Exploring Rich Evidence for Maximum Entropy-based Question Answering

Dissertation

zur Erlangung des Grades

der Doktorin der Ingenieurwissenschaften

der Naturwissenschaftlich-Technischen Fakultaet II

- Physik und Mechatronik -

der Universitaet des Saarlandes

von

Dan Shen

Saarbruecken

2008

Tag des Kolloquiums:	05.12.2008
Dekanin/Dekan:	Univ.-Prof. Dr. rer. nat. C. Becher
Mitglieder des Pruefungsausschusses:	Univ.-Prof. Dr.-Ing. C. Xu Univ.-Prof. Dr. D. Klakow Univ.-Prof. Dr. M. Crocker Dr.-Ing. O. Farle

Abstract

Exploring Rich Evidence for Maximum Entropy-based Question Answering

Dan Shen

Open domain automated Question Answering (QA) aims to automatically answer users' questions in spoken language. I propose a maximum entropy-based ranking model which effectively integrates various features, including orthographic, lexical, surface pattern, syntactic and semantic features for the answer extraction.

To effectively capture syntactic evidence, I present two methods: dependency relation pattern methods and dependency relation path correlation method. Both methods overcome the problems arising from the divergences of lexical representations between question and answer sentences. I experimentally demonstrate that both methods greatly outperform the state-of-the-art syntactic answer extraction methods on TREC datasets.

To capture semantic evidence, I propose an automatic method to incorporate FrameNet-style semantic role information. The graph-theoretic framework goes some way towards addressing coverage problems related with FrameNet and formulates the similarity measure of semantic structures as a graph matching problem. Experimental results show that the FrameNet-based semantic features may further boost the performance on the answer extraction module.

Furthermore, I propose a maximum entropy-based ranking model to incorporate all captured information. As a result, the model using the optimal feature combination achieves top-ranked performance among all of the participants world-wide in the most recent TREC evaluation.

Abstract

Forschung vom reichem Abweis fuer Maximal Entropie-basisbezogene Frage-Antwort

Dan Shen

Domaenen-unabhaengige automatische Frage-Antwort (QA) is zur automatische Antwort auf die Fragen der Benutzer in muendliche Sprache. Ich stelle eine maximal Entropie-basisbezogen Ranking Modul auf, das tatsaechlich integriert verschiedene Features, inkl. orthographisches, lexikalisches, Oberflaeche Muster, syntaktisches und semantisches Features fuer die Antwort Extraktion.

Um eine tatsaechliche Erfassung der syntaktische Beweise zu erhalten, ich repraesentiere zwei Methoden: Abhaengigkeit Beziehung Muster und Abhaengigheit Beziehung Pfad Zusammenhang. Ich demonstriere versuchsweise, dass die beide Methoden die modernste syntaktische Antwort Extraktion Methoden on TREC Datensatz uebertrifft.

Um die semantische Beweise zu erfassen, ich stelle eine automatische Methode auf, dadurch wird semantische Rolle Information in FrameNet-Art inkorporiert. Das experimentell Ergebnis dass die FrameNet-basisbezogene semantische Features die Leistung on Antwort Extraktion Modul.

Darueber hinaus stelle ich einen maximal Entropie-basisbezogenen Ranking Modul, um alle erfasste Information zu inkorporieren. Als Ergebnis, der Modul, der die optimale Feature Kombination benutzt, erreicht top-ranked Leistung unter alle Teilnehmer weltweit in letzte TREC Bewertung.

Contents

List of Figures

List of Tables

Acknowledgments

This thesis could not have been completed without the support, direction and love of many people. I appreciate all the contributions they have made for the thesis.

First of all, I would like to thank my supervisor, Prof. Dietrich Klakow for his support and patience during the past four years. He has been the best friend and advisor a student could hope for. His insight and inspiration for novel and practical research formed the foundation of this dissertation. He supervised me figure out the big picture in my research and continuously advise me the detailed formulation of my work. I enjoyed all the motivating discussions to orientate my research towards automatic question answering area. Without his guidance, support, care, and patience, I could not have been where I am today.

I would also like to express my deepest appreciation to my co-supervisor, Dr. Geert-Jan Kruijff, for his endless support, continuous encouragement and valuable suggestions on this research.

I have been blessed to have Dr. Mirella Lapata as my supervisor during the visiting in Edinburgh University. She is the best female researcher I have ever met. I am greatly indebted to her for providing me such precious opportunity of working with her. She provided me lots of great ideas and a stimulating environment during that winter. I enjoyed every motivating and efficient discussion on semantic role labeling and question answering problems. What I very much owe to her are not only the academic training she gave me, but also the way she taught me to deal with various challenges on my career path, especially for a female researcher. She selflessly shared with me her invaluable experience in both work and life, which will accompany and motivate me for the whole life.

I am thankful to my thesis reviewer: Prof. Hans Uszkoreit for his critical reading of the thesis and constructive comments that enabled me to clarify the problems more deeply and more extensively.

Throughout my doctoral study, I benefited from numerous discussions with my friends and colleagues in the Spoken Language Systems (LSV) Lab. They shared their wisdom and views from the perspective of their fields of expertise. I am grateful to all of them for their time and assistance: Andreas Merkel, Michael Weigand, Jochen L. Leidner, Stefan Kazalski, Saeedeh Momtazi, Fang Xu, Irene Cramer and Barbara Rauch. Andreas Merkel and Michael Weigand are the colleagues I most closely worked with during the past years. We put continuous efforts together on building the Alyssa QA system. It would have been impossible to obtain such achievement without them. I am especially impressed by those tough days when we struggled for system bugs and caught TREC deadline. It has been great pleasure working in such a warm team. Furthermore, I also wish to thank Diana Schreyer for her always kind help in coordinating all administrative stuffs in LSV, Dietmar Kuhn for his always in-time help in solving all computer-related problems for me.

The years at Saarland would not have been nearly as much fun without the other current and former members of the International Post-Graduate College. Special thanks go to Prof. Matthew Crocker, Prof. Manfred Pinkal, Prof. William Barry and Dr. Frank Keller for their insightful feedback on my work. They inspired me to think of my research more deeply and extensively. I am also grateful to Dr. Sebastian Pado for running Shalmaneser system and helping me complete one of my experiments. Moreover, I owe my most direct thanks to my longtime friends Dr. Yi Zhang and Zhiping Zheng. They helped me on things too many to mention. I am also grateful to Claudia Verburg for her always in-time help in making my life easier in Germany. They made these four years a wonderful experience in my whole life. Last, I gratefully acknowledge the financial support of The Deutsche Forschungsgemeinschaft (DFG) in the form of a research scholarship. I would also like to express my gratitude to the Department of Computational Linguistics which provided me an excellent environment and facilities to study and research.

Special gratitude goes to my husband Feng Ni for his unconditional love, for putting up with the countless hours I spent on my thesis work, and for being there when I needed it. Frankly speaking, it is quite tough to be the spouse of a graduate student, but he went through the last four years without any complaints.

Finally, I owe my foremost thanks to my parents, Peiyu Shen and Lanyun Fang, for their immeasurable love and support which enabled me to succeed. No words in any natural language would be sufficient to thank my parents for all they have done for me. This thesis is dedicated to them. Similarly, I would like to thank my entire family from my parents-in-law to my sister-in-law and brother-in-law for their love, encouragement and support in my Ph.D. journey.

To my beloved husband, Feng Ni.

To my parents, Pei-Yu Shen and Lan-Yun Fang.

Chapter 1

Introduction

1.1 Motivation

Automated open-domain Question Answering (QA) represents an advanced technology of Natural Language Processing (NLP), a branch of both artificial intelligence and information retrieval. It aims to automatically answer users' questions in spoken language.

Question answering is primarily deliberated as an improvement from document retrieval to information retrieval. It is motivated by the belief of users' demands on a better tool for document retrieval practice. Firstly, users want to reduce time and effort in formulating effective queries. Secondly users prefer getting answer snippets to their questions directly rather than finding answers by reading all relevant documents. QA technology aims to significantly save users' time by means of using a series of advanced text processing followed document retrieval to pinpoint exact information.

Research in QA is catalyzed by a series of competitive evaluations conducted by the Text REtrieval Conference (TREC) (Voorhees, 2005; Dang, Lin, and Kelly, 2006). Most of researchers in QA community develop their systems according to the task definition of TREC. TREC divides fact-oriented questions into three types: factoid, list and definition. A factoid question, such as "*Who is the mayor of San Francisco?*", requires

an exact phrase "*Gavin Newsom*" as the answer; A list question, such as "*List the names of chewing gums.*" asks for a set of answer instances; While factoid and list questions cover specific aspects of a target, a definition question, such as "*Who is Aaron Copland?*" expects a summary of all important facts related to the target. I will only focus on answering factoid questions in the thesis.

A typical factoid QA system usually consists of several basic modules: 1. Question Processing (QP) module finds useful information from questions, such as expected answer type and key words. 2. Information Retrieval (IR) module searches a large document collection to retrieve a set of relevant sentences using the key words mined from the questions. 3. Answer Extraction (AE) module further analyzes the retrieved sentences using the information provided by the QP module and identifies answer phrases. These modules are connected as a pipeline in QA architecture. Each of the modules has certain impacts on overall performance. It is very difficult to thoroughly investigate all aspects of a QA system. In this thesis, I mainly study the answer extraction and its effect on the question answering.

In recent researches, answer extraction trends to be more and more complex. To better understand texts, current QA systems widely apply natural language processing techniques, such as named entity recognition, parsing, semantic analysis and logic proving. Sometimes, knowledge from external resources, such as WordNet and the Web are also used to supplement data sparseness in a corpus. The various techniques and resources provide indicative features to find proper answers. These features are further integrated by using a pipeline structure, a scoring function or machine learning methods. While QA systems have shown great success in TREC evaluations, there are still several weaknesses that should be addressed to enhance performance:

1. **Lack of flexibility in matching**. There are generally two key factors to pinpoint answers: 1. check whether the semantic categories of answer candidates accord with questions. For example, the proper answer should be at least an expres-

sion of "*DATE*" for the question "*When was 'Cold Mountain' written?*". Most QA systems employ a fine-grained named entity to recognize the semantic categories of answer candidates. 2. More important, proper answers should be supported by their contexts in sentences. There is an observation that the more common structures question and the context of answer candidate share, the more supportive the answer candidate will be. Therefore, appropriate matching between question and answer sentence is the key issue to the answer extraction. To our knowledge, the matching has been conducted on various levels: surface text level (Soubbotin, 2001; Ravichandran and Hovy, 2002; Wu et al., 2005), syntactic level (Harabagiu et al., 2003; Kaisser and Becker, 2004; Cui et al., 2004) and semantic level (Narayanan and Harabagiu, 2004; Sun et al., 2005; Kaisser, 2006). Although there is full of success on the surface level, I find a lack of work addressing flexible matching in syntactic and semantic spaces. Since syntactic and semantic analysis may supply the deeper understanding of texts, employing syntactic and semantic evidence using appropriate methods reinforces the more accurate identification of answers.

2. **Lack of integration of knowledge**. Most AE modules are on the basis of individual evidence separately, such as surface pattern matching (Soubbotin, 2001; Ravichandran and Hovy, 2002; Wu et al., 2005), syntactic matching (Harabagiu et al., 2003; Kaisser and Becker, 2004; Cui et al., 2004) and Web knowledge (Wu et al., 2005; Kaisser and Becker, 2004). Even if some modules capture multiple evidence, they deal with the evidence simply using a scoring function (Xu, A.Licuanan, and Weischedel, 2003; Kaisser and Becker, 2004; Bos, 2006). The scoring function is formalized as a linear interpolation where the weights of features are heuristically decided by human. When more and more features are captured, the scoring function definitely has its limitation. A desirable system should assign optimal weights to features based on prior knowledge from training data.

Therefore, machine learning techniques will greatly contribute to the integration of rich evidence for the answer extraction.

1.2 Methods

To address above problems, I propose a maximum entropy-based ranking model which effectively integrates various features including orthographic, lexical, surface pattern, syntactic and semantic features.

Considering how to effectively capture syntactic evidence for the answer extraction, I present two methods: dependency relation pattern method and dependency relation path correlation method. Both methods are motivated by the observation that the context of a proper answer in an answer sentence often has similar syntactic structure as a question. However, the two methods apply different measures to calculate similarity between syntactic structures.

Dependency relation pattern method represents a syntactic structure as a dependency relation pattern and regards similarity measure as pattern matching. It follows the hypothesis that the more common individual relations two patterns share, the higher matching score they have. A string kernel method is adapted to partially match patterns, which solves the low coverage problem arising from exact pattern matching in a certain extent. However, the hypothesis has certain limitation due to syntactic relation variations. One meaning is often represented as different syntactic relation combinations, such as "*subject*" relation might turn to "*apposition*" relation in two sentences with the same meaning. In this case, the individual dependency relations of two patterns are totally different and a pattern matching method, regardless of exact matching or partial matching, will unfortunately fail.

Dependency relation path correlation method is further proposed as a backup of the dependency relation pattern method. The method assumes that syntactic structures

are similar if their individual relations are highly correlated. A dynamic time warping algorithm is employed to align two syntactic structures based on the correlations of individual relations. The individual relation correlations are estimated using mutual information measure on training set.

This kind of syntactic evidence overcomes the problems arising from the divergences of lexical representations between question and answer sentences, such as long surface distance and word ordering alternation. More important, they may complement each other since they lean towards different performance aspects (the dependency relation pattern method prefers precision while the dependency relation path correlation method prefers recall). To our best knowledge, these kinds of syntactic evidence haven't been well explored by previous statistical-based answer extraction methods.

In addition to syntactic evidence, I incorporate semantic evidence into the answer extraction. I propose an automatic method to effectively incorporate **FrameNet-style semantic role information**. The way of conducting semantic role assignment mainly relies on the comparison of dependency relation paths attested in FrameNet annotations and raw text. I formalize the search for an optimal role assignment as an optimization problem in a bipartite graph. This formalization allows finding an exact, globally optimal solution. The graph-theoretic framework goes some way towards addressing coverage problems related with FrameNet and formulates the similarity measure of semantic structures as a graph matching problem.

Based on the various similarity measures between question and contexts of answer candidates, a **maximum entropy-based ranking model** is proposed to incorporate all of the captured information. Experiments are conducted on previous TREC data. In Chapter 7, I evaluate the effectiveness of individual features respectively and further report the system performance on various feature combinations.

1.3 Contributions

In this thesis, I mainly contribute to three aspects:

- **Syntactic and Semantic matching methods**. I present formal statistical models and similarity measures to realize syntactic and semantic structure matching between question and answer sentences. The key to a QA system is to find appropriate similarity metrics between a question and answer sentences. I evaluate the effectiveness of these matching methods. The experiments show that both of the methods I propose achieve the better performance than state-of-the-art and have positive effect on the whole QA system. The generic matching methods can be easily extended to other applications that utilize the meaning comparison of two sentences, such as textual entailment, opinion mining and information retrieval.

- **Maximum Entropy-based Ranking model**. I study how to effectively integrate rich evidence into a statistical model. I present and compare two views (classification -based view and ranking-based view) to model the task according to maximum entropy theory. Experiments on previous TREC datasets demonstrate the effectiveness of the models. Moreover, the maximum entropy-based ranking model which integrates all captured features achieves top-ranked performance among all of the participants worldwide in the most recent TREC evaluation.

- **Automated Question Answering System**. I contribute to realize a practical question answering system. In addition to the exploration of the answer extraction, I also develop the other key components in the Alyssa QA system ranging from algorithm exploration, architecture design to module implementation. I mainly put efforts on the modules including question processing module, sentence annotation module, answer extraction module and answer validation module.

1.4 Guide to The Thesis

Chapter 2 In this chapter I review the TREC task definition and main technologies year by year and discuss related approaches in question answering, ranging from inference-based approaches to Web-based approaches. Looking at the roadmap of TREC is not only of historical value, but also reveals general issues and future direction in question answering. Furthermore, the discussion of related work also supplies readers the basic knowledge how the task is modeled and which techniques and resources are used.

Chapter 3 In this chapter, I present the architecture of the Alyssa QA system. I discuss in detail the basic modules that are not covered in later chapters. I leave the Answer Extraction Module that embeds syntactic and semantic analysis, statistical modeling in the next three chapters.

Chapter 4 This chapter introduces two methods to effectively capture syntactic evidence into answer extraction, including dependency relation pattern Method and dependency relation path correlation Method. To our knowledge, the syntactic evidence between proper answers and question key words hasn't been well explored in previous statistical Answer Extraction Module.

Chapter 5 This Chapter proposes a graph-theoretic framework to effectively incorporates FrameNet-style semantic role information into answer extraction. This framework allows us to find an exact, globally optimal solution of semantic role labeling. Furthermore, it goes some way towards addressing coverage problems related with FrameNet and allows us to formulate answer extraction as a graph matching problem.

Chapter 6 In this chapter, I discuss how to model answer extraction task on the basis of Maximum Entropy theory. We view the task in two ways: a classification problem and

a ranking problem. I will give theoretic and experimental comparisons of the two models. And more importantly, I will discuss how to incorporate the syntactic and semantic evidence explored in the previous two chapters into the Maximum Entropy models.

Chapter 7 This chapter reports systematic evaluation results of our answer extraction module including the investigation of individual approaches and the overall effectiveness for their combination. As mentioned above, the conclusions that can be drawn from these experiments are less general since we use a particular question answering system, but nevertheless it provides some useful insights in the interaction between answer extraction module and other components for a concrete system.

Chapter 8 In the last chapter, I draw some overall conclusions for the key issues addressed in this thesis and present possible future work in the end of the thesis.

Chapter 2

Background

Research in open-domain question answering is catalyzed by a series of competitive evaluations conducted by the Text Retrieval Conference. Most of researchers in QA community setup and develop their systems according to the task definition of TREC. From year to year, following a long direction of QA research, TREC revises the definition and evaluation measure which increase the difficulty of the task step by step. In this section, I will firstly introduce the history of practical question answering task in TREC, where some definitions and evaluation measures will be also adopted in this thesis. Next, the main representative approaches for finding answers will be reviewed.

2.1 Question Answering at TREC

2.1.1 Text Retrieval

Text retrieval technology targets to find relevant information in large stores of electronic documents. The first research conference, the International Conference on Scientific Information, devoted to the subject was held in 1958. Since then the problem has continued to grow as more information is created in electronic form and more people gain electronic access. The advent of the hypertextually-networked document collection, World Wide

Web, where anyone can publish anything, is a graphic illustration of the need for effective retrieval technology.

The Text REtrieval Conference (TREC) is an ongoing series of workshops designed to supply the infrastructure for large-scale evaluation of text retrieval methodologies, thereby accelerating its transfer into commercial sector. The series is co-sponsored by the U.S. National Institute of Standards and Technology (NIST) and Advanced Research and Development Activity (ARDA) center of the U.S. Department of Defense. TREC began in 1992 as a part of the TIPSTER Text program. Up to now, there have been 16 workshops. Participants in the workshops have been drawn from the academic, commercial, and government organizations from all over the world. Approximately, more than ninety groups from twenty different countries are evolved in TREC.

Various research topic related to text retrieval, such as seeking information/behavior in the blogosphere, searching data of an organization, answering natural language questions, filtering spam, are considered in TREC. They are divided into different tracks and the respective evaluations are conducted. For each track, NIST releases data sets and test problems to participating groups. After several days for running systems, NIST fairly evaluates the results of the participants using uniform scoring. Furthermore, TREC workshop provides an opportunity for the participating researchers to collect and discuss thoughts and ideas and present current and ongoing research work.

TREC claims that their efforts have accomplished a great deal: within the first six years of the workshops, the effectiveness of retrieval systems approximately doubled. A variety of large test collections have been built for both traditional and hoc retrieval and related tasks. The challenges have inspired a large body of research publications. And moreover, many technologies originally developed in TREC are later incorporated into many of the world's commercial products.

Table 2.1: Task definition of TREC QA Track

	99	00	01	02	03	04	05	06
Answer Type	Answer Snippet			Exact Answer				
Document Collection	TREC	TREC+TIPSTER		AQUAINT				
Question Type	Factoid		Factoid+List		Factoid+List+Definition			
Nil Question	No		Yes					
Question in Series	No				Yes			
Num of Questions	200	693	500		351	530	567	
Evaluation Measure	MRR			CWS	Accuracy			

2.1.2 Question Answering Track

Question Answering as a research endeavor has a long history in Text Retrieval community. NIST has continued to organize QA Track annually from its inception in 1999. The focus of the QA track is to extract answers for users' natural-language questions from large collections of open-domain, natural-language texts.

Up to 2007, QA tracks have been organized for nine years. From year to year, the task is defined more and more difficult regarding type of questions, answer type, document collection, question type and evaluation measurement, etc. Table refTAB:Task definition of TREC QA Track summarizes the definitions of the TREC QA Track from 1999 to 2007. The following is a brief view of the task definition and its main technologies used in TREC each year.

TREC-1999 TREC-8 QA track (Voorhees, 1999) is the first large-scale evaluation of domain independent question answering systems. The task in 1999 was defined to retrieve small snippets of texts that contain actual answer to a question rather than ranked documents traditionally returned by text retrieval systems. Document collection that QA systems worked on consisted of newspaper articles including the Financial Times, the Los Angeles Times and the Foreign Broadcast Information Service. There totally were 528,000 documents containing information on a wide variety of subjects. While the subjects of questions were not restricted, the type of questions (called "factoid questions"

later) was limited to fact-based and short-answer-expected. Participants were given 200 such questions. Each question was guaranteed to have at least one document in the collection that may explicitly answer the question.

All processes of QA systems were required to be strictly automatic and participants were not permitted to further develop their systems once they received test questions. After running the systems without any human intervention, the participants returned top-5 ranked answer snippets per question. An answer snippet was limited to either 50 or 250 bytes and manually evaluated by human assessors. The score computed for a submission was the Mean Reciprocal Rank (MRR) defined as follows.

$$MRR = \begin{cases} \frac{1}{R} & , \quad R \text{ is the rank of the correct answer in topN responses} \\ 0 & , \quad \text{no correct answer in topN responses} \end{cases}$$

An individual question received a score equal to the reciprocal of the rank at which the first correct response was returned, or 0 if none of the five response contained a correct answer. The score for a submission was then the mean of all individual questions' reciprocal ranks.

The general retrieval strategy for most QA systems in TREC-8 is as follows. The systems firstly attempted to classify questions according to their expected answer types. The only evidence used for question classification was the question word, such as a question beginning with "*when*" implied a time designation would be needed. Next, the systems retrieved a small set of relevant documents using standard document retrieval technology. The systems performed shallow named entity recognition of the returned documents to detect phrases with the same type as the expected answer type of the question. If such phrases were found sufficiently close to the question words, the systems returned the snippets with appropriate length surrounding the phrases as responses. If no phrase with appropriate answer type was found, the system would fall back to best-matching-passage techniques.

TREC-2000 While TREC-9 QA task (Voorhees, 2000) essentially had the same definition as TREC-8, there were some minor differences. The document collection that year was the set of news articles on the combined set of TIPSTER/TREC disks. In particular, it included the AP newwire, the Wall Street Journal, the San Jose Mercury News, the Financial Times, the Los Angeles Times and the Foreign Broadcast Information Service. Both the document collection and test set of questions in TREC-9(979,000 documents and 693 questions) were larger than those in TREC-8(528,000 documents and 200 questions). A more substantive difference was the source of the test questions. The majoyrity of questions in TREC-8 were developed by either participants or NIST assessors specifically for the track, which made the questions somewhat unnatural and also the task easier since the questions and the target documents shared the same vocabulary. For the TREC-9 task, NIST obtained two query logs (Encarta log from Microsoft and log for Excite Search Engine) and used those as a source of questions. The switch to "real" questions, rather than assessor constructed questions, made TREC-9 much more difficult than TREC-8. For example, real users asked vague questions such as "*Who is Colin Powell?*" and "*Where do lobsters like to live?*". These kinds of questions were substantially harder for both QA systems to answer and assessors to judge.

Broadly speaking, participants in TREC-9 didn't develop entirely new technologies for the task, but refined the individual steps of their TREC-8 systems. They were better at classifying questions to expected answer types and used a wider variety of methods for finding the entailed answer types in retrieved passages. Moreover, word semantic variations were incorporated into QA systems. Many systems used WordNet as an external source of related word expansion for queries and as a means of determining whether the entities recognized from the paragraphs matched the expected answer types of the questions. In addition, the best performing system, Falcon system from the Southern Methodist University, attempted to incorporate the more linguistic-motivated techniques to fully understand the meaning of the questions and the paragraphs. They

integrated semantic form unification, logical prover and successive feedback loops as follows. Firstly, they searched the relevant paragraphs progressively using the various combinations of query terms and their synonyms / morphological derivations. Next, the retrieved paragraphs were parsed into semantic forms and a unification procedure was performed between the question semantic forms and the paragraph semantic forms. If the unification succeeded, the semantic forms were further translated into logical forms and logical reasoning in the form of an abductive back-chaining from answers to questions was conducted. If the proof succeeded, the answer would be returned. Otherwise, another query with semantic variations would be generated and more paragraphs would be retrieved.

TREC-2001 For the third year of QA track (Voorhees, 2001), while the overall goal remained the same as the previous years, new conditions to increase the realism and difficulty of the task were introduced. The track was divided into three sub-tasks (main task, list task, context task). In the main task, questions were no longer guaranteed to have an answer in document collection. Recognition of no answer is an important ability for operational systems to possess since returning an incorrect answer usually harms more than not returning an answer at all. For a TREC-2001 test question, a correct answer was known to exist if assessors found it during question development or if participating systems returned a correct and supported response to the question. Otherwise, the question would be regarded as a NIL question. The participating systems were required to return ”NIL” if they believed no answer was present. In the list task, the systems assembled a set of snippets as the response for a question, requiring the ability to distinguish among snippets found in multiple documents. Furthermore, the context task required the systems to track discourse objects through a series of questions.

The final test set consisted of 500 questions which were filtered from MSNSearch logs and AskJeeves logs. Almost all systems performed the variants of the strategies seen in the earlier TRECs. They used an external lexicon resource, usually WordNet, to ver-

ify the types of candidate responses. While some systems attempted full understanding of questions and answer sentences, increasingly some systems started to try shallower, data-driven approaches. The data-driven approaches relied on simple pattern matching methods on very large corpora (like web) by assuming that in a large enough data source a correct answer was usually repeated often enough to distinguish it from incorrect ones that occasionaly matched the patterns. For NIL question detection, only five runs had the accuracy greater than 0.25 (Accuracy is computed as the number of questions for which NIL is correctly returned divided by the total number of questions for which NIL is returned). It showed that evaluating confidences of answer responses and recognizing ”no answers” were very challenging for almost all of the systems.

TREC-2002 TREC 2002 QA (Voorhees, 2002) track made significant changes to the task definition as compared with the earlier QA tracks. It still contained two tasks, including main task and list task, as TREC 2001, but systems were required to return exact answers in TREC 2002. That is, the answer string returned by a system in response to a question consisted of a complete answer and nothing else, in contrast to the earlier years when the text snippet simply containing the answer was allowed. Judging only exact answers correct forced the systems to demonstrate that they knew precisely where the answer located in a snippet. Given an answer string, four judgement values were given by human assessors:

- **wrong**: the answer string does not contain a correct answer or the answer is not responsive.
- **not supported**: the answer string contains a correct answer but the document returned does not support the answer.
- **not exact**: the answer string contains a correct answer and the document supports it, but the answer string contains more than just the answer or misses the bits of the answer.

- **right**: the answer string consists of exactly a correct answer and have correct supporting document.

From this year, the QA track started to use AQUAINT corpus as document collection. the AQUAINT corpus is a collection of English news texts, which may be obtained from the Linguistic Data Consortium [1] as catalog number LDC2002T31. The collection is comprised of documents from three different sources: the AP newswire from 1998-2000, the New York Times newswire from 1998-2000 and the English portion of the Xinhua News Agency from 1996-2000. There are approximately 1,033,000 documents and 2 gigabytes of texts in the collection.

In addition to the requirement for exact answers and the adoption of new document collection, the TREC 2002 main task had another significant change from the earlier QA tasks. Systems were limited to on response per question instead of top 5. Thus, the scoring metric was also changed. A new evaluation measure in the main task, the confidence-weighted score (CWS), was used to test a system's ability to recognize when it had found a correct answer. Within the submission file returned by a system, the questions were ordered from the most confident response to the least confident response. That is, the question for which the system was most confident to return a correct answer was ranked first and the question for which the system was least confident was ranked last. Given such ranking, an analog of document retrieval's uninterpolated average precision were computed. The measure rewarded the system for a correct answer early in the ranking more than the one late in the ranking. More formally, if there were Q questions in the test set, the confidence-weighted score was defined as follows:

$$CWS = \frac{1}{Q}\sum_{i=1}^{Q}\frac{\text{number correct in first } i \text{ ranks}}{i}$$

The final test questions(500 questions) for the main task were from MSNSearch logs and AskJeeves logs. Different from the 2001 test set where one quarter of questions

[1] Linguistic Data Consortium: `www.ldc.upenn.edu`

were definition questions such as "*Who is Duke Ellington?*" and "*What are polymers?*", this year's test set didn't contain such questions any more. Since the requirement of exact answers resulted QA systems increasingly complex over those in the earlier years, there was little in common across the systems this year. Most systems classified an incoming question into semantic types according to a predefined ontology of question types as the first step. The ontology ranged from small sets of broad categories, such as PERSON, LOCATION, ORGANIZATION and DATE which corresponded to the predefined named entity types in the Message Understanding Conference(MUC), to highly detailed hierarchical schemes, such the subtypes like CITY, CAPITAL and MOUNTAIN were further considered in addition to the LOCATION type. Once the question type was determined, the systems performed a type-specific processing. Many systems in TREC 2002 used external data sources such as name lists, gazetteers, movie and book databases, to match exact answers in relevant text snippets. Furthermore, systems started to capture Web information in various ways. Some systems used the Web as primary source to find answers. Then they mapped the answers back to the AQUAINT Corpus to get their supporting documents. Other systems did the reverse: used the AQUAINT Corpus as the primary source of answers and then verified the answers with the Web Data. Still other systems used the Web as one of the primary sources. At last they fused the answers from the various sources to select final response.

TREC 2002 showed an increase in the number of systems using shallow, data-driven approaches to question answering in contrast to systems attempting full and deep understanding of texts. Both approaches were well-represented in TREC 2002. The top-scoring system found answers by using logic representations of texts and a formal proof scheme. The second-scoring system relied on an extensive set of surface patterns where each pattern was built from the simpler component structure. The detailed technologies for question answering will be discussed in Section 2.2.

As to the creation of question ranking for confidence-weighted measure, almost all of the systems regarded question type as a factor since some question types were obviously easier to answer than others. Some systems predicated an answer candidate for a question with certain probability. Once the probability was comparable across questions, it can also be used to rank the questions. A few systems used a training set of question and answer pairs from the previous years to learn optimal weights of features and predicted confidence on this year's questions. Many systems ranked NIL responses last since they couldn't find an answer rather than they were sure no answer existed.

TREC-2003 Different from the main task of TREC 2002 where only factoid questions were asked, the main task of TREC 2003 (Voorhees, 2003) involved three types of questions, including factoids, lists and definitions. Each question was tagged with its type.

For each factoid question (totoally, there were 413 questions), systems were required to return one exact answer or NIL as a response. This response would be manually judged to *incorrect*, *not supported*, *inexact* or *correct* by assessors. The total score for the factoid questions is accuracy, the fraction of the responses judged as correct.

For each list question (Totally, there were 37 questions), systems were required to assemble a final response from the information located in multiple documents, such as "*List the names of chewing guns.*". The list question could be simply treated as a shorthand for asking the same factoid question multiple times; the set of answers from the factoid question was the appropriate response for the list question. TREC didn't specify a target number of answer instances to return for the list questions and didn't consider the order of the answer instances. Judgments of *incorrect*, *not supported*, *inexact*, *correct* were made individually for each answer instance as for the factoid questions. The instance precision (IP) and instance recall (IR) for a list question were calculated according to the assessor judgments. Let S be the number of correct answers, D be the number of correct, distinct instances returned by a system, and N be the total number of instances returned by the system. Then $IP = D/N$ and $IR = D/S$. F measure combining the

precision IP and the recall IR with equal weight were used to score a list question:

$$F = \frac{2 \times IP \times IR}{IP + IR}$$

The score for all of list questions was the average F score.

Definition questions ask for the most interesting information snippets about a topic, such as "*Who is Colin Powell?*". These questions occur relatively frequently in logs of web search, which increasingly draw attentions of QA communities. Systems were required to return an unordered set of information snippets as a response for each definition question (totally 50 questions are released). Each snippet was presumed to be a facet in the definition of the target. There were no limits placed on either the length of a snippet or the number of snippets. Judging the quality of system's responses was done in two steps. Firstly, assessors created a list of "information nuggets" about the target. An information nugget was a fact on which the assessors could make a binary decision as to whether a response contained the nugget. The assessor further decided which nuggets were vital. The vital nuggets must appear in a definition. Secondly, the assessors made conceptual match between the system responses and the information nuggets by ignoring word and syntactic differences. If the response matched the vital nuggets, it would be rewarded. If the response matched the not-vital nuggets, it would be neither rewarded nor penalized. And if the response didn't match any nuggets in the list, it would be penalized. The final score of a definition question was measured using F-measure defined as follows:

Let r be the number of vital nuggets returned in a response;

a be the number of acceptable (non-vital but on the list) nuggets returned in a response;

R be the total number of vital nuggets in the assessor's list;

len be of the number of non-white space characters in an answer string summed over all answer strings in the response;

Then $recall = r/R$

$$allowance = 100 \times (r + a)$$

$$precision = \frac{allowance}{len}$$

$$F(\beta = 5) = \frac{\beta^2 \times precision \times recall}{(\beta^2 + 1)precision + recall}$$

The final score for the main task run was computed as a weighted average of the three component scores:

$$FinalScore = 1/2 \times FactoidScore + 1/4 \times ListScore + 1/4 \times DefScore$$

The overall approach for answering factoid questions kept unchanged from the previous years. Systems generally identified the expected answer types of the questions as the first processing step, next retrieved relevant documents or sentences using traditional information retrieval technologies, and then extracted answers by performing a match between the question words and the retrieved sentences. While the overall approach remained the same, individual groups continued to refine their modules for each step to increase coverage and accruacy. The detailed techniques they used will be discussed in Section 2.2

For list questions, no specific techniques were developed. Most groups used exactly their factoid QA modules for the list questions, changing only the number of responses returned. Some systems returned the answers whose scores were above an empirically determined threshold. Other systems returned the answers whose score were within an empirically determined fraction of the top result's score.

Different techniques were developed to answer definition questions. Most systems first retrieved passages or sentences about a target using a recall-oriented search engine. Subsequent processing reduced the amount of materials returned. Pattern-based methods were widely used to locate definition-contents in texts. These patterns were either hand-constructed or learned from a training corpus, such as the gloss of WordNet. Some systems also explored to eliminate redundant information using either word overlap measures or document summarization techniques.

TREC-2004 TREC 2004 (Voorhees, 2004) still contained factoid, list and definition questions, but the questions were grouped into different series, where each series was associated with a target. The questions in a series asked for the information about the target. In addition, the last question in a series was an explicit ”other” question, which was to ask for additional interesting snippets about the target that were not covered by the preceding questions in the series. The ”other” question was roughly equivalent to definition questions. The reorganization of questions into series had an important benefit. Each series could be regarded as the abstraction of an information dialog in which users were trying to define a target. The target and earlier questions in a series provided the context for the current question. Context processing is an important element for question answering systems to possess although it hasn't been successfully incorporated into existing open domain question answering systems. As the first step to consider context processing in question answering, TREC simplified the task by adding the constraints that answers of previous questions were not mentioned in later questions. This appeared as a stilted conversational style comparing with true dialog. Furthermore, systems were required to process series independently from one to another and process an individual series in question order, that is, systems couldn't look ahead and use later questions to help answer earlier questions. The targets defined in TREC 2004 included people, organization and other entities. The same evaluation methodology as TREC 2003 were used in TREC 2004.

The overall approach to answer factoid, list and definition questions remained unchanged for the past several years. The only new difficulty of TREC this year was context processing since a question in a series didn't necessarily explicitly include the target of the series. For document/passage retrieval phase, most systems simply appended the target to query. It was an easy but effective strategy since the target indicated the correct domain of the question and most of retrieval methods treated the query as a set of keywords. There were a variety of approaches performing detailed processing of the question to address the difficulty. One common approach was to replace all pronouns in the question with the target. While for the majority of the questions, pronouns actually refered to the targets, a few questions use definite noun phrases rather than pronouns to refer to the targets, such as "*Q: Where is the company located?*", the phrase "*the company*" refers to the target "*Rohm and Haas*". The other approaches tried varying degrees of anaphora resolution to appropriately resolve references in the questions. However, it was hard to judge how much benefit these systems received from the more extensive processing.

TREC-2005 The task definition of TREC 2005 (Voorhees, 2005) was mainly the same as that of 2004. There were only two minor differences:

1. Targets of TREC 2005 questions could also be events in addition to persons, organizations and other entities in TREC 2004. It suggested that pronouns or definite noun phrases of the questions couldn't be simply replaced with the targets since the approach might result in the ungrammatical questions. For example, the phrase "*the disaster*" of the question "*How many crewmen were lost in the disaster?*" couldn't be replaced with the target "*Russian submarine Kursk sinks*". In addition, answers couldn't be readily found by simply looking up the targets in Wikipedia or other pre-compiled Web resources.

2. TREC 2005 further required systems to consider dependencies between the ques-

tions in addition to dependencies between the questions and the targets. Answers from the preceding questions in a series were probably mentioned in the later questions. For example, the phrase "*the winner*" of the question "*What country did the winner represent?*" refered to the answer of the previous question "*Who won the crown?*". Therefore, the TREC 2005 task was more difficult than the TREC 2004 task, which resulted in the lower scores for most of participating systems.

The overall approaches to answer factoid, list and definition questions remained unchanged for the past several years. One observation was that more and more systems turned to exploit size and redundancy of the Web to help find answers. Some searched on the Web to find answers and then projected the answers back to the AQUAINT Corpus to find supporting documents. Others found answer candidates in the AQUAINT Corpus and then used the Web to rerank them. Furthermore, some groups attempted to develop special strategies for answering list questions rather than simply reusing their factoid-answering systems. They adopted bootstrapping strategies to find additional items from initial seed items.

TREC-2006 TREC 2006 (Dang, Lin, and Kelly, 2006) made only one minor change from TREC 2005. The implicit time frame for a question phrased in present tense was the date of the latest document in a document collection rather than any documents returned with answers. Thus, systems were required to give the most up-to-date answers supported by the document collection. It brought the TREC task closer in line with question-answering in real world, where users would prefer the best answers to their questions rather than any answers found in any documents. The evaluation also equally weighted factoid, list and definition questions. Like TREC 2005, overall approaches were not changed so much. Most groups were still working on their existing systems by adding more refining work.

2.2 Approaches to Question Answering

Today's automatic question-answering systems are revolutionizing textual information processing. By combining complex natural language processing techniques, sophisticated linguistic representations and advanced machine learning methods, QA systems can find exact answers to a wide variety of natural language questions in a large collection of unstructured texts. The process of providing a brief and precise answer to a question is quite different from the task of information retrieval and information extraction, although it depends on both as important components. With the increase of difficulty in QA tasks, ranging from extracting answer-contained snippets to extracting exact answers, from answering individual questions in isolation to answering a series of interrelated questions, more and more linguistic-motivated analysis and external resources are incorporated in QA systems. Systems have to resolve dependency of questions in a series, bridge word gaps between questions and answers, pinpoint exact answers, take into consideration of syntactic and semantic roles of words, rank answers better, justify answers and so forth. All of these are beyond document retrieval techniques. Especially for factoid question answering, the use of answer type ontology and natural language processing technique has demonstrably yielded significant gains over pure IR techniques. In the following, I'll review various representative approaches of building practical factoid question answering systems.

2.2.1 Inference-based Approaches

Language Computer Corporation group (Harabagiu et al., 2001; Moldovan et al., 2002; Harabagiu et al., 2003; Moldovan et al., 2004; Harabagiu et al., 2005) conduct automated reasoning by integrating a logic prover, COGEX into QA system. The overall approach is to transform questions and candidate answer passages into logic representations and use world knowledge and linguistic axioms to find which answers are acturally correct.

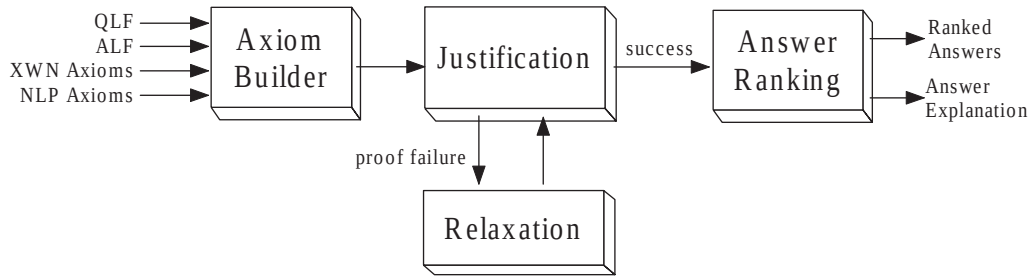


Figure 2.1: Architecture of the COGEX logic prover

The architecture of the LCC inference-based module is shown in Figure 2.1. COGEX captures syntax-based relationships provided by logic representation of questions (QLF) and candidate answer passages (ALF). In addition, it needs world knowledge axioms (XWN axioms) to link questions concepts to answer concepts and NLP axioms to represent semantically equivalent lexical patterns.

The main procedure of COGEX is described as follows: firstly, **Axiom Builder** converts logic forms for questions, answer passages and WordNet glosses into axioms. Based on parse tree patterns in questions and answers, additional NLP axioms are built to supplement existing general NLP axioms. Once the axioms are complete and loaded, **Answer Justification** begins. If a proof fails, **Relaxation Module** is invoked. The purpose of the Relaxation module is twofold:

1. to compensate from errors in text parsing and logic form transformation;

2. to detect correct answers when NLP and XWN axioms fail to provide necessary inferences.

During the Relaxation, the argument-to-predicate assignments in a question are incrementally uncoupled, the proof score is reduced and the justification is re-attempted. The loop between the answer justification and the relaxation module continues until the proof succeeds or the proof score is below a predefined threshold. When all of answer candidates are processed, the answer candidates are ranked according to their proof scores and

the answer justification results are further ouputted from COGEX. The detailed operation of COGEX is discussed in (Moldovan et al., 2003).

Naturally, performing automated reasoning in the context of QA is very complex. One of main challenges is logic representation of open texts. A text logic form is an intermediate step between syntactic parse and deep semantic parse. It represents syntax-based relationships:

- syntactic objects
- syntactic subjects
- prepositional attachments
- complex nominals
- adverbial/adjectival adjuncts

The process of transforming a question or an answer passage to logic forms is illustrated in Figure 2.2. The QLF and ALF are produced in a three-layered first order logic representation.

1. The first layer relies on syntactic parse and named entity recognition.
2. The second layer relies on the recognition of semantic relations processed by a semantic parser.
3. The third layer represents temporal contextual information produced by temporal ordering of events, anchoring events in time intervals and normalizing temporal expressions.

More details regarding the three-layer transformation are discussed in (Moldovan and Rus, 2001; Bixler, Moldovan, and Fowler, 2005; Moldovan, Clark, and Harabagiu, 2005).

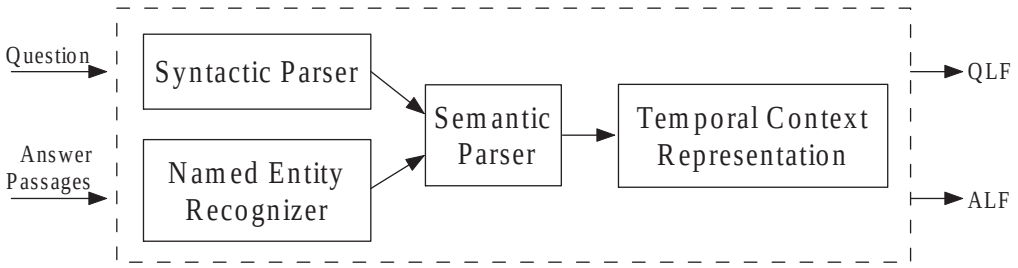


Figure 2.2: Three-Layered Logical Form Transformation

Besides the QLF and ALF, world knowledge axioms and NLP axioms also participate in the answer proof. The world knowledge axioms are necessary to conceptually link questions and answers. They are derived from the glosses of WordNet and eXtended WordNet by transforming the glosses into logic forms. For example, the gloss definition of the concept "*sport_NN#1*" is "*an active diversion requiring physical exertion and competition*", which yields the logic representation:

active_JJ(x1) & diversion_NN(x1) & require_VB(e1,x1,x2) & or_CC(x2,x3,x4) & physical_JJ(x3) & exertion_NN(x3) & competition_NN(x4)

A much improved source of the world knowledge axioms is obtained by mining connection paths between SynSets of WordNet up to a certain distance (Moldovan and Novischi, 2002). Lexical chains, which are sequences of semantically related words, are established between SynSets in different hierarchies. The words along a path are topically related. For example, the path between "*zygote_NN#1*" to "*nuclues_NN#1*" is

zygote_NN#1 → HYPERNYM → cell_NN#1 → HAS_PART → nucleus_NN#1

As to the NLP axioms, they are manually constructed to reflect equivalence classes of linguistic patterns. The considered equivalence phenomena include:

Complex Nominals and Coordinated Conjunctions Noun phrases with full proper names may be equal to their abbreviated forms, such as "*Internet browser Mosaic*" refers to "*Mosaic*"; "*Microsoft*" to "*Microsoft Corp.*".

Appositions Two noun phrases in an apposition are equal with each other, such as "*Italian Andrea Pfister*" is equal to "*designer*" in "*... Italian Andrea Pfister, designer of the 1979 'bird cage' shoe that spawned millions of plastic imitations, ...*".

Possessives Noun phrases may substitute the use of a possessive "*'s*" by using an "*of*" or "*by*" preposition, such as "*Wright brother's first flight*", "*the first flight of Wright brothers*" and "*a 120-foot flight by Wright brothers*" refer to the same.

Prepositions Prepositions may be grouped into equivalence classes depending on expected answer types of questions. In location seeking questions, the prepositions "*at*" and "*in*" are often interchangable. Similarly for "*in*" and "*into*", such as in question "*Q: What body of water does the Colorado River flow into?*" and candidate answer sentence "*... the Colorado River flowed in the Salton trough about 130 miles ...*", the preposition "*into*" and "*in*" take in the same meaning.

Part-of Relations in Location Questions A location question may get a correct answer by referring to a part-of relation for the location. For example, for the question "*Where is Devil's Tower?*" and the answer sentence "*... Devil Tower in the northeast corner of Wyoming ...*", "*Wyoming*" is identified as answer only if system identifies "*Devil Tower*" lies in a part of "*Wyoming*".

As one of the deepest linguistic-motivated method in modern QA systems so far, the inference-based method achieves high accurate but low recall. The high failure rate is mainly due to insufficient axioms. In addition, the method takes long processing time. LCC addresses these issues by adding the logic prover to other modules as an augment but not replacement. The experiment on TREC 2002 questions (415 questions) shows that 415 questions are correctly answered with the logic prover by comparing the original system (317 questions are correctly answered without the logic prover).

2.2.2 Statistical-based Approaches

(Ittycheriah, Franz, and Roukos, 2001; Ittycheriah and Roukos, 2002) develop a statistical-based question answering model. They argue that mathematical framework is tractable and can provide the performance close to state-of-the-art without any specific system tuning efforts. They derive a formal mechanism which incorporates sentence selection, question typing and answer selection into one uniform model rather than separates them into independent components. They model the distribution $p(c|a,q)$, which attempts to measure the "correctness" c given the answer a and the question q. The modeling paradigm uses a classification point of view, as opposed to a generative model where $p(a|q)$ is directly modeled. They argue that the $p(a|q)$ paradigm may suffer from insufficient training data. Furthermore, they introduce a hidden variable representing the class of the answer e (answer tag/named entity type). The model is designed as follows,

$$\begin{aligned} p(c|q,a) &= \sum_e p(c,e|q,a) \\ &= \sum_e p(e|q,a)p(c|e,q,a) \end{aligned}$$

Where, $p(e|q,a)$ is **Answer Tag Model** which predicts the entity that both question and answer satisfy. By assuming that the question and the answer are independent and conditionally independent of the entity, the answer tag model turns to be

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

Where, $p(e|a)$ is a traditional **named entity recognition problem** and $p(e|q)$ is a **question typing problem**.

$p(c|e,q,a)$ is **Answer Selection Model**. Given the question q, the answer candidate a and the entity e predicted by the answer tag model, the correctness of the configuration is predicted. Six feature sets including entity features, definition features, linguistic features, web features, statistical machine translation features and answer patterns, are incorporated into the answer selection model. The details of the features and the models are discussed in (Ittycheriah et al., 2002).

(Ravichandran, Hovy, and Och, 2003) further contrast two answer extraction models based on a Maximum Entropy paradigm. The first model regards the answer extraction as a classification problem $p(c|q, a)$ as (Ittycheriah et al., 2002), while the second model views the task as a re-ranking problem $p(a|q, \{a_1, a_2, ...a_n\})$. They obtain good baseline performance for question answering by only using 4 basic features (frequency, expected answer class, question word absent and word matching). They also argue that QA system modeled as a re-ranking problem significantly outperforms that modeled as a classification problem.

(Xu et al., 2002) select answers by estimating $p(c|q, a)$. They use three features to represent the question q and the answer a.

- The first feature (VS) is a boolean feature judging whether a satisfies the verb argument of q.
- The second feature is a pair of integers (m, n), where m is the number of content words in common between q and the context of a, and n is the total number of content words in q.
- The third feature (T) is the answer type of q. It is based on the assumption that some types of questions (e.g. "*Person*") are more likely to result in correct answers than other types (e.g. "*Animal*").

They pre-compute $p(c|VS)$, $p(c|(m, n))$ and $p(c|T)$ from training data and fit them into a mixture model as:

$$p(c|q, a) = 1/3 \times p(c|VS) + 1/3 \times p(c|(m, n)) + 1/3 \times p(c|T)$$

So far, all of the above methods estimate the correctness of individual answer candidates separately and don't take into account the relevance information between the answer candidates. (Ko, Mitamura, and Nyberg, 2007) propose a probabilistic graphical model for answer ranking. The model estimates the probability of the correctness

of an answer candidate given multiple answer relevance features and answer similarity features. The task is modeled using logistic regression as follows:

$$
\begin{aligned}
& p(c(a_i)|q, a_1, a_2, ...a_n) \\
& \quad \approx \quad p(c(a_i)|rel_1(a_i), ..., rel_{K_1}(a_i), sim_1^{'}(a_i), ..., sim_{K_2}^{'}(a_i)) \\
& \quad = \quad \frac{exp\left(\alpha_0 + \sum_{k=1}^{K_1} \beta_k rel_k(a_i) + \sum_{k=1}^{K_2} \lambda_k sim_k^{'}(a_i)\right)}{1 + exp\left(\alpha_0 + \sum_{k=1}^{K_1} \beta_k rel_k(a_i) + \sum_{k=1}^{K_2} \lambda_k sim_k^{'}(a_i)\right)} \\
& \text{where, } sim_k^{'}(a_i) = \sum_{j=1(j \neq i)}^{N} sim_k(a_i, a_j)
\end{aligned}
$$

In the above model, $rel_k(a_i)$ is a feature function used to produce an answer relevance score for an individual answer candidate a_i. $sim_k(a_i, a_j)$ is a scoring function used to calculate answer similarity between a_i and a_j. $sim_k^{'}(a_i)$ represents one similarity feature for an answer candidate a_i and is obtained by summing $N-1$ answer similarity scores to represent the similarity of one answer candidate to all other candidates. The parameters α, β_k and λ_k are estimated from training data by maximizing log likelihood. In particular, the Quasi-Newton algorithm is used. Multiple resources are used to generate answer relevance scores and answer similarity scores.

Answer Relevance Feature It consists of knowledge-based features and data-driven features. The knowledge-based features are extracted from some publicly available gazetteers (the Tipster Gazetteer, the CIA World Factbook and 50states.com) providing geographic information, and semantic ontology (WordNet) containing the relationship information between words and general meaning types. Data-driven features are extracted from Wikipedia by calculating tf.idf score and Google by computing the minimum number of words between question keywords and answer candidates.

Answer Similarity Feature It is calculated using multiple string distance metrics, such

as Levenshtein, Jaro-Winkler and Cosine similarity, and a list of synonyms from WordNet, Wikipedia and the CIA World FactBook.

2.2.3 Pattern-based Approaches

(Soubbotin, 2001) is the first work to successfully exploit pattern-based approach for question answering. They predefine patterns of textual expressions for certain types of questions. The presence of the patterns in answer sentences may provide evidence of proper answers. The way that they define the indicative patterns is totally heuristic and inductive. Each pattern is represented as a sequence of strings. For example, the patterns for the query type "*when_born*" are as follows:

1. capitalized word; parenthesis; four digits; dash; four digits; parenthesis
2. capitalized word; + "in" + four digits + "born"

Since the pattern-based approach of (Soubbotin, 2001) achieves very promising performance in TREC 2001 evaluation, it starts to draw more attention in QA community. Following that, a series of related explorations are conducted.

(Zhang and Lee, 2007) propose a Web-based pattern mining and matching approach to question answering. For each type of question, textual patterns are automatically learned from the Web using the previous TREC data as training examples. The textual patterns are assessed by the concepts of support and confidence measures, which are borrowed from data mining community. Given an unseen question, the patterns are utilized to extract and rank plausible answers on the Web. They define 22 questions classes. Each class has several templates formulated as regular expressions which indicate the possible appearance of the question class. For instance, some of the templates in the "*ACRONYM*" class are as the following, where, "_Q_" stands for question key phrases.

1. What is _Q_
2. What is the meaning of _Q_
3. What the (?:acronym|abbreviation) _Q_ (?:stands for|means)
4. What _Q_ (?:stands for|means)
5. What the initials _Q_ (?:stands for|means)

For each question class, a set of textual patterns are learned from the Web. Some of the discovered textual patterns of the "*WHO-IS*" class are shown in the following, where "_A_" indicates potential answer position and two special symbols "_<_" and "_>_" indicate the head of snippet and the tail of snippet respectively.

patterns	confidence
1. , _A_ became _Q_	0.09
2. _<_ _A_ was _Q_	0.11
3. _Q_ was _A_ ,	0.05
4. _A_ made history as _Q_	1.00
5. by _A_ (_Q_	0.66
6. _Q_ , _A_ _>_	0.14

(Ravichandran and Hovy, 2002) learn surface patterns automatically from the Internet by using a bootstrapping process. They calculate the precision of each surface pattern and the average precision of each question type. They further use the precision scores to select the patterns and obtain an optimal pattern set. The final pattern set is then applied to find answers to new questions.

To learn the surface patterns from the Web, for each question, they submit the question terms and the answer as query to a search engine and download top 1000 web documents provided by the search engine. Next, they pass each sentence through a suffix tree constructor. They use suffix trees for extracting all substrings of all lengths along with counts. Suffix trees can be processed in time linear on the size of corpus and more

importantly, they don't restrict the length of substrings. The above procedure is repeated for all questions of the same question type. As a result, for the question type "*BIRTH-DATE*", the most common substrings of extracted sentences are:

1. born in <ANSWER> , <NAME>
2. <NAME> was born on <ANSWER>
3. <NAME> (<ANSWER> -
4. <NAME> (<ANSWER> -)

Where, the tag <NAME> and <ANSWER> represent question term and answer term respectively.

Next, they calculate precision of each pattern by the formula $P = C_a/C_o$, where, C_a is the number of patterns with answer term present; C_o is the total number of patterns. When extracting answers for a new question with pattern matching, they use the precision scores to rank answer candidates.

(Wu et al., 2005) propose a supervised learning method to automatically generate patterns from the Web and represent them a format of regular expressions which can handle up to 4 question terms. Question types are defined as the abstracts of ordering chunk sequences. For example, the questions "*When did the United States enter the World War II?*" and "*When did Amtrak begin operations?*" belong to the question type "*when_do_np_vp_np*". The format of answer patterns are defined as regular expressions with various chunk terms, such as "*NP*", "*VP*", "*VPN*", "*ADVP*","*be*", "*in*", "*of*", etc. If more than one noun phrase occur in a question, they index the noun phrases according to their occurring order in the question. The following are the examples of answer patterns for the question type "*when_do_np_vp_np*":

1. NP1 VP NP2 in <Date>([^ <>]+?)</Date>
2. NP2 in <Date>([^ <>]+?)</Date>.{1,15} NP1
3. <Date>([^ <>]+?)</Date> NP1 VP.{1,15} NP2
4. NP1's NP2 in <Date>([^ <>]+?)</Date>

They have more than 50 question types. The number of answer patterns varies with the question types. Some question types have up to 500 patterns. The patterns with the scores greater than 0.5 are applied in pattern matching.

To generate answer patterns for certain question type, they use TREC11, TREC12 and TREC13 questions and answers as training question-answer pairs. For each question-answer pair, they choose top50 documents retrieved by Google. Then, the patterns are generated from the documents by replacing question noun phrases with a symbol, such as "NP_1", "NP_2". Each extracted pattern is scored by its precision Pr_i and frequency F_i as follows:

$$\begin{aligned} Pr_i &= \frac{C_i}{A_i + \varepsilon} \\ &\quad (\varepsilon = 0 \text{ if } A_i \geq 5; \varepsilon = 1 \text{ if } 3 \leq A_i \leq 4; \varepsilon = 2 \text{ if } A_i \leq 2) \\ F_i &= \frac{A_i}{\sum_{k=1}^{n} A_k} \\ S_i &= Pr_i \times (1 + F_i) \end{aligned}$$

where, $A_i(i = 1, 2, ..., n)$ is the frequency of the pattern

$C_i(i = 1, 2, ..., n)$ is the frequency of the pattern which leads to correct answer

Furthermore, they find that supervised iterative optimization is necessary to get the more accurate pattern distribution. They apply the patterns whose scores are greater than a threshold to sample questions and manually evaluate returned answers. For some questions, correct answers which haven't been included in original training question-answer pairs will be found and added to the training set. This process is repeated until no correct answers are found any more.

2.2.4 Machine Translation-based Approaches

(Echihabi and Marcu, 2003) introduce a probabilistic noisy-channel model for question answering. Given a set of question-answer pairs (Q, S_A), they train a probabilistic model for estimating the conditional probability $P(Q|S_A)$. Once the parameters of the model

are learned, given a question Q and the set of sentences Σ returned by an IR engine, one can find the sentence $S_i \in \Sigma$ and the answer in its $A_{i,j}$ by searching for the $S_{i,A_{i,j}}$ that maximizes the conditional probability $P(Q|S_{i,A_{i,j}})$. The core idea of the noisy-channel model is to make explicit mapping from answer sentence parse trees to question parse trees. It can be formulated as the computation of alignment probabilities for target strings (questions) given source strings (answer sentences). The model is nothing but a one-to-one reproduction of sentences which has been widely used in Statistical Machine Translation (SMT) area. They use a publicly available SMT system - GIZA package to automatically compute the Viterbi alignment probabilities between flattened answer parse trees and question trees. In their experiment, they state that the noisy-channel system outperform state-of-the-art rule-based systems that took many person years to develop. It is remarkable that a SMT system can do so well in a totally different context - open domain question answering. Furthermore, building dedicated systems that employs the more sophisticated, QA-motivated generative stories is likely to yield significant improvements.

(Wang, Smith, and Mitamura, 2007) build on the idea that questions and correct answer sentences relate to each other via loose but predictable semantic and syntactic transformations. They propose a probabilistic quasi-synchronous grammar, inspired by the one proposed for machine translation and parameterized by a mixture of a robust non-lexical syntax alignment model(Base Model) with a lexical-semantics log-linear model. The Base Model mainly considers three factors, including POS labels, named entity labels and dependency relation labels, when aligning question words to answer sentence words. The lexical-semantics-driven model further considers thirteen classes of WordNet relations between words, including "*identical-word*", "*synonym*", "*antonym*", "*hypernym*", "*hyponym*", "*derived form*", "*morphological variation*", "*verb group*", "*entailment*", "*entailed-by*", "*see-also*", "*causal relation*" and "*q-word*" relations. They incorporate such relation features into a log-linear model. Finally, the model learns soft

alignments as a hidden variable in discriminative training. Experimental results using TREC dataset are shown to significantly outperform traditional syntactic tree matching methods.

2.2.5 Web-based Approaches

The vast amount of information available on the World Wide Web makes it an attractive external resource for question answering (Clarke, Cormack, and Lynam, 2001; Clarke et al., 2001; Breck et al., 2001; Dumais et al., 2002; Lin, 2002). Web data is viewed as an enormous collection of unstructured, flat texts with tremendous amount of data redundancy. Its immense size qualitatively changes the nature of question answering task as compared to the same task on close corpora, such as news paper texts, encyclopedias, etc. Due to the data redundancy on the Web, any piece of information might be stated in a variety of ways in different documents. Let's borrow an example from (Lin, 2002): considering the question "*Who killed Lincoln?*", there are two possible answer sentences:

> 1. John Wilkes Booth killed Lincoln.
> 2. John Wilkes Booth is perhaps America's most infamous assassin. He is best known for firing the bullet that ended Abraham Lincoln's life.

Obviously, the answer could be much more easily extracted from the sentence(1) than the passage(2). QA turns to be easier if answer sentences are stated as the simple reformulations of questions. In this case, simple techniques, such as keyword-based retrieval and surface pattern-based matching, can perform well although they are not good enough on small corpora. The larger the text collection is, the greater the possibility of having simple statement is. Therefore, data redundancy can be used as a surrogate for sophisticated natural language techniques.

Moreover, with the increase of data size, the quality and credibility of individual documents are decreased. Some documents are poorly written, or contain incorrect in-

formation. As a result, answers extracted from a single document might not be trustable enough to be globally correct. Data redundancy alleviates this issue since multiple occurrences of an answer in different documents lead to the higher credibility level.

Considering the above benefits, more and more QA systems incorporate Web resource. The Web resource is mainly used in the two modules regarding the whole system architectures: Answer Extraction Module (Wu et al., 2003; Wu et al., 2004; Wu et al., 2005; Kaisser and Becker, 2004) and Answer Validation Module (Harabagiu et al., 2005; Xu et al., 2002).

As the most representative work of using the Web resource in Answer Extraction Module, the QA system of Fudan University (Wu et al., 2003; Wu et al., 2004; Wu et al., 2005) is solely built on the Web data. They retrieve relevant snippets from the Web using Google and find answers from the snippets. To construct queries for Google search, they parse questions using LinkParser and then extract four constituents from the parsed questions: subject, predicate, object and adverbial modifier. Next, queries are formulated according to the constituents of the questions. For example, they splitted the question "*What book did Rachel Carson write in 1962?*" into the following constituents:

1. Rachel Carson - subject
2. wrote - predicate
3. in 1962 - adverbial modifier

The following queries from tight to loose are formulated from the above constituents.

1. "Rachel Carson wrote" "in 1962"
2. "Rachel Carson wrote" "in 1962"
3. "Rachel Carson" wrote in 1962
4. Rachel Carson wrote in 1962

From relevant snippets, they extract answers simply using a surface pattern matching method. Different from other methods, they abstract question phrases using different

classes:

- Q_Quotation: the quotation parts in a question;
- Q_Focus: the key words representing the object or event which a question asks about;
- Q_NamedEntity: the name entities in a question;
- Q_Verb: the main verb of a question;
- Q_BNP: the noun phrases of a question.

The different classes of question phrases are assigned different weights in surface patterns. Surface pattern matching scores are calculated by incorporating the weights.

(Kaisser and Becker, 2004) manually construct 157 surface patterns and make strict and fussy surface pattern matching on the Web data. They further use a Google Fallback Mechanism to backup the surface pattern matching methods. The Google Fallback Mechanism exploits n-gram information matching on Google snippets. Final scoring function is designed as the linear interpolation of the strict surface pattern matching, the fussy surface pattern matching and the Google fallback matching. The experiments on TREC 2004 data show that the fallback mechanism performs better than the surface pattern matching methods.

Besides directly finding answers from the Web, researches also use the Web data in Answer Validation Module. It may effectively overcome the locally correct answer problem. (Harabagiu et al., 2005) explore "Web-boosting" features based on a web strategy that utilizes general linguistic patterns to construct a series of search engine queries. Once an answer candidate from TREC collection also occurs in web documents, an additional feature capturing web redundancy information will be fired to boost the answer candidate. As a result, the feature leads to another ranking of the answer candidates produced by the original Answer Extraction Module. The evaluation results on TREC 2005 data show that "web-boosting" provide an added value of 69/331 to final factoid score.

(Xu et al., 2002) apply the Web data to supplement TREC corpus. For efficiency consideration, they look for answers in Top 100 Google hits for a Web search and confine whole web pages to short summaries in order to further reduce processing cost. Two measures are proposed to incorporate the Web information:

1. The confidence of the answer A found from the Web is a function of the question type T and the answer frequency F in Top 100 Google summaries. Specifically,

$$\begin{aligned} p(correct|Q,A) &= p(correct|T,F) \\ &= p(correct|T) \times 0.5 + P(correct|F) \times 0.5 \end{aligned}$$

   where, $T =$ question type, $F =$ frequency of A in Google summaries

2. The confidence of the answer A is a function of its frequency F in Top 100 Google summaries and a Boolean variable $INTREC$, which is true if and only if A is also returned from the TREC corpus. Specifically,

$$p(correct|Q,A) = p(correct|F, INTREC)$$

As a result, they also confirm the positive findings reported in the earlier studies (Dumais et al., 2002). The experiments on TREC 2002 data show that answer frequency in Top 100 Google summaries is a strong predictor of answer correctness.

2.2.6 Paraphrasing-based Approaches

Different from previous methods which mainly focus on the analysis of retrieved sentences using linguistic processing, statistical tools or external resources, paraphrasing-based approaches put more efforts on question analysis. They observe that the more alternative queries representing surface forms of answer sentences are generated from questions, the more likely proper answers occur in the retrieved sentences. The variations of queries are automatically generated by using paraphrasing techniques.

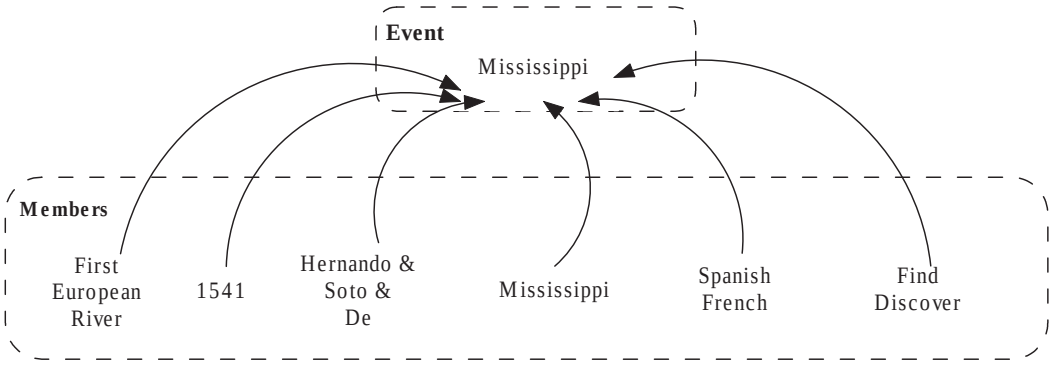


Figure 2.3: Example of the event constructed from the question "*What Spanish explorer discovered the Mississippi River?*"

(Yang and Chua, 2002; Yang et al., 2003) propose an event mining-based method for question processing. The method is summarized as the following steps:

1. To bridge semantic gaps between query space and document space, they integrate the knowledge of pre-retrieved TREC documents, Web, WordNet and manually constructed Ontology to extract additional evidence for an original query. Therefore, the new query contains terms that are related to the local/lexical contexts of the knowledge resources.

2. They perform event construction to discover different elements of an event and further mine relationships between elements which are represented as association rules.

3. They employ the event knowledge to perform boolean query formulation.

4. Given the newly formulated queries, relevant documents are retrieved and answers are extracted by matching association rules between the elements.

Figure 2.3 shows the example of the event constructed from the question "*What Spanish explorer discovered the Mississippi River?*".

(Kaisser, 2006; Kaisser, Scheible, and Webber, 2006) propose a natural language generation-based question analysis. They generate potential answer templates

on the basis of FrameNet paradigm and use the answer templates to induce queries. The queries are sent to Google search engine and the retrieved snippets are further analyzed. If a snippet matches one of the answer templates, the answer will be extracted straight. For example, given the question "*Who purchased YouTube?*", the frame "*Commerce_buy*" is evoked by the question predicate "*purchase*". Furthermore, the question key phrases ("*who*" and "*YouTube*") are assigned with the frame elements ("*FE_Buyer*" and "*FE_Goods*") respectively.

Who → "*FE_Buyer*" purchase → "*Lexical_Unit*" YouTube → "*FE_Goods*"

Using FrameNet annotated sentences, they generate the following potential answer templates.

1. ANSWER[NP] (has|have|had) purchased YouTube
2. YouTube (was|were) purchased by ANSWER[NP]
3. ANSWER[NP] (has|have|had) bought YouTube
4. YouTube (has|have|had) been bought by ANSWER[NP]
5. ANSWER[NP-Genitive] purchase of YouTube
6. YouTube (has|have|had) been sold to ANSWER[NP]
7. YouTube (has|have|had) been retailed to ANSWER[NP]

They consider not only various lexical units in one frame but also lexical units in interrelated frames. In the above example, the lexical units "*purchase.v*","*buy.v*" and "*purchase.n*" (shown in the answer templates $\{1, 2, 3, 4, 5\}$) are considered since they are evoked by the frame "*Commerce_buy*". Moreover, the lexical units "*sell.v*" and "*retail.v*" (shown in the answer template $\{6, 7\}$) are also considered since they are evoked by the frame "*Commerce_sell*" which has the "*is_perspectivized_in*" relation with the original frame "*Commerce_buy*".

Lastly, answers are extracted by matching the potential answer templates and expected answer types.

2.2.7 Knowledge-based Approaches

Besides exploring a unified model, such as inference model, statistical model, machine translation model and paraphrasing model, to integrate various evidences around answer candidates, some researchers only use simple scoring functions to incorporate these evidences and rank answer candidates. We call them Knowledge-based approaches since they mainly put their efforts on mining informative evidence/knowledge rather than developing mathematic models.

Amsterdam Textual QA System, Tequesta (Monz and Rijke, 2001), build Answer Extraction Module based on dependency structure matching. The dependency structures are formed by three types of basic constituents, such as noun phrase(NP), prepositional phrase(PP) and verb group(VG). VG is the head of a dependency structure; NP and PP in its vicinity are the arguments or modifiers of the VG. They generate dependency structures of questions and answer sentences respectively. Then they make comparison between the question structures and the answer sentence structures. Given two structures, the comparison involves two steps

- Checking whether the VGs of the structures match;
- Checking the overlap between the arguments of the structures.

Finally, answer candidates are ranked by a heuristic scoring function which takes into account the above two comparison steps.

BBN (Xu et al., 2002; Xu, A.Licuanan, and Weischedel, 2003) develop Answer Extraction Module according to the degree of matching between questions and answer sentences. The matching is defined on the syntactic and semantic levels. In addition, several additional heuristic rules are defined to penalize answers for a variety of reasons:

- Expected answer type matching: for the question "*How long did the Manson trial last?*", the answer candidate "*20 miles*" is semantically mismatched.

- Vagueness penalty: for the question "*Where is Luxor?*", the answer "*on the other side*" is too vague.

- Negation penalty: for the question "*Who invented the electric guitar?*", there is a negation in the candidate sentence "*Fender did not invent the electric guitar*".

To perform the above penalization, they incorporate external knowledge including WordNet hierarchy, internal quantity, calendar conversion routines and abbreviation routines.

(Bos, 2006) explore linguistically-principled knowledge in Answer Extraction Module. They use Combinatory Categorical Grammar (CCG) to generate syntactic structures of questions and potential answer contexts, and formalize the matching of questions and answer sentences according to Discourse Representation Structure(DRS). The key idea of the module is to use semantics to prune answer candidates, thereby exploiting lexical resources such as WordNet and NomLex to facilitate the selection of answers. The following information is considered in the module:

- Synonyms and hyponyms for nouns and verbs derived from WordNet;

- Hyponyms for nouns harvested from corpora using lexical patterns;

- Nominalization rules generated from NomLex;

- Specialized knowledge, such as attributes (colour, shape), and geographical knowledge (continent, state, country, capital);

- A couple of hand-crafted general inference rules.

Chapter 3

Architecture of the Question Answer System

We, Spoken Language Systems of Saarland University, develop a statistically-inspired open-domain Question Answering research system (Alyssa). The system serves as an experimental platform for QA researchers and allows carrying out a wide range of individual module experiments easily and flexibly. The focus of the thesis, Answer Extraction, is setup on the basis of the Alyssa system. Therefore, before discussing the Answer Extraction methods in Chapter 4 5 and 6, I will give a brief overview how the Alyssa system works in this chapter.

The Alyssa system consists of six basic modules, including Question Processing Module, Document Retrieval Module, Sentence Retrieval Module, Sentence Annotation Module, Answer Extraction Module and Answer Validation Module. The modules are connected through a pipeline structure, as shown in Figure 3.1. A user question firstly undergoes the Question Processing Module, a phase in which several steps are involved independently. The type of the question is determined and a series of linguistic analysis is carried out, including key phrase extraction, syntactic parsing and semantic analysis. Moreover, a query is constructed from the question and is run against the Document

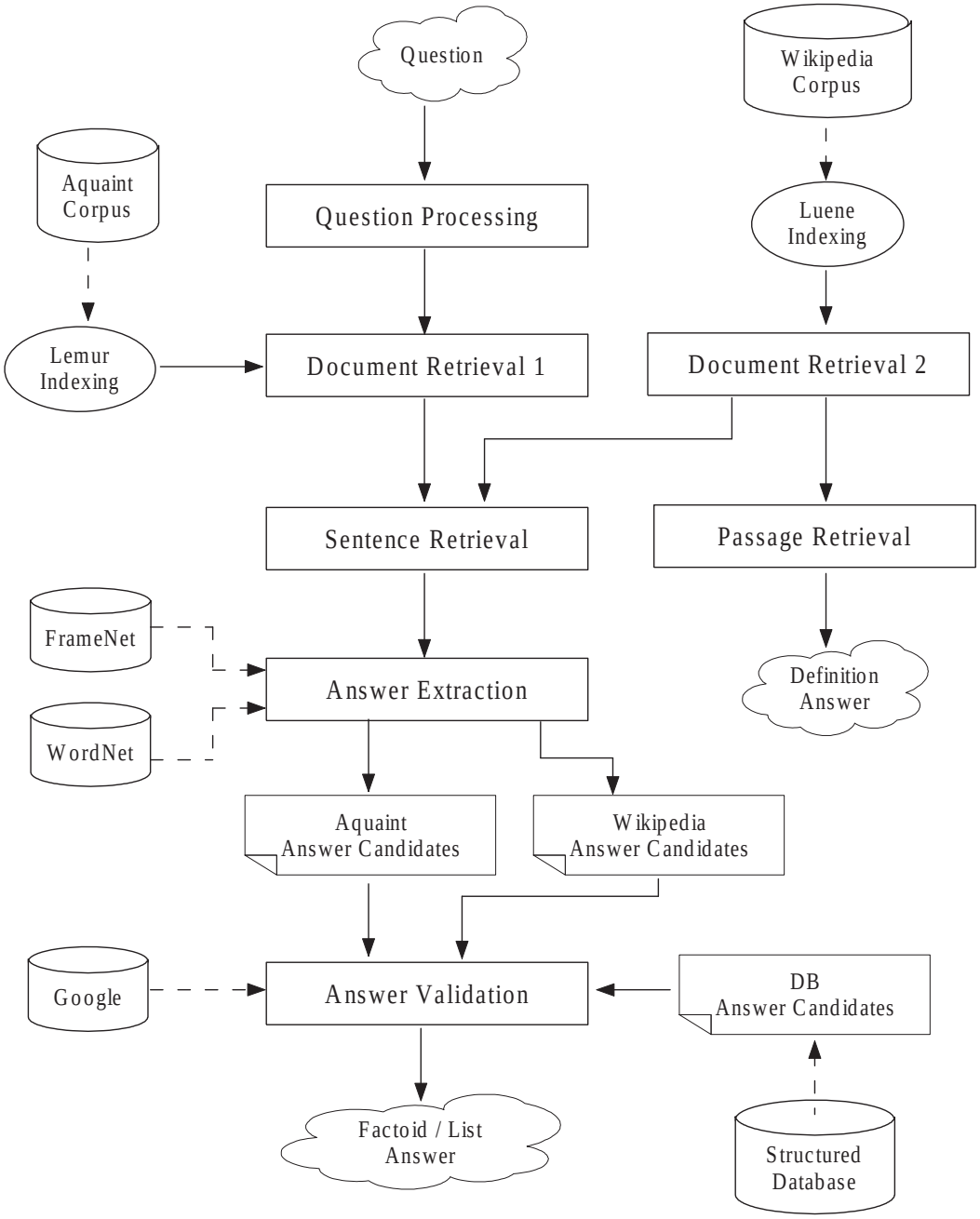


Figure 3.1: Architecture of the Alyssa Question Answering System

Retrieval Module on the Aquaint and Wikipedia indexes. An optional co-reference step allows pronouns or NPs to be replaced by their antecedents. The Sentence Retrieval Module is implemented based on language modeling techniques, and extracted relevant sentences undergo further linguistic analysis in the Sentence Annotation Module before being fed into the Answer Extraction Module. The Answer Extraction Module collects various features from the contexts of answer candidates and integrates them in a supervised machine learning model. Finally, the answer candidates are validated in the Answer Validation Module by further incorporating evidence from Web and structured Database.

As an experimental platform, most of the modules are still under construction. The main functions and methods of the current version are introduced as follows.

3.1 Question Processing Module

Question Processing Module provides a series of question analysis, including question preprocessing, expected answer type identification, key phrase extraction and extension, surface pattern matching, syntactic and semantic structure generation.

3.1.1 Question Preprocessing

After question tokenization, we apply a simple anaphora resolution strategy to relate the question target to one of the question key words. The strategy is conducted in the following three steps:

1. The pronouns of the question are replaced with the question target.

2. For each noun phrase in the question, if it has the same head word as the target and the shorter length than the target, it will be replaced with the target.

3. If the question hasn't contained the target after the previous steps, a noun phrase in the question is chosen heuristically and replaced with the target.

ABBREVIATION
abbreivation, expression abbreviated

ENTITY
animal, body, color, creative, currency, diseases, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word

DESCRIPTION
definition , description, manner, reason

HUMAN
group, individual, title, description

LOCATION
city, country, mountain, other, state

NUMERIC
code, count, date, distance, money, order, other, period, percent, speed, temperature, size, weight

Figure 3.2: Expected Answer Type Taxonomy

3.1.2 Expected Answer Type Identification

The identification of expected answer's semantic categories is crucial for narrowing down answer searching space. For example, for the question "*Which terrorist organization claimed responsibility for the massacre?*", once we know the answer is an "*ORGANIZATION*" name, we may get the answer from the smaller candidate set which only contains organization names. There is no doubt that the correct identification of expected answer types will greatly ease the work of further modules. We use the answer type taxonomy as proposed in (Li and Roth, 2002). It includes 6 coarse and 50 fine grained semantic classes, as shown in Figure 3.2.

Based on the taxonomy, the expected answer type (EAT) identification is formalized to a classification task by using a Bayes classifier with language models. In terms

of classification paradigm, we start with the Bayes classifier:

$$c^* = argmax_c P(Q|c)P(c)$$

The Bayes classifier is known to produce the minimum number of misclassifications if the correct probabilities are known (Q is the question and c is the question type). The probability $P(Q|c)$ can easily be calculated using a language model (LM) trained on all questions of the class c. The major advantage of the language modeling approach is that we can draw on a vast amount of available techniques to estimate and smooth probabilities even if there is very little training data available. We use the 5500 questions provided by the Cognitive Computing Group as University of Illinois at Urbana Champaign [1] for training. On average, there are only about 100 training questions per question type. Specifically, we evaluate absolute discounting, linear interpolation and Dirichlet prior as smoothing techniques. It turns out that the absolute discounting in a variant known as Kneser-Ney smoothing for bigram language models achieves the best results.

The prior $P(c)$ is considered as a unigram language model as well. However, as all classes are seen sufficiently often (that is at least 4 times), there is no smoothing issue at all and relative frequencies are used.

The experiments on TREC 10 data set show that the approach achieve the performance of 80.8% accuracy which outperform all of the systems in current literatures, such as Naive Bayes (67.8%), Neural Network (68.8%), SNoW (75.8%), Decision Tree (77.0%) and SVM (80.2%). Andreas Merkel (Merkel and Klakow, 2007b; Merkel and Klakow, 2007c) gives the detailed information about the approach.

3.1.3 Key Phrase Extraction and Extension

Key phrases (verbs and noun phrases) are extracted from questions by searching in chunk trees. We apply Abney's Chunker (Abney, 1989) to generate the chunk trees of questions.

[1] The training questions of Expected Answer Type Identification is available in `http://l2r.cs.uiuc.edu/\~cogcomp/Data/QA/QC/`

Individual words in key phrases are further semantically expanded by using external resources, such as WordNet and Wikipedia. The word expansion somehow fills in lexical gaps between questions and answer sentences. Moreover, it is the basis of the tolerant question phrase mapping algorithm which will be discussed in Section 3.5.1. Common words and proper words are treated differently in view of the expansion. For the common words, we consider their morphological, format and semantic variations respectively.

Morphological Variation It indicates the inflections of nouns/verbs and expands a word using the words which share the same lemma. For example, for the question "*How many Olympic gold medals did Carl Lewis win?*", we expand the verb "*win*" with its nominal "*winner*", which leads to the mapping from the question verb "*win*" to the word "*winner*" in the answer sentence "*Carl Lewis, winner of nine Olympic gold medals, thinks that* "; For the question "*Where do Rhodes scholars study?*", we expand "*scholar*" with "*scholarships*", which leads to the mapping from the noun phrase "*Rhodes scholars*" to the phrase "*Rhodes scholarships*" in the answer sentence "*Rhodes scholarships provide two or three years study at University of Oxford in England.*". The morphological alternations are found based on a stemming algorithm and "derivationally related forms" in WordNet (Miller, 1990).

Format Variation It copes with special characters, such as "-", "_", "&". For example, "*Ice-T*" is expanded as "*Ice T*" and "*IceT*"; "*Abercrombie & Fitch*" is expanded as "*Abercrombie and Fitch*" and "*Abercrombie Fitch*".

Semantic Variation It considers the synonyms of words. Some types of semantic relations, such as hypernym, hyponym and entailment, enable the retrieval of synonyms. For example, for the question "*Who invented the electric guitar?*", we expand the verb "*invent*" using its direct hypernym "*create*". We search these semantic relations in WordNet and eXtended WordNet by using the same semantic path finding algorithm as (Moldovan and Novischi, 2002). For efficiency, the

search depth is set to 2 in our system.

For the proper words, such as Person and Organization names, we expand them using their alternative names such as full names or abbreviations. This information is collected from the link redirection information of Wikipedia articles. For example, the titles "*United States*", "*US*", "*U.S.*" and "*USA*" in Wikipedia actually refer to one article. We can find the reference information in the "Redirection:" sessions of Wikipedia article XML files [2]. Finally, we build a dictionary for the entity name expansion by collecting and processing all of the redirection information in the whole Wikipedia site. Let show an example of the expansion: for the question "*When did Jack Welch become chairman of General Electric?*", we expand "*General Electric*" with "*GE*", which makes it possible to extract the proper answer "*April 1981*" from the sentence "*Welch became GE's chief executive in April 1981, so the date will mark his 20th anniversary.*". Figure 3.3 shows another example of the question word expansion, where, the common word "*headquartered*" is expanded with its morphological variations "*v. headquarter*" and "*n. headquarters*" and the proper name "*IMF*" is expanded with its full name "*International Monetary Fund*".

3.1.4 Surface Pattern Matching

Considering that there are questions with very high frequency to be asked in TREC, we build question patterns to map high frequent questions to classes and extract answers for the questions using answer patterns. In this section, I will briefly discuss question pattern matching and answer pattern matching respectively.

Different from (Kaisser and Becker, 2004) and (Wu et al., 2005), question classes are defined in terms of semantic meaning but not syntactic structures. A question pattern consists of three elements:

[2]The Wikipedia articles are available to download from `http://en.wikipedia.org/wiki/Wikipedia:Database_download`

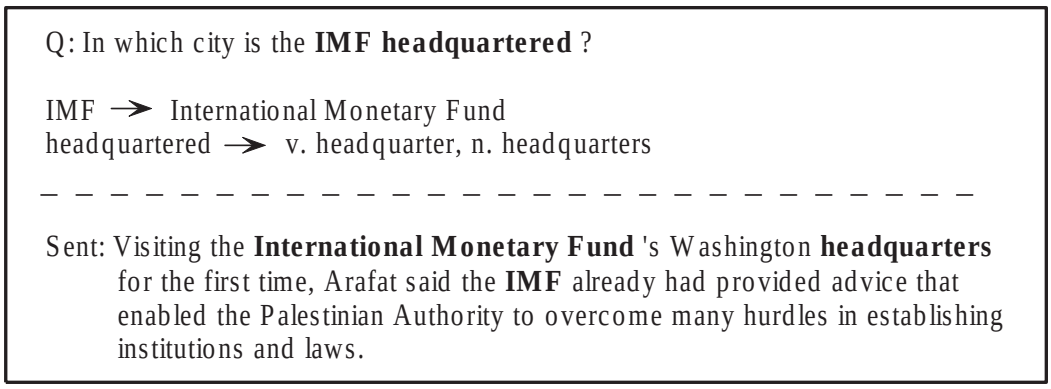


Figure 3.3: An example of question word expansion

- **BFORM** is the basic chunk sequence form of the question pattern;
- **CONS** contains a set of constraints which question pattern matching is required to subject to;
- **SLOTS** is session which connects question pattern matching to answer pattern matching. It contains a set of slot assignment. Each slot assignment records which question key chunks will be used to fill in the corresponding slots of answer patterns.

Figure 3.4 shows a set of question patterns for the question class "*SYNONYM*". Given a question, it is passed to all question classes one by one. For each class, the question is compared with each question pattern (QPTN) in the class. Firstly, the question chunk sequence is matched to the basic form (BFORM) of the pattern. If it matches, we further check whether the constraints (CONS) of the pattern are satisfied by the key chunks of the question. Once all constraints are satisfied, the question is matched to the pattern and classified into the corresponding class. Furthermore, the slot session (SLOTS) of the pattern records which key chunks of the question will be used to fill in the corresponding slots of answer patterns. For example, given a question "*What does AARP stand for?*",

the question chunk sequence "*what do np0 vb0?*" is matched to the basic form "*what do np0 vb0*" of the pattern in Figure 3.4. Moreover, the question satisfies the constraint that the key chunk "*vb0*" belongs to one of the phrases "*mean|(translate to)|(refer to)|(stand for)*" defined in the variable session (VARS). As a result, we match the question to the question pattern and classify it into the "*SYNONYM*" class.

In the current version, 31 question classes, as shown in Figure 3.5 are considered in the system. We manually build a set fo question patterns for each class. In TREC 2006, 92 among total 403 factoid questions are matched to one of the question classes.

Besides question patterns, we also build answer patterns (APTN) to extract answers for the questions which belong to one of the predefined question classes. An answer pattern indicates expected answer position in surface sentences. Answer patterns are represented as regular expressions over tokens, containing three variables:

- **slot** is bound to the key chunks of questions. A question chunk, expected by certain slots, is assigned in the slot session of the corresponding question patterns. For example, in the second question pattern of Figure 3.4, "*slot0*" expects the question key chunk "*np1*" while in the third question pattern, "*slot0*" expects the question key chunk "*np0*".

- **var** is a set of special alternative words, which are usually shared by various patterns and also used in the question patterns. For instance, in Figure 3.4, "*var0*" is set the value as "*name|nickname|alternate|abbreviation|acronym|expansion*".

- **ANSWER** indicates expected answer.

Figure 3.6 shows a set of answer patterns for the question class "*SYNONYM*". Patterns are manually authored for the system. However, TREC 2006 results show that the coverage is not satisfactory since only 12 questions can be correctly answered by the surface pattern matching. The results motivate us to explore deeper linguistic analysis and incorporate more external resources for Answer Extraction.

```
<CLASS key="SYNONYM">
<QPTN_SET>
    <QPTN>
        <BFORM>(who|what) be np0 \?</BFORM>
        <CONS>
        </CONS>
        <SLOTS>
            <SLOT key="slot0">np0</SLOT>
        </SLOTS>
    </QPTN>
    <QPTN>
        <BFORM>what be np0 (of|for) np1 \?</BFORM>
        <CONS>
            <CON key="np0">var0</CON>
        </CONS>
        <SLOTS>
            <SLOT key="slot0">np1</SLOT>
        </SLOTS>
    </QPTN>
    <QPTN>
        <BFORM>what be np0 's np1 \?</BFORM>
        <CONS>
            <CON key="np1">var0</CON>
        </CONS>
        <SLOTS>
            <SLOT key="slot0">np0</SLOT>
        </SLOTS>
    </QPTN>
    <QPTN>
        <BFORM>what do np0 vb0 \?</BFORM>
        <CONS>
            <CON key="vb0">var1</CON>
        </CONS>
        <SLOTS>
            <SLOT key="slot0">np0</SLOT>
        </SLOTS>
    </QPTN>
    <QPTN>
        <BFORM>what be np0 vb0</BFORM>
        <CONS>
            <CON key="vb0">var2</CON>
        </CONS>
        <SLOTS>
            <SLOT key="slot0">np0</SLOT>
        </SLOTS>
    </QPTN>
<QPTN_SET>
<VARS>
    <VAR key="var0">name|nickname|alternate|abbreviation|acronym|expansion</VAR>
    <VAR key="var1">mean|(translate to)|(refer to)|(stand for)</VAR>
    <VAR key="var2">call|name</VAR>
</VARS>
</CLASS>
```

Figure 3.4: Example of question patterns for ”*SYNONYM*” class.

Question Class	Example
WHO_CREATE	Who discovered prions?
WHAT_CREATE	What did Edward Binney and Howard Smith invent in 1903?
WHEN_CREATE	When was the International Criminal Court established?
WHERE_CREATE	When was the Black Panthers organization founded?
WHAT_BE_ORG_OF_PERSON	What record company is Fred Durst with?
WHO_BE_PRESIDENT_OF_ORG	Who is AARP's top official or CEO?
WHO_BE_MEMBER_OF_ORG	Who are the members of Insane Clown Posse?
WHEN_BORN	When was James Dean born?
WHERE_BORN	Where was James Dean born?
WHEN_DIE	When did Franz Kafka die?
WHERE_DIE	Where did Franz Kafka die?
HOW_DIE	What did James Dean die of?
HOW_OLD_DIE	How old was Jean Harlow when Jean Harlow died?
WHERE_BURY	Where is Jean Harlow buried?
NATIONALITY	What is minstrel Al Jolson's nationality?
OCCUPATION	What was Gordon Gekko's profession?
WHO_MARRY	Who is Tom Cruise married to?
WHO_FATHER	Who was Horus father?
WHO_MOTHER	Who was Horus mother?
WHERE_LIVE	Where does Jennifer Capriati live?
WHERE_ORG_LOCATE	Where is AARP's headquarters?
SYNONYM	What does AARP stand for?
PRODUCT	What kind of business is Abercrombie & Fitch?
WHO_BE_IN_EVENT	Who was the on-board commander of the submarine Kursk?
WHEN_EVENT_HAPPEN	When was the first Crip gang started?
WHEN_EVENT	What year was Alaska purchased?
WHERE_EVENT_HAPPEN	In what country did the game of croquet originate?
WHERE_EVENT	Where was the Miss Universe 2000 contest held?
PRIZE	What prizes or awards has Frank Gehry won?
HOW_MANY_MEMBER	How many seats are in the cabin of a Concorde?
SPECIFICATION	What color are UPS trucks?

Figure 3.5: Question classes and examples

```
<CLASS key="SYNONYM">
<APTN_SET>
    <APTN>slot0( ,)?( who|which)? be born ANSWER , </APTN>
    <APTN>slot0( ,)?( who|which)? be( \\w+)? (call|know as) ANSWER , </APTN>
    <APTN>slot0 (, whose|'s)( \\w+|,|\(|\)){0,5} (var0) be ANSWER</APTN>
    <APTN>slot0 (be|,) (var0) (of|for) ANSWER</APTN>
    <APTN>slot0 \(( born)? ANSWER \)</APTN>
    <APTN>slot0 \[( born)? ANSWER \]</APTN>
    <APTN>slot0 ,( \\w+|,|\(|\)){0,5} know as ANSWER</APTN>
    <APTN>change name from ANSWER to( \\w+|,|\(|\)){0,5} slot0</APTN>
    <APTN>ANSWER( ,)?( who|which)? be born slot0 , </APTN>
    <APTN>ANSWER( ,)?( who|which)? be( \\w+)? (call|know as) slot0 , </APTN>
    <APTN>ANSWER (, whose|'s)( \\w+|,|\(|\)){0,5} (var0) be slot0</APTN>
    <APTN>ANSWER (be|,) (var0) (of|for) slot0</APTN>
    <APTN>ANSWER \(( born)? slot0 \)</APTN>
    <APTN>ANSWER \[( born)? slot0 \]</APTN>
    <APTN>ANSWER ,( \\w+|,|\(|\)){0,5} know as slot0</APTN>
    <APTN>change name from slot0 to( \\w+|,|\(|\)){0,5} ANSWER</APTN>
</APTN_SET>
</CLASS>
```

Figure 3.6: Example of answer patterns for ”*SYNONYM*” class.

3.1.5 Syntactic and Semantic Structure Generation

We generate syntactic structures with MiniPar (Lin, 1994) and represent a syntactic structure as a set of dependency relation paths. We further generate FrameNet-style semantic structures with the model proposed in Chapter 5 and represent a semantic structure as a bipartite graph. Question syntactic and semantic structures will be further used in Answer Extraction model by matching the corresponding structures of answer candidates. They are the focus of the thesis, therefore, I will discuss in detailed in the Chapter 4 and 5 respectively.

3.2 Document Retrieval Module

The Lemur Toolkit for Language Modeling and Information Retrieval [3] is used for document retrieval. Queries as well as the AQUAINT Corpus are stemmed with Porter stemmer and no stop-word removal is done. As mentioned above, we choose a language modeling-based approach for the retrieval step using unigram distributions. The smoothing method we use is Bayesian smoothing with Dirichlet priors. As shown in (Hussain, Merkel, and Klakow, 2006), the smoothing with Dirichlet prior performs best in context of document retrieval experiments and even outperforms traditional information retrieval techniques like Okapi and TFIDF. (Hussain, Merkel, and Klakow, 2006) also suggests an optimal smoothing parameter for TREC question sets which we used in the Alyssa system as well. After the retrieval step, we fetch the best N relevant documents and send them to Sentence Retrieval Module. Currently, we set $N = 60$ because the number is most sufficient in previous TREC runs to get about 90% of answers within the relevant documents.

[3] The Lemur Toolkit is available to download in `http://www.lemurproject.org/`

3.3 Sentence Retrieval Module

Before starting sentence retrieval, we firstly run a sentence boundary detection to identify possible ends of sentences. Next, the sentences as well as the queries are stemmed using Porter stemmer. Parallel to the stemming process, we expand the queries and the sentences. If the expected answer type of a query is DATE then the token "DATE" is added at the end of the query. The sentences are prepared in almost the same manner. Here, patterns are used to identify possible occurrences of time and date information. Due to the expansion, possible answer candidates for the expected answer type "DATE" are ranked higher. To get an optimal score for these kinds of queries we introduce a weighting scheme and experimentally determine the specific weight.

Again an unigram language modeling based technique is used to rank the sentences. In detail we use Bayesian smoothing with Dirichlet prior which is given by the following formula:

$$p_\mu(w|d) = \frac{c(w;d) + \mu p(w|C)}{\sum_w c(w;d) + \mu}$$

Where $c(w;d)$ means the count of the word w in the sentence d, C is the collection of sentences and μ is the smoothing parameter.

We choose this kind of smoothing because it has already performed promising for document retrieval. As the smoothing parameter, we choose the interpolation weight $\mu = 100$ by searching in complete parameter space. Experiments show that this method actually performs better than Jelinek-Mercer linear interpolation and absolute discounting.

In addition to the unigram language modeling, we try other language modeling methods for the sentence ranking task. We use the LSVLM toolkit, which is the Language Modeling toolkit from LSV [4] and implements standard language modeling techniques. We decide to use this particular toolkit in the Sentence Retrieval Module rather

[4] Lehrstuhl fuer Sprachsignalverarbeitung

than the Lemur Toolkit due to the flexibility. It is easy to switch between various language models for interpolation or to manipulate vocabulary. In our case, we close the vocabulary over the queries to get the better performance as described in (Merkel and Klakow, 2007a).

Another preparation step is to include a dynamical list of stop-words. The list consists of four most commonly used terms of the complete sentence collection. However, these words are not removed but just get a smaller score. Again a weighting scheme is used to optimally score the stop-words.

Finally, query words, such as "*what*", "*when*", "*who*", are removed. In most cases the query words have no meaning in terms of searching for relevant sentences so removing them gives the higher score for the rest of possible relevant query words.

3.4 Sentence Annotation Module

Sentence Annotation Module conducts a series of linguistic analysis on the retrieved sentences, such as named entity recognition, noun phrase chunking and dependency parsing. As discussed in Section 3.1.2, the Question Processing Module identifies the expected answer types of questions. The answer type taxonomy includes 6 coarse and 50 fine grained semantic classes. Therefore, the Sentence Annotation Module is required to recognize the 50 types of named entities in sentences. We apply different strategies to cope with different named entity types.

For **HUMAN** and **LOCATION** classes, we use Lingpipe [5], a public available named entity recognizer, to identify *PERSON*, *ORGANIZATION* and *LOCATION* names. Since Lingpipe doesn't intend to distinguish the sub-types of *LOCATION*, we use a trigger-word-based method to judge the sub-types. For example,

- The sub-type "*MOUNTAIN*" is triggered by the words "*peak*", "*mountain*" and

[5]The Lingpipe software is available to download in `http://www.alias-i.com/lingpipe`

"*mount*";

- The sub-type "*OCEAN*" is triggered by the words "*gulf*", "*sea*", "*bay*" and "*ocean*";
- The sub-type "*LAKE*" is triggered by the words "*reservoir*", "*lough*", "*lake*" and "*dam*";

For **NUMERIC** names, a rule-based method is firstly used to identify numeric expressions in texts. Next, the numeric expressions are classified into the sub-types of the *NUMERIC* according to their units. For example,

- The units "*meter*", "*mile*" and "*lightyear*" represent the sub-type "*DISTANCE*";
- The units "*kelvin*" and "*celsius*" represent the sub-type "*TEMPERATURE*";
- The units "*ounce*" and "*kg*" and "*ton*" represent the sub-type "*WEIGHT*";

Analogously, the rule-based method is also used to recognize *DATE* and *ABBREVIATION* names.

For **ENTITY** names, two strategies are employed:

- Since some types, such as "*COLOR*", "*LANGUAGE*", "*CURRENCY*" and "*RELIGION*", have relatively limited size of entities, we create the complete entity lists of the types from Wikipedia.
- Some types, such as "*DISEASES*", "*CREATIVE*", "*FOOD*", "*TECHNIQUE*" and "*EVENT*" are more difficult to recognize since they neither have limited size of entities nor have obvious rules to capture. They might be recognized by the support of trigger word lists or comprehensive knowledge databases. We haven't handled them in the current version and put them in our ongoing work schedule.

Moreover, our system is disable to recognize **DESCRIPTION** names as well.

S: Black quarterbacks who succeeded in a pro-style passing offense, like the University of Washington's Warren Moon, were asked to switch positions .

Chunking Results:

[BNP Black quarterbacks] who [VB succeeded] in [BNP a pro-style passing offense] , like [NP [BNP_ORG the University of Washington] 's [BNP_PER Warren Moon]] , were asked to [VB switch] [BNP positions] .

Parsing Results:

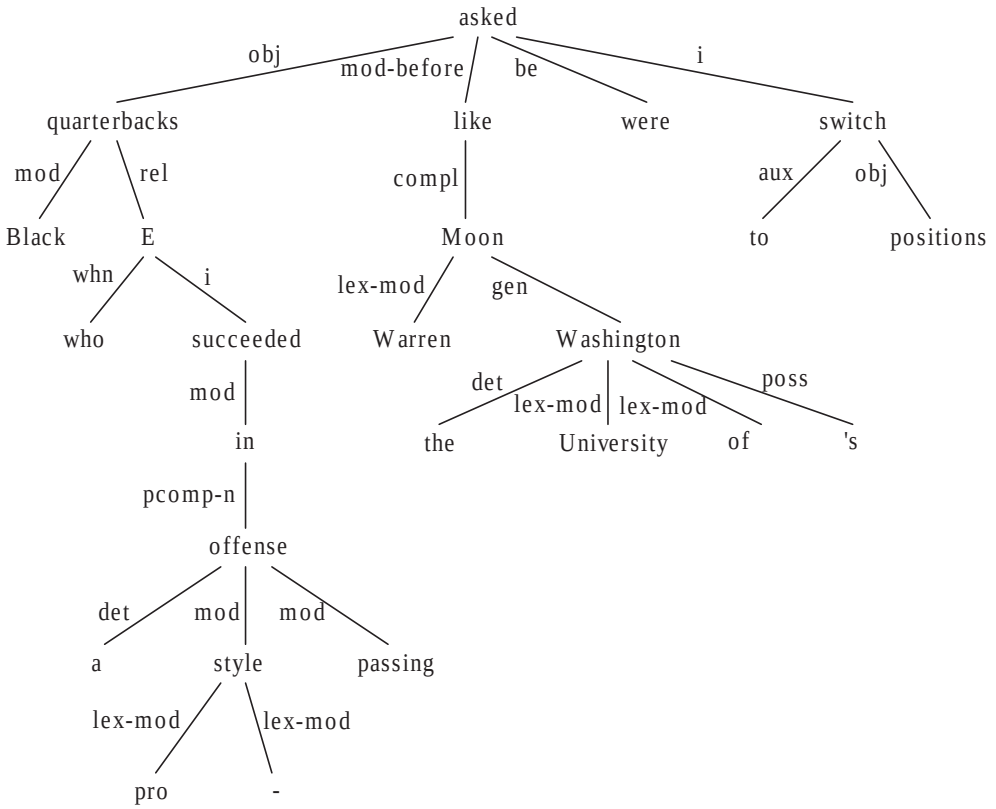


Figure 3.7: Example of named entity recognition, chunking and parsing results of a raw sentence.

After recognizing named entities in sentences, we further chunk the noun phrases of the sentences with Abney's chunker (Abney, 1989) and parse the sentences with MiniPar (Lin, 1994), a fast and robust parser for grammatical dependency relations.

Figure 3.7 shows an example of named entity recognition, chunking and parsing results of a raw sentence. In the named entity recognition result, "*the University of Washington*" is recognized as "*ORGANIZATION*" name and "*Warren Moon*" is recognized as "*PERSON*" name. In the chunking result, the noun phrases and verbs are annotated with square brackets. The tags in the square brackets are explained as follows:

- **NP** is noun phrase;
- **BNP** is base noun phrase. It is defined as the smallest noun phrase in which no other noun phrases are embedded.
- **VB** is verb or verb phrase.

In addition, the parsing result is represented as a tree structure. Each node represents a word and the edge between two nodes indicates the dependency relation between the nodes, such as "*example*" is surface subject ("*example*") of the word "*example*". Totally, there are 42 dependency relations predefined in the Minipar.

3.5 Answer Extraction Module

Answer Extraction Module works on annotated sentences to pinpoint answers by using more linguistic-motivated analysis. Since QA turns to find exact answers rather than text snippets in recent years, answer extraction becomes more and more crucial.

Considering the goal of the Alyssa system is to handle TREC-style factoid questions, we don't spend much efforts to deal with the following question types:

- Complicated questions which consist of multiple sentences;

- Unfactoid questions. The questions expect answers to be verb phrases, such as the answer of the question "*What is the mission of International Finance Corporation?*" is "*promote private sector investment in developing countries*" , or sentences, such as the answer of the question "*How did Cincinnati get its name?*" is "*The society was named for Cincinnatus, a Roman patriot who returned to his farm after saving his city in battle, and this city, in turn, was named for the society.*", or paragraphs, such as the question "*What are the opening words of the Declaration of Independence?*".

On the summary, the question is required to be only one sentence and the corresponding answer is supposed to be a noun phrase rather than the other granularities of texts. Considering noun phrases (NP) as well as verbs (VB) are the most informative elements of sentences, we define these phrases as the basic units to analyze.

- When mapping question words to relevant sentences, we conduct phrase-based mapping rather than word-based mapping;
- Dependency relations between two phrases are analyzed, such as $< NP_1, NP_2 >$ and $< NP, VB >$. However, dependency relations within a noun phrase are not considered, such as the mod relation between an adjective word and its head noun.
- All of the noun phrases in sentences are regarded as answer candidates.

3.5.1 Question Phrase Mapping

Most of Sentence Retrieval and Answer Extraction Modules follow the assumption that a question has word-level or phrase-level overlapping as large as possible with its relevant sentences. The hypothesis supervises the Alyssa system to judge how relevant of sentences to answer a question (Sentence Retrieval Module) and which part of the relevant sentences may exactly answer the question (Answer Extraction Module). Therefore, before we conduct answer extraction, we detect which is the word or phrase overlapping

between a question and its relevant sentences. We call this process as Question Phrase Mapping which is to map the noun phrases and verbs in a question to relevant sentences. During the mapping, we consider morphological, format, semantic and proper name variations between individual words. These variations are detailedly discussed in Section 3.1.3. On the basis of individual word mapping, we propose two methods for phrase mapping: Weighted Edit Distance Method and Approximate Phrase Mapping Method.

Weighted Edit Distance Method Weighted Edit Distance Method is to find the similarity between two phrases by computing the minimal cost of operations needed to transform one phrase into the other, where, an operation is an insertion, deletion, or substitution action. Different from commonly-used edit distance algorithm (Levenshtein, 1965), the weighted edit distance method defines the more flexible cost function which incorporates the variations of individual words.

According to the observation of the task, we set the substitution costs of the variations as follows:

1. Identical words have cost 0;
2. Words with the same morphological root have cost 0.2;
3. Words with the hypernym or hyponym relations have cost 0.4;
4. Words in the same SynSet have cost 0.6;
5. Words with subsequence relations have cost 0.8;
6. otherwise, words have cost 1.

Approximate Phrase Mapping Method Approximate Phrase Mapping Method separates a noun phrase into a set of heads $H = \{h_1, ..., h_i\}$ and a set of modifiers $M = \{m_1, ...m_j\}$. The following heuristic rules are applied to judge heads and modifiers:

- If a noun phrase is a named entity, all words are heads.
- The last word of a noun phrase is head.
- The Rest words are modifiers.

The similarity between two noun phrases $Sim(NP_q, NP_s)$ is defined as the linear interpolation of head similarity and modifier similarity.

$$Sim(NP_q, NP_s) = \lambda Sim(H_q, H_s) + (1 - \lambda)Sim(M_q, M_s)$$

$$Sim(H_q, H_s) = \frac{\sum_{h_i \in H_q} \sum_{h_j \in H_s} Sim(h_i, h_j)}{|H_q \bigcup H_s|}$$

$$Sim(M_q, M_s) = \frac{\sum_{m_i \in M_q} \sum_{m_j \in M_s} Sim(m_i, m_j)}{|M_q \bigcup M_s|}$$

Furthermore, the similarity between two heads $Sim(h_i, h_j)$ are defined as:

- $Sim = 1$, if $h_i = h_j$ on morphological variations;
- $Sim = 1$, if $h_i = h_j$ on format variations;
- $Sim = SemSim(h_i, h_j)$

Particularly, the similarity between two modifiers $Sim(m_i, m_j)$ only incorporates morphological and format variations; the similarity between two heads of named entities $Sim(h_i^{ne}, h_j^{ne})$ incorporates morphological, format and proper name variations, such as the name "*Sacajawea*" has the proper name variation "*Sacagawea*". Moreover, verb similarity measure $Sim(v_1, v_2)$ is the same as the head similarity measure $Sim(h_i, h_j)$.

3.5.2 Answer Candidate Ranking

After mapping question key phrases to answer sentences, we conduct answer extraction on the mapped relevant sentences. We regard answer extraction as an answer candidate

ranking task. Rich evidence, including orthographics, syntactics and semantics are captured around answer candidates and further incorporated into a Maximum Entropy-based ranking model. Finally, a list of top-ranked answer candidates is returned. This part is the focus of the thesis and will be detailedly discussed in Chapter 4 5 and 6

3.6 Answer Validation Module

Answer Validation Module validates extracted answers using additional resources, such as knowledge databases and Web data. The Alyssa system conducts two kinds of validation: Knowledge-based Validation and Web-based Validation.

3.6.1 Knowledge-based Validation

The motivation to add a knowledge-based validation to the existent system is two-fold:

- We observe that there are certain types of recurring questions which cannot be answered by the existent Answer Extraction Module. Many errors can be ascribed to the inability of the named-entity tagging to recognize some prominent name types, such as "*MOVIE*", "*BOOK*" and "*SONG*" names. It results in the absence of correct answer candidate from the Answer Extraction Module. The knowledge-based validation is expected to supply additional answer candidates beyond the Answer Extraction Module.

- Answers extracted from structured knowledge bases might enable to pick correct answers from a ranked list of answer candidates generated by the Answer Extraction Module which - due to its statistic nature - might not be optimal ranking. Hopefully, the knowledge-based validation may amend the ranking of the existent Answer Extraction Module.

The design of the knowledge-based validation is described below.

Firing Question Patterns Each question is matched against a set of manually defined firing patterns. Each firing pattern is associated with some simple binary/ternary relation, such as "*x-isMovieStarringActor-y*", where all arguments except one are already instantiated - the missing argument is to be retrieved from knowledge bases, e.g. "*x-isMovieStarringActor-y* {*x=?,y=Christopher Reeves*}". For establishing these relations we only use lexical information and named-entity tagging which have already been processed for each question during question processing. Only those questions for which a pattern is fired are processed further. The remaining questions are exempt from the validation.

Searching in Knowledge Bases The incomplete relation is translated to a query to knowledge bases. Currently we use the following resources:

- **International Movie Database (IMDB)**: The knowledge base is used not only for the questions asking for movie and television appearances of actors/actresses but also as a source of general biographic data of virtually any famous person, such as "*place/date of birth*", "*cause of death*", "*real/full name*", "*marital status*", etc.
- **Discogs.com**: We use this database for retrieving discographies of musical artists.
- **CIA World Fact Book**: This popular resource contains various factoid information of the states in the world.

Re-Ranking Answers We conduct the follow three re-ranking strategies according to various search results.

- If an answer fails to be retrieved from any knowledge bases, we keep using the original ranked-answer candidate list.
- If an answer is returned by the knowledge base searching but also is found in the original answer candidate list, the answer will be moved up to the first position of

the list.

- If an answer is returned by the knowledge base searching but is not found in the original list, the extra answer will be inserted into the list at the first position. Furthermore, a back-projection schema is carried out to find a document support the answer in document collection.

We evaluate the knowledge-based validation on TREC 2004, 2005 and 2006 factoid and list questions. Fortunately, we manage to achieve improved performance on all of the TREC years.

3.6.2 Web-based Validation

(Breck et al., 2001; Clarke, Cormack, and Lynam, 2001) state that answer redundancy in an enormous collection of unstructured, flat texts has a strong correlation with answer correctness. However, the redundancy evidence can't be successfully obtained from Aquaint Corpus due to its limited size. Answers are only returned from one document in the Corpus. The web-based validation is mainly motivated to complement the Answer Extraction Module by further incorporating the redundancy evidence of ranked answer candidates. In addition, another motivation is that the single instance of an answer candidate may not provide sufficient justification since it might be only locally correct in a document. But the multiple occurrence of an answer candidate in various documents indicates the credibility of the answer to be correct.

The vast amount of information available on the Web makes it an attractive resource for Answer Validation. The core idea is that the volume of available web data is large enough to supply proper answers multiple times and in multiple contexts varying from complicated and implicit contexts where sophisticated natural language processing techniques are required to simple and explicit contexts where only surface pattern matching methods may work well.

We develop a web-based validation component to further validate top-ranked answer candidates returned by the Answer Extraction Module. It uses the frequency of the candidates within the Web data to boost most likely answers. We access the Web data by using Google Search Engine. Various queries ranging from loose to tight are generated. For example, for the question, "*Where is Merrill Lynch headquartered?*", the following queries including Bag-of-Word query, Noun-Phrase-Chunk query and Declarative-Form query are constructed:

- **Bag-of-Word**: Merrill Lynch headquartered
- **Noun-Phrase-Chunk**: "Merrill Lynch" headquartered
- **Declarative-Form**: "Merrill Lynch is headquartered"

The declarative forms of questions are heuristically constructed. We manually develop about 20 patterns for the transformation.

For the first two queries, the top 3 snippets Google returned are the following:

- **Merrill Lynch** rose to prominence on the strength of its brokerage network ... Florida U.S., Global Private Client, Regional **Headquarters** for Latin American ...
- With their help, he also arranged to host 150 teachers and administrators at **Merrill Lynch headquarters** for an allday staff meeting.
- Access **Merrill Lynch headquarter**'s address and subsidiary locations, along with insight related to Merrill Lynch executives, competitors, and operations.

For the third query, Google returns the snippets below:

- Today's **Merrill Lynch is headquartered** in lower Manhattan, not far from its original location on Wall Street.

- **Merrill Lynch is headquartered** in New York.
- **Merrill Lynch is headquartered** in the World Financial Center.

It is obvious that the Declarative-Form query (tightest) leads to the most explicit answers than the Bag-of-Word and Noun-Phrase-Chunk queries. On the other hand, it is also more probable to fail to get snippets from Google. Considering answer candidates from tighter queries are more reliable, we assign different weights to the answer candidates evoked by different queries. The weights for the Bag-of-Word, Noun-Phrase-Chunk and Declarative-Form queries are set to 1:2:5. For each query, top 50 snippets are used for validation.

As to the counting of answer candidate's frequency, specially for quantity-seeking questions, such as "*How many bombers were killed in the London terror bombing attacks?*", the frequency of the answer candidate "*4*", is calculated by matching the head noun-jointed phrase "*4 bombers*" in Google snippets. For each question, top 10 answer candidates from Answer Extraction Module are validated.

We evaluate the contribution of the web-based validation on TREC 2006 factoid questions (403 questions). The Answer Extraction module returns ranked answer candidates, where 175 questions are correctly answered considering top 10 answer candidates of each question while only 83 questions are correctly answered considering top 1 answer candidates. We further validate and re-rank the top 10 answer candidates (175 questions should be the upper bound of the web-based validation). Finally, 122 questions are correctly answered by the web-based validation considering top 1 answer candidates. It shows that the web-based validation significantly improves performance by 46.9% based on the Answer Extraction Module.

In the Alyssa System, I mainly contribute to the Question Processing Module, the Sentence Annotation Module, the Answer Extraction Module and the Answer Validation Module (especially to the web-based validation). In this Chapter, I gave a brief introduction about my efforts on these modules respectively. In the following chapters, I will

detailedly present the Answer Extraction Module which is the focus of the thesis. My exploration on the Answer Extraction Module mainly consists of two-fold:

1. How to capture syntactic and semantic features?

2. How to make prediction according to these features?

Chapter 4

Syntactic Evidence for Answer Extraction

4.1 Motivation

In order to find proper answers, evidence like expected answer type and surface textual pattern is extracted from answer sentences and incorporated in Answer Extraction Module with a pipeline structure, a scoring function or statistical-based methods. However, the evidence extracted only from plain texts might not be sufficient to identify proper answers. For example:

1. For the question "*What are pennies made of?*", expected answer type is unknown;

2. For the question "*Who was the first American in space?*", surface patterns may not detect long distance relations between the question key phrase "*the first American in space*" and the proper answer "*Alan Shepard*" in the sentence "*... that carried Alan Shepard on a 15-minute suborbital flight in 1961, making him the first American in space.*";

3. For the question "*When did the Port Arthur Massacre occur?*", if surface patterns

strongly depend on word ordering, it might fail to find the proper answer "*1996*" in the sentence "*... Australian Prime Minister John Howard moves to enforce the national uniform gun laws forged after the 1996 Port Arthur massacre.*".

To overcome the problems arising from the divergences of lexical representations, such as long surface distance and word ordering alternation, deep linguistic analysis like syntactic analysis, trends to be explored and evidence on complicated data representations is extracted. Syntactic-based method is motivated by the observation that the context of proper answer in answer sentence often has similar syntactic structure with question. For example, in both the question "*What did Alfred Nobel invent?*" and the answer sentence "*... in the will of Swedish industrialist Alfred Nobel, who invented dynamite.*", the phrase "*Alfred Nobel*" is the "*subject*" of the verb "*invent*" and the proper answer "*dynamite*" ("*What*" in the question) is the "*object*" of the verb "*invent*".

Considering how to effectively capture syntactic evidence for the Answer Extraction, we propose two methods: Dependency Relation Pattern Method and Dependency Relation Path Correlation Method. To our best knowledge, the syntactic evidence between proper answers and question key words hasn't been well explored by previous statistical-based Answer Extraction Module. The following Section will introduce some representative work of syntactic-based answer extraction.

4.2 Related Work

In recent TREC evaluation, most of top ranked QA systems explored syntactic evidence in Answer Extraction. We briefly summarize the main usages below.

LCC (Harabagiu et al., 2003) explored syntactic relations, such as subject, object, prepositional attachment and adjectival/adverbial adjuncts, based on logic form transformation which converted plain texts to logic representations. A logic prover further worked on the logic representations to justify answer candidates. The prover achieved

accurate results but suffered from a coverage problem due to insufficient world knowledge/NLP axioms and inference rules. In addition, it took long processing time. Therefore, this method was only used as compensation (like Answer Validation) rather than a replacement of Answer Extraction.

ISI (Echihabi et al., 2003) extracted verb-argument relations, such as "subject-verb" and "verb-object", in answer sentences and compared them with those in questions. IBM's Maximum Entropy-based model (Ittycheriah and Roukos, 2002) integrated a rich feature set, including word co-occurrence scores, named entities and dependency relations. For the dependency relations, they considered a set of predefined relations by partial matching. BBN (Xu et al., 2002) also incorporated verb-argument relations.

However, the above QA systems only focused on certain relation types, such as verb-argument relations, and extracted them using heuristic rules. Therefore, the extraction of such relations was limited in very local contexts of answer nodes, such as parent or sibling nodes, and didn't involve long range dependencies. Furthermore, they concerned the relations only to certain types of question words, such as verbs. Actually, variable types of question words may have different indicative relations with proper answers.

(Kaisser and Becker, 2004) matched questions into predefined patterns, such as the question "*When did Jack Welch retire from GE?*" was matched to the pattern "*When+did+NP+Verb+NP|PP*". For each question pattern, there was a set of syntactic structures (called answer patterns) for potential answers. Candidate answers were ranked by matching the syntactic structures. This method worked well on TREC questions. However, it is costing to manually construct the question and answer patterns.

(Tanev, Kouylekov, and Magnini, 2004; Wu et al., 2005) compared syntactic relations between questions and answer sentences. (Tanev, Kouylekov, and Magnini, 2004) reconstructed a basic syntactic template tree for a question, in which one of the nodes denoted expected answer position. Then, answer candidates were ranked by matching their syntactic trees to the question template tree. Furthermore, the matching was weighted

by lexical variations. (Wu et al., 2005) combined n-gram proximity search and syntactic relation matching. Firstly, candidate answers were filtered by n-gram proximity and only top N candidate answers were kept for syntactic relation matching. Next, question tree and answer candidate tree were matched from node to node.

Although the above systems considered various types of syntactic relations and applied various methods to compare the relations between questions and answer sentences, they shared the common hypothesis that proper answers are more likely to have the same syntactic relations as questions. For example, in the question "*Who founded the Black Panthers organization?*", where, the question word "*who*" has the dependency relation "*subj*" with the verb "*found*" and "*subj obj nn*" with the phrase "*Black Panthers organization*", in the sentence "*Hilliard introduced* ***Bobby Seale****, who co-founded the Black Panther Party here ...*", the proper answer "*Bobby Seale*" has the same relations with most of question phrases. These methods achieved high precision, but poor recall due to syntactic relation variations. One meaning is often represented as different syntactic relation combinations. In the above example, appositive relation frequently appears in answer sentences, such as "*Black Panther Party co-founder* ***Bobby Seale*** *is ordered bound and gagged ...*" and indicates the proper answer "*Bobby Seale*" although it is asked in different way in the question.

(Cui et al., 2004) proposed an approximate dependency relation matching method for both passage retrieval and answer extraction. The similarity between two relations was measured by their co-occurrence rather than exact matching. They stated that their method effectively overcomed the limitation of previous exact matchs exact matching methods. Lastly, they used the sum of similarities of all path pairs to rank candidate answers, wing methods. Lastly, they used the sum of similarities of all path pairs to rank candidate answers, which was on the basis of the assumption that all paths had equal weights. However, it might not be true. For example, in the question "*What book did Rachel Carson write in 1962?*", the phrase "*Rachel Carson*" looked like more important

than "*1962*" since the former was the question topic and the latter was a constraint for the expected answer. In addition, lexical variations were not well considered and a weak relation path alignment algorithm was used in their work.

In this Chapter, we explore more comprehensive syntactic relation-based methods: Dependency Relation Pattern Method and Dependency Relation Path Correlation Method and respectively evaluate the contributions of the mothods in Chapter 7.3 of the thesis.

4.3 Dependency Relation Pattern Method

In this section, we propose a dependency relation pattern method for answer extraction. The method is motivated by the observation that there are predictable/limited-size answer syntactic structures associated with certain question structures. The method aims to detect which syntactic structures frequently occur in questions; for certain question syntactic structure, what kinds of answer syntactic structures are expected. More importantly, how to pinpoint the answer by matching the extracted answer syntactic structures when a unseen question is given.

Syntactic structures are represented as dependency relation pattern (Ptn) in our task. Different from textual patterns, dependency relation patterns capture word relations based on syntactic representations rather than surface texts. Therefore, they may get the deeper understanding of word relations and capture long range dependency between words regardless of their ordering and distance in surface texts.

4.3.1 Dependency Relation Pattern Extraction

A dependency relation pattern $Ptn(w_1, w_2)$ is the smallest dependency subtree which covers the key words w_1 and w_2 in a dependency tree. We represent the pattern as a dependency relation sequence by traversing from the w_1 node to the w_2 node in the tree.

Each relation in the sequence is linked by the symbols indicating upward or downward movements through the tree. For example, in Figure 4.1, the relation sequence from the answer candidate node "*211,456 miles*" to the question word node "*the moon*" is "*pred_U s_D*", to the node "*Earth*" is "*pred_U i_U mod_D pcomp-n_D appo_D mod_D pcomp-n_D*" where "_U" and "_D" indicate upward and downward movements through the tree respectively.

Q1980: How far is [the moon] from [Earth] ?
S: At its perigee , the closest approach to [Earth] , [the moon] is [221,456 miles] away .

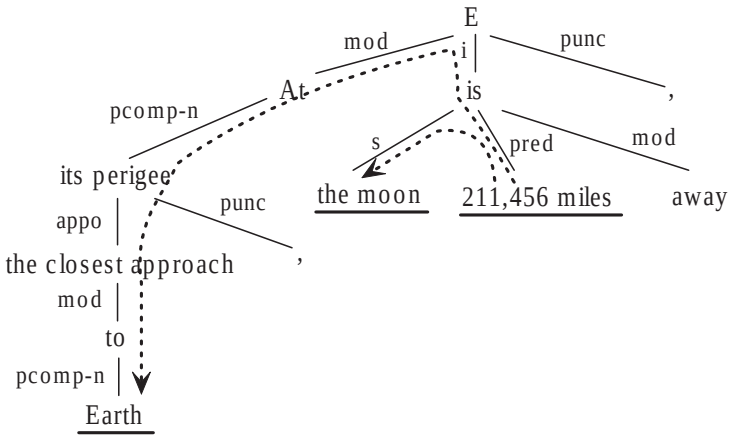


Figure 4.1: Example of dependency relation sequences. The directed paths in dot line are the dependency relation sequences from the answer candidate node "*211,456 miles*" to the question key word nodes "*the moon*" and "*Earth*"

We automatically construct dependency relation patterns from training questions and their corresponding answer sentences respectively. For each question, we extract dependency relation patterns Ptn^q between question word, such as "*who*", "*what*", "*when*" and question key phrases. The question word is replaced with "*EAP*", which indicates expected answer position. Analogously, for each candidate sentence, we extract relation patterns Ptn^a between proper answer and mapped question phrases. As the result of the pattern extraction, we get a set of question patterns $QSet(Ptn_i^q), (i = 1, ..., N)$, where each Ptn_i^q is associated with a set of answer patterns $ASet_i(Ptn_j^a), (j = 1, ..., M_i)$. N is the total number of question patterns and M_i is the number of answer patterns for

the question pattern $DepPtn_i^q$. Some pattern examples are shown in Table 4.1. The following briefly describes our pattern extraction algorithm in training data.

For each question q in training data,

1. Question Processing Module extracts a set of key phrases $\{w_1^q, w_2^q, ..., w_i^q\}$ of q, as described in Section 3.1.3;
2. Regard question word, such as "*who*", "*when*", "*where*", as expected answer position EAP of q;
3. Extract dependency relation pattern $Ptn^q(EAP, w_i^q)$ for each w_i^q. The $Ptn^q(EAP, w_i^q)$ is represented as the relation sequence from the EAP node to the w_i^q node in the dependency tree of q;
4. Add the $Ptn^q(EAP, w_i^q)$ into question pattern set $QSet$;
5. Extract answer patterns for the $Ptn^q(EAP, w_i^q)$ in the following steps:
   For each sentence s in the answer sentence set of q,
   a. Map w_i^q into w_i^s of s ($w_i^q = w_i^s$), as described in Section 3.5.1;
   b. Extract dependency relation pattern $Ptn^a(A, w_i^s)$ covering the proper answer node A and w_i^s node in the dependency tree of s;
   c. Add $Ptn^a(A, w_i^s)$ into the answer pattern set $ASet$ of the $Ptn^q(EAP, w_i^q)$.

4.3.2 Dependency Relation Pattern Scoring

The answer patterns extracted in section 4.3.1 are scored by support and confidence measures. Support and confidence measures are most commonly used to evaluate association rules in data mining area. The support of a rule is the proportion of times the rule applies. The confidence of a rule is the proportion of times the rule is correct. In our task, we score an answer pattern by measuring the strength of the association rule from the answer

Table 4.1: Examples of Dependency Relation Patterns

Ptn^q	subj_U		
		Sup.	Conf.
$ASet(Ptn^a)$	subj_U	0.02	0.55
	s_U	0.01	0.60
	appo_D	0.007	0.67
	person_U	0.006	0.66
	nn_D	0.005	0.74

Ptn^q	pcomp-n_U mod_U		
		Sup.	Conf.
$ASet(Ptn^a)$	pcomp-n_U mod_U	0.06	0.50
	pcomp-n_U mod_U pcomp-n_U mod_U	0.02	0.30
	pcomp-n_U mod_U obj_U	0.008	0.36
	pcomp-n_U mod_U i_D	0.006	0.33

Ptn^q	obj_U		
		Sup.	Conf.
$ASet(Ptn^a)$	obj_U	0.04	0.46
	pcomp-n_U mod_U	0.02	0.33
	mod_D pcomp-n_D	0.01	0.43
	nn_U obj_U	0.007	0.33

Ptn^q	lex-mod_U		
		Sup.	Conf.
$ASet(Ptn^a)$	pcomp-n_U mod_U	0.02	0.25
	subj_D	0.008	0.25
	num_U	0.008	0.73
	appo_D	0.006	0.71
	nn_D	0.006	0.50

pattern to proper answer (the pattern is matched $\Rightarrow$ the answer is correct). Let Ptn^a be one of answer patterns in an answer pattern set $ASet(Ptn^a)$.

$$\text{support}(Ptn^a) = \frac{\text{the number of times } Ptn^a \text{ is matched and } ac \text{ is correct}}{\text{the total number of times patterns in } ASet \text{ are matched}}$$

$$\text{confidence}(Ptn^a) = \frac{\text{the number of times } Ptn^a \text{ is matched and } ac \text{ is correct}}{\text{the total number of times } Ptn^a \text{ is matched}}$$

We score all of the answer patterns. A pattern is removed from the set if its support value is less than the threshold t_{sup} or its confidence value is less than the threshold t_{conf}. In the experiment, we set t_{sup} 0.005 and t_{conf} 0.2. Table 4.1 lists the support and confidence values of the pattern examples.

4.3.3 Dependency Relation Pattern Matching

After extracting and scoring patterns from training data, we further discuss how to pinpoint answers by matching the patterns. Actually, the pattern sets may not be sufficient enough to cover all of unseen cases since we construct them from a limited training data set. Therefore, exact pattern matching might suffer from data sparseness. In this section, we will propose a string kernel-based method to partially match answer patterns. Section 7.3.2 will further report experimental comparison of the extact pattern matching and the partial pattern matching.

Since dependency relation patterns are represented as relation sequences, such as "*pred_U i_U mod_D pcomp-n_D appo_D mod_D pcomp-n_D*", and each character of the sequence is an individual dependency relation between two nodes, string kernel is easily adapted to matching patterns by calculating the similarity between two sequences.

(Haussler, 1999) proposed the first piece of work to describe a convolution kernel over strings. (Lodhi et al., 2000) applied string kernel to text classification. (Leslie,

Eskin, and Noble, 2002) further proposed a spectrum kernel, which was simpler and more efficient than previous string kernels, for protein classification problem. In their tasks, string kernels achieved the better performance than human-defined features.

String kernel is based on the observation that the more common subsequences two strings share, the more similar they are. The string kernel we used is similar to (Lodhi et al., 2000). It is an inner product in the feature space generated by all subsequences of length k. A k-length subsequence is any ordered sequence of k characters occurring in the string though not necessarily contiguously. The degree of contiguity of one subsequence determines how much weight it will have in the comparison. For example, the subsequence "*bad*" ($k = 3$) is present in the string "*badge*", "*band*" and "*bland*", but with different weights. An exponentially decaying factor λ (set 0.5 in the experiment) of their full length l is used to weight subsequences. It emphasizes those occurrences that are close to contiguous. For the above example, the weight of "*bad*" in "*badge*" is λ^3 ($l = 3$); the weight of "*bad*" in "*band*" is λ^4 ($l = 4$); the weight in "*bland*" is λ^5 ($l = 5$). Given two strings s and s', the string kernel function is formalized as follows:

$$SK(s, s') = \sum_{u \subseteq s \wedge u \subseteq s'} (\lambda^{l_s(u)} \times \lambda^{l_{s'}(u)})$$

where, $l_s(u)$ and $l_{s'}(u)$ are the full length of u in the strings s and s' respectively. Finally, the similarity between s and s' are defined as the normalized string kernel value.

$$Sim(s, s') = \frac{SK(s, s')}{\sqrt{SK(s, s)} \times \sqrt{SK(s', s')}}$$

A direct computation of this feature vector would involve a prohibitive amount of computation even for the modest value of k, since the dimension of feature space grows exponentially with k. (Lodhi et al., 2000) further described a strategy to efficiently calculate the inner product based on a dynamic programming technique. Different from (Lodhi et al., 2000), the characters (individual relations) of the string in our task are linked with each other. Therefore, the matching between two subsequences need consider the linking information. Two identical substrings will not only have the same individual

relations but also have the same linking symbols. For efficiency, we only consider the subsequence of length 2 ($k = 2$). Figure 4.2 shows an example of the string kernel measure for the relation sequences "*pcomp-n_U mod_U obj_U*" and "*pcomp-n_U mod_U i_D*".

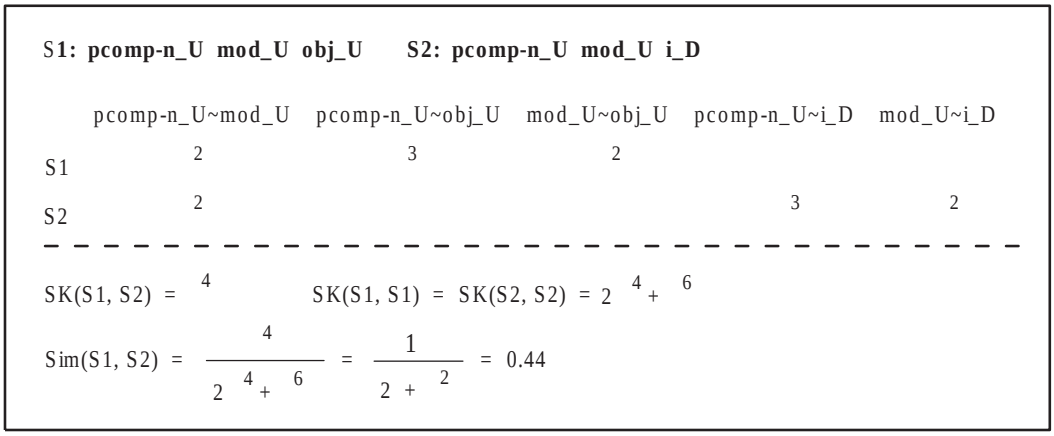


Figure 4.2: An example of the string kernel measure for the relation sequences "*pcomp-n_U mod_U obj_U*" and "*pcomp-n_U mod_U i_D*"

4.4 Dependency Relation Path Correlation Method

In Section 4.3, we propose a dependency relation pattern method. It achieves high accuracy, but poor coverage due to limited size of patterns. Although it applies partial matching method, such as string kernel, to somehow prevent pattern sparseness problem, they still follow the hypothesis that the more common individual relations two dependency relation patterns have, the more matching score they are. The hypothesis has limitation due to syntactic relation variation. One meaning is often represented as different syntactic relation combinations. For example, in the question "*Who founded the Black Panthers organization?*", where, the question word "*who*" has the dependency relations "*subj_U*" with the verb "*found*" and "*subj_U obj_D nn_D*" with the phrase "*Black Panthers organization*". In the corresponding answer sentence "*Black Panther Party co-*

founder ***Bobby Seale*** *is ordered bound and gagged ...*", the appositive relation "*appo_U*" appear in stead of "*subj_U*" to indicate the proper answer "*Bobby Seale*". In this case, the individual dependency relations of two relation paths are totally different and pattern matching method, whatever exact matching or partial matching, will unfortunately fail on it. In this section, we propose a dependency relation path correlation method as a back-off of the dependency relation pattern method.

For each question, dependency relation paths are defined and extracted from the question and its answer sentences. The paths from the question and the answer sentences are paired according to question phrase mapping (Section 3.5.1). Next, correlation between two paths of each pair is calculated by employing a dynamic time warping algorithm. The calculation of the path correlation relies on individual relation correlations, which are estimated from a set of training path pairs. We discuss how the method performs in detailed below.

4.4.1 Dependency Relation Path Extraction

Dependency relation path is defined as a structure $P =< N_1, R, N_2 >$ where, N_1, N_2 are two phrases and R is a relation sequence $R =< r_1, ..., r_i >$ in which r_i is one of the predefined dependency relations. Totally, there are 42 relations defined in MiniPar. The relation sequence R between N_1 and N_2 is extracted by traversing from the N_1 node to the N_2 node in a dependency tree. Individual relations in a sequence are linked by the symbols indicating upward or downward movements through the tree.

For each question, we extract relation paths between question word, such as "*who*", "*what*", "*when*" and question key phrases. The question word is further replaced with "*EAP*", which indicates expected answer position. Analogously, for each candidate sentence, we extract relation paths between answer candidates and mapped question phrases. Figure 4.3 shows the relation paths extracted for a sample question and its answer sentence.

Q: [**What**] [**book**] did [**Rachel Carson**] [**write**] in [**1962**] ?

N1(EAP)	R	N2
What	det_U	book
What	det_U obj_U subj_D	Rachel Carson
What	det_U obj_U	write
What	det_U obj_U mod_D pcomp-n_D	1962

S: [**Rachel Carson**] 's [**1962**] [**book**] " [**Silent Spring**] " said dieldrin causes mania .

N1(AC)	R	N2
Silent Spring	title_U	book
Silent Spring	title_U gen_D	Rachel Carson
Silent Spring	title_U num_D	1962

Figure 4.3: Dependency relation paths for a sample question and sentence. *EAP* indicates expected answer position; *AC* indicates answer candidate.

Next, the relation paths extracted in a question and its answer sentences are paired according to phrase similarity measure. For two relation paths P_i and P_j which are extracted from a question and its answer sentence respectively, if $Sim(N_{i1}, N_{j1}) > 0$ and $Sim(N_{i2}, N_{j2}) > 0$, the P_i and P_j are paired as $< P_i, P_j >$. The question phrase "*EAP*" is mapped to the answer candidates in the sentence. The similarity between two phrases were discussed in Section 3.5.1. Figure 4.4 further shows the paired relation paths which are presented in Figure 4.3.

4.4.2 Dependency Relation Path Correlation

Comparing a proper answer and other wrong candidate answers, we assume that relation paths for the proper answer are more correlated to the corresponding paths in question. So, for each path pair $< P_1, P_2 >$, we measure the correlation between the paths P_1 and

N1 (EAP/AC)	Rq	Rs	N2
Silent Spring	det_U	title_U	book
Silent Spring	det_U obj_U subj_D	title_U gen_D	Rachel Carson
Silent Spring	det_U obj_U mod_D pcomp-n_D	title_U num_D	1962

Figure 4.4: Paired dependency relation paths for a sample question and sentence. *EAP* indicates expected answer position; *AC* indicates answer candidate.

P_2.

We derive the correlations between paths by adapting a dynamic time warping (DTW) algorithm (Sakoe and Chiba, 1971; Itakura, 1975). The algorithm has been widely applied in the area of speech recognition (Rabiner, Rosenberg, and Levinson, 1978). Given point-by-point distance measurement between individual characters, DTW is to find an optimal alignment between two sequences which minimizes the accumulated distance. A sketch of the adapted algorithm is as follows.

Let $R_1 =< r_{11}, ..., r_{1n} >, (n = 1, ..., N)$ and $R_2 =< r_{21}, ..., r_{2m} >, (m = 1, ..., M)$ denote two relation sequences to be matched. R_1 and R_2 consist of N and M relations respectively. $R_1(n) = r_{1n}$ and $R_2(m) = r_{2m}$. $Cor(r_{1n}, r_{2m})$ denotes the correlation between two individual relations r_{1n} and r_{2m}, which is estimated by a statistical model during training (Section 4.4.3). Firstly, we convert the correlation measure $Cor(r_{1n}, r_{2m})$ to the distance measure $Dist(r_{1n}, r_{2m})$ using the following formula:

$$Dist(r_{1n}, r_{2m}) = 1 - Cor(r_{1n}, r_{2m}) \text{ where, } Cor(r_{1n}, r_{2m}) \in [0, 1]$$

Given the distance $Dist(r_{1n}, r_{2m})$ for each pair of relations (r_{1n}, r_{2m}) within R_1 and R_2, the goal of DTW is to find a path, $m = map(n)$, which map n onto the corresponding m such that the accumulated distance $Dist^*$ along the path is minimized.

$$Dist^* = \min_{map(n)} \left\{ \sum_{n=1}^{N} Dist(R_1(n), R_2(map(n)) \right\}$$

Figure 4.5 shows a visualized alignment path between two relation sequences

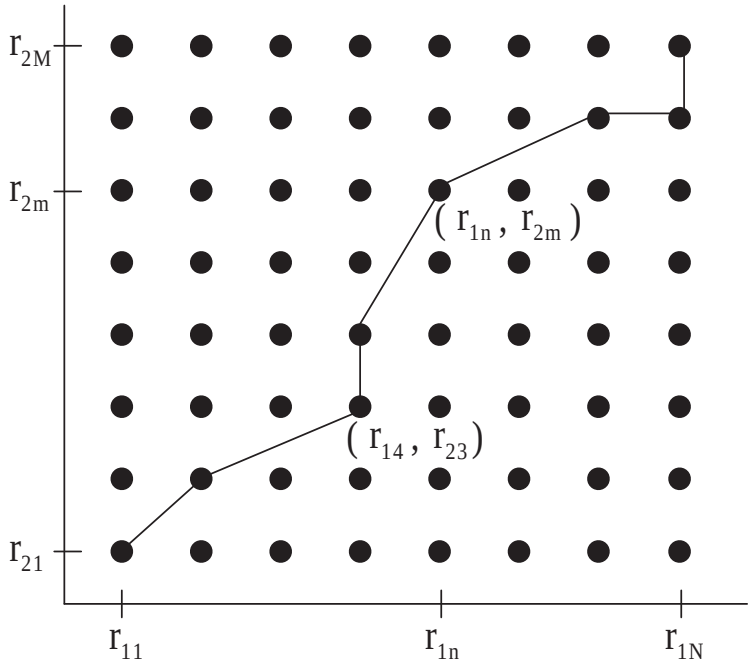


Figure 4.5: A visualized alignment path between two relation sequences $R_1 =< r_{11}, ..., r_{1n} >, (n = 1, ..., N)$ and $R_2 =< r_{21}, ..., r_{2m} >, (m = 1, ..., M)$ in the dynamic time warping algorithm

R_1 and R_2 in the DTW algorithm. An especially powerful technique for determining the optimum path $m = map(n)$ is the method of dynamic programming. Using the technique, the accumulated distance $Dist_A$ to any grid point (n, m) can be recursively calculated as

$$Dist_A(n, m)$$
$$= Dist(r_{1n}, r_{2m}) + \min\{Dist_A(n-1, m), Dist_A(n-1, m-1), Dist_A(n, m-1)\}$$

where, $Dist_A(n, m)$ is the minimum accumulated distance to the grid point (n, m). We find the solution $Dist^* = Dist_A(N, M)$.

The overall distance measure has to be normalized as longer sequences normally give higher scores. So, the distance between two sequences R_1 and R_2 is calculated as

$$Dist(R_1, R_2) = \frac{Dist^*}{\min(N, M)}$$

Finally, we convert the distance measurement to the correlation measurement using

$$Cor(R_1, R_2) = 1 - Dist(R_1, R_2)$$

Figure 4.6 shows an example of correlation measure between two relation sequences R_1 and R_2 using the DTW algorithm. Figure 4.6(a) lists the distance between the individual relations r_{1n} and r_{2m}. Figure 4.6(b) lists the accumulated distance to any grid point (n, m). We find the optimal path shown in Figure 4.6(c) with the following steps:

1. The accumulated distance from the starting position to any position in the first column is computed.

2. The minimum accumulated distance from the starting position to any position in second column is computed.

3. Repeat the step 2 and obtain the minimum accumulated distance from the starting position to every position in every column.

4. The overall distance $Dist^*$ between the relation sequences R_1 and R_2 is the value in the right-top grid.

5. Normalize the overall distance $Dist^*$.

6. Convert the distance measure $Dist(R_1, R_2)$ to the correlation measure $Cor(R_1, R_2)$.

Finally, the correlation between the relation sequences R_1 and R_2 is measured in Figure 4.6 (d).

According to the correlations between two relation sequences, we define the correlation between two relation paths P_1 and P_2 as

$$Cor(P_1, P_2) = Cor(R_1, R_2) \times Sim(N_{11}, N_{21}) \times Sim(N_{12}, N_{22})$$

	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}
r_{26}	0.2	0.3	0.6	0.3	0.2
r_{25}	0.7	0.2	0.1	0.2	0.3
r_{24}	0.8	0.3	0.0	0.3	0.4
r_{23}	0.7	0.2	0.1	0.2	0.3
r_{22}	0.5	0.0	0.3	0.0	0.1
r_{21}	0.1	0.4	0.7	0.4	0.3

(a) Distance between individual relations

	r_{11}	r_{12}	r_{13}	r_{14}	r_{15}
r_{26}	3.0	1.1	0.9	0.6	0.6
r_{25}	2.8	0.8	0.3	0.4	0.7
r_{24}	2.1	0.6	0.2	0.5	0.8
r_{23}	1.3	0.3	0.2	0.4	0.7
r_{22}	0.6	0.1	0.4	0.4	0.5
r_{21}	0.1	0.5	1.2	1.6	1.9

(b) Accumulated Distance

n	m = map(n)
1	1
2	2
3	3, 4
4	5
5	6

(c) Alignment mapping

$$Dist^* = Dist_A(N, M) = 0.6$$

$$Dist(R_1, R_2) = \frac{Dist^*}{Min(N, M)}$$

$$= 0.12$$

$$Cor(R_1, R_2) = 1 - Dist(R_1, R_2)$$

$$= 0.88$$

Figure 4.6: An example of correlation measure between two relation sequences R_1 and R_2 using the Dynamic Time Warping Algorithm

where, $Sim(N_{11}, N_{21})$ and $Sim(N_{12}, N_{22})$ are the phrase mapping score when pairing two paths, as described in Section 3.5.1. If two phrases are absolutely different $Cor(N_{11}, N_{21}) = 0$ or $Cor(N_{12}, N_{22}) = 0$, the paths may not be paired since $Cor(P_1, P_2) = 0$.

Table 4.2: Examples of high-correlated dependency relations and their correlation score

Relation 1	Relation 2	Correlation Score
subj_U	sc_D	6.72E-4
vrel_D	pred_D	2.08E-2
obj_U	appo_U	2.99E-4
pcomp-n_D	appo_D	1.87E-3
pred_U	obj_U	1.62E-3
mod_D	appo_U	2.10E-3
i_D	appo_D	3.49E-3
head_D	post_D	7.28E-3

4.4.3 Individual Relation Correlation Estimation

In the above section, we have described how to measure path correlations. The measure requires individual relation correlations $Cor(r_1, r_2)$ as inputs. We apply a statistical method to estimate the relation correlations from a set of training path pairs. The training data collecting will be described in Section 7.1.

For each question and its answer sentences in training data, we extract the relation paths between "*EAP*" and key phrases in the question and the paths between proper answer and mapped question phrases in the sentences. After pairing the paths, correlation of two individual relations is measured by their bipartite co-occurrence in all training path pairs. Mutual information measure (Cui et al., 2004) is employed to calculate the relation correlations.

$$Cor(r_i^Q, r_j^S) = log \frac{\sum \alpha \times \delta(r_i^Q, r_j^S)}{f_Q(r_i^Q) \times f_S(r_j^S)}$$

where, r_i^Q and r_j^S are two relations in question paths and answer sentence paths respectively. $f_Q(r_i^Q)$ and $f_S(r_j^S)$ are the numbers of occurrences of r_i^Q in question paths and r_j^S in sentence paths respectively. $\delta(r_i^Q, r_j^S)$ is 1 when r_i^Q and r_j^S co-occur in a path pair, and 0 otherwise. α is a factor to discount the co-occurrence value for long paths. It is set to the inverse proportion of the sum of path lengths of the path pair. Table 4.2 shows

examples of high-correlated dependency relations and their correlation score.

Chapter 5

Semantic Evidence for Answer Extraction

5.1 Motivation

Recent years have witnessed significant progress in developing methods for automatic identification and labeling of semantic roles conveyed by sentential constituents.[1] The success of these methods, often referred to collectively as *shallow semantic parsing* (Gildea and Jurafsky, 2002), is largely due to the availability of resources like FrameNet (Fillmore, Johnson, and Petruck, 2003) and PropBank (Palmer, Gildea, and Kingsbury, 2005), which document the surface realization of semantic roles in real world corpora.

More concretely, in the FrameNet paradigm, the meaning of predicates (usually verbs, nouns, or adjectives) is conveyed by *frames*, schematic representations of situations. Semantic roles (or *frame elements*) are defined for each frame and correspond to salient entities present in the evoked situation. Predicates with similar semantics instantiate the same frame and are attested with the same roles. The FrameNet database lists the surface syntactic realizations of semantic roles, and provides annotated example

[1] The approaches are too numerous to list; we refer the interested reader to Carreras and Màrquez (Carreras and Màrquez, 2005) for an overview.

sentences from the British National Corpus. For example, the frame *Commerce_Sell* has three core semantic roles, namely *Buyer*, *Goods*, and *Seller* — each expressed by an indirect object, a direct object, and a subject (see sentences (5.1a)–(5.1c)). It can also be attested with non-core (peripheral) roles (e.g., *Means, Manner*, see (5.1d) and (5.1e)) that are more generic and can be instantiated in several frames, besides *Commerce_Sell*. The verbs "*sell*", "*vend*", and "*retail*" can evoke this frame, but also the nouns "*sale*" and "*vendor*".

(5.1) a. [Lee]$_{Seller}$ sold a textbook [to Abby]$_{Buyer}$.

b. [Kim]$_{Seller}$ sold [the sweater]$_{Goods}$.

c. [My company]$_{Seller}$ has sold [more than three million copies]$_{Goods}$.

d. [Abby]$_{Seller}$ sold [the car]$_{Goods}$ [for cash]$_{Means}$.

e. [He]$_{Seller}$ [reluctantly]$_{Manner}$ sold [his rock]$_{Goods}$.

By abstracting over surface syntactic configurations, semantic roles offer an important first step towards deeper text understanding and hold promise for a range of applications requiring broad coverage semantic processing. Question answering (QA) is often cited as an obvious beneficiary of semantic role labeling (Gildea and Jurafsky, 2002; Palmer, Gildea, and Kingsbury, 2005; Narayanan and Harabagiu, 2004). Faced with the question "*Q: What year did the U.S. buy Alaska?*" and the retrieved sentence "*S: . . . before Russia sold Alaska to the United States in 1867*", a hypothetical QA system must identify that "*United States*" is the *Buyer* despite the fact that it is attested in one instance as a subject and in another as an object. Once this information is known, isolating the correct answer (i.e., "*1867*") can be relatively straightforward.

Although conventional wisdom has it that semantic role labeling ought to improve answer extraction, surprising little work has been done to this effect (see Section 5.2 for details) and initial results have been mostly inconclusive or negative (Sun et al., 2005; Kaisser, 2006). There are at least two good reasons for these findings. First, shallow

semantic parsers trained on declarative sentences will typically have poor performance on questions and generally on out-of-domain data. Second, existing resources do not have exhaustive coverage and recall will be compromised, especially if the question answering system is expected to retrieve answers from unrestricted text. Since FrameNet is still under development, its coverage tends to be more of a problem in comparison to other semantic role resources such as PropBank.

This chapter proposes an automatic method to effectively incorporates FrameNet-style semantic role information. The way we conduct semantic role assignment is conceptually simple and does not require extensive feature engineering. A key feature of our approach is the comparison of dependency relation paths attested in FrameNet annotations and raw text. We formalize the search for an optimal role assignment as an optimization problem in a bipartite graph. This formalization allows us to find an exact, globally optimal solution. The graph-theoretic framework goes some way towards addressing coverage problems related with FrameNet and allows us to formulate answer extraction as a graph matching problem.

In the following section I will provide an overview of existing work on question answering systems exploiting semantic role-based lexical resources. Then I will present our semantic role-based method.

5.2 Related Work

Question answering systems have traditionally depended on a variety of lexical resources to bridge surface differences between questions and potential answers. WordNet (Miller, 1990) is perhaps the most popular resource and has been employed in a variety of QA-related tasks ranging from query expansion, to axiom-based reasoning (Moldovan et al., 2003), passage scoring (Paranjpe, Ramakrishnan, and Srinivasa, 2003), and answer filtering (Leidner et al., 2003). Besides WordNet, recent QA systems increasingly rely on

syntactic information as a means of abstracting over word order differences and structural alternations (e.g., passive vs. active voice). Most syntax-based QA systems (Wu et al., 2005) incorporated some means of comparison between the tree representing a question with the subtree surrounding an answer candidate. The assumption here is that appropriate answers are more likely to have syntactic relations in common with their corresponding questions. Syntactic structure matching has been applied to passage retrieval (Cui et al., 2005) and answer extraction (Shen and Klakow, 2006).

Narayanan and Harabagiu (Narayanan and Harabagiu, 2004) were the first to stress the importance of semantic roles in answering complex questions. Their system identified predicate argument structures by merging semantic role information from PropBank and FrameNet. Expected answers were extracted by performing probabilistic inference over the predicate argument structures in conjunction with a domain specific topic model. Sun et al. (Sun et al., 2005) incorporated semantic analysis in their TREC05 QA system. They used ASSERT (Pradhan et al., 2004), a publicly available shallow semantic parser trained on PropBank, to generate predicate-argument structures which subsequently formed the basis of comparison between question and answer sentences. They found that semantic analysis didn't boost performance due to the low recall of the semantic parser. Kaisser (Kaisser, 2006) proposed a question paraphrasing method based on FrameNet. Questions were assigned semantic roles by matching their dependency relations with those attested in FrameNet annotations. The assignments were used to create question reformulations which were submitted to Google for answer extraction. Their semantic role assignment module was not probabilistic, it relied on strict matching, and run into severe coverage problems.

In line with previous work, our method exploits syntactic information in the form of dependency relation paths together with FrameNet-like semantic roles to smooth lexical and syntactic divergences between questions and answer sentences. Our approach is less domain dependent and resource intensive than Narayanan and Harabagiu (Narayanan

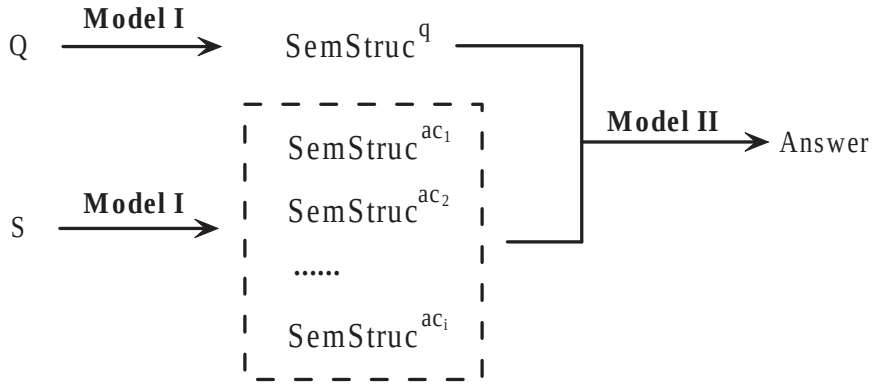


Figure 5.1: Architecture of Semantic Role Method

and Harabagiu, 2004), it solely employs a dependency parser and the FrameNet database. In contrast to Kaisser (Kaisser, 2006), we model the semantic role assignment and answer extraction tasks numerically, thereby alleviating the coverage problems encountered previously.

5.3 Semantic Role Method

5.3.1 Problem Formulation

We have briefly summarized the architecture of the Alyssa QA system in Chapter 3 before formalizing the mechanics of FrameNet-based Semantic evidence. The goal of the semantic evidence is to tackle lexical and syntactic divergences between question and answer sentences. It allows the tolerant matching of lexical terms referring to common concepts and unifies various syntactic representations into identical semantic structures. The core idea of this method is that question and answer sentences are normalized to FrameNet-style semantic representations and answer is retrieved by selecting the candidate whose semantic structure is most similar to the question. Figure 5.1 shows the architecture of the Semantic Role Method.

Semantic structures for questions and sentences are automatically derived using the model described in Section 5.3.2 (Model I). A semantic structure $SemStruc = \langle p, Set(SRA) \rangle$ consists of a predicate p and a set of semantic role assignments $Set(SRA)$. p is a word or phrase evoking a frame F of FrameNet. A semantic role assignment SRA is a ternary structure $\langle w, SR, s \rangle$, consisting of frame element w, its semantic role SR, and score s indicating to what degree SR qualifies as a label for w.

For a question q, we generate a semantic structure $SemStruc^q$. Question words, such as "*what*", "*who*", "*when*", etc., are considered expected answer phrases (EAPs). We require that EAPs are frame elements of $SemStruc^q$. Words involved in semantic role assignments of $SemStruc^q$ are the key words/phrases of the question.

Likely for each answer candidate ac in answer sentences, we derive its semantic structure $SemStruc^{ac}$ and assume that ac is a frame element of $SemStruc^{ac}$. Words in semantic role assignments of $SemStruc^{ac}$ are the key words/phrases of the sentence which are mapped from key words/phrases of the question as discussed in Section 3.5.1. A special mapping is defined from EAP to ac.

Question and answer semantic structures are compared using a model based on graph matching detailed in Section 5.3.3 (Model II). We calculate the similarity of all derived pairs $\langle SemStruc^q, SemStruc^{ac} \rangle$ and select the candidate with the highest value as the answer for the question.

5.3.2 Semantic Structure Generation

Our method crucially exploits the annotated sentences in the FrameNet database together with the output of a dependency parser. Our guiding assumption is that sentences that share dependency relations will also share semantic roles as long as they evoke the same or related frames. This is motivated by much research in lexical semantics (e.g., (Levin, 1993)) hypothesizing that the behavior of words, particularly with respect to the expression and interpretation of their arguments, is to a large extent determined by their

meaning. We first describe how predicates are identified and then introduce our model for semantic role labeling.

Predicate Identification Predicate candidates are identified using a simple look-up procedure which compares POS-tagged tokens against FrameNet entries. For efficiency reasons, we make the simplifying assumption that questions have only one predicate which we select heuristically:

1. Verbs are preferred to other parts of speech;
2. If there is more than one verb in the question, preference is given to the verb with the highest level of embedding in the dependency tree;
3. If no verbs are present, a noun is chosen.

For example, in the question "*Who beat Floyd Patterson to take the title away?*", the words "*beat*", "*take away*" and "*title*" are identified as predicate candidates and "*beat*" is selected the main predicate of the question. For answer sentences, we require that the predicate is either identical or semantically related to the question predicate (see Section 5.3.3). In the example given above, the predicate "*beat*" evoques a single frame (i.e.,*Cause_harm*). However, predicates often have multiple meanings thus evoquing more than one frame. Knowing which is the appropriate frame for a given predicate impacts the semantic role assignment task; selecting the wrong frame will unavoidably result in erroneous semantic roles. Rather than disambiguating polysemous predicates prior to semantic role assignment, we perform the assignment for each frame evoqued by the predicate and default to the frame whose semantic roles yield the highest score.

Semantic Role Assignment Before describing our approach to semantic role labeling we define dependency relation paths. A relation path R is a relation sequence $\langle r_1, r_2, ..., r_L \rangle$, in which r_l ($l = 1, 2, ..., L$) is one of predefined dependency relations

with suffix of traverse direction. An example of a relation path is $R = \langle subj\_U, obj\_D \rangle$, where the subscripts $\_U$ and $\_D$ indicate upward and downward movement in trees, respectively. Given an unannotated sentence whose roles we wish to label, we assume that words or phrases w with a dependency path connecting them to p are frame elements. Each frame element is represented by an *unlabeled* dependency path R_w which we extract by traversing the dependency tree from w to p. Analogously, we extract from the FrameNet annotations all dependency paths R_{SR} that are *labeled* with semantic role information and correspond to p. We next measure the compatibility of labeled and unlabeled paths as follows:

$$s(w, SR) = \max_{R_{SR} \in M} \left[sim^* \left(R_w, R_{SR} \right) \cdot P(R_{SR}) \right]$$

where M is the set of dependency relation paths for SR in FrameNet, $sim\left(R_w, R_{SR}\right)$ is the similarity between paths R_w and R_{SR} weighted by the relative frequency of R_{SR} in FrameNet ($P(R_{SR})$). We consider both core and non-core semantic roles instantiated by frames with at least one annotation in FrameNet. Core roles tend to have more annotations in FrameNet and consequently are considered more probable.

We measure *sim* (R_w, R_{SR}), by adapting a string kernel $SK(s, s')$ to our task.

$$sim^*(R_w, R_{SR}) = \frac{SK(R_w, R_{SR})}{\sqrt{SK(R_w, R_w)} \times \sqrt{SK(R_{SR}, R_{SR})}}$$

Our hypothesis is that the more common substrings two dependency paths have, the more similar they are. The string kernel we used is similar to Leslie (Lodhi et al., 2000) and are discussed in Section 4.3.3 in detailed. It is defined as the inner product of weighted common dependency relation subsequences between R_w and R_{SR}. For efficiency, we only consider 2-length subsequences ($k = 2$). Weight of a subsequence is defined as the sum of the weights of its individual relations. Individual Relations are weighted by a metric akin to $tf \cdot idf$ which measures the degree of association between a candidate SR and the dependency relation r present in the subsequence.

$$weight_{SR}(r) = f_r \cdot \log\left(1 + \frac{N}{n_r}\right)$$

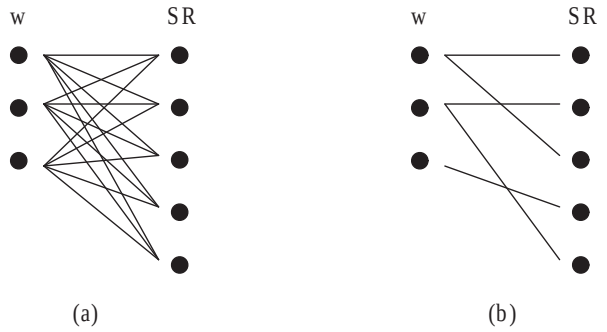


Figure 5.2: Sample original bipartite graph (a) and its subgraph with edge covers (b). In each graph, the left partition represents frame elements and the right partition semantic roles.

where f_r is the frequency of r occurring in SR; N is the total number of SRs evoked by a given frame; and n_r is the number of SRs containing r.

For each frame element we generate a set of semantic role assignments $Set(SRA)$. This initial assignment can be usefully represented as **a complete bipartite graph** in which each frame element (word or phrase) is connected to the semantic roles licensed by the predicate and vice versa. (see Figure 5.2a). Edges are weighted and represent how compatible the frame elements and semantic roles are. Now, for each frame element w we could simply select the semantic role with the highest score. However, this decision procedure is *local*, i.e., it yields a semantic role assignment for each frame element independently of all other elements. We therefore may end up with the same role being assigned to two frame elements or with frame elements having no role at all. We remedy this shortcoming by treating the semantic role assignment as a *global* optimization problem.

We formalize search for the best SR assignment set as an optimization problem in a bipartite graph. The bipartite graph optimization is a flexible and intuitive framework to tackle divergences of SR assignments arising from inexact dependency relation path matching. Moreover, the optimization algorithm, introduced in (Pado and Lapata, 2006), is well-understood and computationally moderate. It is usually phrased as a maximiza-

tion problem over sum of edge weights of a graph. The most important factor to consider in the optimization is the choice of admissible alignments, which impose constraints on SR assignments, such as don't leave target node unassigned, or allow one-to-many assignments. (Pado and Lapata, 2006) lists three admissible alignments (total alignments, edge covers and perfect matching).

Based on observation of the task, we choose edge covers, which is more restrictive than total alignments and less than perfect matching. In a subgraph with edge covers, each node is adjacent to at least one edge. All source and target nodes are forced to participate in assignments. It certainly indicates that edge covers cannot account for unassigned nodes on either side and one-to-many assignments are modeled in both directions. A sample original bipartite graph and its subgraph with edge covers are shown in Figure 5.2. In the figure of the subgraph with edge cover, all source and target nodes are adjacent to an edge and one source node is connected to several target nodes.

According to the choice of the optimization algorithm, we model the interaction between *all* pairwise labeling decisions as a **minimum weight bipartite edge cover problem**. The best assignments with edge covers are obtained using global optimization, since choices of edges are not independent with each other. This calculation of optimal edge covers has been investigated by (Eiter and Mannila, 1997; Cormen, Leiserson, and Rivest, 1990) in context of distance metrics for point sets. They show that edge cover optimization problem can be reduced to perfect matching by adding an auxiliary bipartite graph and constructing a new graph which has the identical number of source and target nodes. As a result, an edge cover is a subgraph of a bipartite graph so that each node is linked to at least one node of the other partition. This yields semantic role assignment for all frame elements and one frame element bearing multiple semantic roles. By inducing such soft labeling we hope to render the matching of questions and answers more robust, thereby addressing to some extent the coverage problems associated with FrameNet. In addition, Edge covers have been successfully applied in several natural language pro-

cessing tasks, including machine translation (Taskar, Lacoste-Julien, and Klein, 2005) and annotation projection (Pado and Lapata, 2006).

Formally, optimal edge cover assignments are the solutions of the following optimization problem:

$$\max_{E \text{ is edge cover}} \prod_{(nd^w, nd^{SR}) \in E} s(nd^w, nd^{SR})$$

where, $s(nd^w, nd^{SR})$ is the compatibility score between the frame element node nd^w and semantic role node nd^{SR}. Edge covers can be computed efficiently in cubic time using the algorithms for the equivalent linear assignment problem. Our experiments use Jonker and Volgenant's (Jonker and Volgenant, 1987) solver.[2]

Figure 5.3 shows the original and optimized semantic role assignments generated by our model for a sample question and answer sentence. Given a question and a sentence:

> "*Q: Who discovered prions?*"
> "*S: 1997: Stanley B. Prusiner, United States, discovery of prions . . .*"

On the question side, we firstly identify "*discover*" as question predicate. Considering "*EAP*" and "*prions*" as frame elements, we extract the unlabeled relation paths for them by traversing the dependency tree from the frame element node to the predicate node. The following is the relation paths for "*EAP*" and "*prions*":

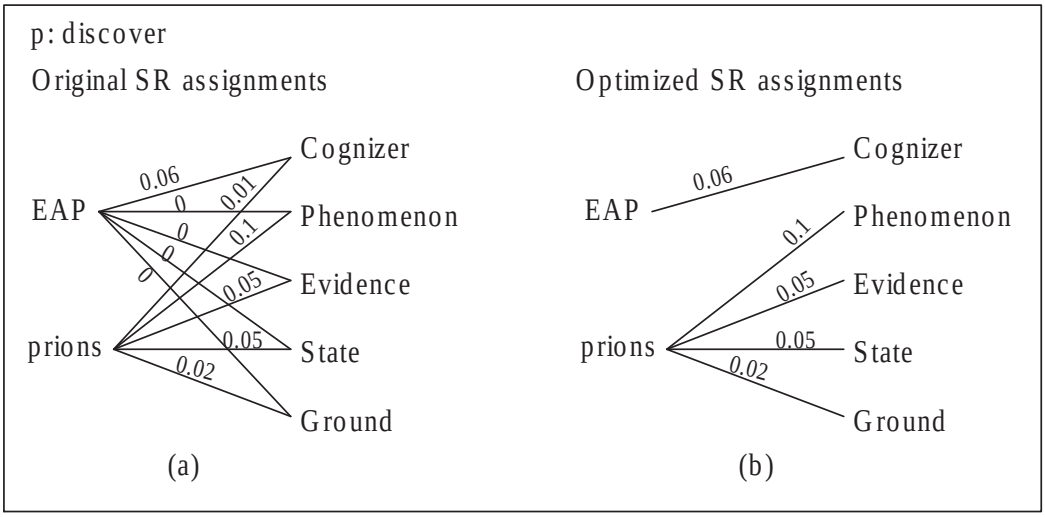
> $R_{EAP} = \langle subj\_U \rangle$
> $R_{prions} = \langle obj\_U \rangle$

Figure 5.4 shows the labeled dependency relation paths R_{SR} for all semantic roles of the predicate "*discover*" in FrameNet. Figure 5.5 shows the weights of individual relations for the predicate "*discover*" in FrameNet. Then we compare the above unlabeled relation paths with the FrameNet labeled paths and assign the edge weights between

[2]The software is available from `http://www.magiclogic.com/assignment.html`.

Q: Who discovered prions?

SemStruc ^{q}



S: 1997: Stanley B. Prusiner, United States, discovery of prions, ...

SemStruc ^{ac} (ac: Stanley B. Prusiner)

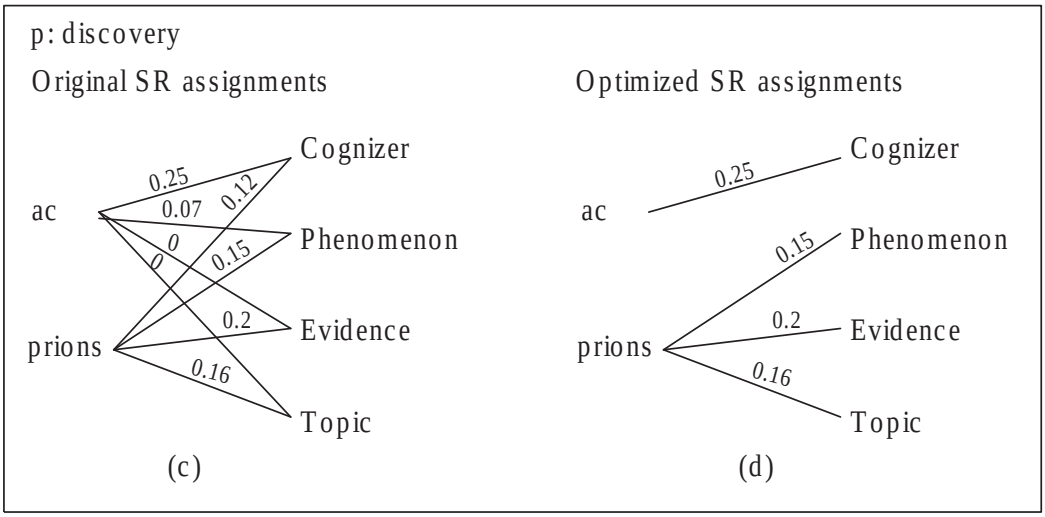


Figure 5.3: Semantic structures induced by our model for an example of question and answer sentence

```
<PRED id="173" name="discover.v" frameid="20" framename="Becoming_aware" sentnumber="376">
    <SR id="87" name="Cognizer" type="Core">
        <SYN score="265">subj_U</SYN>
        <SYN score="236">s_U</SYN>
        <SYN score="15">subj_U conj_U</SYN>
        <SYN score="13">s_U mod_D i_D</SYN>
        ...
    </SR>
    <SR id="88" name="Phenomenon" type="Core">
        <SYN score="282">obj_U</SYN>
        <SYN score="21">s_U i_U rel_U obj_U</SYN>
        ...
    </SR>
    <SR id="89" name="Ground" type="Peripheral">
        <SYN score="50">mod_U obj_U</SYN>
        <SYN score="25">mod_U</SYN>
        <SYN score="5">mod_U pcomp-n_U mod_U obj_U</SYN>
        ...
    </SR>
    <SR id="90" name="State" type="Peripheral">
        <SYN score="8">i_U rel_U obj_U</SYN>
        <SYN score="3">i_U mod_U</SYN>
        ...
    </SR>
    <SR id="91" name="Evidence" type="Peripheral">
        <SYN score="13">mod_U</SYN>
        <SYN score="6">mod_U obj_U</SYN>
        ...
    </SR>
    ...
</PRED>
```

Figure 5.4: Labeled Dependency Relation Paths for the predicate "*discover*" in FrameNet

```
<PRED id="173" name="discover.v" frameid="20" framename="Becoming_aware" sentnumber="376">
    <SR id="87" name="Cognizer" type="Core">
        <R name="sc_D" weight="6.496414920651304" />
        <R name="vrel_U" weight="5.058004558392399" />
        <R name="gen_U" weight="2.3978952727983707" />
        <R name="mod_D" weight="30.91292598208609" />
        <R name="pred_U" weight="7.795697904781566" />
        <R name="obj_D" weight="15.342732830145827" />
        <R name="fc_U" weight="2.365372081092811" />
        <R name="comp1_D" weight="10.394263873042087" />
        <R name="pred_D" weight="1.0116009116784799" />
        ...
    </SR>
    <SR id="88" name="Phenomenon" type="Core">
        <R name="mod_D" weight="11.516580267835995" />
        <R name="head_D" weight="10.228488553430552" />
        <R name="pred_U" weight="14.29211282543287" />
        <R name="s_D" weight="3.897848952390783" />
        <R name="appo_U" weight="1.7047480922384253" />
        <R name="sc_U" weight="3.4094961844768505" />
        <R name="nn_U" weight="7.193685818395112" />
        <R name="obj1_U" weight="3.897848952390783" />
        ...
    </SR>
    ...
</PRED>
```

Figure 5.5: Weights of Individual Dependency Relations for the predicate "*discover*" in FrameNet

```
<PRED id="174" name="discovery.n" frameid="20" framename="Becoming_aware" sentnumber="37">
    <SR id="87" name="Cognizer" type="Core">
        <SYN score="6">gen_U</SYN>
        <SYN score="4">mod_U</SYN>
        <SYN score="2">mod_U s_U subj_D</SYN>
        <SYN score="2">subj_U obj_D</SYN>
        ...
    </SR>
    <SR id="88" name="Phenomenon" type="Core">
        <SYN score="9">mod_U</SYN>
        <SYN score="6">nn_U</SYN>
        <SYN score="3">i_U comp1_U</SYN>
        <SYN score="2">nn_U s_U subj_D</SYN>
        ...
    </SR>
    <SR id="91" name="Evidence" type="Peripheral">
        <SYN score="1">mod_U</SYN>
    </SR>
    <SR id="963" name="Topic" type="Core">
        <SYN score="6">mod_U</SYN>
        <SYN score="1">mod_U s_U obj_D</SYN>
        ...
    </SR>
    ...
</PRED>
```

Figure 5.6: Labeled Dependency Relation Paths for the predicate "*discovery*" in FrameNet

```
<PRED id="174" name="discovery.n" frameid="20" framename="Becoming_aware" sentnumber="37">
    <SR id="87" name="Cognizer" type="Core">
        <R name="pred_D" weight="1.8325814637483102" />
        <R name="mod_U" weight="1.7851484105136781" />
        <R name="mod_D" weight="4.828313737302301" />
        <R name="subj_U" weight="5.497744391244931" />
        <R name="subj_D" weight="2.7488721956224653" />
        <R name="obj_U" weight="3.2188758248682006" />
        ...
    </SR>
    <SR id="88" name="Phenomenon" type="Core">
        <R name="mod_U" weight="2.6777226157705174" />
        <R name="pred_D" weight="2.7488721956224653" />
        <R name="subj_U" weight="0.9162907318741551" />
        <R name="subj_D" weight="5.497744391244931" />
        <R name="s_D" weight="1.8325814637483102" />
        ...
    </SR>
    <SR id="91" name="Evidence" type="Peripheral">
        <R name="mod_U" weight="0.22314355131420976" />
    </SR>
    ...
</PRED>
```

Figure 5.7: Weights of Individual Dependency Relations for the predicate "*discovery*" in FrameNet

w and SR, as shown in (a) of Figure 5.3. Finally, we reduce SR assignments ((a) of Figure 5.3) to an optimal subgraph with edge covers ((b) of Figure 5.3)

Analogously on the sentence side, we first identify "*discovery*" as sentence predicate corresponding to the question predicate "*discover*". Considering "*Stanley B. Prusiner*" as an answer candidate, we extract its unlabeled relation path. Moreover, we extract the relation paths for the key phrases which are mapped from the question key phrases, such as "*prions*", as follows:

$$R_{AC} = \langle subj\_U, s\_D, appo\_D \rangle$$
$$R_{prions} = \langle pcomp - n\_U, mod\_U \rangle$$

Figure 5.6 shows the labeled dependency relation paths of the predicate "*discovery*" in FrameNet. Figure 5.7 shows the weights of individual relations. (c) and (d) of Figure 5.3 shows the original and optimized semantic structures of the answer candidate "*Stanley B. Prusiner*".

5.3.3 Semantic Structure Matching

We measure the similarity between a question and its candidate answer by matching their predicates and semantic role assignments. Since SRs are frame-specific, we prioritize frame matching to SR matching. Two predicates match if they evoke the same frame or one of its hypernyms (or hyponyms). The latter are expressed by the "*Inherits From*" and "*Is Inherited By*" relations in the frame definitions. If the predicates match, we examine whether the assigned semantic roles match. Since we represent SR assignments as graphs with edge covers, we can also formalize SR matching as a graph matching problem.

The similarity between two graphs is measured as the sum of similarities between their subgraphs. We first decompose a graph into subgraphs consisting of one frame element node w and a set of SR nodes connected to it. The similarity between two

subgraphs $SubG_1$, and $SubG_2$ is then formalized as:

$$Sim(SubG_1, SubG_2) = \sum_{\substack{nd_1^{SR} \in SubG_1 \\ nd_2^{SR} \in SubG_2 \\ nd_1^{SR} = nd_2^{SR}}} \frac{1}{|s(nd^w, nd_1^{SR}) - s(nd^w, nd_2^{SR})| + 1}$$

Where, nd_1^{SR} and nd_2^{SR} are semantic role nodes connected to a frame element node nd^w in $SubG_1$ and $SubG_2$, respectively. $s(nd^w, nd_1^{sr})$ and $s(nd^w, nd_2^{SR})$ are edge weights between two nodes in corresponding subgraphs. Our intuition here is that the more semantic roles two subgraphs share for a given frame element, the more similar they are and the closer their corresponding edge weights should be. Edge weights are normalized by dividing by the sum of all edges in a subgraph.

Chapter 6

Maximum Entropy-based Answer Extraction Model

6.1 Maximum Entropy Model

Maximum entropy ideas are the simplest way of modeling all that is known and assume nothing about that which is unknown. Given a collection of events, to avoid making any bold assumption without empirical justification, the best way we can do is to assume that each event is equal probable subject to a set of certain constraints, that is, the probability distribution for the events is uniformed. With the increase of the contraint complexity, we encounter two difficulties at once. Firstly, what exactly is meant by ”uniform” and how can we measure the uniformity of a model? Secondly, having determined a suitable answer to the first question, how do we go about finding the most uniform model? By using the term ”maximum entropy”, we may answer both of the questions. The maximum entropy based formulism is to seek a model satisfying all of known constraints and meanwhile, predicating probability distribution as uniform as possible. This idea is pioneered and extended by (Jaynes, 1983; Berger, Pietra, and Pietra, 1996; Pietra, Pietra, and Lafferty, 1997) in the areas of statistical modeling and natural language processing.

As in (Jaynes, 1983), Jaynes' writes:

> *... the fact that a certain probability distribution maximizes entropy subject to certain constraints representing our incomplete information, is the fundamental property which justifies use of that distribution for inference; it agrees with everything that is known, but carefully avoids assuming anything that is not known. It is a transcription into mathematics of an ancient principle of wisdom.*

So far, Maximum entropy principles have been applied to a variety of problems including:

- Sentence Boundary Detection (Reynar and Ratnaparkhi, 1997)
- Named Entity Detection (Borthwick et al., 1998)
- Phrase Chunking (Koeling, 2000)
- Parsing (Ratnaparkhi, 1999)
- Natural Language Ambiguity Resolution (Ratnaparkhi, 1998)
- Machine Translation (Berger, Pietra, and Pietra, 1996)
- Language Modeling (Rosenfeld, 1994)
- Answer Type Classification (Ittycheriah, Franz, and Roukos, 2001; Ittycheriah et al., 2002)

A thorough description of maximum entropy was presented in (Berger, Pietra, and Pietra, 1996). In this chapter, a brief presentation of the maximum entropy formulation is presented and proofs are left to the references.

6.1.1 Maximum Entropy Principle

Maximum Entropy model is a method of estimating the conditional probability that, given a context x, the process will output y. We denote by $p(y|x)$ the conditional probability that the model assigns to y in the context of x. As described in (Berger, Pietra, and Pietra, 1996), we also use $p(y|x)$ to denote an entire conditional probability distribution provided by the model, where, y and x are regarded as placeholders rather than specific instantiations. $\mathcal{P}$ is defined as the set of all conditional probability distributions. Thus a model $p(y|x)$ is just an element of $\mathcal{P}$.

To get the optimal distribution $p(y|x)$, we collect a large number of training samples (x_1, y_1), (x_2, y_2),...,(x_N, y_N). Moreover, according to empirical knowledge of certain task, we define a set of constraints $\mathcal{C} = \{c_1, ...c_n\}$, where n is the number of the constraints. Firstly, the model $p(y|x)$ is required to satisfy all of the constraints. Subject to this requirement, there still are more than one possible probability distribution. Among all of the possible models p, the most uniform distribution is selected. A mathematical measure of the uniformity of a conditional probability distribution $p(y|x)$ is provided by conditional entropy:

$$H(p) \equiv -\sum_{x,y} \tilde{p}(x)p(y|x)\log p(y|x)$$

Maximum entropy-based model chooses a model p_* that maximizes the entropy $H(p)$ subject to the constraints $\mathcal{C}$.

$$p_* = \arg\max_{p \in \mathcal{C}} H(p)$$

This is a constrained optimization problem. The method of Lagrange multipliers from the theory of constrained optimization is applied. It transforms the primary problem (constrained optimization problem) to the dual problem (unconstrained optimization problem) by incorporating Lagrange multipliers λ_i for each constraint c_i. By the transformation, the searching of the model p_* which maximizes the entropy $H(p)$ turns to be

the searching of the real-valued vector $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ which maximizes the likelihood $\Psi(\lambda)$. In the following two sections, we will discuss how to represent constraints in the model and how to estimate the values of each λ_i ($\lambda_i \in \{\lambda_1, \lambda_2, ..., \lambda_n\}$.

6.1.2 Representing Constraints

For certain task, people always empirically or statistically collect evidence about events to make correct prediction. One way to represent the evidence is to encode useful facts as features. Effective features may successfully abstract and differentiate events and make system conduct proper prediction as easy as possible. A feature f is a binary/real valued function on events. For example, in automatic PERSON recognition task, to express the fact that one word belongs to a PERSON name when it follows the word "*Mr.*", we can introduce the feature function:

$$f(x,y) = \begin{cases} 1 & \text{if } y \text{ is PERSON name and } x \text{ follows } \textit{Mr.} \\ 0 & \text{otherwise} \end{cases}$$

The expected value of f with respect to the empirical distribution $\tilde{p}(x,y)$ is denoted as

$$\tilde{p}(f) \equiv \sum_{x,y} \tilde{p}(x,y) f(x,y)$$

Where, the empirical probability distribution $\tilde{p}(x,y)$ can be simply counted from training events as follows:

$$\tilde{p}(x,y) \equiv \frac{1}{N} \times \text{ number of times that } (x,y) \text{ occurs in the training events}$$

Since we regard the expected value of f on training data $\tilde{p}(f)$ as the most useful statistics we get so far, we acknowledge its importance by requiring our model to accord with it. The expected value of f that the model $p(y|x)$ assigns to is calculated as:

$$p(f) \equiv \sum_{x,y} \tilde{p}(x) p(y|x) f(x,y)$$

where, $\tilde{p}(x)$ is the empirical distribution of x on training events.

We impose a constraint for f that the feature expectation $p(f)$ is equal to the empirical feature expectation $\tilde{p}(f)$.

$$p(f) = \tilde{p}(f)$$

Suppose that we are given n feature functions, each constraint is defined as an equation $p(f_i) = \tilde{p}(f_i)$ between f_i expected value in the model $p(f_i)$ and its expected value in the training data $\tilde{p}(f_i)$. Finally, constraint set in Maximum Entropy model is defined as follows:

$$\mathcal{C} \equiv \{p \in \mathcal{P} | p(f_i) = \tilde{p}(f_i) \text{ for } i \in \{1, 2, ..., n\}\}$$

After defining the constraints $\mathcal{C}$ and constructing the dual problem $\Psi(\lambda)$ of the primary maximum entropy problem $p_* = \arg\max_{p \in \mathcal{C}} H(p)$, we transform the constrainted optimization problem to the searching of the real-valued vector $\{\lambda_1, \lambda_2, ..., \lambda_n\}$ which maximizes the likelihood $\Psi(\lambda)$.

$$\begin{aligned} \Psi(\lambda) &= \sum_{x,y} \tilde{p}(x, y) \log p(y|x) \\ &= -\sum_{x} \tilde{p}(x) \log Z_\lambda(x) + \sum_{i} \lambda_i \tilde{p}(f_i) \end{aligned}$$

where, $Z_\lambda(x)$ is a normalizing constant determined by the requirement that $\sum_y p_\lambda(y|x) = 1$ for all x:

$$Z_\lambda(x) = \sum_y \exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

Once we find the optimal $\lambda^* = \arg\max_\lambda \Psi(\lambda)$, the object function can be explicitly calculated as follows:

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)} \exp\left(\sum_i \lambda_i f_i(x, y)\right)$$

Next section, we will discuss how to search the optimal λ^*.

6.1.3 Parameter Estimation

Generalized iterative Scaling (GIS) is an optimization method specifically tailored to the maximum entropy problem by (Darroch and Ratcliff, 1972). This algorithm is applicable whenever the feature functions $f_i(x, y)$ are nonnegative. Furthermore, the GIS procedure requires the constraints that the sum of all feature values remains a constant C for each event:

$$\sum_i f_i(x, y) = C$$

If it is not the case, choose C to be

$$C = \max_{x,y} \sum_{i=1}^{n} f_i(x, y)$$

and add a correction feature f_l, where l = n + 1, such that

$$f_l(x) = C - \sum_{i=1}^{n} f_i(x)$$

The GIS algorithm is briefly described in the following:

---

1. Start with $\lambda_i = 0$ for all $i \in \{1, 2, ..., n\}$
2. Do for each $i \in \{1, 2, ..., n\}$
   (a) Let $\Delta\lambda_i$ be the solution to
   $$\Delta\lambda_i = \frac{1}{C} \log \frac{\tilde{p}(f_i)}{p_\lambda(f_i)}$$
   (b) Update the value of λ_i according to: $\lambda_i \leftarrow \lambda_i + \Delta\lambda_i$
3. Go to step 2 until all the λ_i have converged

6.1.4 Gaussian Prior Smoothing

Since we regard Maximum Entropy modeling as maximum likelihood training for exponential model, like other maximum likelihood methods, it is prone to overfitting of training data. Therefore, we have to consider how to smooth our model. Many smoothing algorithms were developed in past years. (Chen and Rosenfeld, 1999) gave a comprehensive survey of maximum entropy smoothing methods, such as interpolated smoothing models (Jelinek-mercer smoothing and Witten-Bell smoothing) and backed-off smoothing models(Katz smoothing, absolute discounting and Kneser-Ney smoothing) and stated that the ME smoothing algorithm proposed by (Lafferty, 1997) performed better than all other algorithms. It uses a Gaussian prior on model parameters and selects maximum posteriori instead of maximum likelihood parameter values. In this section, we briefly discuss how the Gaussian prior smoothing works for Maximum Entropy model and leave proofs to the references.

Recall that maximum entropy model is to search the optimal λ^* that maximize the log-likelihood $\Psi(\lambda)$ of the training data.

$$\Psi(\lambda) = \sum_{x,y} \tilde{p}(x,y) \log p(y|x)$$

With the Gaussian prior, which we take to have diagonal covariance, the function $\Psi(\lambda)$ turns to be:

$$\begin{aligned} \Psi'(\lambda) &= \Psi(\lambda) + \sum_{i=1}^{n} \log \left(\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{\lambda_i^2}{2\sigma_i^2} \right) \right) \\ &= \Psi(\lambda) - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} + \text{const} \end{aligned}$$

In the above formular, there are two terms. The former one $\Psi(\lambda)$ prefers the models similar to the training data and more uniform. The latter one $\sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2}$ penalizes the models that have many large λ_i values.

The Gaussian prior algorithm adds little computation to existing maximum entropy training. The original update of each λ_i is to take $\lambda_i^{(t+1)} \leftarrow \lambda_i^{(t)} + \Delta\lambda_i^{(t)}$, where

$\Delta\lambda_i^{(t)}$ satisfies the equation.

$$\tilde{p}(f_i) = p_\lambda(f_i)\exp(\Delta\lambda_i C)$$

With the Gaussian prior, the equation is replaced with

$$\tilde{p}(f_i) = p_\lambda(f_i)\exp(\Delta\lambda_i C) + \frac{\lambda_i + \Delta\lambda_i}{\sigma_i^2}$$

A simple and effective way to search $\Delta\lambda_i$ is Newton's method. It computes the solution α_* of an equation $g(\alpha_*) = 0$ iteratively by the recurrence

$$\alpha_{n+1} = \alpha_n - \frac{g(\alpha_n)}{g^{'}(\alpha_n)}$$

This method is also used in Improve Iterative Scaling (Pietra, Pietra, and Lafferty, 1997) for parameter estimation in Maximum Entropy model.

6.2 Answer Extraction Model

Considering the advantage of Maximum Entropy model, we apply it to question answering. Given a question q and a set of answer candidates $\{ac_1, ac_2, ...ac_N\}$, the task is to select the best answer ac^* from the answer candidate set $ac^* \in \{ac_1, ac_2, ...ac_N\}$. It can be viewed as a classification problem and a ranking problem respectively under Maximum Entropy mechanism.

6.2.1 Answer Candidate Classification

In the classification view, we present each $\langle q, ac\rangle$ pair to the model which classifies it as either correct (true) or incorrect (false) based on evidence (features). In this case, we model $p(c|q, ac)$, where, $c = \{true, false\}$ signifies the correctness of the answer candidate ac with respect to the question q.

Give N answer candidates $\{ac_1, ac_2, ...ac_N\}$ for a question q, the probability $p(c|q, ac)$,($ac \in \{ac_1, ac_2, ...ac_N\}$) for each $\langle q, ac\rangle$ pair is modeled independently of

other such pairs. Thus, the N pairs are presented to the classifier as independent events. Once each probability $p(c|q, ac)$ is computed, the system will select the best answer ac^* based on the following decision rule:

$$ac^* = \arg \max_{ac \in \{ac_1, ac_2, ...ac_N\}} p(true|q, ac)$$

The above decision rule requires the comparison of the probabilities $p(true|q, ac)$. However, the probabilities are modeled as independent events in the classifier and hence the training criterion doesn't make them directly comparable.

When using Maximum Entropy to model the classification problem, we firstly define M feature functions $f_m(q, ac)$, $(m = 1, 2, ...M)$ to effectively characterize the task. The feature functions will be detailed described in Section 6.3. After estimating the parameters $\lambda_{m,c}$ for each feature function $f_m(q, ac)$ on training data, the probability $p(c|q, ac)$ will be calculated as follows:

$$p(c|q, ac) = \frac{\exp\left[\sum_{m=1}^{M} \lambda_{m,c} f_m(q, ac)\right]}{\sum_{c' \in \{true, false\}} \exp\left[\sum_{m=1}^{M} \lambda_{m,c'} f_m(q, ac)\right]}$$

where, $\lambda_{m,c}$, $(m = 1, ..., M;\ c = \{true, false\})$ are the model parameters which are trained with Generalized iterative Scaling(GIS) algorithm (Section 6.1.3) and smoothed with Gaussian Prior Smoothing algorithm (Section 6.1.4).

6.2.2 Answer Candidate Ranking

In the ranking view, we directly model the probability of an answer candidate ac for a question q in answer candidate set $\{ac_1, ac_2, ..., ac_N\}$. The model aims to predicate $p(ac|q, \{ac_1, ac_2, ..., ac_N\})$. This view requires the following decision rule to select the most promising answer:

$$ac^* = \arg \max_{ac \in \{ac_1, ac_2, ...ac_N\}} p(ac|q, \{ac_1, ac_2, ..., ac_N\})$$

Table 6.1: Comparison between classification model and ranking model. Q is the number of questions; N is the number of answer candidates for a question; M is the number of feature functions.

	# Events	# Classes	# Parameters
Classification	$Q \times N$	2	$2M$
Ranking	Q	N	M

Comparing with the classification model, the ranking model makes the probability $p(ac|q, \{ac_1, ac_2, ..., ac_N\})$ directly comparable against each other, by incorporating it into the training criterion.

Using the same feature functions $f_m(q, ac)$, $(m = 1, 2, ...M)$ as the classification task, the probability $p(ac|q, \{ac_1, ac_2, ..., ac_N\})$ will be calculated as follows:

$$p(ac|q, \{ac_1, ac_2, ..., ac_N\}) = \frac{\exp\left[\sum_{m=1}^{M} \lambda_m f_m(q, ac)\right]}{\sum_{ac' \in \{ac_1, ac_2, ..., ac_N\}} \exp\left[\sum_{m=1}^{M} \lambda_m f_m(q, ac')\right]}$$

Where, λ_m, $(m = 1, ..., M)$ are the model parameters. Note that the parameters are defined as λ_m in the ranking model, whereas as $\lambda_{m,c}$ in the classification model. This is because in the classification model, each feature function $f_m(q, ac)$ has different weights associated with different classes ($\lambda_{m,true}$ and $\lambda_{m,false}$ respectively). Therefore, the classification model has as twice parameters as the ranking model.

Another difference between the models occurs in event construction. Suppose there are Q questions in training data and each question has N answer candidates, the classification model will handle $Q \times A$ events and two classes ($true$ and $false$) per event while the ranking model will have Q event and N classes $(ac_1, ..., ac_N)$ per event. So the event space of the classification model is much larger than that of the ranking model. Table 6.1 summarizes the difference between the classification model and the ranking model.

6.3 Features

Surface Features We incorporate four types of surface features into Maximum Entropy model.

- **Expected Answer Type Matching Features**: If the semantic category of answer candidate accords with the expected answer type (EAT) of question, EAT feature fires. The identification of question expected answer type was discussed in Section 3.1.2 and the recognition of answer candidate semantic category was discussed in Section 3.4.

- **Orthographic Features**: They capture the surface format of answer candidate, such as capitalizations, digits and lengths, etc. We expect to judge what a proper answer looks like from word format point of view since the semantic category of answer candidate naturally might not be correctly recognized all the time.

- **POS Features**: For certain question type, if the words in answer candidate belong to certain POS type, one POS feature fires. It is also expected to backup the fail of semantic category recognition.

- **Surface Pattern Matching**: Considering that there are questions with very high frequency to be asked in TREC, we build question patterns to map high frequent questions to classes and extract answers for the question classes using answer patterns. Surface pattern matching is discussed in Section 3.1.4. Once question and answer pattern matching succeeds, one feature of suface pattern matching fires.

Table 6.2 lists some examples of surface features. All of them are binary features. In addition, many other features, such as the answer candidate frequency, can be extracted based on the Sentence Retrieval output and are thought as an indicative evidence for the Answer Extraction (Ittycheriah and Roukos, 2002). However, in this thesis, we

Table 6.2: Surface Features

Features	Examples	Explanation
EAT matching	EAT_DAT	ac type matchs the EAT (DATE) of question
	EAT_PERSON	ac type matchs the EAT (PERSON) of question
	EAT_DISTANCE	ac type matchs the EAT (DISTANCE) of question
Orthographic	SSEQ_Q	ac is a subsequence of question
	CAP_EAT_LOC	ac is capitalized and the EAT is LOCATION
	LNGlt3_EAT_PER	ac length is less than 3 and the EAT is PERSON
POS	CD_EAT_NUM	syn. tag of ac is CD and the EAT is NUMBER
	NNP_EAT_PER	syn. tag of ac is NNP and the EAT is PERSON
Surface Pattern Matching	SUR_PTN	question and answer pattern matching succeeds

are to focus on the Answer Extraction Module independently, so we do not incorporate such features in the current model.

Dependency Relation Features The extraction of dependency relation information is discussed in Chapter 4, which consists of dependency relation pattern matching and dependency relation correlation. Both of them are on the basis of the comparison between dependency relations of question and answer sentence.

Dependency relation pattern matching was discussed in Section 4.3. We respectively extract question and answer patterns from training data. These patterns will be used to exact answer for an unseen question. We firstly match the unseen question to the question patterns. Once we get the matched question pattern, the answer patterns evoked by the question pattern will be further matched to pinpoint proper answers. A string kernel, calculating the similarity between two sequences, is used to tolerant answer pattern matching in stead of exact matching. The feature value is set as the answer pattern matching score. The experiments (Section 7.3.2) will evaluate the coverage of the pattern sets and the performance of the two pattern matching methods.

Q: What party led Australia from 1983 to 1996?

Target:	party
Topic:	Australia
Constraint:	1983; 1996
Verb:	lead

Figure 6.1: Examples of question phrase types

Dependency relation correlation was discussed in Section 4.4. Dependency relation paths in question are firstly paired with paths in answer sentence according to question key word mapping. Then a dynamic time warping algorithm is applied to align the paired relation paths and calculate their correlation. The correlation of two paths relied on the correlation of individual relations which are statistically estimated from training data. Finally, the correlation score is used as feature value. Two facts are considered to affect relation path comparison: **question phrase type** and **path length**. For each question, we divide question phrases into four types: target, topic, constraint and verb. Figure 6.1 shows an example of each question phrase type.

- **Target** is a kind of word which indicates the expected answer type of a question, such as "*party*" in "*What party led Australia from 1983 to 1996?*".
- **Topic** is the event/person that a question is talking about, such as the word "*Australia*" in the above example question. Intuitively, it is the most important phrase of a question.
- **Constraint** is the other question phrase except topic, such as "*1983*" and "*1996*".
- **Verb** is the main verb of a question, such as "*lead*".

Furthermore, since shorter path indicates closer relation between two phrases, we discount path matching score by dividing the score by the question path length. Lastly, we

sum the discounted path matching score for each type of question phrases and fire it as a feature, such as

- **Target_Ptn=p**, where "*p*" is the pattern matching score for question *target* words.
- **Topic_Cor=c**, where "*c*" is the path correlation value for question *topic* words.

Totally, there are 8 dependency relation features to fire for each answer candidate, including Target_Ptn, Topic_Ptn, Constraint_Ptn, Verb_Ptn, Target_Cor, Topic_Cor, Constraint_Cor and Verb_Cor.

Semantic Structure Matching Features FrameNet-style semantic role information was discussed in Chapter 5. We present an automatic method for semantic role assignment which is based on the comparison of dependency relation paths attested in FrameNet annotations and raw texts. We formalize the search for an optimal role assignment as an optimization problem in a bipartite graph. This formalization allows us to find an exact globally optimal solution. In addition, the soft labeling is enabled in the optimization which goes some way towards addressing coverage problem related with FrameNet. Finally, semantic structure matching is formulated as a graph matching problem. The matching score is used as the value of semantic structure matching feature.

Finally, the ME-based ranking model incorporate the surface features, dependency relation features and semantic structure matching features to rank answer candidates.

Chapter 7

Evaluation

7.1 Experiment Setting

We apply the Answer Extraction (AE) module to the TREC QA task. The goal of the AE module is to identify exact answers for questions from candidate sentence collections. The AE module accepts questions and their relevant sentences as input and returns a set of ranked answers as output. The performance of the AE module is evaluated using the mean reciprocal rank (MRR). Furthermore, we also list the percentages of the correct answers in terms of the top 1, top5 and top10 answers returned.

Here we summarize the processing steps of the Alyssa system preceding the AE module. A user question firstly undergoes the Question Processing Module (Section 3.1), a phase in which several steps are involved independently. The expected answer type of the question is identified and a series of linguistic analysis is carried out, including key phrase extraction and extension, surface pattern matching, syntactic and semantic structure generation. Moreover, a query is constructed from the question and is run against the Document Retrieval Module (Section 3.2) on the Aquaint indexes. The Sentence Retrieval Module (Section 3.3) is implemented based on language modeling techniques, and the extracted relevant sentences undergo further linguistic analysis in the Sentence Anno-

tation Module (Section 3.4) including named entity recognition, noun phrase chunking and dependency parsing. Before the sentences are fed into the AE Module, the Question Phrase Mapping Module (Section 3.5.1) is applied to detect which is the word or phrase overlapping between the question and the relevant sentences by considering morphological, format, semantic and proper name variations of individual words. Finally, all the information supplying the evidence of proper answer is integrated into the AE Module. We assume that all the noun phrases attested in the relevant sentences are answer candidates. A Maximum Entropy-based model is applied to rank the answer candidates according to their semantic category, surface, syntactic and semantic evidence, proposed in Chapter 4 and 5. The top-ranked answer candidates are finally passed to the Answer Validation Module (Section 3.6) which is based on the evidence from the Web and structured databases.

All of our experiments are performed on the TREC99–07 factoid questions. We exclude NIL questions since TREC doesn't supply proper answers for them. We train the AE module on the questions of TREC99-03 and test it on the questions of TREC04-07. The test is separately conducted on the TREC years in order to make our results comparable with other related work. The documents and sentences for TREC99-01 questions are from the TREC Corpus[1] and for TREC02-07 questions are from the AQUAINT Corpus[2] The following steps are used to generate the gold standard data set:

1. Retrieve relevant documents for each question according to TREC judgments;

2. Select the sentences containing proper answers and at least one question key word from the relevant documents;

3. Manually check the sentences, remove the unsupported ones and tag proper answers in the rest sentences according to TREC answer patterns.

[1]TREC Corpus includes the AP newswire, the Wall Street Journal, the San Jose Mercury News, the Financial Times, the Los Angeles Times and the Foreign Broadcast Information Service

[2]Aquaint Corpus consists of English newswire texts and is used as the main document collection in official TREC evaluations.

Table 7.1: Statistics of TREC questions

	TREC	# Questions	# Not-Nil Questions	# Answer Sentences
Training Data	1999	200	193	402
	2000	693	667	2659
	2001	500	433	2628
	2002	500	444	1380
	2003	413	362	1063
	total	2306	2099	8132
Test Data	2004	230	203	698
	2005	362	327	1662
	2006	403	386	1235
	2007	360	344	1001
	total	1355	1260	4596

Table 7.1 shows the statistics of the gold standard data sets. Totally, there are 2099 training questions and 1260 test questions. On average, given a question, there are about 4 relevant sentences containing proper answers in the Corpora. In the other word, there are not many chances to find the proper answers in the Corpora. It indicates that the answer extraction on the Corpora will be much harder than that on the Web where the proper answers might occur in lots of documents and are supported by various contexts.

Obviously the quality of relevant sentence set has strong impact on the performance of the answer extraction. It is meaningless to evaluate the AE module on the questions whose relevant sentences don't contain any proper answers. To our knowledge, most of the existing QA systems lose about half of the questions in the Sentence Retrieval stage. In this thesis, we evaluate the AE module on the following sentence sets corresponding to different quality levels.

- Gold Standard Sentence Set (**GSSet**): It assumes that the Sentence Retrieval Module gets 100% precision and 100% recall. This set has the best quality. Each sentence in the *GSSet* contains a proper answer. Although this setup is somewhat idealized, it allows us to evaluate the AE Module in more detailed.

- Sentence Retrieval Output (**SRSet**): It uses the top N ranked sentences returned by a real Sentence Retrieval Module. This set has the worst quality. A portion of questions can't be answered using the sentence set. Section 7.2 will show how many questions are lost in the Document Retrieval and Sentence Retrieval Modules respectively in the Alyssa system. The value will be regarded as the up-bound of the AE Module.

- **GSSRSet**: We add the sentence retrieval output to the gold standard sentence set (*GSSRSet* = *GSSet* ∪ *SRSet*). The *GSSRSet* thus includes at least one correct sentence (100% recall) per question as well as wrong/unsupported sentences. Using this setting, we can make sure it is the fault of the AE module when an answer is not found. And meanwhile, we may also test the capability of the AE module on handling noisy data.

7.2 Performance of Document and Sentence Retrieval

In the pipeline structure of a QA system, performance of one module will be confined by performances of previous modules. Before evaluating the AE Module, we firstly investigate the performances of its previous modules: the Document Retrieval Module and the Sentence Retrieval Module. The questions failed in the two modules will totally lose the chance to be correctly answered at last. The more documents or sentences are returned, the larger possibility proper answers are contained. On the other hand, this will also increase the difficulty of the AE module.

Table 7.2 shows the performance of the document retrieval according to various N values. The parameter N indicates the number of documents the module returns for a question. Since the performance doesn't increase much more when $N > 60$, we pass the 60 top ranked documents to the Sentence Retrieval Module. Table 7.3 shows the performance of the sentence retrieval according to various M values. The parameter M

Table 7.2: Perfomance of document retrieval on TREC04-07 questions: Number of questions of which the N top ranked documents contain proper answers; numbers in parentheses are accuracy.

	TREC04		TREC05		TREC06		TREC07		overall	
Total	203		327		386		344		1260	
Top 100	190	(93.6)	305	(93.3)	356	(92.2)	292	(84.9)	1143	(90.7)
Top 80	188	(92.6)	299	(91.4)	352	(91.2)	287	(83.4)	1126	(89.4)
Top 60	185	(91.1)	294	(89.9)	332	(86.0)	276	(80.2)	1087	(86.3)
Top 40	181	(89.2)	281	(85.9)	322	(83.4)	271	(78.8)	1055	(83.7)
Top 20	167	(82.3)	260	(79.5)	295	(76.4)	231	(67.2)	953	(75.6)

indicates the number of sentences the module returns for a question. We set the parameter $M = 100$. These top ranked sentences will be further fed into the AE Module.

7.3 Syntactic Methods

7.3.1 Overall Performance

In order to evaluate the effectiveness of the two syntactic methods: Dependency Relation Pattern Method (Section 4.3) and Dependency Relation Path Correlation Method (Section 4.4) in the answer extraction, we compare them with the state of the art. The evidence captured by a syntactic method is incorporated into a Maximum Entropy-based ranking model along with the other common surface features, as described in Section 6.3. The Maximum Entropy model is then trained on TREC99-03 questions and ranks answer candidates in test stage. We evaluate the MRR, Top1, Top5 and Top10 performances of the ranked answers. Totally, six answer extraction methods are evaluated for comparison:

- **Density**: Density-based method is used as baseline. It prefers answer candidates which have shorter surface distance to question phrases.

Table 7.3: Perfomance of sentence retrieval on TREC04-07 questions: Number of questions of which the M top ranked sentences contain proper answers; numbers in parentheses are accuracy.

	TREC04		TREC05		TREC06		TREC07		overall	
Total	203		327		386		344		1260	
Top 200	171	(84.2)	282	(86.2)	316	(81.9)	231	(67.2)	1000	(79.4)
Top 180	171	(84.2)	281	(85.9)	314	(81.3)	230	(66.9)	996	(79.0)
Top 160	170	(83.7)	277	(84.7)	313	(81.1)	230	(66.9)	990	(78.6)
Top 140	170	(83.7)	275	(84.1)	308	(79.8)	230	(66.9)	983	(78.0)
Top 120	168	(82.8)	271	(82.9)	303	(78.5)	226	(65.7)	968	(76.8)
Top 100	166	(81.8)	266	(81.3)	295	(76.4)	225	(65.4)	952	(75.6)
Top 80	166	(81.8)	258	(78.9)	290	(75.1)	220	(64.0)	934	(74.1)
Top 60	160	(78.8)	252	(77.1)	274	(71.0)	208	(60.5)	894	(71.0)
Top 40	154	(75.9)	241	(73.7)	255	(66.1)	201	(58.4)	851	(67.5)
Top 20	140	(69.0)	213	(65.1)	217	(56.2)	180	(52.3)	750	(59.5)

- Syntactic Distance (**SynDist**): *SynDist* considers the length of dependency relation path from answer candidate to question key phrase. The shorter a relation path is, the closer relationship the words of the path have.

- Strict Syntactic Structure Matching (**StrictMatch**): Strict relation matching proposed by (Tanev, Kouylekov, and Magnini, 2004; Wu et al., 2005) is on the basis of the assumption that the more common individual relations two syntactic structures share, the more similar they are. We implement it by adapting the relation correlation measure in Section 4.4.3. In stead of learning invidual relation correlations during training, we predefine them as: $Cor(r_1, r_2) = 1$ if $r_1 = r_2$; 0, otherwise.

- Approximate Syntactic Structure Matching (**ApprMatch**): Approximate relation matching (Cui et al., 2004) aligns two relation paths using fuzzy matching and ranks answer candidates according to the sum of all path similarities. This method was briefly described in Section 4.2.

Table 7.4: Performance of syntactic methods on *GSSet*

TREC		Density	SynDist	StricMatch	ApprMatch	DepPtn	DepCor
04	MRR	82 (40.4)	88 (43.3)	110 (54.2)	117 (57.6)	135 (66.5)	136 (67.0)
	Top1	73 (36.0)	76 (37.4)	98 (48.3)	106 (52.2)	128 (63.1)	126 (62.1)
	Top5	112 (55.2)	114 (56.2)	134 (66.0)	140 (69.0)	145 (71.4)	150 (73.9)
	Top10	117 (57.6)	118 (58.1)	139 (68.5)	147 (72.4)	156 (76.8)	160 (78.8)
05	MRR	111 (33.9)	130 (39.8)	172 (52.6)	176 (53.8)	222 (67.9)	197 (60.2)
	Top1	101 (30.9)	114 (34.9)	156 (47.7)	157 (48.0)	205 (62.7)	174 (53.2)
	Top5	131 (40.1)	164 (50.2)	198 (60.6)	206 (63.0)	235 (71.9)	245 (74.9)
	Top10	147 (44.3)	170 (52.0)	201 (61.5)	219 (67.0)	245 (74.9)	249 (76.1)
06	MRR	133 (34.5)	142 (36.8)	177 (45.9)	186 (48.2)	234 (60.6)	221 (57.3)
	Top1	118 (30.6)	122 (31.6)	162 (42.0)	166 (43.0)	224 (58.0)	201 (52.1)
	Top5	164 (42.5)	182 (47.2)	208 (53.9)	216 (56.0)	258 (66.8)	258 (66.8)
	Top10	185 (47.9)	198 (51.3)	228 (59.1)	228 (59.1)	263 (68.1)	272 (70.5)
07	MRR	129 (37.5)	133 (38.7)	166 (48.3)	169 (49.1)	200 (58.1)	188 (54.7)
	Top1	106 (30.8)	108 (31.4)	141 (41.0)	139 (40.4)	190 (55.2)	168 (48.8)
	Top5	167 (48.5)	171 (49.7)	207 (60.2)	207 (60.2)	216 (62.8)	215 (62.5)
	Top10	172 (50.0)	187 (54.4)	213 (61.9)	213 (61.9)	231 (67.2)	235 (68.3)
all	MRR	455 (36.1)	493 (39.1)	625 (49.6)	648 (51.4)	791 (62.8)	742 (58.9)
	Top1	398 (31.6)	420 (33.3)	557 (44.2)	568 (45.1)	747 (59.3)	669 (53.1)
	Top5	574 (45.6)	631 (50.1)	747 (59.3)	769 (61.0)	854 (67.8)	868 (68.9)
	Top10	621 (49.3)	673 (53.4)	781 (62.0)	807 (64.0)	895 (71.0)	916 (72.7)

- Dependency Relation Pattern (**DepPtn**): It is the method proposed in Section 4.3. Dependency relation patterns are firstly extracted from training questions and answer sentences. Given a unseen question for testing, the patterns are partially matched using string kernel.

- Dependency Relation Path Correlation Method (**DepCor**): It is the method proposed in Section 4.4. Different from *ApprMatch*, ME-based ranking model is implemented to incorporate path correlations which assign different weights for different paths respectively. Furthermore, phrase mapping score is incorporated into the path correlation measure.

Table 7.5: Performance of syntactic methods on *GSSRSet*

TREC		Density	SynDist	StricMatch	ApprMatch	DepPtn	DepCor
04	MRR	52 (25.6)	64 (31.5)	90 (44.3)	96 (47.3)	132 (65.0)	138 (68.0)
	Top1	40 (19.7)	47 (23.2)	81 (39.9)	88 (43.3)	125 (61.6)	128 (63.1)
	Top5	73 (36.0)	90 (44.3)	105 (51.7)	132 (65.0)	146 (71.9)	151 (74.4)
	Top10	86 (42.4)	95 (46.8)	108 (53.2)	139 (68.5)	156 (76.8)	160 (78.8)
05	MRR	74 (22.6)	95 (29.1)	135 (41.3)	149 (45.6)	218 (66.7)	195 (59.6)
	Top1	54 (16.5)	72 (22.0)	114 (34.9)	128 (39.1)	202 (61.8)	171 (52.3)
	Top5	102 (31.2)	126 (38.5)	167 (51.1)	181 (55.4)	231 (70.6)	248 (75.8)
	Top10	112 (34.3)	132 (40.4)	177 (54.1)	198 (60.6)	245 (74.9)	249 (76.1)
06	MRR	96 (24.9)	106 (27.5)	143 (37.0)	152 (39.4)	233 (60.4)	219 (56.7)
	Top1	69 (17.9)	76 (19.7)	125 (32.4)	131 (33.9)	222 (57.5)	199 (51.6)
	Top5	131 (33.9)	143 (37.0)	170 (44.0)	181 (46.9)	260 (67.4)	255 (66.1)
	Top10	139 (36.0)	158 (40.9)	195 (50.5)	197 (51.0)	264 (68.4)	271 (70.2)
07	MRR	79 (23.0)	92 (26.7)	134 (39.0)	142 (41.3)	203 (59.0)	188 (54.7)
	Top1	60 (17.4)	73 (21.2)	105 (30.5)	110 (32.0)	192 (55.8)	169 (49.1)
	Top5	106 (30.8)	120 (34.9)	175 (50.9)	183 (53.2)	218 (63.4)	214 (62.2)
	Top10	130 (37.8)	132 (38.4)	178 (51.7)	189 (54.9)	231 (67.2)	234 (68.0)
all	MRR	301 (23.9)	357 (28.3)	502 (39.8)	539 (42.8)	786 (62.4)	740 (58.7)
	Top1	223 (17.7)	268 (21.3)	425 (33.7)	457 (36.3)	741 (58.8)	667 (52.9)
	Top5	412 (32.7)	479 (38.0)	617 (49.0)	677 (53.7)	855 (67.9)	868 (68.9)
	Top10	467 (37.1)	517 (41.0)	658 (52.2)	723 (57.4)	896 (71.1)	914 (72.5)

Table 7.6: Performance of syntactic methods on *SRSet*

TREC		Density	SynDist	StricMatch	ApprMatch	DepPtn	DepCor
04	MRR	31 (15.3)	47 (23.2)	58 (28.6)	61 (30.0)	71 (35.0)	72 (35.4)
	Top1	18 (8.9)	26 (12.8)	38 (18.7)	41 (20.2)	53 (26.1)	54 (26.6)
	Top5	47 (23.2)	69 (34.0)	79 (38.9)	79 (38.9)	97 (47.8)	99 (48.8)
	Top10	62 (30.5)	78 (38.4)	82 (40.4)	84 (41.4)	106 (52.1)	108 (53.2)
05	MRR	56 (17.1)	74 (22.6)	89 (27.2)	94 (28.7)	115 (35.1)	110 (33.7)
	Top1	36 (11.0)	43 (13.1)	60 (18.3)	69 (21.1)	80 (24.5)	78 (23.9)
	Top5	93 (21.4)	108 (33.0)	120 (36.7)	123 (37.6)	165 (50.5)	172 (52.6)
	Top10	108 (33.1)	136 (41.6)	141 (43.1)	143 (43.7)	183 (56.0)	190 (58.1)
06	MRR	70 (18.1)	84 (21.8)	100 (25.9)	102 (26.4)	117 (30.2)	113 (29.3)
	Top1	42 (10.9)	56 (14.5)	72 (18.7)	73 (18.9)	89 (23.1)	85 (22.0)
	Top5	102 (26.4)	117 (30.3)	129 (33.4)	130 (33.7)	157 (40.7)	160 (41.5)
	Top10	138 (35.8)	157 (40.7)	167 (43.3)	167 (43.3)	185 (47.9)	189 (49.0)
07	MRR	49 (14.2)	62 (18.0)	74 (21.5)	79 (23.0)	93 (27.1)	92 (26.6)
	Top1	28 (8.1)	40 (11.6)	56 (16.3)	61 (17.7)	67 (19.5)	64 (18.6)
	Top5	71 (20.6)	87 (25.3)	96 (27.9)	97 (28.2)	123 (35.8)	130 (37.8)
	Top10	100 (29.1)	110 (32.0)	113 (32.8)	116 (33.7)	152 (44.2)	153 (44.5)
all	MRR	206 (16.3)	267 (21.2)	321 (25.5)	336 (26.7)	396 (31.4)	387 (30.7)
	Top1	124 (9.8)	165 (13.1)	226 (17.9)	244 (19.4)	289 (22.9)	281 (22.3)
	Top5	313 (24.8)	381 (30.2)	424 (33.7)	429 (34.0)	542 (43.0)	561 (44.5)
	Top10	408 (32.4)	481 (38.2)	503 (39.9)	510 (40.5)	626 (49.7)	640 (50.8)

Table 7.4, 7.5 and 7.6 show the performances of the six methods on different sentence sets. The main observations from the tables are as follows:

1. The performance comparison of the methods are consistent on the different data sets (*GSSet*, *GSSRSet* and *SRSet*) and the different years (04, 05, 06 and 07). We analyze the results on the *GSSet* (Table 7.4) as an example. Actually, the same finding may also be obtained from the *GSSRSet* (Table 7.5) and the *SRSet* (Table 7.6). From Table 7.4, we find that the syntactic methods *SynDist*, *StrictMatch*, *ApprMatch*, *DepPtn* and *DepCor* significantly improve MRR by 3.0%, 13.5%, 15.3%, 26.7% and 22.8% over the baseline method *Density*. The improvements may benefit from the various exploration of syntactic information. It shows that syntactic evidence indeed helps a lot in the answer extraction. *DepPtn* outperforms all the other syntactic-based methods (*SynDist*, *StrictMatch* and *ApprMatch*) by about 23.7%, 13.2% and 11.4% MRR improvement. *DepCor* outperforms *SynDist*, *StrictMatch* and *ApprMatch*) by about 19.8%, 9.3% and 7.5% MRR improvement. The strict matching (*StrictMatch*) often fails due to the variation of syntactic representations. To some extent, such variation may be captured by the approximate matching (*ApprMatch*) using more relaxed matching. However, the *ApprMatch* still follows the assumption that two sequences are more similar if only they share more common individual relations. The dependency relation pattern method (*DepPtn*) further solves this problem with pattern matching. Answer patterns represent a set of elementary syntactic formats of expected answers. A partial matching with string kernel is performed to find more potential syntactic expressions of proper answers. In stead of requiring common individual relations in the matching, the dependency relation correlation method (*DepCor*) further estimates correlations of individual relations between questions and answer sentences using a MI-based statistical method. Then a DTW algorithm is applied to align two relation paths according to the correlation scores. Since the *DepCor* doesn't follow the

strict assumption any more, we expect it will achieve better coverage without large reduction of precision. The experiment on the *GSSet* shows that the *DepPtn*(62.8% MRR and 59.3% Top1) performs better than the *DepCor* (58.9% MRR and 53.1% Top1) in MRR and Top1 evaluation. However, when Top5 or Top10 answer candidates are returned, the *DepCor* (68.9% Top5 and 72.7% Top10) outperforms the *DepPtn* (67.8% Top5 and 71.0% Top10). It indicates that the *DepPtn* contributes more to the precision while the *DepCor* to the recall. It gives us an illumination that incorporating both of them might achieve better performance. This result will be shown in Table 7.11 of Section 7.4.4.

2. A surprising finding is that our methods (*DepPtn* and *DepCor*) achieve quite similar performance on the *GSSet* and the *GSSRSet* although the *GSSRSet* contains much more noisy sentences. It proves the great capability of our methods on the identification of supportive sentences. Unfortunately, the *Density* and the *SynDist* methods can't perform on the *GSSRSet* as well as the *GSSet*. Table 7.6 further shows the performance of the answer extraction on the real data set (*SRSet*). It is much worse than those on the *GSSet* and the *GSSRSet*. One obvious reason is that the sentence retrieval fails to get the sentences containing proper answers. As show in Table 7.3, we evaluate the sentence retrieval module by judging whether a sentence contains a proper answer phrase or not. It indicates that 81.8% on TREC2004, 81.3% on TREC2005, 76.4% on TREC2006 and 65.4% on TREC2007 are the up-bound performance of the answer extraction. However, even if a returned sentence contains proper answer, we can't automatically judge whether it is supportive. For those unsupportive sentences which might not have any similarity with questions, our methods still won't get any chance to pinpoint proper answers. For this reason, we think the real up-bound will be even below the above numbers.

3. The tables also show that there exists a large gap (about 10%) between Top1 and

Table 7.7: Coverage of question and answer pattern matching on *SRSet*

Data	Total	Exact QPtn Match		Exact APtn Match		Partial APtn Match	
TREC04	203	185	(91.2)	105	(51.9)	150	(73.8)
TREC05	327	286	(87.5)	159	(48.6)	249	(76.3)
TREC06	386	345	(89.3)	206	(53.3)	292	(75.7)
TREC07	344	312	(90.8)	161	(46.7)	248	(72.1)
overall	1260	1128	(89.5)	631	(50.1)	939	(74.5)

Top5 performance. It indicates that although our methods can accurately identify a small set of potential answer candidates, it is weak to decide which the best one is. The reason might be that our methods only focus on local evidence, such as the evidence extracted from a single sentence, without using any additional resources. This finding motivates us to construct a web validation module as a complement of the answer extraction by further incorporating answer redundancy evidence from Web data. We re-rank the Top 5 or 10 answer candidates using the web validation. The final results will be shown in Section 7.5.

7.3.2 Coverage and performance of dependency relation patterns

Since the *DepPtn* achieves the best MRR among all of the methods as shown in Table 7.4, 7.5 and 7.6, we try to find how many questions can actually benefit from the *DepPtn*. We firstly investigate the coverage of the question and answer patterns respectively. Next, we attempt to judge whether partial answer pattern matching is more effective than exact matching.

Question patterns represent the most frequent syntactic structures in questions. Each question pattern corresponds to a set of answer patterns which indicates basic syntactic structures of proper answers for the question pattern. The question and answer patterns are constructed from training data with the procedure of Section 4.3.1. Given a unseen question, firstly, we match it to the question patterns. Once it matches, we fur-

Table 7.8: Performance of dependency relation pattern matching methods on *SRSet*

TREC		Exact APtn Match		Partial APtn Match	
04	MRR	62	(30.5)	71	(35.0)
	Top1	46	(22.7)	53	(26.1)
	Top5	82	(40.4)	97	(47.8)
	Top10	84	(41.4)	106	(52.2)
05	MRR	96	(29.4)	115	(35.1)
	Top1	72	(22.0)	80	(24.5)
	Top5	126	(38.5)	165	(50.5)
	Top10	145	(44.3)	183	(56.0)
06	MRR	104	(26.9)	117	(30.2)
	Top1	78	(20.2)	89	(23.1)
	Top5	135	(35.0)	157	(40.7)
	Top10	170	(44.0)	185	(47.9)
07	MRR	82	(23.8)	93	(27.1)
	Top1	61	(17.7)	67	(19.5)
	Top5	113	(32.8)	123	(35.8)
	Top10	122	(35.5)	152	(44.2)
all	MRR	344	(27.3)	396	(31.4)
	Top1	257	(20.4)	289	(22.9)
	Top5	456	(36.2)	542	(43.0)
	Top10	521	(41.3)	626	(49.7)

ther pinpoint answers by matching the corresponding answer patterns. Since syntactic structures of TREC factoid questions are relatively simple, we decide to conduct exact question pattern matching (**Exact QPtn Match**). As shown in Table 7.7, the exact question pattern matching may cover 89.5% unseen questions. Comparing with the questions, syntactic structures of answer sentences are diversiform. Moreover, the answer patterns may not be sufficient enough to cover all of unseen cases since we construct them from a limited training data set. Therefore, exact answer pattern matching (**Exact APtn Match**) might suffer from data sparseness. In Section 4.3.3, we propose a string kernel-based method to partially match the answer patterns (**Partial APtn Match**). As shown in Table 7.7, the *Exact APtn Match* only has 50.1% coverage while the *Partial APtn Match* may increase the coverage to 74.5%.

Furthermore, we evaluate the effectiveness of the two answer pattern matching methods: *Exact APtn Match* and *Partial APtn Match* on the real data set *SRSet*. As shown in Table 7.8, the *Partial APtn Match* outperforms the *Exact APtn Match* by 4.1% MRR. It might be due to the higher coverage of the *Partial APtn Match*.

7.4 Semantic Role Method

7.4.1 FrameNet

FrameNet is a lexicon resource for English (Baker, Fillmore, and Lowe, 1998) based on Frame Semantics. In FrameNet, a scenario is abstractly represented as a frame, such as the frame *Becoming_aware* describes the situation that *"COGNIZER adds some PHENOMENON to their model of the world"*. A predicate, which usually is a verb, noun or adjective, is referred to a frame if it describes the situation of the frame. For example, the predicates *detect.v*, *discover.v*, *discovery*, *find.v*, *notice* are associated with the frame *Becoming_aware* in FrameNet. It may cope with lexical variations between questions and answer sentences. Frame elements (FEs) are further defined in a frame

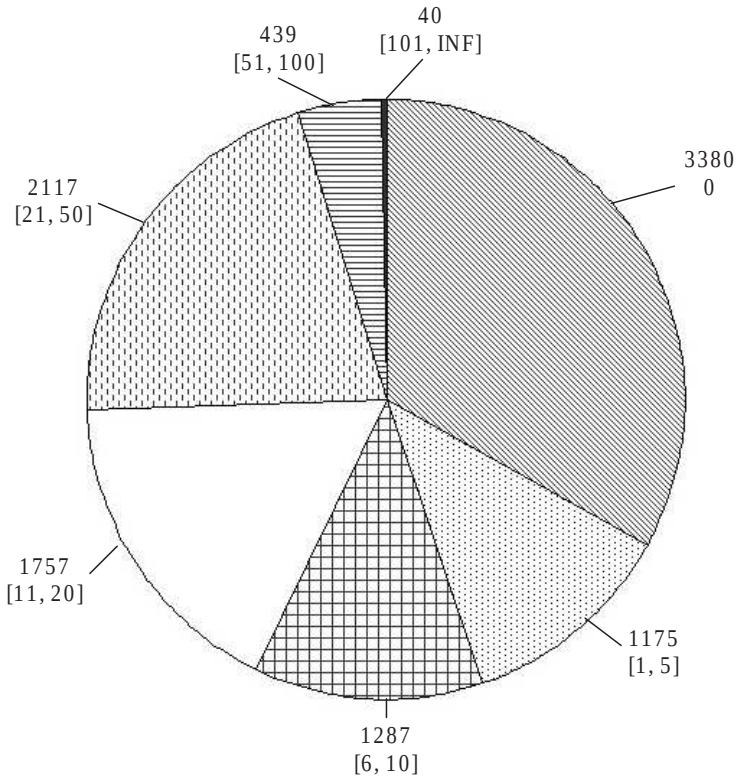


Figure 7.1: Distribution of numbers of predicates and annotated sentences; each subpie lists the number of predicates (above) with their corresponding number of annotated sentences (below)

as semantic roles of predicates of the frame. Therefore, they are frame specific. Each predicate is supported by a set of sentences with FE annotations. We find that the same FE of a frame may have various syntactic realizations. For example, FE *COGNIZER* frequently has *subj* dependency relation with predicate *discover.v* and *gen* relation with *discovery.n*. Therefore, constructing semantic structure on syntactic level may unify the syntactic variation between questions and answer sentences.

We use the FrameNet V1.3 lexical database. It contains 10,195 predicates grouped into 795 semantic frames and 141,238 annotated sentences. Figure 7.1 shows the number of annotated sentences available for different predicates. As can be seen, there are 3,380 predicates with no annotated sentences and 1,175 predicates with less than 5 annotated sentences. It might bring low coverage problem when generating semantic structures. Since FrameNet is still in construction, frames / predicates / annotated sentences are not complete. All FrameNet sentences, questions, and answer sentences were parsed using MiniPar (Lin, 1994), a robust dependency parser.

7.4.2 Baseline

We compare the semantic role method proposed in Chapter 5 to the answer extraction method that exploits solely syntactic information without making use of FrameNet or any other type of role semantic annotations. Section 7.3 evaluates the syntactic methods (*DepPtn* and *DepCor*) separately. The experiments show that the *DepPtn* contributes more to the precision while the *DepCor* to the recall. So we hope their combination will further enhance the performance. Here, we will evaluate the performance of the combination and regard it as the first baseline for the semantic role method.

Our second baseline employs Shalmaneser (Erk and Padó, 2006), a publicly available shallow semantic parser[3], for the role labeling task instead of the graph-based model

[3]The software is available from `http://www.coli.uni-saarland.de/projects/salsa/shal/`.

presented in Section 5.3.2. The software is trained on the FrameNet annotated sentences using a standard feature set (see Carreras and Màrquez (2005) for details). It may automatically identify semantic frames and assign semantic roles for free texts. We use Shalmaneser to parse questions and answer sentences. The parser makes hard decisions about the presence or absence of a semantic role. Unfortunately, this prevents us from using our method for semantic structure matching (see Section 5.3.3) which assumes a soft labeling. We therefore come up with a simple matching strategy suitable for the parser's output. For question and answer sentences matching in their frame assignment, phrases bearing the same semantic role as the EAP are considered answer candidates. The latter are ranked according to word overlap (i.e., identical phrases are ranked higher than phrases with no overlap at all).

7.4.3 FrameNet Coverage

As mentioned in Section 5.3.2 we extract dependency relation paths by traversing a dependency tree from frame element nodes to predicate nodes. We used all dependency relations provided by MiniPar (42 in total). In order to increase coverage, we combine all relation paths for predicates that evoke the same frame and are labeled with the same POS tag. For example, "*found*" and "*establish*" are both instances of the frame *Intentionally_create* but the database does not have any annotated sentences for "*found.v*". In default of not assigning any role labels for "*found.v*", our model employs the relation paths for the semantically related "*establish.v*".

Before reporting the performance, we firstly try to answer the following questions: (1) How does the incompleteness of FrameNet impact QA performance on the TREC data sets? In particular, we want to examine whether there are questions for which in principle no answer can be found due to missing frame entries or missing annotated sentences. (2) Are all questions and their corresponding answers amenable to a FrameNet-style analysis? In other words, we want to assess whether questions and an-

swers often evoke the same or related frames (with similar roles). This is a prerequisite for semantic structure matching and ultimate answer extraction.

Our results are summarized in Table 7.9 which records the number of questions to be answered for the TREC04–07 datasets (Total). We also give information regarding the number of questions which are in principle *unanswerable* with a FrameNet-style semantic role analysis.

The column **NoFrame** shows the number of questions which don't have an appropriate frame or predicate in the database. For example, there is currently no predicate entry for "*sponsor*" or "*sink*" (e.g., "*Q: Who is the sponsor of the International Criminal Court?*" and "*Q: What date did the Lusitania sink?*").

The column **NoAnnot** refers to questions for which no semantic role labeling is possible because annotated sentences for relevant predicates are missing. For instance, there are no annotations for "*win*" (e.g., "*Q: What division did Floyd Patterson win?*"), for "*hit*" (e.g., "*Q: What was the Beatles' first number one hit?*") or for "*visit*" (e.g., "*S: The Hale-Bopp comet's visit to the earth - just once every 4,200 years, will give...*").

This problem is not specific to our method which admittedly relies on FrameNet annotations for performing the semantic role assignment (see Section 5.3.2). Shallow semantic parsers trained on FrameNet would also have trouble assigning roles to predicates for which no data is available.

Finally, the column **NoMatch** reports the number of questions which cannot be answered due to frame mismatches. Consider "*Q: What does AARP stand for?*" whose answer is found in "*S: The American Association of Retired Persons (AARP) qualify for discounts...*". The answer and the question evoke different frames; in fact here a semantic role analysis is not relevant for locating the right answer.

As can be seen NoMatch cases are by far the most frequent. The number of questions remaining after excluding NoFrame, NoAnnot, and NoMatch are shown under the Rest heading in Table 7.9.

Table 7.9: Number of questions which cannot be answered using a FrameNet style semantic analysis; numbers in parentheses are percentages of Total (NoFrame: frames or predicates are missing; NoAnnot: annotated sentences are missing, NoMatch: questions and candidate answers evoke different frames.

Data	Total	NoFrame		NoAnnot		NoMatch		Rest	
TREC04	203	47	(23.2)	14	(6.9)	67	(33.0)	75	(36.9)
TREC05	327	70	(21.4)	23	(7.0)	145	(44.3)	89	(27.2)
TREC06	386	85	(21.9)	26	(6.7)	151	(39.2)	124	(32.1)
TREC07	344	76	(22.1)	25	(7.3)	151	(43.9)	92	(26.7)
overall	1260	278	(22.1)	88	(7.0)	514	(40.8)	380	(30.2)

These results indicate that FrameNet-based semantic role analysis applies to approximately 30.2% of the TREC data. This means that an answer extraction module relying solely on FrameNet will have poor performance, since it will be unable to find answers for more than half of the questions being asked. Frame mismatches could be detected relatively easily as well as cases where frames or predicates are absent from the database or lack annotated sentences. Actually, the affected question-answer pairs could be analyzed using more traditional answer extraction methods or a different style of semantic role analysis (e.g., PropBank). However, we leave this to future work. Next, we subsequently report results on questions which meet with none of the above problems (see Rest in Table 7.9) and examine whether our semantic method brings any performance improvements on this limited dataset which is admittedly favorable towards a FrameNet style analysis.

7.4.4 Performance

This section is trying to answer the third question: Do the semantic role method introduced in this thesis bring any performance gains over state-of-the-art shallow semantic parsers or more conventional syntactic QA systems? Recall that our graph-based model is designed especially for the answer extraction task.

Table 7.10: Performance of semantic methods on subset of *SRSet* (see the *Rest* column in Table 7.9).

TREC		DepPtn+DepCor		SemParse		SemMatch	
04	MRR	30	(40.0)	16	(21.3)	37	(49.3)
	Top1	23	(30.7)	11	(14.7)	32	(42.7)
	Top5	39	(52.0)	22	(29.3)	44	(58.7)
	Top10	42	(56.0)	22	(29.3)	44	(58.7)
05	MRR	40	(44.9)	13	(14.6)	46	(51.7)
	Top1	34	(38.2)	10	(11.2)	41	(46.1)
	Top5	49	(55.1)	19	(21.3)	52	(58.4)
	Top10	56	(62.9)	21	(23.6)	56	(62.9)
06	MRR	43	(34.7)	19	(15.3)	51	(41.1)
	Top1	37	(29.8)	15	(12.1)	50	(40.3)
	Top5	53	(42.7)	22	(17.7)	56	(45.2)
	Top10	63	(50.8)	22	(17.7)	63	(50.8)
07	MRR	32	(34.8)	11	(11.9)	35	(38.0)
	Top1	24	(26.1)	9	(9.8)	31	(33.7)
	Top5	41	(44.6)	16	(17.4)	43	(46.7)
	Top10	43	(46.7)	16	(17.4)	45	(48.9)
all	MRR	145	(38.2)	59	(15.5)	169	(44.5)
	Top1	118	(31.1)	45	(11.8)	154	(40.5)
	Top5	182	(47.9)	79	(20.8)	195	(51.3)
	Top10	204	(53.7)	81	(21.3)	208	(54.7)

Table 7.10 shows the results of our semantic role method (**SemMatch**) together with two baseline systems. The first baseline (**DepPtn+DepCor**) uses only syntactic information, whereas the second baseline (**SemParse**) uses Shalmaneser, a state-of-the-art shallow semantic parser for the role labeling task. As can be seen, the *SemMatch* is significantly better than both the *DepPtn+DepCor* and the *SemParse*, whereas the latter is significantly worse than the *DepPtn+DepCor*.

Although promising, the results in Table 7.10 are not very informative, since they show the performance gains on partial data. Instead of using the semantic role method on its own, we next combine it with the syntactic method *DepPtn+DepCor*. If FrameNet is

Table 7.11: Performance of the semantic methods on *SRSet* (see *Total* column in Table 7.9).

TREC		DepPtn+DepCor		+SemParse		+SemMatch	
04	MRR	72	(35.6)	63	(31.0)	80	(39.1)
	Top1	56	(27.6)	41	(20.2)	65	(32.0)
	Top5	100	(49.3)	91	(44.8)	102	(50.2)
	Top10	112	(55.2)	103	(50.7)	114	(56.2)
05	MRR	140	(42.8)	124	(37.9)	148	(45.1)
	Top1	110	(33.6)	91	(27.8)	123	(37.6)
	Top5	172	(52.6)	158	(48.3)	179	(54.7)
	Top10	200	(61.2)	192	(58.7)	205	(62.7)
06	MRR	121	(31.3)	101	(26.2)	130	(33.5)
	Top1	96	(24.9)	71	(18.4)	102	(26.4)
	Top5	160	(41.5)	143	(37.0)	165	(42.7)
	Top10	191	(49.5)	178	(46.1)	194	(50.3)
07	MRR	101	(29.4)	85	(24.7)	109	(31.8)
	Top1	78	(22.7)	57	(16.6)	83	(24.1)
	Top5	142	(41.3)	129	(37.5)	147	(42.7)
	Top10	159	(46.2)	150	(43.6)	163	(47.4)
all	MRR	434	(34.4)	373	(29.6)	467	(37.1)
	Top1	340	(27.0)	260	(20.6)	373	(29.6)
	Top5	574	(45.6)	521	(41.3)	593	(47.1)
	Top10	662	(52.5)	623	(49.4)	676	(53.7)

indeed helpful for QA, we would expect a combined method to yield better performance over a purely syntactic answer extraction module. The two methods are combined as follows. Given a question, we first pass it to the semantic role method; if an answer is found, our job is done; if no answer is returned, the question is passed on to the *DepPtn+DepCor*. Our results are given in Table 7.11. The *+SemMatch* and *+SemParse* are ensemble systems using the *DepPtn+DepCor* together with the semantic method proposed in this paper and Shalmaneser respectively. We also compare these methods against the *DepPtn+DepCor* on its own.

We can now attempt to answer our third question concerning our model's perfor-

Table 7.12: Overall Performance of the Alyssa QA system on *SRSet*

Module		TREC04		TREC05		TREC06		TREC07		all	
AE	MRR	80	(39.1)	148	(45.1)	130	(33.5)	109	(31.8)	467	(37.1)
	Top1	65	(32.0)	123	(37.6)	102	(26.4)	83	(24.1)	373	(29.6)
	Top5	102	(50.2)	179	(54.7)	165	(42.7)	147	(42.7)	593	(47.1)
	Top10	114	(56.2)	205	(62.7)	194	(50.3)	163	(47.4)	676	(53.7)
WV (Top5)		89	(43.8)	130	(39.8)	117	(30.3)	99	(28.8)	435	(34.5)
WV (Top10)		82	(40.4)	123	(37.6)	117	(30.3)	97	(28.2)	419	(33.3)

mance on the real TREC data *SRSet*. Our experiments show that a FrameNet-enhanced answer extraction module significantly outperforms a similar module that uses only syntactic information (compare the *+SemMatch* and the *DepPtn+DepCor* in Table 7.11). Another interesting finding is that the shallow semantic parser *+SemParse* performs considerably worse in comparison to our graph-based model *+SemMatch* and the syntactic method *DepPtn+DepCor*. Inspection of the parser's output highlights two explanations for this. First, the shallow semantic parser has difficulty assigning accurate semantic roles to questions (even when they are reformulated as declarative sentences). And secondly, it tends to favor precision over recall, thus reducing the number of questions for which answers can be found. A similar finding is reported in (Sun et al., 2005) for a PropBank trained parser. In addition, we find that the syntactic method combination *DepPtn+DepCor* (34.4% MRR) indeed outperforms the separate use of them (31.4% MRR of the *DepPtn* and 30.7% MRR of the *DepCor*).

7.5 Final Performance of Alyssa QA System

As describe in Section 3.6.2, we develop a Web Validation(WV) Module to further validate top ranked answer candidates from the Answer Extraction(AE) Module. It uses frequency of candidates within the Web data to boost most likely answers. We access

the Web data by using the Google Search Engine. Various queries ranging from loose to tight are generated and different weights are assigned to different queries. The weights for the Bag-of-Word, Noun-Phrase-Chunk and Declarative-Form queries are set to 1:2:5. For each query, top 50 snippets returned by Google are used for validation.

We evaluate the final performance of the Alyssa QA system. As shown in Table 7.12, validating Top 5 answer candidate leads to the best performance. Finally, the Alyssa QA system achieves 43.8%, 39.8%, 30.3% and 28.8% accuracy on TREC04, 05, 06 and 07 respectively.

Chapter 8

Conclusion

8.1 Methods

In this thesis, I address two problems (flexible matching problem and knowledge integration problem) in current factoid question answering systems and propose methods to solve the problems theoretically and practically. The module applies maximum entropy ranking theory to effectively integrate various evidence, including orthographic, lexical, surface pattern, syntactic and semantic features. The features are captured by conducting appropriate matching between question and answer sentences on various levels including surface text level, syntactic level and semantic level.

I propose two methods: dependency relation pattern method (*DepPtn*) and dependency relation path correlation method (*DepCor*) to capture syntactic evidence. Both methods are motivated to reduce divergences of lexical representations between two sentences and measure similarity based on their syntactic representations. The *DepPtn* represents a syntactic structure as a dependency relation pattern and regards similarity measure as pattern matching. It assumes that syntactic structures are similar if they share a large portion of common individual relations. A string kernel method is adapted to match patterns partially. Due to the coverage problem of the *DepPtn*, the *DepCor* is

further proposed as its backup. The method assumes that syntactic structures are similar if their individual relations are highly correlated. A dynamic time warping algorithm is applied to align two syntactic structures based on the correlations of their individual relations. The correlation values are estimated using mutual information measure on training set.

In order to capture semantic evidence, I present a graph-based model to incorporate FrameNet style role semantic information effectively. The way of conducting semantic role assignment mainly relies on the comparison of dependency relation paths attested in FrameNet annotations and raw text. The search for an optimal role assignment is formalized as an optimization problem in a bipartite graph. This formalization allows finding an exact, globally optimized solution. The graph theory framework goes some way towards addressing coverage and recall problems related to FrameNet and formulates the similarity measure of semantic structures as a graph matching problem.

Based on the various similarity measures between question and contexts of answer candidates, I present a maximum entropy-based model to incorporate all of the captured information. The answer extraction task is considered as an answer candidate ranking problem under the maximum entropy mechanism. The ranking model aims to predicate the conditional probability $p(ac|q, \{ac_1, ac_2, ..., ac_N\})$, where, the probability of an answer candidate ac for a question q in answer candidate set $\{ac_1, ac_2, ..., ac_N\}$ is directly modeled.

8.2 Results

I apply the answer extraction module to the TREC QA task and perform experiments on the TREC99-07 factoid questions (TREC99-03 for training and TREC04-07 for test).

As to the syntactic evidence, I obtain the following findings:

1. The experiments show that the *DepPtn* and the *DepCor* significantly outperform

four state-of-the-art syntactic-based methods by up to 15.1% in MRR. The performance comparison of the methods are consistent on the different sentence sets (*GSSet*, *GSSRSet* and *SRSet*) and the different years (04, 05, 06 and 07).

2. It is observed that the *DepPtn* performs better than the *DepCor* in MRR and Top1 evaluations while the *DepCor* outperforms the *DepPtn* in Top5 and Top10 evaluations. It indicates that the *DepPtn* contributes more to the precision while the *DepCor* to the recall. Finally, the method *DepPtn* + *DepCor* which combines both of them further boosts the performance by up to 3.7% MRR.

3. The *DepPtn* and the *DepCor* achieve quite similar performance on different sentence sets (*GSSet* and *GSSRSet*) although the *GSSRSet* contains much more noise. It proves the great capability of the methods on identifying supportive sentences.

4. The large gap (about 10%) between Top1 and Top5 performance indicates the weakness of our system on recognizing top1 answers. On the other hand, it also illuminates the use of a web validation module to re-rank the answer candidates returned by the answer extraction.

5. The *DepPtn* achieves the best MRR performance among all syntactic methods, which benefits from the partial answer pattern matching method. This method outperforms the exact matching by 4.1% MRR due to the higher coverage.

As to the semantic evidence, the experiments demonstrate that the proposed semantic matching method can be effectively combined with the syntactic methods to obtain performance superior (+2.7% MRR) to the latter when used on their own. The method also significantly outperforms a shallow semantic parser trained on the FrameNet annotated corpus. The consistent performance gains on coverage and recall might be due to the adopted graph theory framework. As a by-product, I provide a detailed analysis of the appropriateness of FrameNet for QA. The results indicate that FrameNet-based

analysis only works for approximately 30.2% of the TREC questions. It motivates the use of semantically informed methods in conjunction with syntactic-based methods.

Finally, integrating of all evidence using a maximum-entropy ranking model achieves 37.1% MRR and 29.6% Top1 performances on the 04-07 TREC questions. The system is further enhanced to 34.5% Top1 performance by using a web validation module to re-rank the top 5 answer candidates returned by the answer extraction module.

8.3 Future Work

I will discuss the roadmap for future research.

1. **Exploring more features in the answer extraction.** In this thesis, I employ surface, syntactic and semantic features to make appropriate matching between question and answer sentence. It aims to predict whether answer candidates are supported by their contexts in sentences. Many more semantic features can be used in experiments by appropriately matching between expected answer types of questions and semantic categories of answer candidates. In the current Alyssa system, I develop a fine-grained named entity recognizer to recognize 50 types of named entities in sentences, as discussed in Section 3.4 by using rule-based and dictionary-based methods. However, it is not trivial to build a comprehensive named entity dictionary manually. It is interesting to explore how to identify more named entities from a huge un-annotated dataset by applying a bootstrapping method, such as active learning. In addition, the problem of name normalization should be solved for better use of redundancy information from answer candidates. For example, normalizing DATE entities written in different forms.

2. **Applying syntactic matching methods to other NLP applications.** I proposed two similarity measurements to match sentences on syntactic representations. While the experiments are designed for factoid QA task in this thesis, they are generic and

can be easily extended to other applications that utilize meaning comparison of two sentences.

- **Information extraction (IE)** Matching problem in IE tasks is quite similar to the one in the factoid answer extraction. In IE tasks, two slots of named entities, such as a PERSON name and an ORGANIZATION name, are regarded as terminal nodes in a syntactic relation sequence and dependency relations between the slots are internal nodes in the sequence. Then, the relation between the entities is predicted by matching the syntactic sequence with those in training data. Since IE systems also suffer from various unseen instances not being strictly matched to training sequences, the soft sequence matching method will be expected to improve recall in IE systems.
- **Opinion Mining (OM)** Most of opinion mining systems use polarity word aggregation methods to predict sentiment orientation of sentences. They measure the impact of a polarity word on a topic by using surface distance. The closer a polarity word is to a topic word, the more important the polarity word is for the topic. Syntactic-based matching may help to enhance the measurement since it considers not only distance of two words but also their syntactic relations. A polarity word and a topic word are regarded as two terminal nodes respectively in a syntactic relation sequence and dependency relations between them as internal nodes. A set of representative relation sequences is built from training data. The impact of a polarity word on a topic will be calculated by matching its relation sequence to the training sequences.

3. **Improving list and definition question answering.** The Alyssa QA system mainly deals with factoid questions. It does not put much effort on other question types, such as list and definition questions. Enhancing the other modules is a reasonable step to take. The Alyssa system processes list questions using the same way as

factoid questions. Answer candidates are considered separately. Actually, incorporating relevance measurement between answer candidates will definitely help. I plan to explore more following this direction. On the other hand, more empirical and formalize models will be proposed to answer definition questions. In addition, it is also interesting to design and build an interactive user interface for the Alyssa system.

4. **Extending the Alyssa system to a new opinion QA system.** An opinion QA system focuses on answering opinion questions, such as "*Which countries would like to build nuclear power plants?*". Different from factoid QA, opinion QA system extracts answers from blog data. Opinion QA system can be extended from existing factoid QA system by enhancing the abilities in the following aspects: 1. The ability of processing low-quality texts since blog articles are much noisier than newswires. 2. The ability of identifying subjective sentences in sentence retrieval module. The sentence retrieval module for factoid QA system is to retrieve the sentences containing most of important question terms. However, in opinion QA system, the sentences are further required to contain users' opinions. It is a binary classification task (subjective or objective sentence classification). 3. The ability of recognizing sentiment orientation of answer candidates in answer extraction module. While factoid answer extraction analyzes whether an answer candidate is supported by its context in a sentence, opinion answer extraction will further judge whether the answer candidate has the same sentiment orientation as question. For example, the questions "*Which rock bands do college students like?*" and "*Which rock bands do college students dislike?*" will get totally different answers in opinion answer extraction. The recognition of sentiment orientation is to solve sentiment (positive, negative or neutral) classification problems.

References

Abney, S. 1989. Parsing by chunks. *The MIT Parsing Volume*.

Baker, C. F., C. J. Fillmore, and J. B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the International Conference on Computational Linguistics(COLING1998)*.

Berger, A. L., S. A. D. Pietra, and V. J. D. Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguisitics*, 22:39–71.

Bixler, D., D. Moldovan, and A. Fowler. 2005. Using knowledge extraction and maintenance techniques to enhance analytical performance. In *Proceedings of the 2005 International Conference on Intelligence Analysis, Washington D. C.*

Borthwick, A., J. Sterling, E. Agichtein, and R. Grishman. 1998. Exploring diverse knowledge sources via maximum entropy in named entity recognition. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics Workshop on Very Large Corpora*, pages 152–160.

Bos, J. 2006. The "la sapienza" question answering system at trec-2006. In *Proceedings of the Text Retrieval Conference(TREC2006), NIST*.

Breck, E., M. Light, G. S. Mann, E. Riloff, B. Brown, P. Anand, M. Rooth, and M. Thelen. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL2001) Workshop on Open-Domain Question Answering*.

Carreras, Xavier and Llu'is Màrquez, editors. 2005. *Proceedings of the CoNLL shared task: Semantic role labelling*.

Chen, S. and R. Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. *Technical Report CMUCS -99-108, Carnegie Mellon University*.

Clarke, C., G. Cormack, and T. Lynam. 2001. Exploiting redundancy in question an-

swering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2001)*.

Clarke, C., G. Cormack, T. Lynam, C. M. Li, and G. McLearn. 2001. Web reinforced question answering (multitext experiments for trec 2001). In *Proceedings of the Text Retrieval Conference(TREC2001), NIST*.

Cormen, Thomas, Charles Leiserson, and Ronald Rivest. 1990. *Introduction to Algorithms*. MIT Press.

Cui, H., K. Y. Li, R. X. Sun, T. S. Chua, and M. Y. Kan. 2004. National university of singapore at the trec-13 question answering. In *Proceedings of the Text Retrieval Conference(TREC2004), NIST*.

Cui, H., R. X. Sun, K. Y. Li, M. Y. Kan, and T. S. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the Annual International ACM SIGIR 2005 Conference on Research and Development in Information Retrieval*, pages 400–407. ACM Press.

Dang, H. T., J. Lin, and D. Kelly. 2006. Overview of the trec 2006 question answering track. In *Proceedings of the Text Retrieval Conference(TREC2006), NIST*.

Darroch, J. and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The annuals of Mathematical Statistics (1972)*, 43:1470–1480.

Dumais, S., M. Banko, E. Brill, J. Lin, and A. Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2002)*.

Echihabi, A., U. Hermjakob, E. Hovy, D. Marcu, E. Melz, and D. Ravichandran. 2003. Multiple-engine question answering in textmap. In *Proceedings of the Text Retrieval Conference(TREC2003), NIST*.

Echihabi, A. and D. Marcu. 2003. A noisy-channel approach to question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics(ACL2003)*.

Eiter, Thomas and Heikki Mannila. 1997. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133.

Erk, Katrin and Sebastian Padó. 2006. Shalmaneser - a flexible toolbox for semantic role assignment. In *Proceedings of the International Conference on Language Resources and Evaluation(LREC 2006)*.

Fillmore, Charles J., Christopher R. Johnson, and Miriam R.L. Petruck. 2003. Background to FrameNet. *International Journal of Lexicography*, 16:235–250.

Gildea, Daniel and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Harabagiu, S., D. Moldovan, C. Clark, M. Bowden, A. Hickl, and P. Wang. 2005. Employing two question answering systems in trec-2005. In *Proceedings of the Text Retrieval Conference(TREC2005), NIST*.

Harabagiu, S., D. Moldovan, C. Clark, M. Bowden, J. Williams, and J. Bensley. 2003. Answer mining by combining extraction techiniques with abductive reasoning. In *Proceedings of the Text Retrieval Conference(TREC2003), NIST*.

Harabagiu, S., D. Moldovan, M. Pasca, M. Surdeanu, R. Mihalcea, R. Girju, V. Rus, F. Lacatusu, P. Morarescu, and R. Bunescu. 2001. Answering complex, list and context questions with lcc's question answering server. In *Proceedings of the Text Retrieval Conference(TREC2001), NIST*.

Haussler, D. 1999. Convolution kernels on discrete structures. Technical report, Technical Report UCS-CRL-99-10, University of California, Santa Cruz.

Hussain, M., A. Merkel, and D. Klakow. 2006. Dedicated backing-off distributions for language model based passage retrieval. In *Proceedings of Hildesheimer Informatik-Berichte, LWA*.

Itakura, F. I. 1975. Minimum prediction residual principle applied to speech recognition. In *Proceedings of IEEE Transactions on Acoustics Speech and Signal Processing*, pages 67–72.

Ittycheriah, A., M. Franz, and S. Roukos. 2001. Ibm's statistical question answering system – trec-10. In *Proceedings of the Text Retrieval Conference(TREC2001), NIST*.

Ittycheriah, A., M. Franz, W. J. Zhu, A. Ratnaparkhi, and R. Mammone. 2002. Question answering using maximum entropy components. In *Proceedings of the Second Conference of the North America Chapter of the Association of Computational Linguistics, Pittsburgh, PA*, pages 33–39.

Ittycheriah, A. and S. Roukos. 2002. Ibm's statistical question answering system – trec-11. In *Proceedings of the Text Retrieval Conference(TREC2002), NIST*.

Jaynes, E. 1983. *Papers on Probability, Statistics, and Statistical Physics*. D. Reidel Publishing Co., Dordrecht-Holland.

Jonker, R. and A. Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38:325–340.

Kaisser, M. 2006. Web question answering by exploiting wide-coverage lexical resources. In *Proceedings of the Eleventh ESSLLI Student Session*.

Kaisser, M. and T. Becker. 2004. Question answering by searching large corpora with linguistic methods. In *Proceedings of the Text Retrieval Conference(TREC2004), NIST*.

Kaisser, M., S. Scheible, and B. Webber. 2006. Experiments at the university of edinburgh for the trec 2006 qa track. In *Proceedings of the Text Retrieval Conference(TREC2006), NIST*.

Ko, J. W., T. Mitamura, and E. Nyberg. 2007. Language-independent probabilistic answer ranking for question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics(ACL2007)*.

Koeling, R. 2000. Chunking with maximum entropy models. In *Proceedings of the Conference on Natural Language Learning (CONLL2000)*, pages 139–141.

Lafferty, J. 1997. *Personal Communication*.

Leidner, Jochen, Johan Bos, Tiphaine Dalmas, James Curran, Stephen Clark, Collin Bannard, Bonnie Webber, and Mark Steedman. 2003. The qed open-domain answer retrieval system for TREC 2003. In *Proceedings of the Text Retrieval Conference(TREC2003), NIST;*, pages 595–599.

Leslie, C., E. Eskin, and W. S. Noble. 2002. The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the Pacific Biocomputing Symposium*.

Levenshtein, V. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848.

Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. University of Chicago Press, Chicago.

Li, Xin and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING2002)*, pages 556–562, Taipei, Taiwan.

Lin, D. K. 1994. Principar—an efficient, broad-coverage, principle-based parser. In *Proceedings of The International Conference on Computational Linguistics(COLING1994)*, pages 42–488.

Lin, J. 2002. The web as a resource for question answering: Perspectives and challenges. In *Proceedings of the third International Conference on Language Resources and Evaluation(LREC 2002)*.

Lodhi, H., J. S. Taylor, N. Cristianini, and C. J. C. H. Watkins. 2000. Text classification using string kernels. In *Proceedings of the Annual Conference on Neural Information Processing Systems(NIPS2000)*, pages 563–369.

Merkel, A. and D. Klakow. 2007a. Comparing improved language models for sentence retrieval in question answering. In *Proceedings of Computational Linguistics in the Netherlands CLIN*.

Merkel, A. and D. Klakow. 2007b. Improved methods for language model based question classification. In *Proceedings of 8th Interspeech Conference*.

Merkel, A. and D. Klakow. 2007c. Language model based query classification. In *Proceedings of 29th European Conference on Information Retrieval (ECIR)*.

Miller, G. A. 1990. Wordnet: an on-line lexical database. *International Journal of Lexicography, Special issue*, 2(4).

Moldovan, D., C. Clark, and S. Harabagiu. 2005. Temporal context representation and reasoning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence(IJCAI2005)*.

Moldovan, D., C. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of the Conference of the North America Chapter of the Association of Computational Linguistics(HLT-NAACL2003)*, pages 87–93.

Moldovan, D., S. Harabagiu, C. Clark, and M. Bowden. 2004. Poweranswer-2: Experiments and analysis over trec 2004. In *Proceedings of the Text Retrieval Conference(TREC2004), NIST*.

Moldovan, D., S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Movischi, A. Badulescu, and O. Bolohan. 2002. Lcc tools for question answering. In *Proceedings of the Text Retrieval Conference(TREC2002), NIST*.

Moldovan, D. and A. Novischi. 2002. Lexical chains for question answering. In *Proceedings of the International Conference on Computational Linguistics(COLING2002)*.

Moldovan, D. and V. Rus. 2001. Logic form transformation of wordnet and its applicability to question answering. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics(ACL2001)*.

Monz, C. and M. D. Rijke. 2001. Tequesta: The university of amsterdam's tex-

tual question answering system. In *Proceedings of the Text Retrieval Conference(TREC2001), NIST*.

Narayanan, S. and S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of International Conference on Computational Linguistics(COLING2004)*.

Pado, S. and M. Lapata. 2006. Optimal constituent alignment with edge covers for semantic projection. In *Proceedings of the International Conference on Computational Linguistics / the Annual Meeting of the Association for Computational Linguistics(COLING/ACL2006)*.

Palmer, Martha, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Paranjpe, Deepa, Ganesh Ramakrishnan, and Sumana Srinivasa. 2003. Passage scoring for question answering via bayesian inference on lexical relations. In *Proceedings of the TREC*, pages 305–210.

Pietra, S. D., V. D. Pietra, and J. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Pradhan, Sameer, Wayne Ward, Kadri Hacioglu, James Martin, and Daniel Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the HLT/NAACL*, pages 141–144, Boston, MA.

Rabiner, L. R., A. E. Rosenberg, and S. E. Levinson. 1978. Considerations in dynamic time warping algorithms for discrete word recognition. In *Proceedings of IEEE Transactions on Acoustics, Speech and Signal Processing*.

Ratnaparkhi, A. 1998. *Maximum Entropy Models for Natural Language Ambiguity Resolution*. Ph.D. thesis, Department of Computer and Information Science, University of Pennsylvania.

Ratnaparkhi, A. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning Journal*, 34(1-3):151–175.

Ravichandran, D. and E. Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics(ACL2002)*.

Ravichandran, D., E. Hovy, and F. J. Och. 2003. Statistical qa - classifier vs. re-ranker: What's the difference? In *Proceedings of the Annual Meeting of the Association for Computational Linguistics workshop on Multilingual Summarization and Question Answering*.

Reynar, J. C. and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentences boundaries. In *Proceedings of the fifth on Applied Natural Language Processing*, pages 16–19.

Rosenfeld, R. 1994. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. Ph.D. thesis, Pittsburgh, PA.

Sakoe, H. and S. Chiba. 1971. A dynamic programming approach to continuous speech recognition. In *Proceedings of Int. Cong. Acoustics*, Budapest,Hungary.

Shen, D. and D. Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics(ACL2006)*.

Soubbotin, M. M. 2001. Patterns of potential answer expressions as clues to the right answer. In *Proceedings of the Text Retrieval Conference(TREC2001), NIST*.

Sun, R. X., J. J. Jiang, Y. F. Tan, H. Cui, T. S. Chua, and M. Y. Kan. 2005. Using syntactic and semantic relation analysis in question answering. In *Proceedings of the Text Retrieval Conference(TREC2005), NIST*.

Tanev, H., M. Kouylekov, and B. Magnini. 2004. Combining linguisitic processing and web mining for question answering: Itc-irst at trec-2004. In *Proceedings of the Text Retrieval Conference(TREC2004), NIST*.

Taskar, Ben, Simon Lacoste-Julien, and Dan Klein. 2005. A discriminative matching

approach to word alignment. In *Proceedings of the HLT/EMNLP*, pages 73–80, Vancouver, BC.

Voorhees, E. M. 1999. The trec-8 question answering track report. In *Proceedings of the Text Retrieval Conference(TREC1999), NIST*.

Voorhees, E. M. 2000. Overview of the trec-9 question answering track. In *Proceedings of the Text Retrieval Conference(TREC2000), NIST*.

Voorhees, E. M. 2001. Overview of the trec 2001 question answering track. In *Proceedings of the Text Retrieval Conference(TREC2001), NIST*.

Voorhees, E. M. 2002. Overview of the trec 2002 question answering track. In *Proceedings of the Text Retrieval Conference(TREC2002), NIST*.

Voorhees, E. M. 2003. Overview of the trec 2003 question answering track. In *Proceedings of the Text Retrieval Conference(TREC2003), NIST*.

Voorhees, E. M. 2004. Overview of the trec 2004 question answering track. In *Proceedings of the Text Retrieval Conference(TREC2004), NIST*.

Voorhees, E. M. 2005. Overview of the trec 2005 question answering track. In *Proceedings of the Text Retrieval Conference(TREC2005), NIST*.

Wang, M. Q., N. A. Smith, and T. Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP2007)*.

Wu, L. D., S. J. Huang, L. You, Z. S. Zhang, X. Li, and Y. Q. Zhou. 2004. Fduqa on trec2004 qa track. In *Proceedings of the Text Retrieval Conference(TREC2004), NIST*.

Wu, L. D., X. J. Huang, Y. Q. Zhou, Y. P. Du, and L. You. 2003. Fduqa on trec2003 qa task. In *Proceedings of the Text Retrieval Conference(TREC2003), NIST*.

Wu, M., M. Y. Duan, S. Shaikh, S. Small, and T. Strzalkowski. 2005. University at albany's ilqua in trec 2005. In *Proceedings of the Text Retrieval Conference(TREC2005), NIST*.

Xu, J. X., A.Licuanan, and R. Weischedel. 2003. Trec2003 qa at bbn: Answering definitional questions. In *Proceedings of the Text Retrieval Conference(TREC2003), NIST*.

Xu, J. X., A. Licuanan, J. May, S. Miller, and R. Weischedel. 2002. Trec2002 qa at bbn: Answer selection and confidence estimation. In *Proceedings of the Text Retrieval Conference(TREC2002), NIST*.

Yang, H. and T. S. Chua. 2002. The integration of lexical knowledge and external resources for question answering. In *Proceedings of the Text Retrieval Conference(TREC2002), NIST*.

Yang, H., H. Cui, M. Maslennikov, L. Qiu, M. Y. Kan, and T. S. Chua. 2003. Qualifier in trec-12 qa main task. In *Proceedings of the Text Retrieval Conference(TREC2003), NIST*.

Zhang, D. and W. Lee. 2007. Web based pattern mining and matching approach to question answering. In *Proceedings of the Text Retrieval Conference(TREC2002), NIST*.