studying the behavior of an individual language model, we select a set of representative parameter values and examine the sensitivity of non–interpolated average precision MAP to the variation in these values. In question answering mean reciprocal rank (MRR) is also widely used. We checked the correlation of MRR and MAP on question answering tasks. For consistency with the document retrieval, we report MAP throughout the chapter.

5.1.3 Experimental Methods

Our experimental methods are language modeling based. We have defined a number of language models using Jelinek–Mercer smoothing techniques.

5.2 System Architecture for Passage Re–Ranking

This section explains the complete architecture of our experimental setup. We defined and tested a series of language models; Table VIII.2 gives a listing of these language models, which are explained in following subsections. Language models described in this section utilize the vocabulary closed with the query and the value of interpolation parameter is varied between zero and one. The main difference between these models is the background collection.

5.2.1 Language Model I ($pdclm$)

This language model is defined as linear interpolation between unigram language models defined on passage and related document collection. Whereas each passage is taken from related retrieved passages (section VIII.3) and the related document collection consists of 500 top ranked documents retrieved (chapter VII), for a given query. As perplexity is given by the formula

$$PP = exp[-\sum_{w} f(w) \log P(w)]$$

where *f(w)* is the relative frequency of words in the query and the probability is

$$P(w) = (1-\lambda)P_{ml}(w|p) + \lambda P(w|dc),$$

where P_{ml} is maximum likelihood of word *w* in passage *p*. Figure VIII.4 explains the complete setup to re–rank passages using this language model.

Standard Tree: It contains statistical information for given passage being ranked. We build one standard tree per passage. This tree is the basis of the passage language model.

Language Model	**Label**
Interpolation between language models for passage and relevant document collection.	$pdclm$
Interpolation between language models for passage and relevant passage collection.	$ppclm$
Interpolation between language models for passage and single document from which passage is taken.	$pdlm$
Interpolation between language models for passage and complete corpus.	$pClm$

Table VIII.2: Labels used for different language models

Background Standard Tree: It consists of statistical information for the complete document collection used for the backing–off language model.

Re–ranker: It is responsible for re–ranking the collection of 500 related passages per query. It utilizes standard tree and background tree containing statistical information required by language models.

Vocabulary: Our word list consists of all the words in document collection closed with words from the query.

5.2.2 Language Model II ($ppclm$)

This language model is similar to $pdclm$ explained above with the related passage collection consisting of 500 top ranked passages retrieved as the background collection. For this language model the probability is

$$P(w) = (1-\lambda)P_{ml}(w|p) + \lambda P(w|pc),$$

5.2.3 Language Model III ($pdlm$)

Here again the language model differs from $pdclm$ in the background collection. The background collection is the single document from which the passage was extracted i.e. the document containing the passage being ranked. For this language model, the probability for calculating the perplexity is

$$P(w) = (1-\lambda)P_{ml}(w|p) + \lambda P(w|d),$$

Figure VIII.5 explains the complete setup to re–rank passages using this language model.

Re–ranker: Same as in section VIII.5.2.1.

Vocabulary: Our word list consists of all the words in the single document containing the passage being ranked, closed with words from query.

Standard Tree: Same as in section VIII.5.2.1.

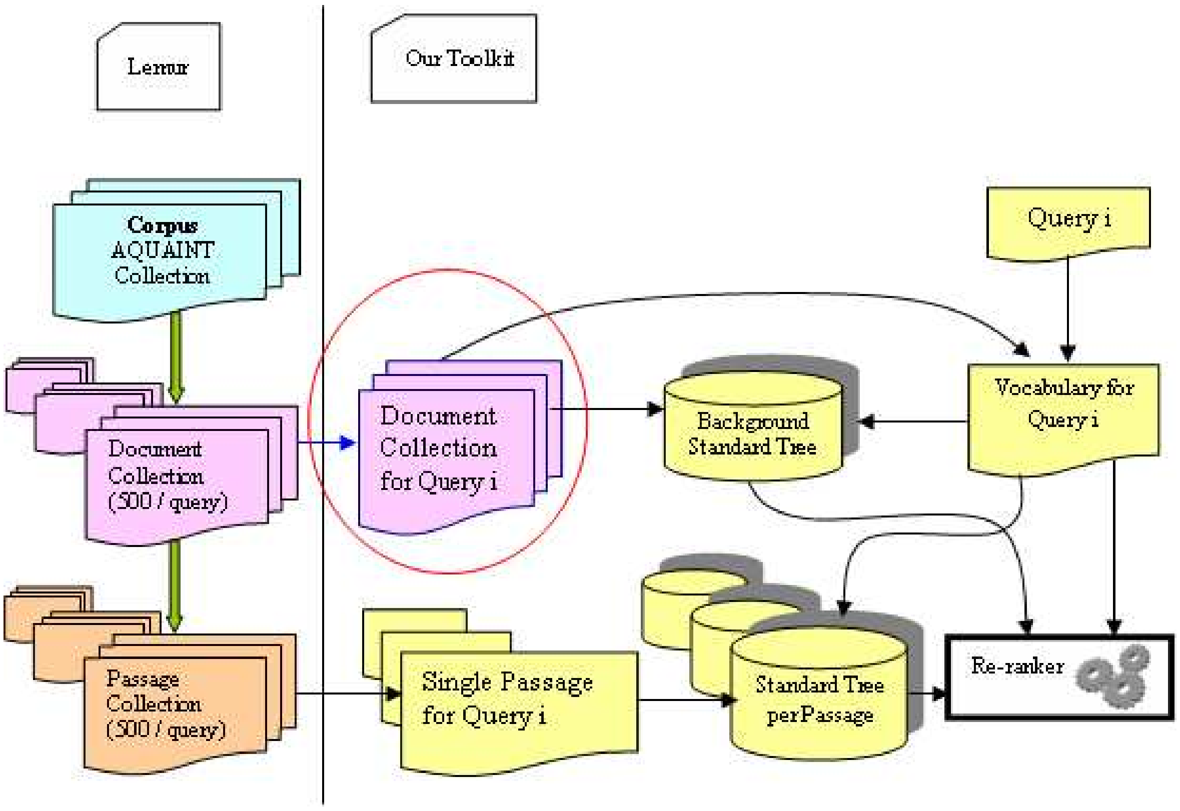

Lemur
Our Toolkit
Corpus AQUAINT Collection
Document Collection (500 / query)
Document Collection for Query i
Background Standard Tree
Query i
Vocabulary for Query i
Passage Collection (500 / query)
Single Passage for Query i
Standard Tree perPassage
Re-ranker

Figure VIII.4: Re–ranking setup for the $pdclm$ language model.

Background Standard Tree: It consists of statistical information for the single document containing the passage being ranked. We build one standard tree per document.

5.2.4 Language Model IV ($pClm$)

For this language model the background collection is the complete corpus (AQUAINT document collection). The probability for calculating the perplexity is

$$P(w) = (1 - \lambda)P_{ml}(w|p) + \lambda P(w|C),$$

5.3 Experimental Results

This section discusses the results of our experiments. Jelinek–Mercer smoothing is used in all of the experiments with the value of λ varied from 0.01 to 0.99. We first reproduce Jelinek–Mercer smoothing results of passage retrieval (section VIII.3) with a $pdclm$ language model. Then other language models are defined and tested to improve baseline.

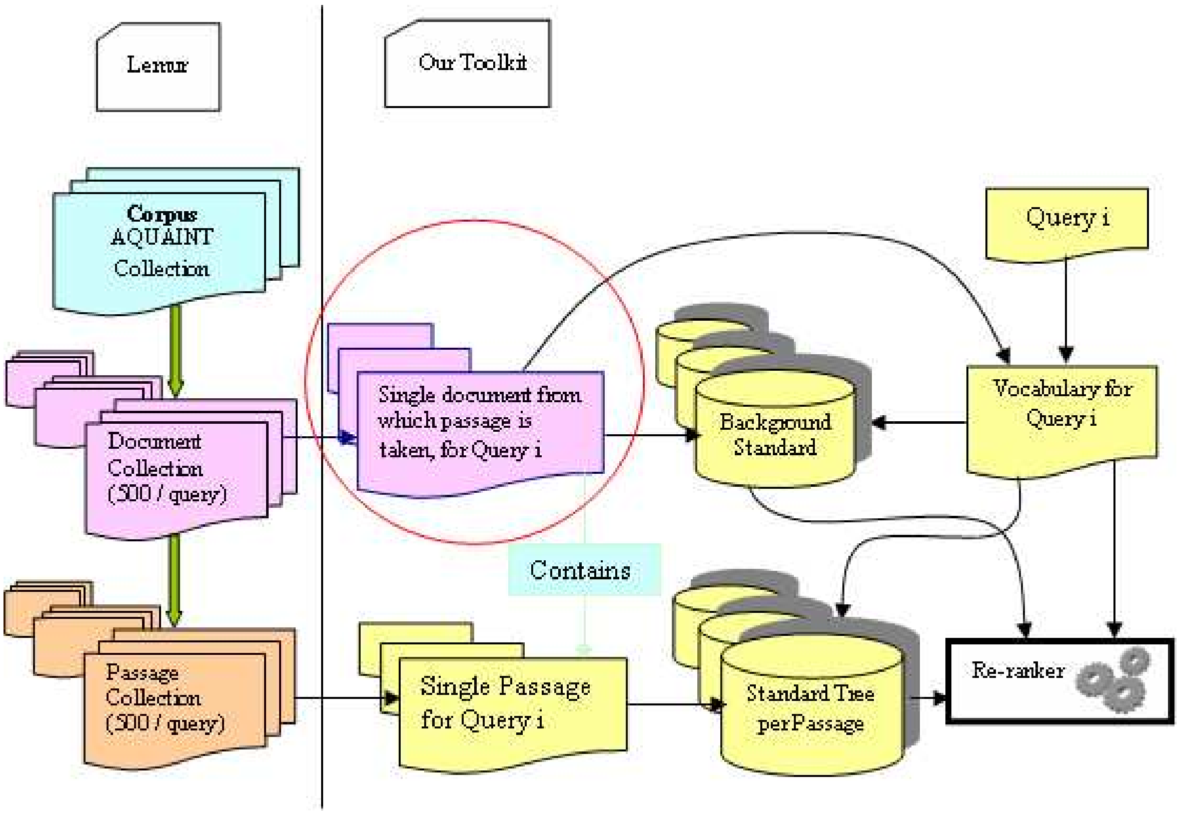

Lemur
Our Toolkit
Corpus AQUAINT Collection
Document Collection (500 / query)
Passage Collection (500 / query)
Single document from which passage is taken, for Query i
Contains
Single Passage for Query i
Background Standard
Standard Tree per Passage
Query i
Vocabulary for Query i
Re-ranker

Figure VIII.5: Dataset flow diagram for the $pdlm$ language model.

5.3.1 Language Model I ($pdclm$)

This language model is a reproduction of the language model with Jelinek–Mercer smoothing used in section VIII.3. It reproduces our previous results, which confirmed the validity of results generated by our language modeling toolkit. The plot in Figure VIII.6 shows the non–interpolated average precision for different settings of λ. It illustrates that the optimal value of λ is near 0.4.

5.3.2 Language Model II ($ppclm$)

Figure VIII.6 shows the results using this language model by line with squares as points. The optimal value for λ is 0.05. According to Zhai and Lafferty (2001) small λ means more emphasis on relative term weighting, which implies that the passage collection has a smaller role in ranking than the passage itself. This might be due to a small size of passages and a variety in topics they discuss. With this language model we observe a 20% decrease over the baseline.

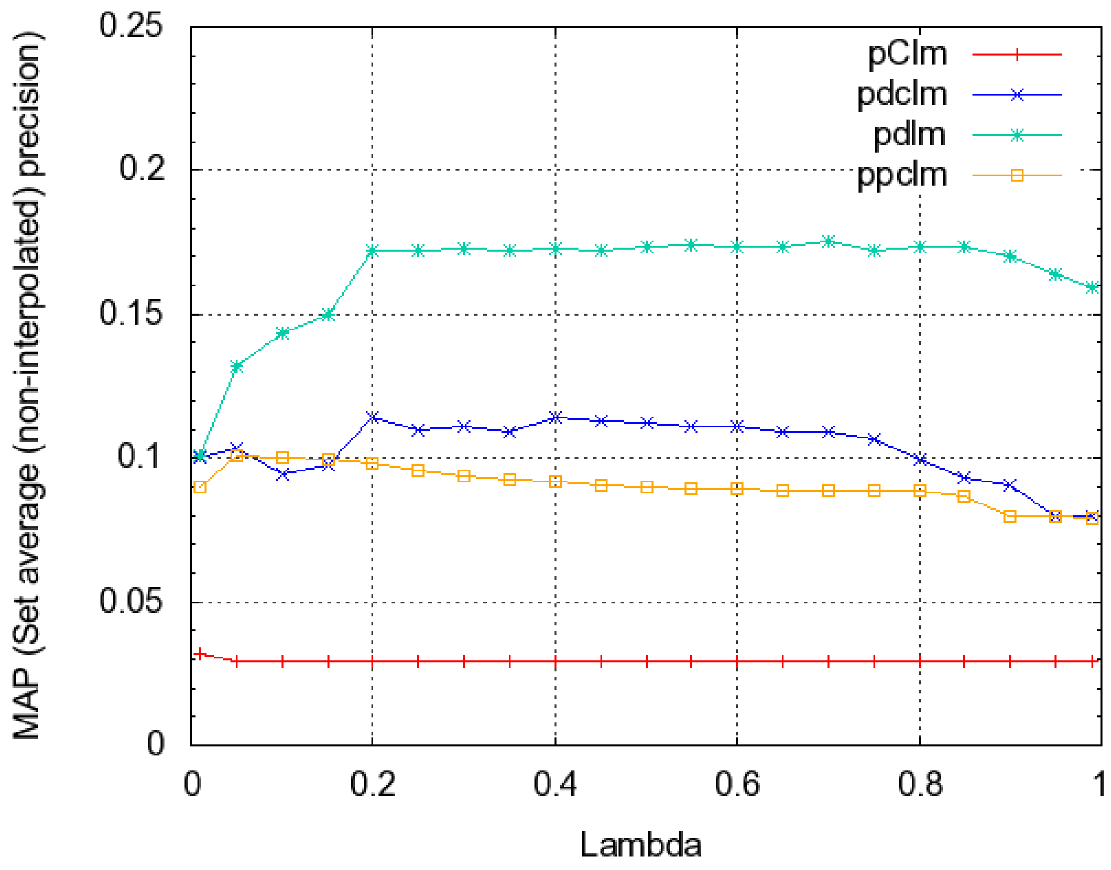


Figure VIII.6: Plot of non–interpolated average precision against λ. Jelinek–Mercer b/w passage and different background collections with λ varying from 0.01 to 0.99.

5.3.3 Language Model III ($pdlm$)

The line with asterisks in Figure VIII.6 shows the results using this language model. The optimal value for λ is 0.70. The value of λ near the middle of the parameter space suggests that both passage and document collection are equally important for ranking. Yet the document is given a bit more importance than the passage, which is quite understandable as passages are of small size and sometimes they miss some related terms from query. With this language model we have more than 38% improvement over the baseline, which is quite a significant figure. This is no surprise in so far as both the document and the passage being used discuss the same topic. The related document size is relatively small compared to the document or passage collection, which also contributes to the improvement in results.

5.3.4 Language Model IV ($pClm$)

Figure VIII.6 shows, using the line with bars, the results using this language model. The optimal value of λ is 0.01. A small λ means more emphasis on relative term weighting, which means that the corpus plays no role in ranking the passages. This is because of the large size of the corpus, with lots of irrelevant terms. It is also clear from Figure VIII.6 that this language model performed worst of all our proposed models.
Table VIII.3 displays the best results of each language model.

6 Conclusion and Future Work

We studied the problem of language model smoothing in the context of passage retrieval for QA Systems and compared it with traditional models, including tf–idf and Okapi. We then examined three popular interpolation–based smoothing methods (Jelinek–Mercer, Dirichlet Prior, and Absolute Discounting), and evaluated them using the AQUAINT retrieval testing collection.
We defined a number of language models based on the Jelinek–Mercer smoothing technique, and found out that interpolation between language models for the passage and the single document from which the passage is extracted provided more than 38% improvement, which is quite significant for QA Systems.
Table VIII.4 lists the best runs for our document retrieval, passage retrieval and re–ranking experiments. Our best performing language model can be used for real QA Systems. We used one of the basic approaches to passage generation. One problem with our approach is that it does not take the topic shift within a passage into consideration. It also does not account for topics which spread over multiple passages. Other more sophisticated passaging techniques could further improve our proposed language model. The language models we proposed and tested are all unigram models. As previous work depicts, higher order

Method	**Lambda**	**MAP**
pdclm	0.40	0.114
ppclm	0.05	0.101
pdlm	0.70	0.176
pClm	0.01	0.032

Table VIII.3: Non–interpolated average precisions for the best run of each language model. Passage re–ranking using the document language model for smoothing improves MAP by 39% over the best result from Lemur.

language models will improve retrieval performance.

Step	**Method**	**Lambda**	**MAP**
Document Retrieval	Dirichlet Prior	μ = 1000	0.254
Passage Retrieval	Dirichlet Prior	μ = 500	0.127
Re–ranking	pdlm	λ = 0.70	0.176

Table VIII.4: Summary of Document and Passage Retrieval results.

It is also very important to study how to exploit the past relevance judgements, the current query and the current database to train the smoothing parameters, since, in practice, it would not be feasible to search the whole parameter space as we did in this thesis. One possibility to determine the parameters automatically could be the use of Leaving–one–out.

Chapter IX

Sentence Retrieval

A retrieval system is a very important part in a question answering framework. It reduces the number of documents to be considered for finding an answer. For further refinement, the documents are split up into smaller chunks to deal with the topic variability in larger documents. In our case, we divided the documents into single sentences. Then a language model based approach was used to re–rank the sentence collection.
For this purpose, we developed a new language model toolkit. It implements all standard language modeling techniques and is more flexible than other tools in terms of backing–off strategies, model combinations and design of the retrieval vocabulary. With the aid of this toolkit we carried out re–ranking experiments with standard language model based smoothing methods. On top of these algorithms we developed some new, improved models including dynamic stop word reduction and stemming. We also experimented with query expansion depending on the type of a query. On a TREC corpus, we demonstrated that our proposed approaches provide a performance superior to the standard methods. In terms of Mean Reciprocal Rank (MRR) we can prove a performance gain from 0.31 to 0.39.
This chapter is mainly based on Merkel and Klakow (2007a).

1 Introduction

The major goal of a question answering (QA) system is to provide an accurate answer to a user question. Compared to a standard document retrieval framework, which just returns relevant documents to a query, a QA system has to respond with an adequate answer to a natural language question. Thus, the process of retrieving documents is just a part of

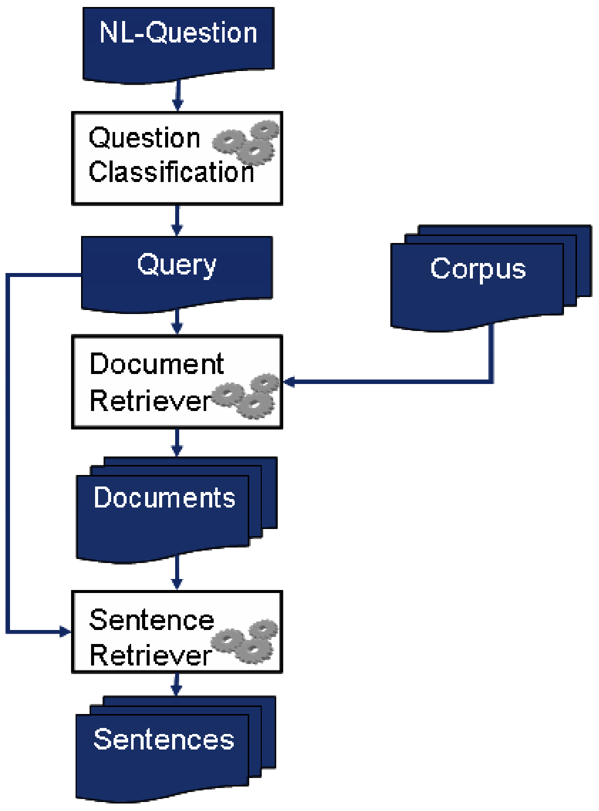


Figure IX.1: A general architecture for question answering systems

a complex sequence. In order to provide the user with an answer, possible candidates have to be extracted from the documents. To simplify this procedure, the text is segmented into smaller passages and a further retrieval step is done. This process is called sentence retrieval, if the passage contains just one sentence.

In this chapter, we describe an experimental setup for comparing different language models in order to improve sentence retrieval within a question answering context. Figure IX.1 shows the general construction of a question answering system. It starts with the analysis of a natural language (NL) question (upper part). Generally, in this *Question Analyzer*, the expected answer type is determined, but it is also possible to make some other deep analyses like part of speech (POS) tagging or named entity recognition. The result is a processed query, which can be used for the following retrieval steps.

The next step is the document retrieval (*Document Retriever*). In a QA system, the retrieval framework is a very crucial part. It is used to decrease the number of documents in a potentially large corpus. This is done in order to reduce the search space in which a correct answer has to be found. It is necessary to reduce the search space because the following components may use long–lasting deep analysis algorithms which strongly depend on the size of the processed corpus. Therefore, it is important to process just the documents which seem relevant to a query to get answers within an appropriate period of time (see chapter VII for more information).

But within such a limited collection, there might still be large documents. Or within single documents some topic changes might occur. If this is the case, again, the following components have to analyse more text than necessary in order to find the correct answer. To overcome this problem, it is essential to further reduce the size of the collection. This can be done by splitting up the text segments into smaller chunks of passages. After dividing the documents, a second retrieval step is necessary in order to re–rank the new passage collection (*Passage Retriever*) using the pre–processed query. By doing so, the corpus size and thus the search space is reduced again (see chapter VIII for a more detailed view on passage retrieval).

In a final step, the passage collection is processed by the *Answer Extractor*. Here, the single passages are analyzed by computing the part of speech, the named entities and other linguistic features. Finally, the most probable answers are selected and returned by the system (cf. chapter II for more information about a general Q&A architecture).

For our experimental setup we did not use the complete general architecture of a question answering system but just the upper part of Figure IX.1. So we skipped the extraction of the most relevant answers.

In the query construction, we used a language model based method to find the expected answer type (cf. chapter VI.1) and some simple techniques to optimize the question for the following retrieval steps. The document retrieval was also done by using a language modeling approach.

Nevertheless, in this chapter, we describe the use of a special case of passage retrieval where we directly split the documents into single sentences. In a next step, a language model (LM) based approach with unigram distributions was applied to re–rank the text chunks. For these purposes, we used the language modeling toolkit developed at our chair. It implements all standard language modeling techniques, like linear interpolation and backing–off models. Its advantage is that it is more flexible than other tools in terms of model combinations, design of the retrieval vocabulary and the smoothing strategies. By means of this toolkit we conducted re–ranking experiments with standard language model based smoothing methods like Jelinek–Mercer linear interpolation, Bayesian smoothing with Dirichlet priors and Absolute Discounting as well as some new, improved models. We focused on investigating refinements which are easy to implement such as ignoring query words, dynamic stopword lists and stemming. We also experimented by modeling the expected answer type of a query into the LM approach.

To make our results comparable to current literature, we evaluated our algorithms on a news texts corpus from the Text REtrieval Conference (TREC) – the AQUAINT corpus. Here, we demonstrate that our proposed algorithms outperform the standard methods in terms of mean reciprocal rank (MRR) by 25%. We can also show that we need to return fewer sentences to achieve equal or even better accuracy. So it is possible to say that we attained our goal, namely to reduce the search space for the following components in a QA framework.

The rest of the chapter is organized as follows: The next section presents some related work. Section IX.3 shows the language model based smoothing methods we used for our experiments. Section IX.4 presents the used datasets as well as the experiments we performed in order to achieve optimal results. Section IX.5 concludes the results.

2 Related Work

As mentioned above, sentence retrieval is just a special case of passage retrieval where the text selection has the size of one sentence. There is also some related work in the area of sentence retrieval for QA systems. One example is Murdock and Croft (2004). They understand the meaning of retrieving sentences as the translation of a user query to a (more or less complex) answer. With this idea, they suppose to overcome the problem of the shortness of sentences to compute a multinomial distribution. Their approach is based on the IBM Model 1 and is smoothed with the corresponding document in addition to the collection. They show a performance gain to the original query–likelihood scoring.

In Losada (2005) language model based approaches for sentence retrieval are compared. They define multinomial and multiple–Bernoulli distributions on top of the query–likelihood approach. Their motivation is the shortness of a sentence. In a multiple–Bernoulli framework the non–query terms are also taken into account. So, they show a significant performance increase compared to a multinomial approach.

An other application for sentence retrieval is the TREC Novelty track (Harman 2002). Here, the task is to reduce the amount of redundant and non–relevant information in a given document set. Normally, this is done by a two–step approach. The first part is to find the relevant sentences according to a query. In a second part, those sentences are selected which contain novel information compared to the retrieved set in the first part. Larkey et al. (2002) and Allan, Wade, and Bolivar (2003) give some examples of how to build such a system. They use three different methods for extracting relevant sentences; a vector based approach using *tf–idf*, a version using the *Kullback–Leibler divergence (KLD)* and an approach using a *Two–Stage Smoothing* model. Because, in contrast to our experiments, they do not find any significant differences between these methods, they decide to use the *tf–idf* approach. From their point of view, the selection of the relevant sentences is the major challenge, so they try to further improve the performance by using known techniques like query expansion, pseudo–relevance feedback and other features. But, again in contrast to our observations, just pseudo–feedback helps to improve the performance.

A major difference to the open–domain question answering is that in the Novelty track, a set of relevant documents is given. So, there is no need to find some relevant documents out of a large corpus first. Allan, Wade, and Bolivar (2003) also show the negative effects when using a real information retrieval system instead of a given document set.

A last significant difference is the kind of processing of the retrieved data. In a question answering system, further steps are the extraction and selection of possible answers out of the sentences. This task is very hard and time–consuming, so it is necessary to keep the set of returned sentences as small as possible.

As already mentioned, a partial implementation of the system can be found in Shen et al. (2006). There, a complete statistically–inspired QA system in the context of the TREC 2006 question answering track is developed.

Chapter VI gives a more specific description of the language model based query classification part we used in our experiments.

3 Methodology

First, we want to introduce the language model based approach proposed by Ponte and Croft (1998) as our information retrieval framework for sentence retrieval[1]. They rank the user

[1]Chapter III gives a more detailed view on language model based information retrieval.

query using a query model, whereas a language model for each document is determined. Then the probability of producing the query with those models is calculated. Following Zhai and Lafferty (2001), applying the Bayes rule results in

$$P(D|Q) \propto P(Q|D)P(D) \tag{IX.1}$$

where $P(D)$ is the prior belief of a document and $P(Q|D)$ is the probability of the query given a document.

We act on the assumption that the prior $P(D)$ is a uniform distribution, so it is equal for all documents and therefore irrelevant for ranking the query. Thus, it will be ignored in further computations. The probability of $P(Q|D)$ is calculated by using language models. This conversion means that we just have to calculate the conditional probability of the user query and the document we intend to rank. This task seems easier than calculating $P(D|Q)$.

Formula IX.1 has a data sparsity problem. Generally, there isn't enough training data to compute language models for a complete query[2]. To overcome this problem we act on the assumption that all words in the query are independent. This independence assumption results in unigram language models as proposed in Zhai and Lafferty (2001):

$$P(Q|D) = \prod_{i=1}^{N} P(q_i|D) \tag{IX.2}$$

whereas N is the number of terms in a query. In our approach the documents are sentences, so we used $P(q_i|S)$ as our experimental baseline, where S is the sentence we intend to score.

In the next sections we will describe how to calculate those probabilities. Because we use a maximum likelihood estimate to calculate $P(w|S)$, it is necessary to smooth them in order to avoid zero probabilities. Chapter III.1 presents more information about this topic.

3.1 Jelinek–Mercer smoothing

The Jelinek–Mercer smoothing method is just a linear interpolation between the maximum likelihood probability and a background collection model. It is defined by

[2]Let's suppose that a query has 7 words in average. Then 7–gram language models have to be computed.

$$P_\lambda(w|S) = (1-\lambda)\frac{N(wS)}{\sum_w N(wS)} + \lambda P(w|C) \tag{IX.3}$$

where $N(wS)$ is the count of word w in sentence S and λ is the smoothing parameter. $P(w|C)$ is the collection model. In our experiments the background collection always consists of the set containing all sentences.

3.2 Absolute Discounting

This smoothing method has its origin in the task of speech recognition. There, it is the most efficient and thus the most commonly used technique. But it was also introduced to the task of information retrieval by Zhai and Lafferty (2001). It results in

$$P_\delta(w|S) = \frac{\max\left(N(wS)-\delta, 0\right)}{\sum_w N(wS)} + \frac{\delta B}{\sum_w N(wS)} P(w|C) \tag{IX.4}$$

whereas $N(wS)$ are the frequencies of w in S and $P(w|C)$ is the collection model of all sentences. δ defines the smoothing parameter to redistribute some probability mass to unseen events. The parameter B counts how often $N(wS)$ is larger than δ.

3.3 Bayesian smoothing with Dirichlet priors

Bayesian smoothing using Dirichlet priors is the approach which performs best according to our question answering task in document retrieval as well as according to our sentence retrieval framework. It is also described by Zhai and Lafferty (2001) and is defined by

$$P_\mu(w|S) = \frac{N(wS) + \mu P(w|C)}{\sum_w N(wS) + \mu} \tag{IX.5}$$

where $N(wS)$ is the frequency of observations of the word w in sentence S. μ is the smoothing parameter. Again, $P(w|C)$ is the collection model containing all sentences. A special case of this method is the *add–epsilon smoothing*, i.e. when a uniform collection model is used.

3.4 Dataset

As dataset for our experiments we used the *TREC 2004 QA collection*. Chapter VII.2.1 presents more details about the used dataset.

The question set for TREC 2004 consists of 351 questions, which are further divided into subsets. Each subset has a unique topic and a set of *factoid*, *list* and *other* questions. For example, a typical *factoid* question is *When was James Dean born?* whereas a *list* question would be *What movies did James Dean appear in?*. The task of the *other* question is mainly to find as many different information concerning the topic as possible.

As evaluation metrics for the results, the Mean Reciprocal Rank (MRR) and the accuracy of the system was used. In this context, accuracy means the percentage of answerable questions using a specific number of returned sentences. For testing the parameters in the query construction, we used the Mean Average Precision (MAP).

4 Experiments

For efficiency reasons we chose a three–step approach for our experiments. First, the user question was analyzed. Therefore, we used the approach described in chapter VI.1 to extract the expected answer type. It specifies a language model based query classification, using a simple Bayes classifier as paradigm. The taxonomy of the classifier takes 6 coarse and 50 fine grained classes into account.

In addition to this, some simple methods were used to further optimize the query for the following retrieval task.

In a second step, the *Lemur Toolkit for Language Modeling and Information Retrieval* was used to carry out a language model based document retrieval. As suggested in Hussain, Merkel, and Klakow (2006), we performed Bayesian smoothing with Dirichlet priors. We fetched the top 50 relevant documents because Shen et al. (2006) showed that this number is sufficient to answer about 90% of questions. After the extraction, we split them up into sentences using the sentence boundary detection algorithm provided by LingPipe. We also used larger passages (Hussain, Merkel, and Klakow 2006), but the sentence–based approach is much more efficient.

The third step was the re–ranking of sentences using the language model based methods described in section IX.3. For these purposes we used the language modeling toolkit developed by our chair (*LSVLM*). It implements all standard language modeling techniques and is more flexible than other tools in terms of backing–off strategies, model combinations and design of the retrieval vocabulary.

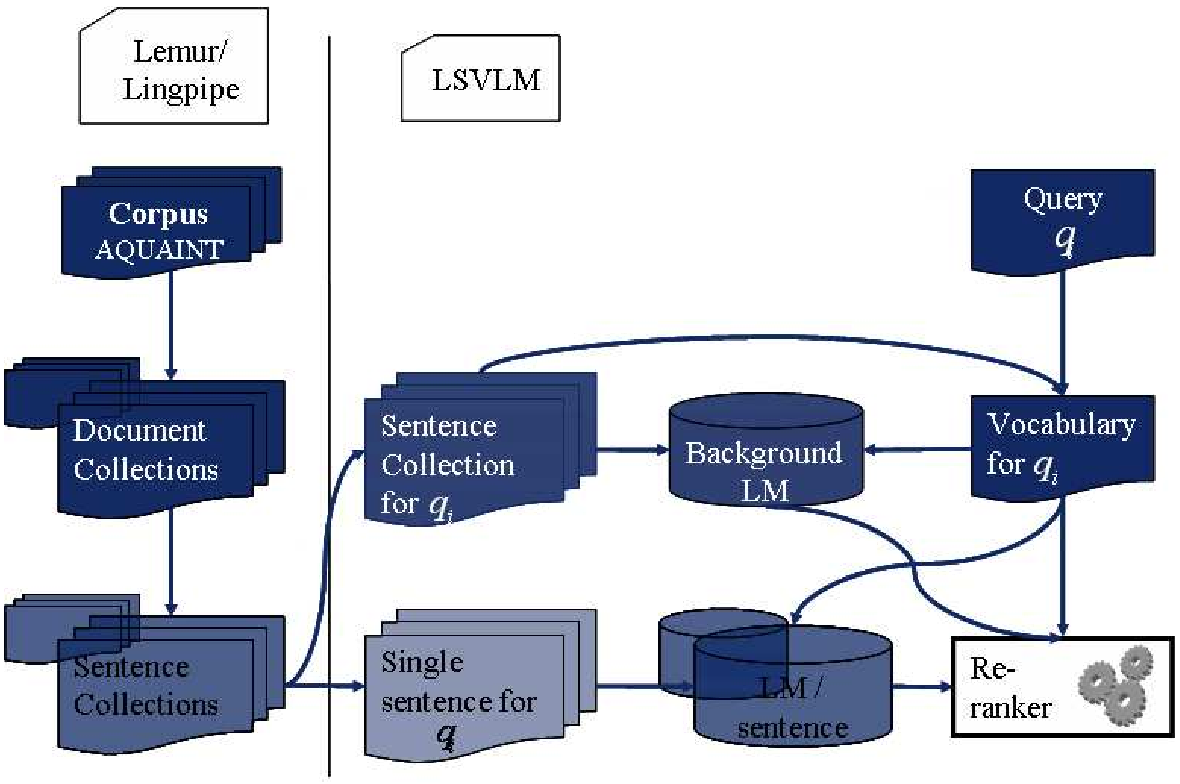


Figure IX.2: The sentence retrieval architecture for our experiments

Figure IX.2 shows the architecture of the sentence retrieval experiments we made. On the left–hand side, the pre–processing steps with standard software are shown. On the top one finds the AQUAINT corpus described in section IX.3.4. Then the document collections we gain by using Lemur can be seen. On the bottom one finds the sentence collections we receive by using the LingPipe toolkit.

On the right–hand side, we show the experimental setup for the *LSVLM* framework. In the middle the background model for each experiment is presented. It consists of the complete sentence collection for a given user query q_i. Out of this collection, a background language model is build. As vocabulary for this model we use the combination of the vocabulary built from the user query q_i and the corresponding sentence collection. So, the vocabulary is closed over the query.
On the bottom left, the single sentences for a query q_i can be seen. These are the sentences we want to re–score in our experiments. Then, again language models are created for each

individual sentence by using the closed vocabulary.

And finally, the two language models are used to calculate a new score. Because of the flexibility of our toolkit, it is possible to easily change the smoothing algorithms and parameters used to get the optimal setting.

In our sentence retrieval experiments, we used tf–idf and OKAPI as standard baseline approaches and linear interpolation (Jelinek–Mercer), absolute discounting and Dirichlet priors as language model based smoothing algorithms (see section IX.3).

The smoothing parameters for the different methods were experimentally defined on the TREC 2003 dataset. For absolute discounting, we took the discounting parameter $\delta = 0.1$, for linear interpolation the smoothing parameter was set to $\lambda = 0.8$ and for Dirichlet prior we set $\mu = 100$.

4.1 Results

In this section, we discuss the results we achieved by using the query construction, the document retrieval and the optimized sentence retrieval steps.[3]

As already mentioned in section IX.4, we first analyzed the user query by extracting the expected answer type[4]. This answer type is used in a later step to optimize the language models in the sentence retrieval step.

In addition to this approach, we used further methods to optimize the query for document and sentence retrieval. In a first step, the topic of the query was included for multiple times. This inclusion was done because, within our language model approach, the repeating of a specific term for multiple times results in a higher score for that term. A higher score means that the included term gets greater importance in that context. Chapter II.3.1 provides more information about this fact.

The last step in the query construction was the subtraction of the query word. In general, this term has no positive effects on the retrieval system and can therefore be ignored. Here, the same argumentation holds as for including the topic for multiple time. By removing the query word, this score will be zero and other, possibly more relevant terms get a higher score.

The effects of these methods on the sentence retrieval framework can be found in section IX.4.1.2.

[3]`http://www.ldc.upenn.edu/Catalog/docs/LDC2002T31/`

[4]Further results can be found in Merkel and Klakow (2007b).

As mentioned above, the *Lemur* toolkit was used to perform document retrieval. Therefore, the queries as well as the AQUAINT corpus were stemmed and no stop–words were removed. Then we chose a language model based approach to retrieve the documents. As smoothing method, we used Bayesian smoothing with Dirichlet priors because chapter VII illustrated that this approach performs best for this task. There, we show that it even provides a performance superior to standard approaches like tf–idf and Okapi. We also suggest an optimal smoothing parameter for this question set which we also used for our experimental setup.
After doing the retrieval, the 50 most relevant documents were fetched and split up into sentences.

4.1.1 Baseline Experiments

This section describes the baseline experiments we conducted before starting our optimization approaches. Figure IX.3 shows the results of those experiments. On the x–axis it presents the number of returned sentences by the system on a logarithmic scale. On the y–axis the accuracy of the system is shown. For example, an accuracy of 0.5 means that 50% of the queries are answerable by the system.
For the standard tf–idf and Okapi baseline experiments we used the *Lemur* toolkit. The figure shows that the tf–idf performs better than Okapi regarding this task. Both approaches were not optimized for these experiments.
In a next step, we used our *LSVLM* toolkit to conduct the baseline experiments with the three standard language model based smoothing approaches (as described in section IX.4).
For a small number of returned sentences (1–50), the linear interpolation (Jelinek–Mercer) and the absolute discounting smoothing perform comparatively bad. In this part the Bayesian smoothing with Dirichlet priors obviously performs better.
In the last segment (50–100 sentences) the Dirichlet prior approach performs somewhat worse than absolute discounting. The best smoothing method for this part is the Jelinek–Mercer interpolation. But this performance gain is not visibly significant.
The figure also shows that all baseline language model based approaches perform better than the standard tf–idf and Okapi methods for this task by large margin.
As already mentioned in section IX.1, in a question answering system we are most interested in getting a high accuracy at a small number of returned sentences. That means, the

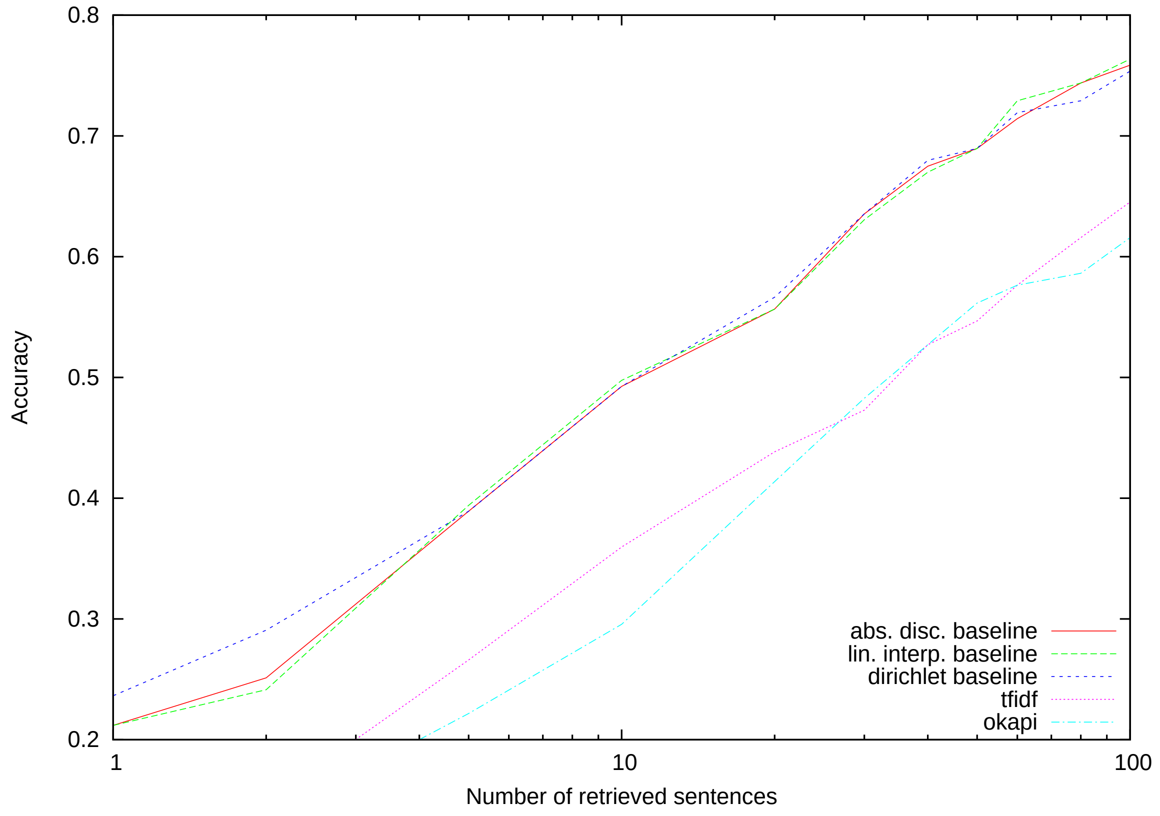


Figure IX.3: Number of retrieved sentences vs. accuracy for baseline experiments

following modules need to process just smaller sets of sentences to reach the same level of accuracy. Thus, the Bayesian smoothing with Dirichlet priors was chosen as a optimization baseline for further experiments.

4.1.2 Improved Smoothing Methods

Figure IX.4 shows the results of the experiments we carried out with optimized language models for the question answering task. Again, on the axis of abscissae the number of returned sentences is plotted on a logarithmic scale, whereas on the ordinate the accuracy of the system is shown (as described in section IX.3.4).

For better comparison of the improvements of the optimization steps, the Dirichlet prior smoothing method is shown as baseline (curve (1)). The other lines show the performance gain of each individual method we added to the baseline. Each new experiment is based on the previous optimization method.

Our first approach is already discussed in section IX.4.1. It is the simple removal of the query word and therefore belongs to the query construction step. Figure IX.4 shows the resulting

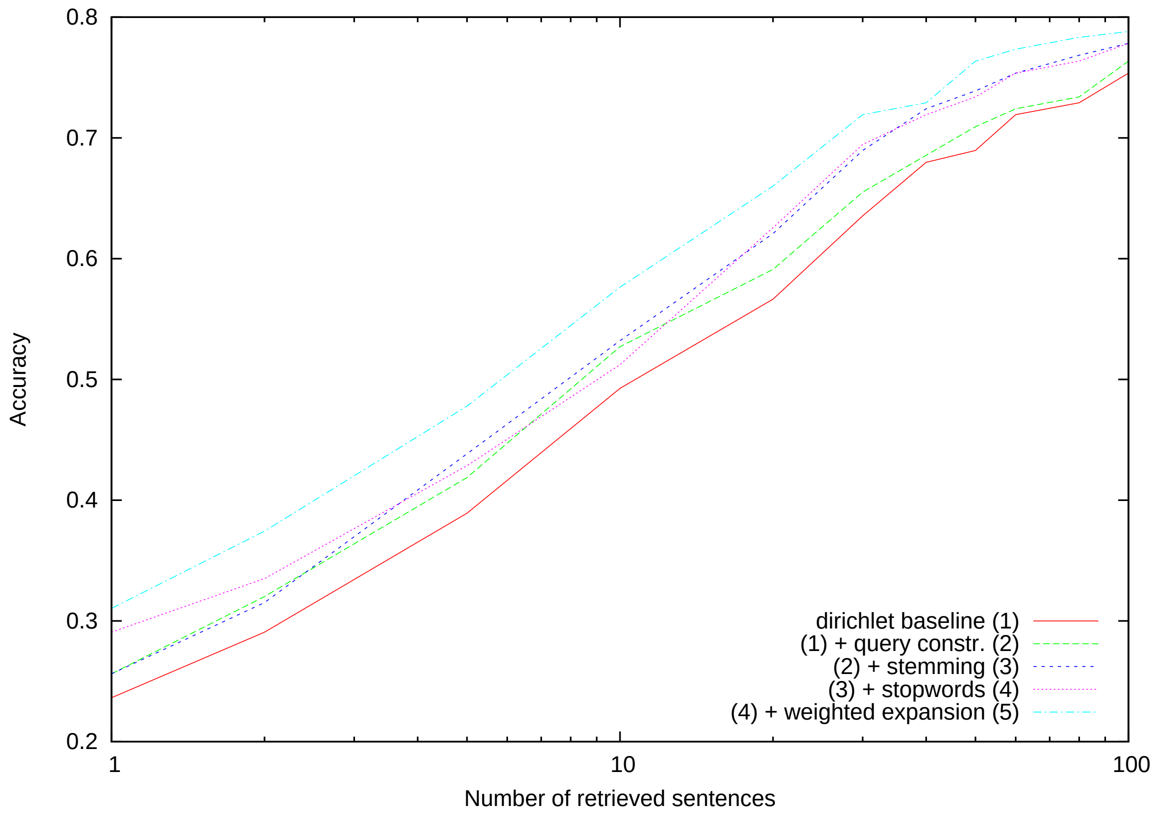


Figure IX.4: Number of retrieved sentences vs. accuracy for optimized smoothing methods

effects on the system. The new curve (2) provides a performance superior to the Dirichlet baseline.

In a second step, we added the Porter stemmer to our experiments. The results are shown in curve (3). As described in relevant literature, this addition also results in a further advance of system accuracy of this specific kind of task.

As a next optimization criterion we used a dynamic stopword list. It was created by selecting the four most commonly used terms of the complete sentence collection. However, those terms were not removed as usual for stop–words but they got just a smaller weight in the language model. This re–weighting is based on the same findings we already discussed in section IX.4.1. The result in Figure IX.4 shows in curve (4) a small performance gain, when looking at a very small number of returned sentences. Besides, the accuracy is nearly equal to the previous step.

For the last optimization experiment, the expected answer type of user query we gained in the query construction, was used to expand the language models with this additional information (see section IX.4.1). This was done by expanding the query and a sentence in dependency on the extracted question type. Thereby, sentences, which match the query type, are ranked higher.

This means, for example, if the expected answer type is *Date*, the term *DATE* is added to the question. Then patterns are used to identify expressions for dates in a sentence. If such a date expression also occurs in a sentence, it is expanded with the *DATE* term as well. After this step, the additional terms are weighted and thus the language model based approach gives a higher rank to sentences which match the corresponding question.

The resulting effects of this last optimization step is also shown in Figure IX.4. Here, curve (5) demonstrates the improvement of performance by adding the weighted expansion. The distribution outperforms all other combined methods by a large margin.

Table IX.1 shows the MRR of the baseline experiments and combination of all optimization steps. Standard Okapi and tf–idf achieved the worst MRR. The Jelinek–Mercer interpolation and absolute discounting baseline perform better with a MRR of 0.29. We found out that the Dirichlet prior baseline again performs a little bit better with a MRR of 0.31. This was the reason why we developed the improved language models on top of this distribution. The table also shows that the combination of all optimization steps (Dirichlet Combined) performs best with an MRR of 0.39. This means that there is an improvement of more than 25% compared to the Dirichlet baseline and, that there is an improvement of more than 34% compared to the other LM based experiments.

5 Conclusion

In this chapter, we showed a language model based framework to perform improved sentence retrieval in a question answering context. The major goal was to improve the accuracy of the system in order to return just a smaller number of relevant sentences. This reduces the search space of the following components in a QA system. Because these components are typically deep–analysis approaches which strongly depend on the size of processing documents, such a step is necessary.

For this purpose, we first analyzed the user query by extracting the expected answer type and by doing some other simple text manipulations.

After a language model based document retrieval step, we split up the documents into smaller text passages in the size of sentences.

Distribution	MRR
Okapi	0.16
tf–idf	0.18
Jelinek–Mercer	0.29
Absolute Discounting	0.29
Dirichlet Baseline	0.31
Dirichlet Combined	0.39

Table IX.1: Mean Reciprocal Rank of baseline and optimized experiments

Then, the *LSVLM* toolkit, a language model based framework we developed at our department, was introduced. With this toolkit, we were able to conduct sentence retrieval experiments in a more flexible way than with other state–of–the–art information retrieval frameworks. We conducted baseline experiments with standard tf–idf and Okapi as well as with language model based smoothing methods like Jelinek–Mercer interpolation, Bayesian smoothing with Dirichlet priors and absolute discounting.

We proved that Dirichlet priors baseline performed best for our task, so we developed our optimization steps on top of this approach.

In several experiments we illustrated that using query word removal, dynamic stopword list weighting and stemming had resulted in a performance gain. In the last experiment, we modeled the expected answer type of a user query into the used language models. This approach performed better than the LM baselines by at least 25%.

We also proved that we had needed to return fewer sentences in order to achieve equal or even better performance in terms of system accuracy. So we attained our goal to reduce the search space for the following components in a QA framework.

Chapter X

Conclusion and Future Work

The motivation of this thesis was to bring the notion of statistical language models to the task of question answering. In particular, we performed experiments with language model based question classification, document-, passage-, and sentence retrieval. We showed that using a statistical approach to question answering can increase the performance of the complete system.

In contrast to today's standard Web search engines, where a set of relevant documents is returned to satisfy a user's information need, a question answering (Q&A) system provides the user with a concise answer. This means, if a user asks for *Who was the founder of the Guinness Brewery?*, the system should answer with *Arthur Guinness* instead with a set of documents which might contain the correct answer.

To understand how a Q&A system works in particular, chapter II gives an overview of the most important modules.

The first stage in finding a correct answer normally is to analyze the natural language question. In this step, the specific type of a question is determined, i.e. the question type of our example *Who was the founder of the Guinness Brewery?* is *Person*. Depending on the used taxonomy, other classes like *Location, Description* or *Entity* are also possible. This module is very important for the complete system, because the question type is used in various other parts of the Q&A engine. For example, the question class often corresponds with the expected answer type and is therefore used to specify possible answer candidates in the answer extraction module. To determine the correct question class complex linguistic methods are used in general. The approach, we used in this thesis, is based on statistical language models and therefore has better performance as we explain later when discussing our experimental results.

This analysis step also contains a simple form of query construction to increase the retrieval performance in succeeding information retrieval modules.

The use of information retrieval (IR) for the task of question answering is controversially discussed in recent literature (cf. Monz (2003)). The main motivation of IR modules is to reduce the search space for the following deep linguistic algorithms because they can perform much more robust and faster when processing a smaller amount of text.

Normally, the task of IR in a question answering systems is splitted into two parts. First, a standard document retrieval is done to fetch the relevant documents out of a possibly large corpus. This step is also called *document pre-fetching* (Monz 2003). Although, for standard document retrieval, traditional retrieval algorithms, like *Okapi25* or *tf-idf* perform best, we showed that for the task of Q&A the language model based approaches perform better.

But because documents in the retrieved collection might still be too large or contain different topics, this set is further divided into smaller text segments. This also leads to the main motivation, because in doing so, the amount of text that will be processed by following algorithms is further reduced and, therefore, the performance is increased.

Hence, *Passage Retrieval* is now standard in most state-of-the-art Q&A systems and there are various approaches for this task. Again, in this thesis, we used a language model based approach to retrieve and re-rank text passages.

If passage retrieval is done by just using one sentence as text size, the method is called *Sentence Retrieval*. We also introduced optimized language models for this task which perform better than traditional retrieval algorithms.

After reducing the set of relevant text fragments in that way, possible answer candidates are selected and re-ranked. Normally, this task needs very complex and time-consuming linguistic techniques, which strongly depend on the size of the processed corpus. That is why we just focus on the upper part of a complete question answering system in this thesis.

In this chapter, we also presented former and state-of-the-art Q&A systems and introduced our statistically inspired approach we implemented to perform at TREC.

To compare our approach with current Q&A systems and to evaluate single modules in this thesis, we additionally introduced some standard evaluation metrics, like *precision, recall* or the *mean average precision*.

The notion of statistical language models (SLM) was presented in chapter III. There, the motivation of LM was introduced as well as some advantages and flaws identified. So, for example, SLM are mathematically well founded and present a simple, well understood framework. But on the other hand, it is also difficult to incorporate some well-known retrieval notions, like relevance or relevance feedback.

Zero-probabilities constitute another disadvantage when estimating SLM. Therefore we explain different smoothing techniques to solve the problem of unseen events. Finally, some further language model frameworks are presented as a basis of improving Q&A modules, like *Log-linear models*.

This chapter also introduces language models for the task of information retrieval. This technique was proposed by Ponte and Croft (1998) and was the fundamental idea of implementing language model based question answering modules. In this context, the connection between the traditional *tf-idf* and SLM is explained and some improved language models are presented which we later used in some of our experiments. After introducing the notion of SLM for IR, a theoretic approach for language model based question answering was given. Here, the problem was taken as classification task and then split into the different subtasks, like question typing, information retrieval and answer extraction.

Chapter IV presents an application for language model based passage retrieval which is part of the question answering system developed at the *SmartWeb* project. This project is sponsored by the Federal Ministry of Education and Research of Germany and wants to provide users with multimodal access to the semantic web. Open-domain Q&A based on Web pages is just one part of this system. The main focus was on presenting information about the soccer Worldchampionship 2006 using multimodal access, like speech and scripture. In this chapter we showed that using language model based approaches for question answering could work in real applications.

Some related work about language model based applications is presented in chapter V, whereas in chapter VI experimental results for question classification and the correlated confidence measures are presented.

As mentioned above question classification is very important for doing Q&A, because the results are later used in various other components. In this thesis, we have introduced a language model based question classification module which uses a Bayes classifier as classification paradigm. As smoothing methods several standard smoothing techniques as well as improved methods were used to overcome the problem of zero-probabilities. We also showed the effects of using different background models for standard smoothing, such as zerogram and unigram models.

In general, we could verify that our improved methods perform better than the standard smoothing techniques. Although, Bayesian smoothing using Dirichlet priors performs best for the task of information retrieval, the absolute discounting is the best method to classify questions.
Regarding the improved smoothing methods, the *Improved Absolute Discounting* performs better than the standard backing-off approach. The best improved algorithm implements a log–linear interpolation using bigram statistics. These results are comparable to other question classification approaches like SVMs, but in terms of time complexity for model training, the language model based approach performs much faster.

The second issue we discussed in this chapter is the notion of confidence measures for question types. The motivation originally came from the theoretic approach for language model based question answering. Here, the probability of a question class has to be calculated.
When regarding the best question type returned by the system, we have proved that a confidence value of 0.9 or higher is sufficient to take the estimated class as correct. Experiments with two different datasets confirmed these results. So, these values can be used in later parts of a Q&A system, for example in order to have more features to select possible answer candidates.

Chapter VII presents our experiments for the task of document retrieval. The first idea was to find an adequate number of documents which should be returned by system. This was done in order to reduce the search space for the following parts of the Q&A system. As already mentioned, some modules use deep linguistic algorithms which strongly depend on the size of the processing corpus and therefore work much faster and more robust if this size is as small as possible.
Hence, our first experiments compared traditional retrieval methods with the language model based approach concerning the differing numbers of returned documents. We showed that the SLM used for the task of document retrieval perform better than traditional approaches, like *Okapi* or *tf-idf*. We also concluded that returning 500 documents is sufficient for this task using a newswire corpus as training collection.
The second set of experiments finally defined the used smoothing parameters by searching

the complete parameter space. As main result, we pointed out that using Bayesian smoothing with Dirichlet priors performed best for the task of document retrieval.

The same experiments were also done for passage retrieval in chapter VIII. Again, first the number of returned documents was fixed and afterwards the smoothing parameters were tuned. The results were very similar, too. We showed, that the language model based approach performed better than using traditional methods and returning 500 text passages was also sufficient. Here, the best standard approach was Dirichlet smoothing.
Further experiments were made using improved language models to re–rank the set of passages. In particular, the main research issue was to find the best–suited background distribution for smoothing. Here, the results showed that using the single relevant document the ranked passage came from as background corpus, the smoothing method performed best.

The final set of experiments was explained in chapter IX in the area of sentence retrieval. Again, the motivation was to improve the accuracy of the system by returning just a smaller number of relevant sentences to reduce the search space of the following components as explained above.
In a first step, we carried out baseline experiments to compare traditional retrieval methods with the language model based approach. We proved that Bayesian smoothing using Dirichlet priors also performed best for the task of sentence retrieval. Hence, further optimization steps were based on this method. Using simple modifications, like topic adding, query word removal, dynamic stopword list weighting and stemming already resulted in a performance gain. In the last optimization experiment, we showed that modeling the expected answer type of a question into the SLM performed better than the language model baselines by at least 25%.

A possible continuation of the presented experiments is the use of class based language models (Brown et al. 1992) for question classification as well as for the retrieval tasks. Using this approach might be promising because it is strongly related to clustering, which yields very good results for the task of information retrieval (Kurland, Lee and Domshlak 2005).
Another idea, we have not experimented with so far is the use of our improved smoothing techniques for doing document-, passage-, and sentence retrieval. This is also a promising approach, because in the case of log–linear models, also higher-order models, like bigrams, are incorporated, which are useful to increase performance in IR (Song and Croft 1999).
In chapter VI, we mentioned that for specific datasets the confidence measures showed

reasonable results for values lower than 0.9. Here, further experiments using question type depending confidence measures should be done.

Finally, further corpora can be used to perform Q&A with the proposed modules. For example, AnswerBus (Zheng 2002) uses the Internet to answer natural language questions. As results, the system returns a set of relevant sentences to the user. This approach can also be realized using the techniques proposed in this thesis. Chapter IV already shows the implementation of the passage retrieval part for using the Web as corpus.

It should be a goal for the future to use Q&A systems not only with predefined newswire data. Therefore, it is a very interesting task to bring the statistical language model based question answering approach to the Internet.

Bibliography

Abney, S.P. (1989): Parsing by chunks, in Berwick, R.C., Abney, S.P., and Tenny, C., editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pp. 257–278. Kluwer Academic Publishers, Boston, USA, 1991.

Agichtein, E. and Gravano, L. (2000): Snowball: Extracting Relations from Large Plain–Text Collections, in *Proceedings of the 5th ACM International Conference on Digital Libraries*, pp. 85–94, San Antonio, United States, 2000.

Allan, J. (1996): Incremental relevance feedback for information filtering, in *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, Zurich, Switzerland, 1996.

Allan, J., Wade, C., and Bolivar, A. (2003): Retrieval and novelty detection at the sentence level, in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, Toronto, Canada, 2003.

Altavista: `http://www.altavista.com`.

Baeza–Yates, R. and Ribero–Neto, B. (1999): Modern Information Retrieval, *Addison Wesley*, 1999.

Baker, C.F., Fillmore, C.J., and Lowe, J.B. (1998): The Berkeley FrameNet Project, *Proceedings of the 36th annual meeting on Association for Computational Linguistics* , Montreal, Canada, 1998.

BBC: The British Broadcasting Company. `http://www.bbc.co.uk`.

Berger, A. and Lafferty, J. (1999): Information retrieval as statistical translation, in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, Berkeley, USA, 1999.

Berry, M.W., Dumais, S.T., O'Brien, G.W. (1995): Using Linear Algebra for Intelligent Information Retrieval, in *SIAM Review*, 37(4):573–595, 1995.

Bikel, D.M., Miller, S., Schwartz, R., and Weischedel, R. (1997): Nymble: a high-performance learning name-finder, in *Proceedings of the fifth conference on Applied natural language processing*, Washington, DC, 1997.

Bos, J. (2006): The L̈a SapienzaQ̈uestion Answering system at TREC–2006, in *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg: NIST, 2006.

Brill, E. (1992): A simple rule-based part of speech tagger, in *Proceedings of the workshop on Speech and Natural Language*, Harriman, New York, 1992.

Brill, E. (1994): Some advances in transformation-based part of speech tagging, in *Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, Seattle, Washington, 1994.

Brin, S. and Page, L. (1998): The anatomy of large–scale hypertextual Web search engine, in *Proceedings of the 7th international World Wide Web Conference*, 1998.

Bronnenberg, W., Bunt, H., Landsbergen, J., Scha, R., Schoenmakers, W., and van Utteren, E. (1980): The question answering system PHLIQUA1, in L. Bolc, editor,*Natural Language Question Answering Systems*, pp. 217–305, MacMillan, 1980.

Brown, P.F., DeSouza, P.V., Mercer, R.L., Della Pietra, V., and Lai, J.C. (1992): Class-based n-gram models of natural language, in *Computational Linguistics*, 18(4):467–479, 1992.

Brown, P.F., Pietra, S.A.D., Pietra, V.J.D., and Mercer, R.L. (1993): The mathematics of statistical machine translation: Parameter estimation, in *Computational Linguistics*, 19:263–311, 1993.

Burke, R., Hammond, K., Kulyukin, V., Lytinen, S., Tomuro, N., and Schoenberg, S. (1997): Question answering from frequently–asked question files: Experiences with the FAQ Finder system, in *AI Magazine*, 18(2):57–66, 1997.

Buckley, C., Salton, G., and Allan, J. (1993): Automatic retrieval with locality information using SMART, in Harman, D.K., editor, *Proceedings of the First Text REtrieval Conference (TREC–1)*, Gaithersburg: NIST, 1993.

Cai, D., Yu, S., Wen, J.R., and Ma, W.Y. (2004): Block-based web search, in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, Sheffield, United Kingdom, 2004.

Callan, J.P. (1994): Passage–level Evidence in Document Retrieval, in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, Dublin, Ireland, 1994.

Callan, J.P., Croft, W.B., and Harding, S.M. (1992): The INQUERY Retrieval System, in *Proceedings of DEXA–92, 3rd International Conference on Database and Expert Systems Applications*, Valencia, Spain, 1992.

Cao, G., Nie, J.Y., and Bai, J. (2005): Integrating word relationships into language models, in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, 2005.

Clarke, C., Cormack, G.V., Kemkes, G., Laszlo, M., Lynam, T., Terra, E., and Tilke, P. (2002): Statistical selection of exact answers (MultiText Experiments for TREC 2002), in *Proceedings of the 11th Text REtrieval Conference (TREC 2002)*, Gaithersburg: NIST, 2000.

Clarke, C., Cormack, G.V., Kisman, D.I.E., and Lynam, T.R. (2000): Question Answering by Passage Selection (MultiText Experiments for TREC–9), in *Proceedings of the Ninth Text REtrieval Conference (TREC–9)*, Gaithersburg: NIST, 2000.

Clarke, C. and Terra, E. (2003): 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*, pp. 427–428, 2003.

CLEF: The Cross-Language Evaluation Forum. `http://www.clef-campaign.org`.

Cleverdon, C.W. (1967): The Cranfield tests on index language devices, in *Aslib Proceedings*, 19:173–192, 1967. Reprint in Sparck Jones, K. and Willet, P.: Readings in Information Retrieval, San Fransisco. Morgan Kaufmann Publishers, 1997.

Corrada–Emmanuel, A., Croft, W.B., and Murdock, V. (2003): Answer Passage Retrieval for Question Answering, *CIIR Technical Report*, University of Massachusetts, 2003.

Cover, T. M. and Thomas, J. A. (2001): Elements of Information Theory, *Wiley Series in Telecommunications*, 2001.

Cui, H., Kan, M.Y., and Chua, T.S. (2004): Unsupervised learning of soft patterns for generating definitions from online news, in *Proceedings of the 13th international conference on World Wide Web*, New York, USA, 2004.

Cui, H., Sun, R., Li, K., Kan, M.Y., and Chua, T.S. (2005): Question answering passage retrieval using dependency relations, in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, 2005.

Darroch, J.N. and Rattcliff, D. (1972): Generalized Iterative Scaling for Log–Linear Models, in *The Annals of Mathematical Statistics*, 43(5):1470-1480, 1972.

Deerwester, S., Dumais, S., Furnas, G.W., Landauer, T.K., Harshman, R. (1990): Indexing by Latent Semantic Analysis, in *Journal of the Society for Information Science*, 41(6): 391-407, 1990.

Gao, J., Nie, J.Y., Wu, G., and Cao, G. (2004): Dependence language model for information retrieval, in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, Sheffield, United Kingdom, 2004.

GeoClef: Evaluation of multilingual Geographic Information Retrieval (GIR) systems. `http://ir.shef.ac.uk/geoclef/`.

Google: `http://www.google.com`.

Graff, D. (2002): The AQUAINT Corpus of English News Text, *Technical Report, Linguistic Data Consortium LCC*, Philadelphia, 2002.

Green, B., Wolf, A., Chomsky, C., and Laughery, K. (1963): Baseball: An automatic question answerer, in E. Figenbaum and J. Fledman, editors, *Computers and Thoughts*, McGraw-Hill, 1963.

Harabagiu, S., Hickl, A., Williams, J., Bensley, J., Roberts, K., Shi, Y., and Rink, B. (2006): Question Answering with LCC's CHAUCER at TREC 2006, in *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg: NIST, 2006.

Harman, D. (2002). Overview of the TREC 2002 Novelty Track, in *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg: NIST, 2002.

Hatcher, E. and Gospodnetić, O. (2004): Lucene in Action, in *Manning Publications Co*, 2004.

Hiemstra, D. (2002): Term-specific smoothing for the language modeling approach to information retrieval: the importance of a query term, in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Tampere, Finland, 2002.

Hiemstra, D., Robertson, S., and Zaragoza, H. (2004): Parsimonious language models for information retrieval, in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, Sheffield, United Kingdom, 2004.

Hussain, M., Merkel, A., and Klakow, D.(2006): Dedicated Backing–Off Distributions for Language Model Based Passage Retrieval, in *Hildesheimer Informatik-Berichte, LWA 2006*, Hildesheim, 2006.

Jelinek, F. (1997): Statistical Methods for Speech Recognition, *Massachusetts Institute of Technology Press*, Cambridge, USA, 1997

Jin, R., Hauptmann, A.G., and Zhai, C. (2002): Title language model for information retrieval, in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Tampere, Finland, 2002

Kaisser, M. and Becker, T. (2004): Question Answering by Searching Large Corpora with Linguistic Methods, in *Proceedings of the Thirteenth Text REtrieval Conference (TREC 2004)*, Gaithersburg: NIST, 2004.

Kaisser, M., Scheible, S., and Webber, B. (2006): Experiments at the University of Edinburgh for the TREC 2006 QA track, in *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg: NIST, 2006.

Kazalski, S. (2007): Evaluation of a Question Answering System Using Newswire and Blog Corpora, in *Bachelor thesis; to be published*, Saarland University, 2007.

Katz, B. (1997): Annotating the World Wide Web Using Natural Language, in *Proceedings of RIAO '97*, Montreal, 1997.

Klakow, D. (1998): Log–Linear Interpolation Of Language Models, in *Proceedings of the 5th International Conference on Spoken Language Processing*, Sydney, Australia, 1998.

Klakow, D. (2003): Sprachmodellierung, Saarland University, 2003.

Klakow, D. (2006a): Using Regional Information in Language Model Based Automatic Concept Annotation and Retrieval of Video, in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings*, Toulouse, France, 2006.

Klakow, D. (2006b): Language Model Adaptation for Tiny Adaptation Corpora, in *Proceedings of the Ninth International Conference on Spoken Language Processing*, Pittsburgh, USA, 2006.

Kleinberg, J. (1998): Authoritative sources in a hyperlinked environment, in *Proceedings of the 9th ACM–SIAM Symposium on Discrete Algorithms*, pages 668–677, San Fransisco, 1998.

Kneser, R. and Ney, H. (1995): Improved backing-off for M-gram language modeling, in *International Conference on Acoustics, Speech, and Signal Processing, ICASSP*, 1(1)181–184, Detroit, USA, 1995.

Kraaij, W. and Spitters M. (2003): Language Models for Topic Tracking, in Croft, W.B. and Lafferty, J., editors *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

Krovetz, R and Croft, W.B. (1992): Lexical ambiguity and information retrieval, in *ACM Transactions on Information Systems (TOIS)*, 10(2)115–141, 1992.

Kurland, O. and Lee, L. (2004): Corpus structure, language models, and ad hoc information retrieval, in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, Sheffield, United Kingdom, 2004.

Kurland, O., Lee, L., and Domshlak, C. (2005): Better than the real thing?: iterative pseudo-query processing using cluster-based language models, in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, 2005.

Lafferty, J. and Zhai, C. (2001): Document language models, query models, and risk minimization for information retrieval, in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, USA, 2001.

Lafferty, J. and Zhai, C. (2003): Probabilistic Relevance Models Based on Document and Query Generation, in Croft, W.B. and Lafferty, J., editors *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

Larkey, L.S., Allan, J., Connell, M.E., Bolivar, A., and Wade, C. (2002): UMass at TREC 2002: Cross Language and Novelty Tracks, in *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, Gaithersburg: NIST, 2002.

Lavrenko, V. (2003): Language Modeling in IR, *Tutorial at the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, Toronto, Canada, 2003.

Lavrenko, V. and Croft, W.B. (2001): Relevance based Language Models, in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, USA, 2001.

Lavrenko, V., Choquette, M., and Croft, W.B. (2002): Crosslingual relevance models, in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Tampere, Finland, 2002.

Lavrenko, V. and Croft, B. (2003): Relevance Models in Information Retrieval, in Croft, W.B. and Lafferty, J., editors *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

Lehnert, W. (1978): The Process of Question Answering: A Computer Simulation of Cognition, in *Lawrence Erlbaum Associates*, 1978.

Lehnert, W. (1994): 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, 1978.

Li, X. and Roth, D. (2002): Learning Question Classifiers, in *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.

Lin, D. (1994): Principar – an efficient, broad–coverage, priciple–based parser, in *Proceedings of COLING*, pp. 482–488, Kyoto, Japan, 1994.

Lin, D. (1998): Dependency–based Evaluation of MINIPAR, in *Workshop on the Evaluation of Parsing Systems*, Granada, Spain, 1998.

Lin, J. and Katz, B. (2003): Question Answering from the Web Using Knowledge Annotation and Knowledge Mining, in *Proceedings of Twelfth International Conference on Information and Knowledge Management*, New Orleans, USA, 2003.

LingPipe: `http://www.alias-i.com/lingpipe/`.

Liu, X. and Croft, W.B. (2002): Passage Retrieval Based On Language Models, in *Proceedings of the eleventh international conference on Information and knowledge management*, McLean, Virginia, 2002.

Liu, X. and Croft, W.B. (2004): Cluster–based retrieval using language models, in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, Sheffield, United Kingdom, 2004.

Llopis, F., Ferrández, A., and Vicedo, J. (2002): Passage selection to improve question answering, in *Proceedings of the COLING 2002 Workshop on Multilingual Summarization and Question Answering*, Taipei, Taiwan, 2002.

Losada, D.E.(2005): Language modeling for sentence retrieval: A comparison between Multiple-Bernoulli models and Multinomial models, in *Proceedings of the Information Retrieval and Theory Workshop*, Glasgow, United Kingdom, 2005.

Lucene Search Engine: `http://lucene.apache.org`.

Magnini, B., Negri, M., Prevete, R., and Tanev, H. (2001): Is it the right answer?: exploiting web redundancy for Answer Validation, *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Philadelphia, USA, 2001.

Manmatha (2003): Applications of Score Distributions in Information Retrieval, in Croft, W.B. and Lafferty, J., editors *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

Manning, C.D., Raghavan, P., and Schütze H. (2007): An Introduction to Information Retrieval, *Cambridge University Press, Preliminary Draft*, Cambridge University, 2007.

Manning, C.D. and Schütze H. (1999): Foundations of Statistical Natural Language Processing, *Massachusetts Institute of Technology*, Cambridge, 1999.

Marcus, M., Santorini, B., and Marcinkiewicz, M.A. (1993): Building a large annotated corpus of English: The Penn treebank, *Computational Linguistics*, 19:313–330, 1993.

Merchant, R.H. (1993): TIPSTER program overview, in *Proceedings of TIPSTER text program (phase 1)*, 1993.

Merkel, A. and Klakow, D. (2007a): Comparing Improved Language Models for Sentence Retrieval in Question Answering, in *Proceedings of Computational Linguistics in the Netherlands CLIN*, Leuven, Belgium, 2007.

Merkel, A. and Klakow, D. (2007b): Language Model Based Query Classification, in *Proceedings of the 29th European Conference on Information Retrieval (ECIR)*, Rome, Italy, 2007.

Merkel, A. and Klakow, D. (2007c): Improved Methods for Language Model Based Question Classification, in *Proceedings of the 8th Interspeech Conference*, Antwerp, Belgium, 2007.

Metzler, D. and Croft, B. (2004): Combining the language model and inference network approaches to retrieval, in *Information Processing and Management: an International Journal*, 40(5):735–750, 2004.

Miller, G.A. (1995): WordNet: a lexical database for English, in *Communications of the ACM*, 38(11)39–41, 1995.

Miller, D.R.H., Leak, T., and Schwartz, R.M. (1999): A hidden Markov model information retrieval system, in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 214–221, Berkeley, California, 1999.

Mittal, V.O. and Witbrock, M.J. (2003): Language Modeling Experiments in Non–Extractive Summarization, in Croft, W.B. and Lafferty, J., editors *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

Mittendorf, E. and Schäuble, P. (1994): Document and passage retrieval based on hidden Markov models, in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, Dublin, Ireland, 1994.

Mlynarczyk, S. and Lytinen, S. (2005): FAQFinder Question Answering Improvements Using Question/Answer Matching, *Proceedings of L&T-2005 – Human Language Technologies as a Challenge for Computer Science and Linguistics*, Poznan Poland,, 2005.

Monz, C. (2003): From Document Retrieval to Question Answering, *ILLC Dissertation Series DS-2003-4*, Amsterdam, 2003.

Murdock, V. and Croft, W.B. (2004): Simple translation models for sentence retrieval in factoid question answering, in *Proceedings of the Information Retrieval for Question Answering Workshop at SIGIR 2004* , Sheffield, United Kingdom, 2004.

Murdock, V. and Croft, W.B. (2005): A translation model for sentence retrieval, in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Vancouver, Canada, 2005.

Neumann, G. and Sacaleanu, B. (2005): DFKI's LT–lab at the CLEF 2005 Multiple Language Question Answering Track, in *Proceedings of the Working Notes, CLEF Cross-Language Evaluation Forum*, Vienna, Austria, 2005.

Ney, H., Essen. U.,and Kneser, R. (1994): On Structuring Probabilistic Dependencies in Stochastic Language Modeling, in *Computer Speech and Language*, 8(1)1-38, 1994.

NYT: The New York Times. `http://www.nytimes.com`.

O'Connor, J. (1980): Answer-passage retrieval by text searching, in *Journal of the American Society for Information Science*, 31(4)227-239, Lehigh University, Bethlehem, 1980.

Otterbacher, J., Erkan, G., and Radev, D.R. (2005): Using Random Walks for Question–focused Sentence Retrieval, in *Proceedings of Human Language Technology Conference Conference on Empirical Methods in Natural Language Processing*, pp. 915-922, Vancouver, Canada, 2005.

Reuters: `http://www.reuters.com`.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1998): The PageRank Citation Ranking: Bringing Order to the Web, *Technical Report*, Stanford University, 1998.

Palmer, M., Gildea, D., and Kingsbury, P. (2005): The Proposition Bank: An Annotated Corpus of Semantic Roles, in *Computational Linguistics*, 31(1):71–106, 2005.

Paşca, M. (2003): Open–Domain Question Answering from Large Text Collections, in *Studies in computational linguistics*, Leland Stanford Junior University, 2003.

Phillips, A. (1960): A question–answering routine, Memo. 16 *Artificial Intelligence Project*, 1960.

Ponte, J.M. and Croft, B. (1998): A Language Modeling Approach to Information Retrieval, in *Proceedings of the 21th annual international ACM SIGIR conference on Research and development in information retrieval*, Melbourne, Australia, 1998.

Ponte, J.M. (1998): A Language Modeling Approach to Information Retrieval, *PhD thesis*, University of Massachusetts, 1998.

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

The NIST PRISE search engine: `http://www-nlpir.nist.gov/works/papers/zp2/zp2.html`.

Reuters: `http://www.reuters.com`.

Roberts, I. (2002): Information Retrieval for question answering, *Master's thesis*, University of Sheffield, 2002.

Robertson, S.E. (1977): The Probability Ranking Principle in IR, in K. Sparck Jones and Peter Willet, editors, *Readings in Information Retrieval*, 1997.

Robertson, S.E. and Spärck Jones, K. (1976): Relevance weighting of search terms, in *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

Robertson, S.E. and Walker, S. (1994): Some Simple Effective Approximations to the 2–Poisson Model for Probabilistic Weighted Retrieval, in K. Sparck Jones and Peter Willet, editors, *Readings in Information Retrieval*, 1997.

Robertson, S.E., Walker, S., Jones, S., Hancock–Beaulieu, M., and Gatford, M. (1994): Okapi at TREC-3, in *Proceedings of the Third Text REtrieval Conference (TREC 1994)*, Gaithersburg: NIST, 1994.

Salton, G., Allan, J, and Buckley, C. (1993): Approaches to Passage Retrieval in Full Text Information Systems, in *Computer Science Technical Reports*, Cornell University, 1993.

Salton, G. and Buckley, C. (1988): Term-weighting approaches in automatic text retrieval, in *Information Processing and Management*, 24(5):513–523, 1988.

Salton, G. and McGill, M. (1983): Introduction to Modern Information Retrieval, *McGraw–Hill*, 1983.

Salton, G., Yang, C.S., and Yu, C.T. (1975): A theory of term importance in automatic text analysis, *Journal of the American Society for Information Sciences*, 26(1): 33–44, 1975.

Scha, R. (1983): Logical Foundations for Question Answering, *PhD thesis*, University of Groningen, 1983.

Schäuble, P. (1993): SPIDER: a multiuser information retrieval system for semistructured and dynamic data, in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, Pittsburgh, USA, 1993.

Schlaefer, N., Gieselmann, P., and Sautter, G. (2006): The EPHYRA QA System at TREC 2006, *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg: NIST, 2006.

Schuler, K.K. (2005): VerbNet: A Broad–Coverage Comprehensive Verb Lexicon, *PhD thesis*, University of Pennsylvania, 2005.

Shannon, C.E. (1948): A mathematical theory of communication, *Bell System Tech*, 27:379–423, 1948.

Shen, D. and Klakow, D. (2006): Exploring Correlation of Dependency Relation Paths for Answer Extraction, in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pp. 889–896, Sydney, Australia, 2006.

Shen, D., Kruijff, G.J., and Klakow, D. (2005): Exploring Syntactic Relation Patterns for Question Answering, in *Proceedings of the 2nd International Joint Conference on Natural Language Processing (IJCNLP)*, Korea, 2005.

Shen, D., Leidner, J., Merkel, A., and Klakow, D (2006): The Alyssa System at TREC 2006: A Statistically-Inspired Question Answering System, in *Proceedings of the 15th Text Retrieval Conference (TREC 2006)*, Gaithersburg: NIST, 2006.

Shen, D., Wiegand, M., Merkel, A., Kaszalski, S., and Klakow, D. (2007): The Alyssa System at TREC QA 2007: Do We Need Blog06?, in *Proceedings of the 16th Text Retrieval Conference (TREC 2007)*, Gaithersburg: NIST, 2007.

Smartweb (2007): Leitinnovation SmartWeb; mobiler, breitbandiger Zugang zum semantischen Web, *Managementzusammenfassung*, DFKI, 2007.

Song, F. and Croft, W.B. (1999): A general language model for information retrieval, in *Proceedings of the eighth international conference on Information and knowledge management*, Kansas City, USA, 1999.

Spärck Jones, K., Robertson, S., Hiemstra, D., and Zaragoza, H. (2003): Language Modeling and Relevance, in Croft, W.B. and Lafferty, J., editors *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

Spitters, M. and Kraaij, W. (2000): A Language Modeling Approach to Tracking News Events, in *Proceedings of the Ninth Text Retrieval Conference (TREC–9)*, Gaithersburg: NIST, 2000.

Suzuki, J., Taira, H., Sasaki, Y., and Maeda, E. (2003): Question classification using HDAG kernel, in *Proceedings of the ACL 2003 workshop on Multilingual summarization and question answering*, 12:61–68, 2003.

Teahan, W.J. and Harper, D.J. (2003): Using Compression–Based Language Models for Text Categorization, in Croft, W.B. and Lafferty, J., editors *Language Modeling for Information Retrieval*, Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

Tellex, S., Katz, B., Lin, J., Fernandes, A., and Marton, G. (2003): Quantitative evaluation of passage retrieval algorithms for question answering, in *Proceedings of the 26th an-*

nual international ACM SIGIR conference on Research and development in information retrieval, Toronto, Canada, 2003.

Thorne, J. (1962): Automatic language analysis, ASTIA 297381 *Final Technical Report*, Arlington, 1962.

van Rijsbergen, C. (1979): Information Retrieval,*Butterworth, 2nd edition*, 1979.

Voorhees, E. (2002): Overview of the TREC 2002 question answering track, in *Notebook of the 11th Text REtrieval Conference (TREC 2002)*, Gaithersburg: NIST, 2002.

Voorhees, E. and Harman, D. (1999): Overview of the Eigth Text REtrieval Conference (TREC8), in *The 8th Text REtrieval Conference (TREC-8)*, Gaithersburg: NIST, 1999.

Voorhees, E. and Harman, D. (2007): TREC: Experiment and Evaluation in Information Retrieval, *National Institute of Standards and Technology, The MIT Press*, Cambridge, 2007.

Wade, C. and Allan, J. (2005): Passage Retrieval and Evaluation, *CIIR Technical Report*, University of Massachusetts, 2005.

Wahlster, W. (2004): SmartWeb, Mobile Applications of the Semantic Web, in *Proceedings of Informatik*, Springer, 2004.

Wahlster, W. (2007): SmartWeb, ein multimodales Dialogsystem für das semantische Web, in Reuse, B. and Vollmar, R., editors, *40 Jahre Informatikforschung in Deutschland*, Heidelberg, Berlin, Springer, 2007.

Weikum, G. (2000): Information Retrieval, *Lecture Slides Winter Semester 200001*, Saarland University, 2000.

Weikum, G. (2005): Information Retrieval and Data Mining, *Lecture Slides Winter Semester 200506*, Saarland University, 2005.

Whittaker, E., Novak, J., Chatain, P., and Furui, S. (2006): TREC2006 Question Answering Experiments at Tokyo Institute of Technology, in *Proceedings of the Fifteenth Text REtrieval Conference (TREC 2006)*, Gaithersburg: NIST, 2006.

Wikipedia. The Free Encyclopedia. `http://www.wikipedia.org`.

Winograd, T. (1977): Five Lectures on Artificial Intelligence, in A. Zampoli, editor, *Fundamental Studies in Computer Science*, North–Holland, 1977.

Woods, W. (1977): Lunar rocks in natural English: Explorations in natural language question answering, in A. Zampoli, editor, *Linguistic Structures Processing*, Elsevier North–Holland, 1977.

Wu, Y., Zhao, J., and Xu, B. (2006): Cluster–based Language Model for Sentence Retrieval in Chinese Question Answering, in *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, Sydney, Australia, 2006.

Yahoo: `http://www.yahoo.com`.

Zhai, C. (2002): Risk Minimization and Language Modeling in Text Retrieval, *PhD thesis*, Carnegie Mellon University, 2002.

Zhai, C. (2005): Statistical Language Models for Information Retrieval, *Tutorial at the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, Salvador, Brazil, 2005.

Zhai, C. and Lafferty, J. (2001). A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval, in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, 2001.

Zhai, C. and Lafferty, J. (2001b). Model-based feedback in the language modeling approach to information retrieval, in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, 2001.

Zhai, C. and Lafferty, J. (2002): Two–Stage Language Models for Information Retrieval, in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, Tampere, Finland, 2002.

Zhai, C. and Lafferty, J. (2006): A risk minimization framework for information retrieval, in *Information Processing and Management: an International Journal*, 24(1):31–55, 2006.

Zhang, D. and Lee, W.S. (2003): Question classification using support vector machines, in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, Toronto, Canada, 2003.

Zhang, D. and Lee, W.S. (2004): A Language Modeling Approach to Passage Question Answering, in *NIST Special Publications: TREC 2003*, Gaithersburg: NIST, 2004.

Zheng, Z. (2002): AnswerBus Question Answering, in *Proceedings of the second international conference on Human Language Technology Research*, San Diego, California, 2002.