Soft Matching for Question Answering

Hang Cui

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the School of Computing

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgments

This thesis would not have been possible without the support, direction and love of a multitude of people. First, I have been truly blessed to have two wonderful advisors on the path of scholarship, whose skills have greatly complemented each other and gave me the unique chance to explore my work in both information retrieval and natural language processing. Tat-Seng Chua, who led me through the four tough years by reminding me of the big picture in my research, is always supportive and accommodating to my ideas. Min-Yen Kan, for his continuous efforts and great patience in discussions and formulations of my work, as well as detailed editing. What I very much owe to my advisors are not only the academic training they have given me, but also the way they have taught me to deal with various challenges on my career path. They selflessly shared with me their invaluable experience in both work and life, which will accompany and motivate me for the whole life.

I have been blessed to have had many supporting my endeavors for scholarship since the very beginning of this work, playing multiple roles for which I am greatly thankful for:

My parents Han-Sheng Cui and Yong-Mei Suo, and my wife Adela Junwen Chen for the moral support. I would not have finished my thesis without their backing locally and remotely.

My thesis committee members: Hwee-Tou Ng, Chew-Lim Tan, Wee-Sun Lee and John M. Prager for their critical readings of the thesis and giving constructive criticism that enabled me to clarify claims and contributions which needed additional coverage in the thesis.

I am also grateful to those who have spent time discussing my work with me and gave their constructive comments. They have helped me think of the problems more deeply and more extensively: Jimmy Lin from the University of Maryland, College Park; Sanda Harabagiu from University of Texas at Dallas; John Tait from University of Sunderland.

I am also indebted to Krishna Bharat and Vibhu Mittal from Google Inc., who provided me precious opportunities of internship at Google, which let me see the opportunities of information retrieval and natural language processing in real world.

Thanks also go to those who kindly allowed me to make use of their software tools to complete the work more efficiently: Dekang Lin for Minipar and Dina Demner-Fushman for the POURPRE evaluation tool;

and finally Line Fong Loo, for her always kind help in co-ordinating all administrative stuffs in my four years in the school of computing.

I am also grateful for the comments from the anonymous reviewers of the papers I have had the privilege of publishing in conferences, workshops and journals. I have been financially supported by the Singapore Millennium Foundation Scholarship (ref no. 2003-SMS-0230) for three years (2003 – 2006), and had been supported by the National University of Singapore graduate scholarship for one year (2002 – 2003).

To my beloved wife, Adela Junwen Chen.

To my parents, Han-Sheng Cui and Yong-Mei Suo.

Contents

Abstract

Soft Matching for Question Answering

Hang Cui

I identify weaknesses in exact matching of syntactic and semantic features in current question answering (QA) systems. Such hard matching may fare poorly given variations in natural language texts. To combat such problems, I develop two soft matching schemes. I implement both soft matching schemes using statistical models and apply them to two components in a QA system. Such a QA system is designed to fulfill the information need of advanced users who search for information in a systematic way. Taking a search target as input, the QA system can produce a summarized profile, or definition, for the target and answer a series of factoid questions about the target.

To build up the QA system, I develop two key components – (1) the definitional question answering system that generates the definition for a given target; and (2) the factoid question answering system that is responsible for answering specific questions. In this thesis, I focus on precise sentence retrieval for these two components and evaluate them component-wise.

To retrieve definition sentences that construct the definition, I apply lexico-syntactic pattern matching to identify definition sentences. Most current systems employ hard matching of manually constructed definition patterns, which may have the problem of low recall due to language variations. To combat this problem, I adopt the soft matching scheme anchored at the search target. In particular, I develop three soft pattern models – a simple baseline model and two formal ones based on the bigram model and the Profile Hidden Markov Model (PHMM), respectively. The soft pattern models generalize pattern matching as the process of producing token sequences. I experimentally show that employing soft pattern models greatly outperforms the system that utilizes hard matching of pattern rules.

To obtain precise answer sentences for a specific factoid question about a target, I examine the dependency relations between matched question words in addition to lexical matching. As the same relations may be phrased differently, I adopt another soft matching scheme. Specifically, I employ a machine translation model to implement this soft matching scheme to compute the similarity between multiple relations. I experimentally demonstrate that the passage retrieval performance is significantly augmented by combining soft relation matching with lexical matching.

The main contribution of this thesis is in developing soft matching schemes to flexibly match both lexico-syntactic patterns and dependency relations, and applying the soft matching models to sentence retrieval for answering definition and factoid questions.

List of Tables

List of Figures

Chapter 1

Introduction

With the advent of Internet, the Web has grown to an enormous knowledge repository, which archives more information than any library on the planet. Facing such a huge virtual library, finding useful information is just like finding a needle in a haystack. Nowadays, searching for information on the Web has become part of people's life. To meet this huge demand, search engines (SE) dominate the attention of people. As the "database of our intentions" (Battelle, 2005), search engines, such as Google[1] and Yahoo![2], help people explore the Web to find useful information.

Despite the great success of Web search engines, people still face the problem of how to find precisely what they really want. Question answering (QA) is one technology that addresses this problem. In contrast to information retrieval, question answering attempts to return exact answers in response to a query. Current QA systems mainly deal with fact-oriented questions that ask for facts about a target, such as a person or an organization. In the Text Retrieval Conference (or TREC, (Voorhees, 2000)), open-domain QA is evaluated on a large news corpus. Fact-oriented questions are divided into three types - factoid, list and definition questions. Factoid questions, such as "When was Aaron Copland born?", require

[1] http://www.google.com

[2] http://www.yahoo.com

exact phrases or text fragments as answers. List questions, like “List all works by Aaron Copland”, ask for a list of answers belonging to the same group. While factoid and list questions cover specific aspects of the target, definition questions expect a summary of all important facets related to a given target. For instance, for the question “Who is Aaron Copland?”, the user may want to know when and where he was born, why he was famous, his main musical works and other supplementary information like his activities as a communist. To answer such a question, a QA system has to identify definitions about the target from the corpus and put them together to form an answer.

The state-of-the-art QA systems have complex architectures. They draw on statistical passage retrieval (Tellex et al., 2003; Chu-Carroll et al., 2004), question typing (Hovy et al., 2001; Chu-Carroll et al., 2004) and semantic parsing (Echihabi and Marcu, 2003; Xu, Licuanan, and Weischedel, 2003). In statistical ranking of relevant passages, to supplement the sparseness in a corpus, current systems also exploit knowledge from external resources, such as WordNet (Harabagiu et al., 2000) and the Web (Brill et al., 2001). Given the statistical techniques employed, the techniques focus on lexical and named entity matching with question terms. As such, it is often difficult for existing QA systems to find answers as they share few words in common with the question. To circumvent this problem, recent work attempts to map answer sentences to questions in other spaces, such as lexico-syntactic patterns. For instance, IBM and ISI have systems that map questions and answer sentences into parse trees and surface patterns (Ravichandran and Hovy, 2002). Echihabi and Marcu (2003) adopted noisy-channel approach from machine translation to align questions and answer sentences based on a trained model.

While current QA systems have shown great success in TREC evaluations, I have identified two weaknesses of these systems that should be addressed to enhance the performance:

1. Not target-aware – Most current QA systems deal with isolated questions without considering the focus of the questions. Note that I refer to the focus here as the search target mainly concerned by user questions. For instance, if a user submits questions about "Aaron Copland", it would be helpful if he or she has some background knowledge about the target. The background knowledge of the target could serve as the context for other questions. Within the context, users could ask questions in a more manifest way and thus more complete and precise answers are expected to obtain. I believe a desirable QA system should be aware of the target of the input questions and be able to help the user build up the context by providing a profile for the search target.

2. Lack of flexibility in matching – While there has been work on semantic matching of words (*e.g.* via WordNet) beyond exact lexical match, there lacks work addressing flexible matching in other spaces, such as pattern matching. Employing other syntactic or semantic features, like textual patterns or semantic relations, reinforces the precise search for answers. However, rigid match often fares poorly due to errors in other tools and variations in natural language texts.

To address these problems, I have proposed an integrated question answering system that is supposed to deal with factual questions about a given target and have implemented two key components for the system. One component, which is called **the definitional question answering system**, analyzes the target and returns a summarized profile for that target. The profile could serve as a definition to help the user build up the context for his or her follow-up questions regarding the target. The other component, which is **the factoid question answering system**, allows the user to ask specific questions about the target. Within this framework, I focus my work on retrieval of accurate answer sentences for definition and factoid questions. A main hypothesis here is that once the right answer sentence is retrieved, it is likely to contain the answer. I will incorporate techniques that go beyond word-based metrics to boost the precision and completeness of sentence retrieval. I believe that the key to a QA system is to find appropriate similarity metrics between the question and the sentences that contain the answer.

1.0.1 Problem Statement

In this thesis, I hypothesize that flexible matching for syntactic and semantic features can improve the performance of sentence retrieval for question answering. I examine statistical similarity metrics to realize flexible matching, which I term as **soft matching**.

Hypothesis: Soft matching of syntactic and semantic features beyond lexical features can improve the performance of sentence retrieval for answering definition and factoid questions as compared to systems that employ only exact match of such features.

To this end, I have devised two soft matching schemes that are used in the sentence retrieval modules of my QA system.

1.1 Soft Matching Schemes

In sentence retrieval, it is crucial to measure how similar a candidate sentence and the query are in terms of different features. In addition to lexical features, one may draw on other features, such as patterns represented by token sequences and relations between words, to capture the similarity. In contrast to exact match of such features, I propose soft matching, which allows approximate match and embodies the similarity measure as the degree of match in terms of certain features.

I introduce two schemes of soft matching in this thesis – one based on one anchor and the other based on a pair of anchors or multiple anchors. I define the elements in the candidate sentence which determine the boundary of the unit to be matched as **anchors**. For instance, to match definition patterns for a certain target, the target is considered as an anchor. Next, I illustrate the two soft matching schemes:

One Anchor Soft matching for one anchor is represented as:

$$\begin{aligned}&\text{Given } \; t_{-L}, \ldots, t_{-2}, t_{-1} \; < Anchor > \; t_1, t_2, \ldots, t_L \\ &\text{calculate } \; Degree_{soft-match}(t_{-L}, \ldots, t_{-2}, t_{-1}) + Degree_{Soft-Match}(t_1, t_2, \ldots, t_L)\end{aligned}$$

where the length of the matching unit is L and t_i is the i^{th} feature in the matching unit. The soft matching is performed on both feature sequences left and right of $< Anchor >$ and the match degrees are combined.

Pair of Anchors Soft matching for a pair of anchors is represented as:

$$\begin{aligned}&\text{Given } \; < Anchor_1 > \; t_1, t_2, \ldots, t_L \; < Anchor_2 > \\ &\text{calculate } \; Degree_{Soft-Match}(t_1, t_2, \ldots, t_L)\end{aligned}$$

The soft matching is conducted on the matching unit between the two anchors. Multiple anchors can be treated as multiple pairs, and thus are only an extension of the case with two anchors.

1.2 The Integrated QA System

To date, most QA systems deal with ad-hoc questions - input questions are assumed to be unrelated. It is natural for users to pose a question and get an answer without being aware of the context of his or her target. However, for advanced users such as professional information analysts, it is rather difficult to grasp really relevant information about a target by asking such ad-hoc questions. In contrast, an analyst prefers to collect information on a target in a more structured way. Imagine such a scenario: While reading news papers, an analyst encounters the name of a terrorist that interests him or her. Without much clue about the terrorist, she first wants to know all the important facets about that person – *e.g.*, origin, education background, activities performed, *etc.* The analyst may not want to ask many questions to retrieve information about every aspect of that person. Rather, the system should be able to provide a summary of the information as a profile. The system learns and builds the context of questions that will retrieve information necessary for the construction of the profile. In the next step, the analyst may come up with some specific questions after reading the profile. She may consult the system for answers on these specific questions, which are about the terrorist. The system should be capable of searching for answers from the relevant information obtained in the first step. In other words, the QA system should be able to perform two classes of tasks (Moldovan et al., 2003) – fusing answers from different documents and answering factual questions.

To achieve this goal, an integrated QA system for given targets is needed. Recently, the National Institute of Science and Technology (NIST) in the United States has started to address this goal in its TREC guidelines. The TREC QA task includes a number of targets and the participating systems are required to answer specific questions about each target, as well as to find all other information related to the targets (Voorhees, 2004; Voorhees and Dang, 2005). In this thesis, I present key components for building such an integrated QA system to address this problem. The QA system, which takes a given target as a query, analyzes the target and generates a profile for this target as the context, within which specific questions about it can be precisely answered. I will not present all required modules, nor evaluate the integrated QA system as a whole. I focus my work on the two key modules of definition extraction and factoid question answering, and evaluate them component-wise.

The core components leverage lexico-syntactic patterns and semantic relations to identify appropriate sentences relevant to a target (see Figure 3.1). Besides the module for sentence retrieval, other modules in the system also impact the final output. These include technologies for question analysis, document retrieval, anaphora resolution, *etc.* These other technologies are beyond the scope of this thesis. I focus on dealing with open-domain questions with little domain-specific knowledge. Interactive QA is also beyond the scope of my thesis.

I chose the news domain for question answering as there is a need for users to get answers for timely questions which cannot be answered by other existing knowledge sources. Furthermore, news is diverse in its content and constantly changing.

1.2.1 Soft Matching in the QA System

In Section 1.1, I sketched two soft matching schemes, which will be further detailed in this thesis. I apply both schemes in my QA system – the one-anchor scheme on soft matching of definition patterns; and the two-anchor scheme on soft matching of dependency relations between words.

To retrieve definition sentences to construct the definition for the search target, I employ textual definition pattern matching to identify definition sentences. Since the definition patterns capture the contexts surrounding the search target in the sentences, the search target is the only anchor to locate the matching units. As such, I apply the one-anchor soft matching scheme to soft pattern matching.

In order to precisely obtain the answer sentence for a factoid question about a target, I examine the similarity between the dependency relations between matched question words as additional evidence for ranking candidate sentences. In this case, the matching units are multiple relations between every pair of matched question terms. Therefore, it is natural to apply the two-anchor scheme for soft relation matching.

I employ different statistical models, which are discussed in Chapters 4, 5 and 6, to implement the soft matching schemes. The features in soft matching vary according to different tasks. For pattern matching, the features are lexical words, syntactic tags and punctuations, while for relation matching, the features are grammatical relations. It is worth pointing out that the soft matching schemes and the corresponding statistical models are generic and can be applied to other scenarios where one may find similar anchor matching schemes, provided that appropriate matching features are used.

1.3 Contributions

In this thesis, I make the following contributions:

Lexico-Syntactic Pattern Matching. I present formal statistical models for realizing soft lexico-syntactic pattern learning and matching. Moreover, I show how to generalize sentences into abstract pattern instances, which facilitate more generic matching. I evaluate the effectiveness of the soft pattern matching models on definition sentence retrieval. The generic soft pattern models can be extended to other applications that utilize textual patterns.

Passage Retrieval. I show how dependency relations between matched question terms help improve the performance of passage retrieval in a QA system. In particular, I adapt dependency relation matching to my soft matching scheme and make use of a statistical translation model to calculate the similarity between multiple relations. Such a technique is also applicable to improving passage retrieval in other retrieval systems.

Question Answering. I present two key components to build a QA system – one component can present the definition (or profile) to the search target and the other answers subsequent specific questions about the target. Such an integrated QA system does not answer questions on an ad-hoc basis. In contrast, it helps the user better understand the target and make the search process more manageable. It is more adaptable to the use of the information system by the advanced users.

1.4 Guide to This Thesis

In Chapter 2, I give the background for question answering. I will review existing work for definitional and factoid QA, as well as the main techniques they employ. Specifically, I first review the work on lexico-syntactic pattern rule induction in information extraction as it is closely related to definition pattern matching in definitional QA. Definition pattern matching is formally identical to pattern matching in information extraction. My soft pattern models are alternative to rule induction techniques. In presenting existing work in definitional QA, I also show related work in domain-specific definition generation and query-dependent summarization. I then review the work in passage retrieval for factoid QA.

In Chapter 3, I present the architecture of the QA system. In addition, I discuss in detail the basic modules that are not covered in later chapters. I leave the modules that embed soft matching technologies in the next three chapters.

In Chapter 4, I show the basic soft pattern model. This model is simple and ad-hoc but embodies the fundamental idea of soft pattern matching. In this section, I also show how to generalize definition sentences and test sentences into pattern instances, which are represented in lexical and syntactic token sequences and are abstract of the sentences. The generalized pattern instances are the basis for soft matching.

In Chapter 5, I present two generic soft pattern models which are derived from formal statistical models. These two models are the formalization of the previous simple model and obtain better performance in evaluations. I will present evaluation results by using these two formal models, comparing with that obtained by the basic soft matching model and hard matching rules.

While the previous two chapters are about soft pattern matching in definitional QA, in Chapter 6, I show the soft matching technique for dependency relations in factoid QA passage retrieval. I present the statistical translation model for calculating the similarity between relation paths (multiple relations) in the parsing trees.

I conclude the thesis in Chapter 7. I will summarize the contributions and point out the limitations of this work. I present possible future work in the end of the thesis.

Chapter 2

Background

2.1 Overview of Question Answering

Different from traditional information retrieval (Salton and McGill, 1984), which gets a list of relevant documents for a given query, QA systems aim to answer a natural language question with the most exact answer. For instance, the question "`Who invented the paper clip?`" in TREC should be answered by the name of the inventor. Question answering tasks in TREC have evolved in the past years. At the beginning, TREC QA track had only simple factoid questions like the above example and requires the systems to answer the questions by a text fragment of 50 bytes. Such questions are usually sufficiently answered by a word or a phrase. In recent TREC, the guideline requires the system to return the exact answer to factoid questions, instead of snippets. From 2003 (Voorhees, 2003b), TREC has introduced two new separate types of questions - list questions and definition questions. The list question answering task requires the systems to assemble the answers to a list question, such as "`What Chinese provinces have a McDonalds restaurant?`", from multiple supporting documents. The answer should be a list of factoid answers to the question. The definition questions are answered by a set of text fragments or sentences, which provide an extended definition to the target, such as "`Who is Colin Powell?`". Instead of being correct or not, the list and definition questions are evaluated based on their precision and recall on the facts to answer the questions.

From TREC-2004, the question answering task integrated definition factoid and list questions into different series, where each series had a target associated with it. The systems are required to give definition to each target and answer the target-related factoid and list questions. I illustrate a question series in Figure 2.1. As is seen in Figure 2.1, each question in a series asked for some information about the target. In addition, the final question in each series was an explicit "other" question, which was to be interpreted as "Tell me other interesting things about this target I don't know enough to ask directly". This last question is roughly equivalent to the definition questions in previous TREC tasks. Each series is a (limited) abstraction of an information dialog in which the user is trying to define the target. The target and earlier questions in a series provide the context for the current question. The construction of TREC question series is very similar to the working process of my proposed QA system, which is able to present the definition of the search target and answer subsequent factoid questions. As such, I will use TREC data set as the evaluation set in my experiments.

```
22 Franz Kafka
22.1 FACTOID  Where was Franz Kafka born?
22.2 FACTOID  When was he born?
22.3 FACTOID  What is his ethnic background?
22.4 LIST  What books did he author?
22.5 OTHER
```

Figure 2.1: A Sample Series in TREC-2004

A typical open domain QA system consists of three modules to perform the information seeking process (Harabagiu, Maiorano, and Pasca, 2003):

1. **Question Processing** - The question processing module captures the semantics embedded in a natural language question and is able to recognize the expected answer type. For instance, given the question "`Who invented the paper clip?`", the expected answer type is `person`. In addition, the keywords in the question are utilized to retrieve documents and passages where the possible answer may lie.

2. **Document and Passage Processing** - The document processing module indexes and retrieves the documents in the data collection based on the keywords given in the questions. QA systems break the retrieved documents down to passages and select the most relevant passages for answer processing.

3. **Answer Processing** - The answer processing module completes the task of finding the exact answer from the relevant passages. It compares the semantics of the answer against those embedded in the question.

In my QA system, as my goal is to obtain sentence-level answers for the questions, I will not discuss the question processing module and the answer processing module. The reason is two-fold: (1) The input to my QA system is a search target and a series of factoid questions about the target. It is not necessary to perform question analysis to search for definitions for the target. (2) As for the factoid questions, I am more interested in improving the passage retrieval process by examining the semantics in both the question and the relevant sentences.

In the rest of this chapter, I give background knowledge in definitional QA and passage retrieval for factoid QA. Before coming to the two types of QA, I will first review lexico-syntactic pattern induction because it is closely related to my soft pattern matching technique for definitional QA.

2.2 Lexico-Syntactic Pattern Induction

As pattern learning and matching comprise a large portion of my work, previous work in lexico-syntactic pattern (or rule) induction is closely related because it generalizes rules represented by regular expressions from annotated training text. Pattern learning algorithms are categorized into pattern generalization on free text and structured documents, as well as automatic wrapper induction (see (Muslea, 1999) for a survey).

Many pattern rule inductive learning systems have been developed for information extraction (IE) on free texts and semi-structured texts. AutoSlog by Riloff (1993) is an early pattern learning system. It employs a set of initial heuristics to identify the interesting part of a given sentence. An extraction pattern is created when a sentence is initiated by any of the heuristics and the pattern consists of the constraint of the matched heuristic and the specific words. For instance, "`<victim> was kidnapped`" is a generated rule by Autoslog for the terrorism domain. AutoSlog-TS (Riloff, 1996) extends Autoslog by adopting a means of exhaustive pattern generation on unannotated text, which is only pre-classified according to scenarios. Autoslog-TS generates all possible patterns around each noun phrase in the training corpus and employs the popularity score (or relevance score) of each pattern to filter out those low-frequency patterns. WHISK (Soderland, 1999) induces multi-slot rules from a training corpus top-down. It was designed to handle text styles ranging from highly structured text to free text. WHISK performs rule induction starting from a randomly selected seed instance. It grows a rule from the seed by starting with an empty rule. With more training instances, the slots of the rule become more specific with words or tokens. Generated rules with more errors than the threshold on the training data are discarded. (LP)2 (Ciravegna, 2001) is a covering algorithm for adaptive IE systems that induce symbolic rules. In (LP)2, the training is performed in two steps: first, a set of tagging rules is learned to identify the boundaries of slots; next, additional rules are induced to correct mistakes in the first step of tagging. Another recent work in rule induction is by Xiao et al. (2003), namely, the GRID system. GRID differs from the previous work in that it examines the global statistics of the tokens in each slot to select the tokens to specialize the slots. The selected tokens should minimize the incurred errors and occur frequently in certain slots. As such, GRID performs more efficient rule induction.

From the above work, we see that IE systems rely on a set of textual rules which are generalized from training examples. These algorithms generalize the context around the target of interest, in terms of syntax and semantics, and abstract contextual constraints. A pattern is matched if each surrounding word of the extraction candidate matches the pattern's corresponding constraint. As such, I say that the pattern matching by the generalized rules is *hard matching*, as it requires an exact, slot-by-slot match. A pattern is not matched if any slot is not matched. Such hard matching often fails when there is mis-match between generalized rules and unseen text due to great variance in natural languages. To circumvent this problem, I will introduce soft pattern matching models in Chapters 4 and 5.

Similar to my soft matching idea, Nahm and Mooney (2001) proposed learning soft matching rules from texts by combining rule-based and instance-based learning. Words in each slot are generalized by traditional rule induction techniques and test instances are matched to the rules by their cosine similarities. Likewise, Snowball (Agichtein et al., 2001) system tries to extract relations, such as the headquarters of companies, from large-scale data on the Web using textual rules that are approximately matched by calculating cosine similarity of the test instances with the rules. While their work embraced the idea of statistical matching, it simplifies the task by performing only lexical matching in slots. Different from their work, my soft pattern matching models consider lexical tokens alongside syntactic features and adopt a probabilistic framework that combines slot content and sequential fidelity in computing the degree of pattern match. In addition, my goal is to propose generic pattern matching models that can be extended to question answering where textual patterns are employed.

2.3 Definitional Question Answering

I categorize the definition extraction systems into two groups – domain-specific definition extraction systems and open-domain definition extraction (or definitional QA) systems. I classify my soft pattern based system in the latter category.

TREC has had a separate competition track on definitional question answering since 2003. Entrants' systems are evaluated over a corpus of over 1 million news articles from various news agencies. The definitional QA task requires the participating systems to extract and return interesting information about a particular person or term, such as "`Who is Vlad the Impaler?`" or "`What is a prion?`" The evaluation of definition questions is based on a manual check of how many answer nuggets (determined by the human assessor) are covered by system responses. Partial credit for answers (Voorhees, 2003a) can be given. Figure 2.2 illustrates an example question from TREC and its corresponding answer nuggets.

TREC assesses definitional QA systems with respect to content precision and recall and does not attempt to judge definitions with respect to fluency or coherence. This is in line with the focus on my work in retrieving relevant definition sentences but differs from the general task of definition generation in which such stylistic criteria matter. As seen in Figure 2.2, content nuggets are categorized into vital pieces of information and okay ones that would be desirable to include in such

```
Qid 1933: Who is Vlad the Impaler?
1933 1 okay 16th century warrior prince
1933 2 vital Inspiration for Bram Stoker 1897 novel "Dracula"
1933 3 okay Buried in medieval monastery on islet in Lake Snagov
1933 4 vital Impaled opponents on stakes when they crossed him
1933 5 okay Lived in Transylvania (Romania)
1933 6 okay Fought Turks
1933 7 okay Called "Prince of Darkness"
1933 8 okay Possibly related to British royalty
```

Figure 2.2: A Sample Definition Question and Answer Nuggets from TREC

extended definitions.

In early TREC, definition questions are mixed with factoid questions and are required to be answered by a phrase as short definition. As such, the systems, such as the FALCON system (Harabagiu et al., 2000) and IBM's system (Prager, Radev, and Czuba, 2001), employed simple, manually constructed patterns to extract proper phrases or hypernyms from WordNet to define the search target.

However, extended definitions are thought to be more useful to users as they incorporate more description and context of the target term, which may better facilitate comprehension. Recent definitional QA systems have applied more sophisticated analyses to retrieve such descriptive sentences. Table 2.1 summarizes the techniques employed by some representative TREC systems that perform well in the official evaluations. An exhaustive listing of techniques on a per system basis is presented in Table A.1 of Appendix A.

From the table, it is clear that definitional QA systems mainly rely on two types of information to identify definitions: definitional linguistic constructs and statistical ranking. Let's examine these two components in more detail.

Table 2.1: Summary of Techniques Employed by TREC Systems

TREC Systems	**Definitional Linguistic Constructs**					**Statistical Ranking**	
	Surface Patterns[a]	Patterns on Parse Trees[b]	Appositives & Copulas[c]	Relative Clauses[d]	Predicates & Verb Phrases[e]	Centroid Vector or Profile[f]	Mining External Definitions[g]
NUS[h] (Yang et al., 2003)	×		×	×		×	
BBN (Xu, Licuanan, and Weischedel, 2003; Xu, Weischedel, and Licuanan, 2004)	×	×	×	×	×	×	×
Columbia (Blair-Goldensohn, McKeown, and Schlaikjer, 2003)		×	×		×	×	
LCC (Harabagiu et al., 2003)	×		×				
MIT (Katz et al., 2004)	×		×	×	×	×	×
IBM PIQUANT (Chu-Carroll et al., 2004; Prager et al., 2003)	×		×	×		×	×
Amsterdam (Ahn et al., 2004)		×	×			×	×
Sheffield (Gaizauskas et al., 2004)	×		×	×		×	×
Korea University (Han et al., 2004)		×	×	×	×	×	×

[a] Lexico-syntactic surface patterns, such as "`<TARGET> , the $NNP`".

[b] Pattern rules for extracting specified constructs from syntactic parse trees for sentences.

[c] Appositives – *e.g.* "*Gunter Blobel, a cellular and molecular biologist, . . .*"
Copulas – *e.g.* "*Stem cell is a cell from which other types of cells can develop.*"

[d] *e.g.* "*. . . Gunter Blobel who won the Nobel Prize for . . .*"

[e] Predicates and verb phrases are mainly for describing a person or special relations. They are identified by a set of specialized verbs, which are often coupled with people's behaviors, such as "born" and "vote".

[f] To construct a centroid vector or profile for each target and use that centroid vector to rank the relevance of candidate sentences or constructs. The centroid vector contains a set of highly relevant words to the target, which could be selected by frequent words in external definitions/biographies or extracted candidate sentences or constructs.

[g] To have other definitions obtained from definitional web sites, such as online biographies and encyclopedias. The relevant words to the target in the corpus are augmented with weights if they also appear in external definitions.

[h] This system is used in TREC-12, before I proposed the soft pattern models.

2.3.1 Definitional Linguistic Constructs

All systems try to identify specific definitional linguistic constructs that mark definitional sentences. Examples of such definitional linguistic constructs include appositives and copulas. Appositives, such as "`Gunter Blobel, a cellular and molecular biologist, ...`", are mostly used in news to introduce a person or a new term. To recognize such linguistic constructs, the systems employ pre-compiled patterns, either on surface text(*e.g.*, BBN, MIT and LCC) or on syntactic parse trees (*e.g.*, Amsterdam, Columbia and Korea University). Definition patterns can also be defined based on specific question patterns and entity classes (Harabagiu et al., 2005). Since surface patterns are more adaptable and easier to deploy without the requirement of task-specific parsing, I discuss only surface textual patterns that are represented in lexical/syntactic tokens. I list some definition patterns in Table 5.2. According to component evaluations (Xu, Weischedel, and Licuanan, 2004; Cui et al., 2004a), definition pattern matching is the most important component in a definitional QA system.

It is worth noting that the patterns employed by current definitional QA systems are equivalent to those that have been used by information extraction (IE) systems, as stated in the previous section. Virtually all definitional QA systems that employ manual patterns (*e.g.*, (Harabagiu et al., 2003; Hildebrandt, Katz, and Lin, 2004)) or automatic rule induction algorithms(*e.g.*, (Peng et al., 2005; Cui et al., 2004a)) are hard pattern matching systems, as their patterns are equivalent to regular expressions and perform slot-by-slot matching.

I identify two drawbacks of using such generalized pattern rules for extracting definitions:

1. **Inflexibility in matching**: As stated, hard matching rules fail to match when there are even small variations between the training instances and the test text, such as extra or missing tokens. Such variations in natural language

text are common in extended definitional sentences and are a hallmark of fluent, well-crafted articles. Similar problems occur in information extraction, but are usually more limited as IE tends to extract domain-specific and task-specific information.

2. **Inconsistent weighting of patterns**: Most systems use statistical metrics to rank the importance of retrieved constructs, but treat each definitional pattern with the same level of importance. However, different definition patterns should be weighted differently. For instance, appositives are the most popular syntactic pattern for definitions, and thus should be weighted heavily. Many systems lack a consistent method to determine the importance of the various definition patterns. The frequency of each pattern can then be utilized when ranking extracted definition candidates.

To circumvent the above problems, I proposed an alternate pattern generation and matching technique, **soft pattern matching**, for definition sentence identification (Cui, Kan, and Chua, 2004; Cui, Kan, and Chua, 2005). Different from current definition patterns, soft pattern models learn holistic definition patterns from all training instances and assign weights to different pattern instances according to their distributions in the training data. More importantly, it does not treat pattern matching simply as a binary decision, but allows partial matching by calculating a generative degree-of-match probability between the test instance and the set of training instances.

The definition of soft patterns encompasses several existing approaches to information extraction. Several graphical models for IE can be viewed as soft pattern matching in this framework. Skounakis et al. (2003) applied hierarchical HMMs to the task of extracting binary relations in biomedical texts. They constructed two HMMs to represent words and phrases, which are two levels of emission units. Earlier work by McCallum et al. (2000) demonstrates the application of Maximum

Entropy Markov Model (MEMM) to segmentation and extraction of FAQs from web documents. These variations of HMMs also model pattern matching as token sequence generation and are able to deal with variations in test instances. However, they cannot be applied to definition pattern matching directly because the topologies they employ are task-specific.

In this work, I focus my discussion on lexico-syntactic patterns used in definitional QA systems. There are other patterns beyond textual patterns. For instance, in TREC 2005, LCC (Harabagiu et al., 2005) employ another two types of pre-compiled patterns - question patterns and entity classes. Question patterns comprise a list of factoid questions which are considered essential nuggets according to the type of the target. Entity classes indicate relevant named entities to the target in the corpus. These two types of patterns could be considered as pre-defined templates for searching for definitions for different targets. Since such template-like patterns need intense manual labor and expertise to construct, I do not consider them in this thesis.

2.3.2 Statistical Ranking

The second common component in many definitional QA systems is statistical ranking to weight the relevance of extracted definition candidates. A commonly-employed method is to construct a centroid vector, or *profile*, for the search target and rank the definition candidates by calculating the similarity between the candidates and the centroid vector. Centroid words are relevant, non-trivial words correlated with the search target. They are selected from the extracted candidates by measuring their co-occurrence with the target or by measuring their corpus frequency in a large set of definitions or biographies available from an external resource.

My centroid ranking method, discussed in Section 3.1.1, is based on the former technique but also generalizes the lexical tokens into syntactic tags to create evidence for more generic patterns. I will discuss how to replace centroid words with their syntactic tags in Section 4.1.

TREC systems (*e.g.*, (Ahn et al., 2004)) also utilize definitions extracted from online encyclopedia and biographical web sites, which provide a much larger and cleaner resource for definitions. External definitions are usually utilized to reinforce the definition candidates from the corpus. The weights of candidates with higher amounts of overlap with the external definitions are thus augmented. I will discuss the use of external definitions in my system in Section 3.1.1.1. However, I will not discuss in detail the evaluations on the use of external knowledge as my focus is on soft matching for QA. I will summarize the observations on experiments with external resources in Section 4.5.6.

2.3.3 Related Work

In this section, I present the existing work that is related to definition extraction. As TREC QA task is for open domain QA, I first review the complementary work in domain-specific definition extraction. Then, I discuss query-dependent summarization, which is pertinent to definition generation because the latter summarizes all relevant information about a target.

2.3.3.1 Domain-Specific Definition Extraction

There has been much work on the extraction of definitions for terms from structured or unstructured text. Identifying a canonical form for abbreviations and acronyms is perhaps the simplest form of definition extraction. Schwartz and Hearst (2003) presented an algorithm that searches definition for acronyms in biomedical text. The algorithm searches for the form “short form (long form)” or “long form (short form)” and examines whether each letter in the short term comes from each word in the long form. Such definitions for abbreviations are relatively simple to identify, and thus it is sufficient to apply only string processing techniques. Zahariev (2003) introduced dynamic programming in matching definitions to handle more complicated acronyms, which may have multiple letters from a single word in the expansion form.

DEFINDER (Klavans and Muresan, 2001) is part of a digital library project and aims to provide readable definitions of medical terms to patients. While developed for a specific domain, the two primary techniques it employed are largely domain-independent: (1) Shallow text pattern analysis – patterns such as "is called" and "is the term used to describe" are utilized to identify definitions. (2) Grammar analysis for recognizing more complex structures like appositives. In their evaluation, Klavans and Muresan (2001) showed that online medical dictionaries have lower coverage compared to the results automatically extracted by DEFINDER (the completeness of online dictionaries varies from $22\% - 76\%$ compared to the extracted definitions). Their results show that automatic definition extraction systems complement manually-constructed dictionaries. I believe that the coverage of standard authoritative sources is lower in the open-domain context as new terms are coined frequently. As such, developing automatic systems for definition generation is indispensable.

As both the accuracy of manually-constructed definitions and the coverage of automatically-extracted definitions are positive qualities, researchers often combine both types of resources. For instance, in (Muresan et al., 2003), glossaries identified from existing web sites and definitions extracted from unstructured text by DEFINDER are integrated to determine conceptual connections between different term databases.

Schiffman et al.'s system (2001) produces biographical summaries (*i.e.* to answer "*who is*" questions). They combined a data-driven statistical method with machine learned rules to generate definitions. The biographical information is identified by appositives and special predicates lead by verbs that are associated with typical actions of people. Likewise, Sarner and Carberry (1988) identified fourteen distinct predicates that are related to definition content, such as those associated with identification, properties and components. The generated definitions were placed in the context of cooperative dialogs. Due to the specific scenario of the use of their system, they weighted predicates to determine which are involved in the definition based on three models: the model of the user's domain knowledge, the model of the user's underlying plan and goal and that of how receptive the user is to various information.

More recently, the ubiquity of the Web has generated interest on finding definitions. Liu et al. (2003) proposed mining topic specific definitions from the Web. The basic idea is to utilize a set of hand-crafted rules to find definition sentences on web pages. They also tried to utilize the structure of web pages to identify sub-topics of each main topic, which could be considered part of extended definition of the main topic.

The above systems automatically extract definitions from plain text or web pages. However, they are domain-specific, *i.e.*, working on only a specific category of terms or on a particular corpus. In contrast, my aim is to present a comprehensive definition generation system that works on news articles and is able to extract definitions for a wide spectrum of terms.

2.3.3.2 Query-Dependent Summarization

Another existing work that is closely related to my work in definitional QA lies in query-dependent summarization because definitional QA can be considered as the process of sentence extraction and summarization based on a specific query, *i.e.*, the target.

Goldstein et al. (1999) presented Maximal Marginal Relevance (MMR) on multi-document summarization. The basic idea is to choose sentences that are closely correlated (or similar) to the query and are different from the sentences that are already in the summary. Their statistical model of sentence selection has been adopted in my sentence summarization module for generating definitions. My variation of MMR will be presented in Section 3.1.2. White et al. (2001) applied an information extraction system to a summarization system based on scenarios, like natural disaster. The IE system extracts specific pieces of information and let the summarization system put them into template-based summaries. The extracted information was also utilized to supplement the scenario templates for summarization. Radev and McKeown (1998) presented a system that can produce a summary of a given event from multiple news sources. In addition to scenario template-based sentence extraction, they incorporated complex techniques in discourse planning and language generation to ensure the coherence of the generated summary. Tombros and Sanderson (1998) applied a summarization system to an information retrieval system such that the users obtain a summary of each retrieved document. The summary helps users locate the target documents more quickly. They relied on the article title, the location of sentences, important terms in the documents and terms biased towards the query to determine which sentences to construct the summary.

However, query-dependent summarization does not apply to definitional QA because the former summarizes all relevant documents while the latter requires the system to extract definition sentences on the target and then summarize them. In other words, definitional QA capitalizes more on the identification of definition sentences.

2.4 Passage Retrieval for Factoid Question Answering

Passage retrieval has been studied in depth in information retrieval (Kaszkiel and Zobel, 1997). It aims to search for more precise and compact text excerpts in response to users' queries, rather than providing whole documents. Passage retrieval is a crucial component in factoid question answering (QA) systems. To answer a specific factoid question about `Louvre`, *e.g.*, "`When was the Louvre transformed into a museum`?" a factoid QA system employs a pipeline structure that consists of several modules to get the short and precise answer: (1) locating the relevant documents, (2) retrieving passages that may contain the answer, and (3) pinpointing the exact answer from candidate passages. I focus on Step 2 because passage retrieval greatly affects the performance of a factoid QA system. If a passage retrieval module returns too many irrelevant passages, the answer extraction module is likely to fail to pinpoint the correct answer due to too much noise. Moreover, a passage can sufficiently answer a factoid question. Lin et al. (2003) showed that users prefer passages to phrase-long answers because passages provide sufficient context for them to understand the answer.

The simplest passage retrieval method, employed by MITRE (Light et al., 2001), counts the number of matched question terms in a passage. Other passage retrieval systems, such as those employed in SiteQ (Lee et al., 2001) and IBM (Prager et al., 2003), are density-based as they take into account the distances between question terms in the candidate passages. IBM's passage retrieval system takes into account WordNet synonyms in addition to word matching. In addition, the system considers the "dispersion measure," which counts the matched words' distance in the passage, and "cluster words," which examine the adjacent words in both the query and the passage. It linearly combines all the measures to weight the passages. SiteQ weights the query terms based on their part-of-speech tags and ranks the passages according to the sum of the weights of matched words, as well as their normalized distance. Hovy et al. (2001) presented the ISI system, which weighs different lexical features including query terms, proper names and stemmed words, to rank the passages.

Tellex et al. (2003) conducted a thorough quantitative component evaluation for passage retrieval algorithms employed by current QA systems. The authors concluded that neglecting crucial relations between words is a major source of false positives for current lexical matching based retrieval techniques. The reason is that many irrelevant passages share the same question terms with correct ones, but the relations between these terms are different from those in the question. We illustrate this by a sample question and some candidate sentences in Figure 2.3, where only sentence S1 contains the correct answer. The other three sentences share many question terms (in italics) but are incorrect.

<**Question**> What percent of the nation's cheese does Wisconsin produce?
<**S1**>**(correct)** In *Wisconsin*, where farmers *produce* roughly 28 *percent* of the *nation*'s *cheese*, the outrage is palpable.
<**S2**>**(Incorrect)** ... the number of consumers who mention California when asked about *cheese* has risen by 14 *percent*, while the number specifying *Wisconsin* has dropped 16 *percent*.
<**S3**>**(Incorrect)** The wry "It's the *Cheese*" ads, which attribute California's allure to its *cheese* _ and indulge in an occasional dig at the Wisconsin stuff" ... sales of *cheese* in California grew three times as fast as sales in the *nation* as a whole 3.7 *percent* compared to 1.2 *percent*, ...
<**S4**>**(Incorrect)** Awareness of the Real California *Cheese* logo, which appears on about 95 *percent* of California *cheeses*, has also made strides.

Figure 2.3: Sample Question and Candidate Passages Illustrate that lexical matching can lead to incorrect answers.

Figure 2.3 shows that a passage retrieval system that relies only on lexical level matching and considers each question term an independent token may fare poorly in real applications. Next, I will review some previous work that tried to incorporate term relationship in the retrieval phase.

2.4.1 Attempts in Previous Work

To extract precise answers, Harabagiu et al. (2003) applied a theorem prover that conducts abductive reasoning over WordNet to derive semantic relationship between words. Other techniques attempt to approximate such relations between words statistically. For instance, some language modeling approaches capture simple dependency relations by using bigrams (*e.g.*, (Song and Croft, 1999)). But these models only capture dependency relations between adjacent words.

To take into account relations between question terms, previous work has applied grammatical or statistical co-occurrence based relations. PiQASso (Attardi et al., 2001) employed a dependency parser and extracts the answer from a candidate sentence if the relations reflected in the question are matched in that sentence. However, that system does not perform well due to low recall resulting from matching relations in only the top ranked sentences. To remedy the recall problem, Katz and Lin (2003) indexed and matched specific relations (*e.g.*, `subject-verb-object`) over an entire QA corpus. However, they performed their evaluation on only a handful of manually constructed questions instead of the community-standard TREC data.

Both the above systems select answers based on strict matching of dependency relations. Strict matching is problematic when conducted on a large corpus because the same relationship is often phrased differently in the parse trees of the question and the answer. For instance, appositive relations can be rephrased using other dependency relations - such as the `whn` (nominal wh-phrase) relation - in the question. As such, strict matching of relations may fare poorly in recall, which is an important consideration in passage retrieval. To address the problem brought by rigid matching of relations, I propose to adopt soft matching of dependency relations between matched question terms to improve factoid QA passage retrieval (Cui et al., 2005). I will discuss my soft relation matching in Chapter 6.

There are methods that model dependency relations statistically at the surface level. For instance, Gao et al. (2004) proposed a language model that captures dependency relations that are learned from training data. They proposed a statistical parsing model that captures dependency relations between words based on the co-occurrences of words in the training data.

Instead of adopting such statistically determined relations, my proposed relation matching method is based on grammatical dependency relations determined by Minipar (Lin, 1998), a fast and robust dependency parser. The reason is threefold: (1) Different from information retrieval, we do not have a large amount of QA data for training. Using relation matching based entirely on statistics could be problematic due to sparse data. (2) QA questions are sentences, which enable us to adopt a dependency parser to extract various types of dependency relations. Such typed relations, which have more accurate meanings in expressing dependency relationships, tend to be of higher differentiating capability in filtering out irrelevant relations. (3) Unlike Gao et al., we seek to build a system with an off-the-shelf parser so that the system and its results are easier to reproduce. Minipar is a free research dependency parser that fulfills this requirement. Minipar has been used in question answering (*e.g.*, (Attardi et al., 2001)) in the past.

Chapter 3

Architecture of the Question Answering System

In this chapter, I present the architecture of my QA system. In particular, I illustrate the subsystem of definitional QA. While I leave the core technologies of soft matching in later chapters, I discuss other modules in the rest of this chapter.

I illustrate the overall architecture of my question answering system in Figure 3.1. This system takes a search target, *e.g.*, “Aaron Copland”, as input and retrieves a set of relevant documents. The sentences in the relevant documents are utilized to produce the definition for the target and to answer specific questions about the target. The core module of the system is the one for answer sentence evaluation, which ranks definition sentences and answer sentences for specific questions. The ranked candidate answer sentences are then fed to the module of answer extraction and summarization to produce the final answer.

In particular, the system performs the following steps to get the answers.

Document retrieval and sentence splitting: Given a search target, it first takes the target as a query and feeds it into a standard document retrieval system (Step (a)). The result is a set of relevant documents about the target.

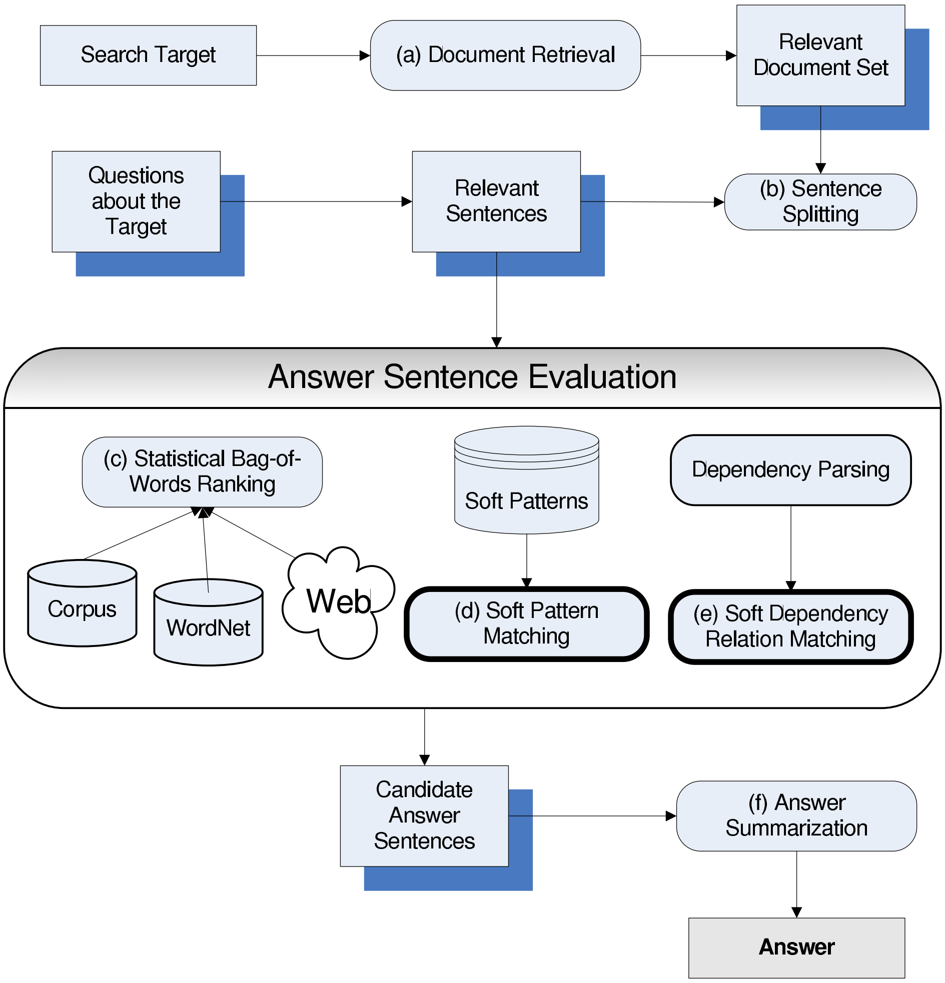

Search Target
(a) Document Retrieval
Relevant Document Set
Questions about the Target
Relevant Sentences
(b) Sentence Splitting
Answer Sentence Evaluation
(c) Statistical Bag-of-Words Ranking
Soft Patterns
Dependency Parsing
Corpus
WordNet
Web
(d) Soft Pattern Matching
(e) Soft Dependency Relation Matching
Candidate Answer Sentences
(f) Answer Summarization
Answer

Figure 3.1: Illustration of the Architecture of the Integrated QA System

The documents are then split into sentences in Step (b).

The retrieved relevant sentences are the basis for subsequent answer sentence evaluation to get two types of answers: (1) definition sentences to construct the definition for the target; and (2) answer sentences to answer the factoid questions about the target. Note that I do not deal with sentences referring anaphorically to the target.

Definition generation: The target-relevant sentences are fed into the definitional QA subsystem and are evaluated by: (c) statistical bag-of-words ranking and (d) soft pattern matching to obtain definition sentences. The statistical bag-of-words ranking can be reinforced by external resources of definitions, such as the Web and WordNet. The definitional QA subsystem includes another module (f) to select top ranked definition sentences to fulfill the pre-specified answer length. I will discuss the definitional QA subsystem in the next section.

Factoid question answering: Given the specific factoid questions about the target, the system employs (c) statistical bag-of-words ranking and (e) soft dependency relation matching module to evaluate the candidate answer sentences. In my system, I utilize sentences to answer the factoid questions, and thus it does not include a module as answer extraction as in other QA systems. The highest ranked sentence is selected as the answer to the question. I will discuss the module of soft dependency relation matching in Chapter 6.

3.1 The Subsystem for Definitional QA

Figure 3.2 shows the architecture of the definitional QA subsystem, which is specialized from the architecture illustrated in Figure 3.1.

Given the target and a set of relevant sentences, the system executes the following steps to construct an appropriate answer.

(1) Pattern instance generalization: I process the retrieved sentences into pattern instances, on which soft definition pattern generation and matching are performed. I first replace the words that are specific to the search targets with their general syntactic (POS or chunk) tags. Remaining words are stemmed. I refer to these remaining lexical words and substituted general syntactic tags as *tokens*. I then take the tokens surrounding the search target

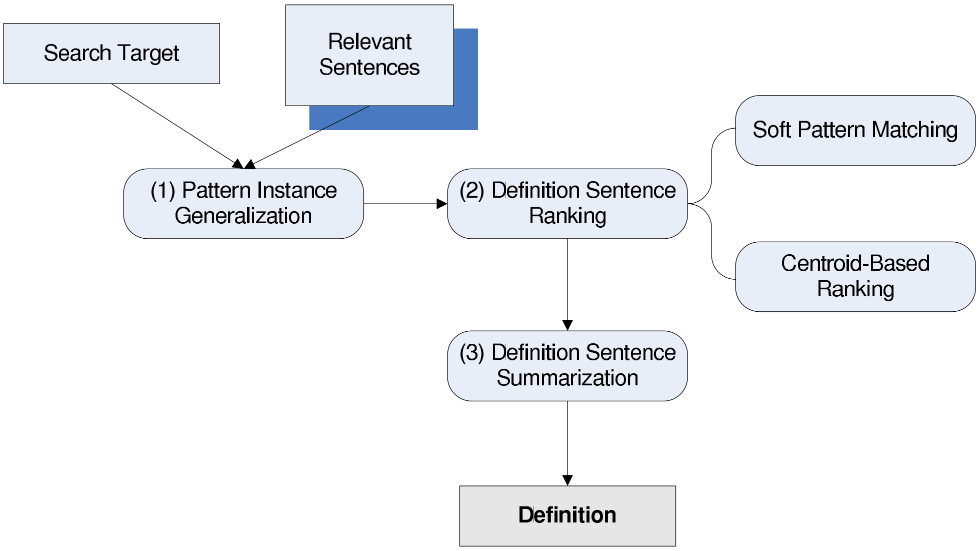

Search Target
Relevant Sentences
(1) Pattern Instance Generalization
(2) Definition Sentence Ranking
Soft Pattern Matching
Centroid-Based Ranking
(3) Definition Sentence Summarization
Definition

Figure 3.2: Illustration of the Architecture of the Definitional QA Subsystem

as pattern instances. Figure 3.3 illustrates several sample pattern instances. I will discuss in detail how to generalize pattern instances in Section 4.1.

```
BE$ discovered by NNP <TARGET> , DT$ NN ,
DT$ JJ <TARGET> , DT$ JJ NN
NNP , known as <TARGET> , BE$ be held
<TARGET> including NN in DT$
<TARGET> BE$ CD$ of DT$
```

Figure 3.3: Sample Pattern Instances Generated after Pre-processing

(2) Definition sentence ranking: The definition sentence retrieval module ranks the input sentences based on how likely they are definition sentences for the target. I rank definition sentences using two features: soft pattern matching and centroid based ranking, which correspond to modules (c) and (d) in Figure 3.1. To rank sentences, I combine the pattern matching and bag-of-words ranking scores using simple linear weights.

(3) Definition sentence summarization: This module is exactly (f) in Figure 3.1. It produces the final definition by selecting from top ranked sentences and removing redundant sentences. Note that the relevance of definition sentences is determined by the pre-defined length of the definition, i.e., the number of selected top ranked sentences.

The key steps of soft pattern matching tasks of generalization (Step 1) and ranking (Step 2) deserve an in-depth discussion and will be described in detail in the following chapters. In the remainder of this section, I discuss bag-of-word relevance approach (part of Step 2) and the final summarization sentence selection (Step 3).

3.1.1 Bag-of-Words Statistical Ranking of Relevance

In order to accumulate as many relevant sentences for the search target as possible, I adopt *centroid ranking*, a bag-of-words statistical ranking technique for weighting the relevance of passages with respect to a given target. Centroid ranking has been applied in summarization by Radev et al. (2004), and in definitional question answering (Xu, Weischedel, and Licuanan, 2004; Cui, Kan, and Chua, 2004).

In multi-document summarization, Radev et al. (2004) select centroid words by taking words that are most representative across documents by computing words' global $TF \times IDF$ weights. However, in the definitional QA context, centroid words must bear very specific information describing the search target. As such, I adopt a local relevance metric of words with respect to the search target based on their co-occurrences with the search target. To assess the importance of each word independent of the search target, I use inverse document frequency (IDF)[1]. A word's local co-occurrence and the global IDF scores are combined to represent the relevance of a word to the search target. The centrality of a word is then implemented

[1] I use the statistics from Web Term Document Frequency and Rank site (http://elib.cs.berkeley.edu/docfreq/) to approximate words' IDF within the corpus.

in our system by the following equation:

$$Centrality_T(w) = -\log \frac{SF(T, w)}{SF(T) \times SF(w)} \times IDF(w) \tag{3.1}$$

where T denotes the search target and w is a candidate word occurring in the context of T. $SF(w_1, w2)$ is the number of sentences that contain both w_1 and w_2 and $SF(w)$ is the number of sentences that contain w. $IDF(w)$ represents the IDF value of w.

Given the input sentences, stopwords are removed and the remaining words are stemmed. Centrality scores for the remaining stemmed words are calculated and those words whose scores exceed a standard deviation over the mean are selected as centroid words.

For each target, the system constructs a centroid vector from the resulting centroid words. Similar to the work by Blair-Goldensohn et al. (2004) and Xu et al. (2004), I then rank the candidate sentences by their similarity with the centroid vector, using cosine similarity. Sentences that are highly ranked are considered candidate definition sentences.

3.1.1.1 External Knowledge

In addition to corpus statistics, I also make use of external definitions for the search targets to supplement centroid word selection. The main reason of utilizing external definitions is that there are only a few occurrences in the corpus for some targets, and thus it would be difficult to obtain reliable co-occurrence statistics for those contextual words. In my system, I make use of two types of external knowledge: task-independent (*e.g.*, general Web search) and task-specific (*e.g.*, definitions from definitional sources). As for task-independent information, I attempt to obtain 200 snippets from Google for each search target. For task-specific information, I retrieve the whole definition text from Answers.com[2], which is an aggregation site for online encyclopedias and biographies, for targets that have entries in these sites. While there are many other Web sites that can be used as sources for definitions, I take this site as a representative sample to examine their impact on the performance. I augment the weight of those words that also occur in the text retrieved by Google or the definitional Web resources:

$$Weight(w) \;=\; \begin{cases} Weight_{centroid}(w)\times(1+\log(SF(w)+1)) & \text{if w occurs in Google snippets} \\ Weight_{centroid}(w)\times(1+\theta) & \text{if w occurs in the external definition} \end{cases} \tag{3.2}$$

where $Weight_{centroid}(w)$ denotes the centroid weight of the word w obtained by Equation 3.1. $SF(w)$ gives the number of snippets that contain the word w while θ is a constant factor. I try different θ values (from 0.2 to 1.0) to optimize the system and set it to 0.6 based on my experiments.

3.1.2 Definition Sentence Summarization

In Step 3, the system constructs the final definition from the ranked candidate sentences. This is done by selecting the top-ranked sentences that suit the length requirement and avoid including redundant content. I adopt a variation of Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) to select non-redundant sentences from the top list of sentences ranked by definition weighting scores. The sentence selection algorithm is presented in Figure 3.4. Different from the approach taken by Carbonell and Goldstein, who ranked all passages with MMR, my method examines sentences in descending ranked order and stops when the length of the definition is satisfied.

[2] http://www.answers.com

```
Input:   ranked_sentences – Rank list of sentences in descending order of definitional scores
         num_selected_sentences – Number of selected sentences in the output list
Output:  selected_sentences – List of selected definition sentences

    Add the first sentence of ranked_sentences to selected_sentences
    Remove that sentence from ranked_sentences
    $N = 1$
    for each sentence stc in the remaining of ranked_sentences
        for each sentence sel_stc in selected_sentences
            $sim = CosineSimilarity(stc, sel\_stc)$
            record the maximum similarity max_sim
        if $max\_sim <$ similarity threshold $\eta$
            add stc into selected_sentences and remove stc from ranked_sentences
            $N = N + 1$
        if $N > num\_selected\_sentences$
            return
    end
```

Figure 3.4: Definition Sentence Summarization Algorithm

Chapter 4

A Simple Soft Pattern Matching Model

In this chapter, I discuss a simple soft pattern matching model. It embodies the one-anchor soft matching scheme as described in Section 1.1. As soft pattern matching is used to identify definition sentences, it takes the target as the anchor and the lexical/syntactic tokens around the target as matching units. This simple soft pattern model embodies the two basic characteristics of soft matching of textual patterns: namely, individual slot match degree and sequential fidelity. Such characteristics, plus the process of generalizing definition sentences into pattern instances, lay the foundations for the formal soft pattern models presented in the next chapter. I also develop a group pseudo-relevance feedback (GPRF) method to automatically label sentences for use in soft pattern generation.

I will present the method of soft pattern generalization and matching in the next section. I then present a method of unsupervised learning of soft patterns by GPRF. I complete the chapter with evaluations of soft pattern matching, compared with both manually constructed and machine learned hard pattern matching.

4.1 Generalization of Pattern Instances

To ensure the generality of learned soft patterns, I first generalize the sentences into abstract pattern instances. Given a group of potential definition sentences, the goal is to learn the local contextual patterns surrounding the given search target. Here, I focus on near window dependencies. This is because that definition sentences are identified mainly by adjacent words and punctuations.

Definition (Pattern Instance) A pattern instance is a token sequence that contains lexical/syntactic tokens left and right to the search target after performing transformation steps of tagging and chunking, selective substitution and window cropping on a candidate sentence.

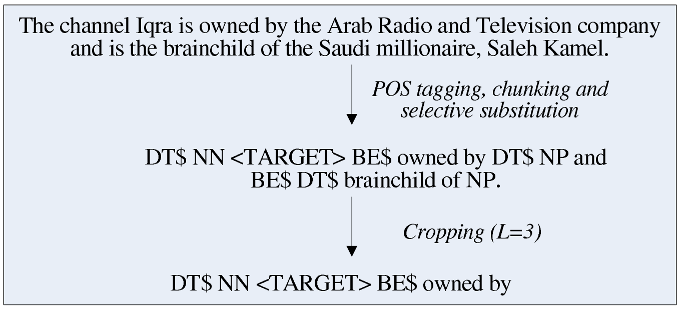

The channel Iqra is owned by the Arab Radio and Television company and is the brainchild of the Saudi millionaire, Saleh Kamel.
POS tagging, chunking and selective substitution
DT$ NN <TARGET> BE$ owned by DT$ NP and BE$ DT$ brainchild of NP.
Cropping (L=3)
DT$ NN <TARGET> BE$ owned by

Figure 4.1: Illustration of Generalization of Pattern Instances

The process of generalizing pattern instances is illustrated in Figure 4.1. It consists of three steps:

1. **Tagging and chunking** – The sentences are first processed with part-of-speech (POS) tagging and chunking by a natural language tagger and chunker[1].

2. **Selective substitution** – Certain lexical items are then selectively substituted for their syntactic classes. The substitution attempts to replace words

[1] I use NLProcessor, a commercial parser from Infogistics Ltd. http://www.infogistics.com/.

that are specifically related to the search term with general tags. The lexical forms of these target-specific words are too specific to the search target to help form general definition patterns and hence are replaced by their part-of-speech tags. Likewise, I perform the same substitution to noun phrases identified by chunking as different scenarios usually do not share the same noun phrase instances. Moreover, I collapse the adjacent syntactic tags of the same type into one. The substitution rules that I use and some examples are listed in Table 4.1.

Table 4.1: Heuristics Used for Selective Substitution

Token	Substitution	Examples (from the example sentence in Figure 4.1)
Any part of the search target	`<TARGET>`	`Iqra` → `<TARGET>`
Target-specific words (centroid words)	Corresponding syntactic classes determined by part-of-speech tags	`channel` → `NN`
Noun phrases by chunking	`NP`	`Arab Radio and Television company` → `NP`
`is, am, are, was, were`	`BE$`	`is` → `BE$`
`a, an, the`	`DT$`	`the` → `DT$`
All numeric values	`CD$`	`63` → `CD$`
Adjectival and adverbial modifiers	*deleted*	
All other words and punctuations	*No substitution*	`owned, by, of` are unchanged

In small-scale experiments, I tested the substitution rules showed above. The substitution rules showed good performance in generalizing the definition sentences and hence enhanced the pattern learning process. I conjectured that better substituion rules may improve the learned soft patterns. It would be nice if such substitution rules could be automatically learned. There has been work in generalizing the symbols used in patterns. Borkar et al. (2001) demonstrated a simple strategy of hierarchical feature selection. They utilized part of the training data as a validation set and prune the specific features. The goal of their method is similar to ours – to generalize the generated patterns. However, we do not adopt the prunning technique because the number of tokens in definitional patterns is large. Instead, we employ a statistical method to determine the specific words to be substituted, which is discussed in the next subsection.

3. **Window cropping** – I only consider the "local context" around `<TARGET>`. The context is modeled as a window centered on `<TARGET>` according to a predefined size L, *i.e.*, the number of tokens on both sides of `<TARGET>`. Thus, we get pattern instances with size $2L + 1$ including the search target.

Determining Substituted Words

In Table 4.1, target-specific words are those non-trivial words that are highly correlated to the search target. Centroid words are defined as target-specific words, selected by the method introduced in Section 3.1.1. As such, centroid words should be replaced by generic tags. Centroid words vary according to search targets. An alternate way to determine target-specific words is to examine the frequent words in the training pattern instances. As definition patterns are supposed to be generic across targets, there are frequently used words, such as "known" and "born", across pattern instances. Pattern instances can be constructed by keeping these common words and substituting other words for generic syntactic tags. However, I do not adopt this alternative as it biases the learned definition patterns towards patterns for certain types of targets because keywords from low-frequency targets' definitions tend to be ignored.

4.2 Constructing Soft Pattern Vector

Accumulating all the pattern instances extracted from the definition sentences and aligning them according to the positions of `<TARGET>`, I obtain a virtual vector representing the soft definition patterns. The pattern vector Pa is denoted as:

$$< slot_{-L}, \ldots, slot_{-2}, slot_{-1}, TARGET, slot_1, slot_2, \ldots, slot_L : Pa >$$

where $slot_i$ contains a vector of tokens with their probabilities of occurrence:

$$< (token_{i1}, weight_{i1}), (token_{i2}, weight_{i2}) \ldots (token_{im}, weight_{im}) : slot_i >$$

Here $token_{ij}$ denotes any token, which could be a word, punctuation or syntactic tag, contained in $slot_i$; and $weight_{ij}$ gives the importance of the j^{th} token to the i^{th} slot. $weight_{ij}$ can thus be expressed as the conditional probability of the token occurring in that slot. Thus it can be approximated by:

$$\Pr(token_{ij}|slot_i) = \frac{f(token_{ij})}{\sum_{s=1}^{m} f(token_{is})} \tag{4.1}$$

where $f(token_{is})$ stands for the number of occurrences of $token_{is}$ within $slot_i$. $\Pr(token_{ij}|slot_i)$ considers only the local weight of $token_{ij}$ to $slot_i$. I also take into account the weight of $token_{ij}$ on condition of other tokens in adjacent slots, which is to be discussed next. Here I only explicitly only model the slot contents independent of sequential information but in the soft matching equation in the next section, sequential information is accounted for. Note that I put words and general syntactic tags in two separate buckets and count their frequencies in the corresponding buckets separately. Syntactic tags typically have a much higher frequency compared to individual words, and would thus skew the distribution if combined with words. As such, I need to separate the two types and estimate each token's unigram probability against its own set. I illustrate the process of constructing soft pattern vectors from pattern instances of training data in Figure 4.2.

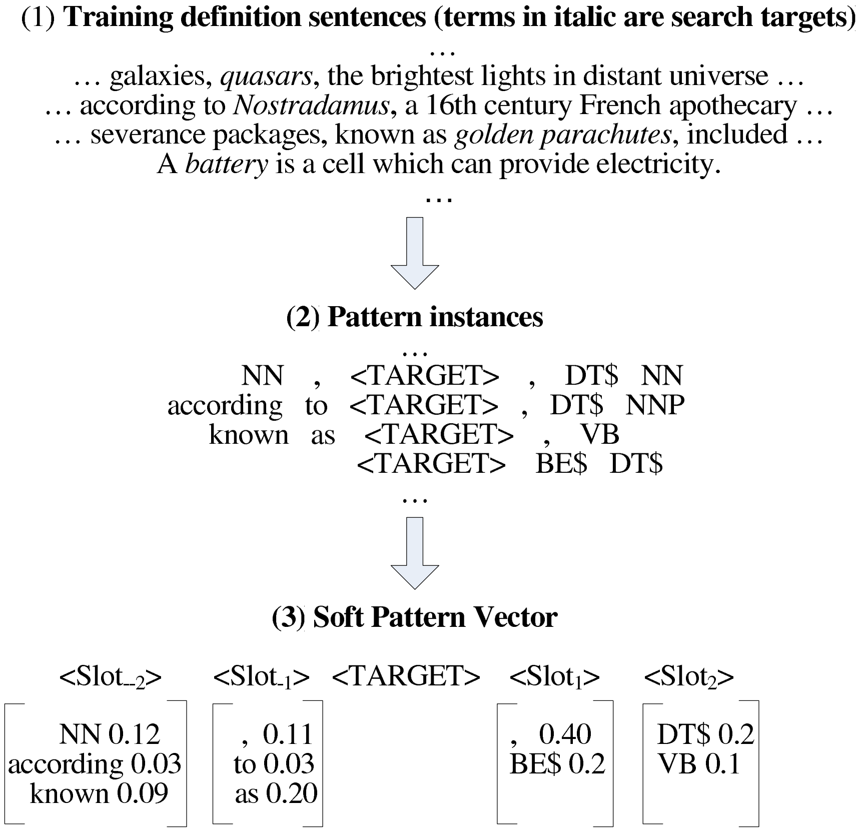

(1) Training definition sentences (terms in italic are search targets)
...
... galaxies, quasars, the brightest lights in distant universe ...
... according to Nostradamus, a 16th century French apothecary ...
... severance packages, known as golden parachutes, included ...
A battery is a cell which can provide electricity.
...
(2) Pattern instances
...
NN , <TARGET> , DT$ NN
according to <TARGET> , DT$ NNP
known as <TARGET> , VB
<TARGET> BE$ DT$
...
(3) Soft Pattern Vector
<Slot-2> <Slot-1> <TARGET> <Slot1> <Slot2>
NN 0.12
according 0.03
known 0.09
, 0.11
to 0.03
as 0.20
, 0.40
BE$ 0.2
DT$ 0.2
VB 0.1

Figure 4.2: Constructing Soft Pattern Vectors

4.3 Soft Pattern Matching

What results from the generalization process is a virtual vector Pa with a set of associated probabilities for slot fillers at each slot. The soft pattern vector Pa is then used to calculate the degree to which a test sentence matches the sentences used to construct the soft patterns. The test sentences are first pre-processed with the identical procedures of POS tagging and chunking, as well as substitution as we did to the labeled definition sentences. Using the same window size L, the token fragment S surrounding the `<TARGET>` is retrieved: $< token_{-L}, \ldots, token_{-2}, token_{-1}, TARGET, token_1, token_2, \ldots token_L : S >$. The matching degree of the test sentence to the generalized definition patterns is measured by the similarity between the vector S and the virtual soft pattern vector Pa. The matching degree is calculated in two parts. The first part calculates the degree of similarity between individual slots, while the second part examines sequence fidelity. In the first part, we compute Pa_weight_{slots} by assuming that all slots are independent of each other. The score is calculated as:

$$Pa\_weight_{slots} = \Pr(S|Pa) = \prod_{i=-L}^{L} \Pr(token_i|slot_i) \tag{4.2}$$

Specifically, I combine all the weights calculated in Equation 4.1 to derive the similarity for independent slots. This equation is very flexible in matching the soft patterns because it considers only individual slots. Even if some slots are missing, it still can give a similarity measure to the definition patterns.

The second part of the matching metric considers the sequence of tokens, to filter out unlikely token sequences to increase precision. I adopt a bigram model to formulate this sequence measure. Specifically, given a token sequence T, we calculate the conditional probability of $\Pr(T|Pa)$ which models how likely the sequence occurs according to the underlying soft patterns. I calculate the sequence probability for the left and the right sequences starting from `<TARGET>`. The probability of the right sequence is calculated as follows:

$$\begin{aligned} \Pr(right\_seq|Pa) &= \Pr(token_1, token_2, \ldots token_L|Pa) \\ &= \Pr(token_1) \Pr(token_2|token_1) \\ &\quad \ldots \Pr(token_L|token_{L-1}) \end{aligned} \tag{4.3}$$

where $\Pr(token_i|token_{i-1})$ is estimated by counting the occurrences of the bigram

$< token_{i-1} \;\; token_i >$ and the unigram $token_{i-1}$ as:

$$\Pr(token_i|token_{i-1}) = \frac{f(< token_{i-1}, token_i >)}{f(token_{i-1})} \tag{4.4}$$

The process for calculating the probability of the left sequence is identical. In addition, $\Pr(token_{-1})$ and $\Pr(token_1)$ can be estimated based on the proportion of occurrences of the token in the immediately left and right slots to `<TARGET>`. The sequence weight of the token vector for the sentence, denoted by Pa_weight_{seq}, consists of the weights of its left sequence and right sequence which are calculated by Equation 4.3:

$$Pa\_weight_{seq} = (1-\alpha)\Pr(left\_seq|Pa) + \alpha \Pr(right\_seq|Pa) \tag{4.5}$$

α is a tunable parameter. Based on my observations of definitions, the right context of the search term is more important in indicating a definition sentence, thus I set α to 0.7. Here, I set the parameters based on personal observations; in Section 5.1, I will show how to estimate the parameters by using expectation maximization (EM) algorithm.

Finally, the aforementioned two similarity weights determine the overall pattern weight of the given sentence:

$$Pattern\_match\_weight = \frac{Pa\_weight_{slots} \times Pa\_weight_{seq}}{length(S)} \tag{4.6}$$

where the length of the fragment S is used as the normalization factor.

4.4 Unsupervised Learning of Soft Patterns by Group Pseudo-Relevance Feedback

In order to perform learning for soft patterns, a set of labeled definition sentences needs to be provided as training instances. While the formal soft pattern models which will be discussed in the next chapter are trained on manually labeled definition sentences, I present an unsupervised learning scheme for soft pattern training based on pseudo-relevance feedback (PRF) in this section. The evaluations in this chapter are accordingly based on the unsupervised learning. The experimental results by supervised learning will be presented in the next chapter. While automatically generated training data is not as accurate as manually labeled, the purpose of this section is to show an alternate way to accumulate training data when there is no sufficient manual labeling available. In addition, I believe the automatically generated data can be a good supplement to the supervised data (Sudo, Sekine, and Grishman, 2001).

The process of the unsupervised soft pattern learning and matching is illustrated in Figure 4.3.

Step 1 automatically ranks sentences from the input documents, using centroid words that are highly correlated with the search target as indicators. To automatically decide whether a sentence is definitional, I use a simple cutoff in which sentences that are ranked more highly are considered definitional. This is similar to the work by Sudo et al. (2001), who proposed unsupervised learning method for pattern discovery by utilizing $TF \times IDF$ weight to select a set of relevant documents and sentences, and then built patterns from them.

I employ a group pseudo-relevance feedback (GPRF) strategy. In standard pseudo-relevance feedback (also known as blind or local feedback) used in document retrieval, for each query, the top n ranked documents are deemed relevant and used to modify the query to retrieve a new set of documents (Buckley et al., 1994). I employ the same technique here: the system takes the top n ($n = 10$) sentences from each question's ranking results and combines these sentences over all questions as (blindly) labeled definition sentences. I then conduct the soft pattern generalization process on these sentences.

Input: a set of questions and corresponding relevant sentences.

1. **First round of ranking (statistical ranking)**: Rank all input sentences statistically. I employ the centroid based ranking (see Section 3.1.1) to accomplish the first round of ranking.
2. **Pseudo-relevance feedback**: Take all the top n ranked sentences ($n = 10$) for each question from the statistical ranking as labeled definition sentences.
3. **Soft pattern construction**: Generalize the pseudo-labeled training sentences from the previous step into pattern instances and construct the soft pattern vector from the pattern instances.
4. **Second round of ranking (incorporating soft pattern matching)**: Re-rank the sentences by linearly combining the statistical centroid based weights and soft pattern matching weights.

Figure 4.3: The Algorithm for Unsupervised Learning of Soft Patterns

It is worth pointing out that I take all the top ranked sentences from a group of questions as a batch of labeled definition sentences which are fed into the pattern generalization module, instead of generalizing patterns from the results of one question. It makes the "blind" labeling process more reliable by constructing large training set to combat data sparseness.

One assumption here is that the top ranked list actually contains enough definition sentences that can be used to obtain good patterns. The other important assumption for group based PRF to work effectively is that the definition patterns derived from different questions are similar, which is reasonable for the domain of news. Thus, although some of the top ranked sentences for each search term are not definitional, the effects of such errors would be mitigated by performing PRF and pattern generalization over the entire group. Moreover, in journalistic text, descriptive sentences often contain essential information about the search term. Therefore some of the definition sentences will rank high by the centroid based method. This is supported by my experiments on TREC data. I observed that 33% of the top ten ranked sentences over a question set of 50 questions from TREC-13 were actually definition sentences (165 of 500). While a 33% accuracy rate may seem low, it is still better than the baseline for performing PRF in (Buckley et al., 1994). The experimental results in later sections show that the use of PRF significantly improves the quality of the resulting soft patterns.

4.5 Evaluations

I report on two separate evaluations to show the effectiveness and adaptability of the soft pattern matching system. The soft patterns are either learned without supervision by adopting group pseudo-relevance feedback or learned through supervision by training on a corpus of crawled news articles. Before coming to discussing the evaluation results, I will first present the evaluation setup, which includes data sets, comparison systems and evaluation metrics.

4.5.1 Data Sets

I employ the TREC-12 definitional question answering data set (Voorhees, 2003b) which includes a question set comprising 50 questions and answer judgments. In addition to the test data, I also accumulated a set of online news articles as training data. The reason of employing external resources for training is that TREC-12 evaluation is the first to adopt definitional QA task, and thus we lack training data from previous data sets. In the next chapter, I will present evaluation results based on the training data coming from preceding tasks. To construct the training corpus, I collected 26 questions about people and other terms from the Lycos search engine, which were the most popular queries issued by users, during a day in September 2003. Most of the questions can be found in the Lycos 50 report[2]. I list the 26 questions in Table A.2 in Appendix A. The questions were submitted to Google to retrieve news articles from eight news sites, including BBC, CNN and USAToday. I set the limit for the number of pages downloaded from each site to 200. The text body of the news pages, embedded between the HTML tags `<P>` and `</P>`, is extracted and preprocessed in the same fashion as was done to the TREC articles.

I asked seven subjects to label all definition sentences. The subjects are postgraduates in computer science. Four of them are native English speakers. They were assigned different groups of sentences and each sentence had two people to label. I keep only those sentences labeled by both assessors as definition sentences. The labeled sentences are processed into 596 positive and 15,442 negative training instances. This corpus of crawled news articles is denoted as "Web corpus".

4.5.2 Comparison Systems Using Hard Matching Patterns

In order to compare the performance of soft pattern matching with hard matching patterns, I employ two systems that use hard pattern rules from either manually construction or machine learning. For these pattern rules, hard matching is performed to match test sentences to the rules.

4.5.2.1 The HCR System

I use the system we developed for the TREC-12's definitional question answering task (Cui et al., 2004b). As the system employed hand-crafted rules, I denote it as HCR. The rules (listed in Table 4.2), partly derived from the previous work (Liu,

[2] http://50.lycos.com

Chin, and Ng, 2003; Harabagiu et al., 2000), were carefully constructed for the TREC corpus. Specifically, HCR differs from the soft matching system in that: (i) it utilized hand crafted rules as in other existing work, instead of the soft pattern matching described in this work; and (ii) it uses regular expressions to match the rules. The system was ranked second according to TREC-12 evaluation, with the F_5 measure of 0.473. Thus I have good reason to believe that HCR is representative of top performing systems in answering definition questions.

Table 4.2: Manually Constructed Rules Used in HCR.

ID	Regular expressions of rules
1	`<TARGET> (who | which | that)* (is | are) (called | known as)*`
2	`<TARGET> , (a | an | the)`
3	`<TARGET> (is | are) (a | an | the)`
4	`<TARGET> , or`
5	`<TARGET> (- | :)`
6	`<TARGET> (is | are) (used to | referred to | employed to | defined as | described as)`
7	`‘‘(.+)’’ by <TARGET>`
8	`(called | known as | referred to) <TARGET>`

4.5.2.2 Hard Pattern Rule Induction by GRID

In addition to manually constructed definition patterns, in the evaluations, I also compare against the system that uses automatically induced rules. Machine-learned patterns may do better at recall by learning from large-scale training data. Machine induced rules are widely used for information extraction (Muslea, 1999). To adapt a rule induction system for information extraction to definition pattern learning, I apply GRID (Xiao, Chua, and Cui, 2004), a state-of-the-art supervised rule induction algorithm. I select GRID for two reasons. First, unlike other rule induction algorithms that start with seed rules (Riloff, 1996) or randomly selected instances (Soderland, 1999), GRID uses corpus-wide distribution statistics to start

```
1. <TARGET> , DT NN
2. <TARGET> , DT NNP
3. <TARGET> , who won
4. <TARGET> , (known | listed) as
5. who BE <TARGET> 's
6. <TARGET> BE DT NN
```

Figure 4.4: Sample Rules Generated by GRID.

the rule induction process. This is likely to fit well with the diversity in definition patterns. Second, GRID utilizes both tokens and coarse-grained tags (*e.g.*, POS and phrase level tags) in learning rules. The rules learned by GRID are represented as regular expressions. I run GRID over the generalized pattern instances from the labeled definition sentences of the Web corpus to induce definitional pattern rules. In total, there is a set of 100 rules generated by GRID. An excerpt of the generated rules is shown in Figure 4.4.

4.5.3 Evaluation Metrics

In order to get comparable evaluation results, I adopt the same evaluation metrics as used in TREC definitional question answering task (Voorhees, 2003a). For each question from the TREC corpus, there is a list of "vital" nuggets and "okay" nuggets for answering this question provided by TREC. Vital nuggets represent the most important facts about the target and should be included in a definition. Okay nuggets contribute to relevant information but are not essential. Using the given answer nuggets as a gold standard, an individual definition is scored using nugget recall (NR) and an approximation to nugget precision (NP) based on length. Long definition answers will be penalized in precision. These scores are combined using the F_β measure with recall being β times as important as precision. I list the official definition of these metrics in Table 4.3. Note that TREC-12 employed F_5 ($\beta = 5$) measure while the subsequent TREC employed F_3 ($\beta = 3$) measure. To be consistent, since I use TREC-12 data set in this evaluation, I list both F_3 and F_5 scores, but will use only F_3 measure afterwards.

Table 4.3: TREC Definition of NR, NP and F_β Measure

r	number of vital nuggets in the system response
R	number of vital nuggets in the gold standard
a	number of okay nuggets in the system response
l	length of the system response

$$NR = \frac{r}{R}$$

$$NP = \begin{cases} 1 & if\ l < 100 \times (r + a) \\ 1 - \frac{l - 100 \times (r+a)}{l} & otherwise \end{cases}$$

$$F_\beta = \frac{(\beta^2 + 1) \times NR \times NP}{\beta^2 \times NP + NR} \qquad \beta = 3,\ 5$$

4.5.4 Effectiveness of Unsupervised Learned Soft Patterns

In this evaluation, I compare the system that uses unsupervised learned soft patterns (SP) by adopting GPRF against the HCR system on the 50 TREC questions. To illustrate the significance of definition patterns, the baseline system uses only the centroid based method to rank sentences. In the SP+GPRF system, 683 pattern instances are extracted from the 500 blindly labeled definition sentences. I vary the window size L from 1 to 5 in soft patterns extraction and matching to study the impact of the distance of contextual slots from the search target. The results of NR, NP and both F_5 and F_3 measures are listed in Table 4.4.

As shown in Table 4.4, we see significant improvements obtained by both the HCR and SP+GPRF systems over the baseline statistical method, with the maximum improvement of 13.45% and 30.03%, respectively, for F_3 measure. It

Table 4.4: Comparison of NR, NP, F_3 and F_5 measures. Percentage of improvement over the baseline is shown in the brackets.

Systems	NR	NP	F_3	F_5
Centroid (Baseline)	0.463	0.169	0.394	0.423
HCR	0.514 (+11.05%)	0.206 (+22.05%)	0.447 (+13.45%)	0.472 (+11.52%)
SP+GPRF ($L = 1$)	0.561 (+21.14%)	0.206 (+21.78%)	0.479 (+21.34%)	0.507 (+19.65%)
SP+GPRF ($L = 2$)	0.601 (+29.74%)	0.221 (+30.94%)	0.513 (+30.03%)	0.539 (+27.20%)
SP+GPRF ($L = 3$)	0.579 (+25.16%)	0.217 (+28.24%)	0.496 (+25.82%)	0.531 (+25.37%)
SP+GPRF ($L = 4$)	0.551 (+19.05%)	0.204 (+20.82%)	0.471 (+19.40%)	0.495 (+16.97%)
SP+GPRF ($L = 5$)	0.557 (+20.33%)	0.204 (+20.45%)	0.475 (+20.40%)	0.484 (+14.35%)

shows that both the hand-crafted hard-coded rules as well as the automatically learned soft patterns are effective in selecting definition sentences. This is in line with my assumption that news articles define a term or person using some textual patterns.

We also see that a window size of 2 performs the best. This shows that definition patterns tend to be restricted to the tokens adjacent to the search term. The performance of the system drops when the window size reaches 4 or greater. Although a larger window size takes more contextual information into account, I believe it also introduces more noise in the distant slots. As phrase chunking and word omission have been done in the soft pattern generation process, I believe that the resulting small windows capture sufficient context.

The unsupervised SP+GPRF system also outperforms the labor intensive HCR system. Over a man-month of time was used to develop the hand-crafted rules through continuous cycle of system coding and performance analysis. Despite a slight drop in precision for some window size settings, the recall and both F_3 and

F_5 measures obtained using our techniques are better than those by HCR, with a maximum improvement of 16.83% for recall and 14.77% for F_3 measure with the window size of 2. A paired t-test gives the p values for the improvements in recall and F_3 measure as 0.069 and 0.007, respectively. I attribute such improvement to the soft matching patterns which are more flexible than hard coded crafted rules and thus are more adaptable to diversified patterns reflected in news. Additional benefit comes from the feasibility of applying GPRF to automatically labeling definition sentences for pattern discovery.

4.5.5 Comparison with Hard Matching Patterns

In this evaluation, I compare the supervised learned soft patterns with hard matching patterns in the definitional QA system. I have two hypotheses concerning the use of definition patterns: (a) Manually-constructed patterns ought to be of high precision but low recall, due to the difficulty in enumerating an exhaustive specification of definition patterns. Machine-learned patterns may do better at recall by learning from large-scale training data. (b) Soft matching patterns should outperform hard matching systems no matter whether the hard patterns are manually constructed or machine learned. To validate these hypotheses, I conduct a series of experiments using the TREC corpus.

Here, I again use HCR as a baseline system. In the second configuration, I replace the manually constructed rules by a set of 100 hard rules in regular expressions generalized by the GRID algorithm over the Web data training set. This set of hard rules is denoted as "GRID HP". The third test explores the use of soft patterns derived from all manually labeled definition sentences from the Web corpus. The resulting group of soft patterns is denoted as "Supervised SP".

To combine statistical weighting with pattern matching, I apply different strategies to hard matching rules and soft patterns: As the match is binary for manually constructed rules and generalized hard rules by GRID, the weight of any sentence that matches a rule has its score multiplied by a constant factor g, which is set to 2; this is the optimum setting that I have ascertained in my validation experiments by varying the setting from 1.2 to 3. When applying soft pattern matching, the sentences are re-ranked by the linear combination of statistical and pattern matching weights. I weight evidence from pattern matching higher (0.6 for pattern matching weight and 0.4 for statistical weight) because I believe that patterns are better able to identify definition sentences.

Table 4.5: Comparison with Hard Patterns. Percentage of improvement over the baseline is shown in the brackets.

Use of Patterns	NR	NP	F_3	F_5
HCR (Baseline)	0.514	0.206	0.447	0.472
GRID HP	0.536	0.222	0.470 (+5.15%)	0.498 (+6.56%)
Supervised SP	0.630	0.243	0.544 (+21.70%)	0.560 (+19.92%)

The evaluation results are presented in Table 4.5. I make the following observations:

1. Machine learned patterns outperform the manually constructed ones. As many of the TREC top-performing systems use manually constructed patterns, they are likely to benefit from automatic pattern learning. We see improvements of 5.15% and 6.56% in both F_3 and F_5 measures over the manually constructed rules when using the generalized hard patterns generated by GRID. When applying the soft matching patterns over the supervised training pattern instances, the improvement rises significantly to 21.70% and 19.92%. This validates my hypothesis that manually constructed rules are often limited in recall. I expect a larger performance gain with more training instances. This will be validated in the evaluations discussed in the next chapter.

2. Soft patterns significantly outperform machine-learned hard patterns. Applying soft patterns over the supervised Lycos pattern instances, the system performs 12.53% better than when using GRID generalized hard rules in F_5 measure. This improvement is statistically significant ($p < 0.01$). I conjecture that soft patterns can better capture infrequent definition patterns as they use all positive instances in the construction of a flexible probabilistic model. Hard-matching rule induction systems may ignore such infrequent data. In addition, strict slot-by-slot matching may miss some positive instances that exhibit minor variations in expressions, which are common to definitions. Soft patterns thus provide a mechanism to overcome these problems.

4.5.6 Additional Evaluations on the Use of External Knowledge

In addition to the experiments reported in this chapter, I conducted more thorough experiments on the use of external knowledge in answering definition questions (Cui et al., 2004a). As described in Section 3.1, in the definitional QA system, the statistical ranking component leverages evidence about the search target from both the corpus and external resources, such as WordNet (Fellbaum, 1999) and the Web. The statistical ranking component identifies significant terms that bear central information on the search target to locate relevant sentences. In order to find more accurate terms to describe the target, this component often employs a variety of external resources to find the basic definition of the target. As the effects of external knowledge deviates from the main theme of soft matching in this thesis, I only summarize the main results here. Part of the experimental results can be found in Tables B.1 and B.2 in Appendix B.

1. Specific Web resources are more useful than general Web resources in helping find more definition sentences. I further divide the external resources into

two categories – general resources, provided in the form of Google snippets and WordNet definitions; and task-specific resources, in the form of existing definitions from Answers.com. The main problem is generic resources provide lower coverage of the search targets and usually offer only relevant information about the targets, instead of direct definitions.

2. Reinforced by using external resources, the performance of GPRF-based unsupervised labeling is comparable to that of supervised learning. This shows that as external knowledge augments the precision of statistical ranking, the quality of blind feedback is improved, and thus it benefits the unsupervised learning of soft patterns.

4.6 Conclusion

In this chapter, I have discussed a set of techniques of how to generalize definition sentences into pattern instances and how to construct soft patterns out of the pattern instances. Soft pattern matching is better suited for capturing the diversity of definition patterns in news. I also introduce the application of group pseudo-relevance feedback (GPRF) to perform automatic labeling of training instances from ranked results. My contribution here is to use GPRF over a large set of input questions to counter noise and data sparseness. The automatically labeled definition sentences are utilized to generalize soft patterns. I conducted two experiments in evaluating soft pattern matching - one based on unsupervised learning by using GPRF and the other on supervised training data. I compared soft pattern matching with hard matching pattern rules, which are manually constructed and automatically generalized by machine learning. The experimental results show that machine learning methods for pattern generation outperform manually constructed patterns used by most current definitional QA systems. More important, soft pattern matching significantly outperforms hard matching rules due to the flexibility in dealing with variations in natural language.

I have shown a GPRF based unsupervised learning scheme for soft pattern construction. In the next chapter, I will present formal soft pattern models using supervised learning by adopting manually labeled training data. However, it is natural to employ the unsupervised learning algorithm to produce more training data as supplementary to the manually generated one in future work.

Chapter 5

Two Formal Soft Pattern Matching Models

In the previous chapter, I presented a basic soft pattern matching model, which aims to overcome the problem of mismatch in pattern matching due to language variations. While the soft matching method was shown to significantly outperform hard matching patterns in a definitional QA system, it computes the degree of match in an ad-hoc manner, and has not been anchored in a theoretically sound framework. In this chapter, I propose two soft matching models to address this problem: one based on bigrams and the other on the Profile Hidden Markov Model (PHMM). Both models provide a theoretically sound method to model pattern matching as a probabilistic process that generates token sequences. I will demonstrate the effectiveness of the models on recent TREC data. The experimental results show that both models significantly outperform state-of-the-art manually constructed hard matching patterns and the previously proposed basic soft pattern matching method, which is not optimized for parameter estimation.

A critical difference between the two models is that the PHMM has more complex topology. As such, it is expected to be able to handle language variations more effectively but requires more training data to converge. I verify this hypothesis experimentally.

In the next sections, I present the two soft pattern models: the bigram model and the Profile HMM, respectively. Like the aforementioned basic soft pattern model, both soft matching models perform training and testing on the basis of pattern instances, which are abstract token sequences representing the original sentences. As the generalization of definition sentences into pattern instances has been discussed in Section 4.1, I proceed directly to present the models. After the discussion of models, I come to the evaluations and discussions.

5.1 Bigram Model

The first soft pattern model I introduce is based on n-gram language models. Language modeling has been extensively studied in speech recognition, part-of-speech tagging and syntactic parsing (Rosenfeld, 2000). N-gram language modeling is one important approach which models local sequential dependencies between adjacent tokens. Trigrams ($n = 3$) are a common choice when large training corpora are available. I use a bigram ($n = 2$) model for soft pattern matching, as there is only a limited amount of training data available. I also remedy problems with sparse data by smoothing n-gram probabilities.

While the original bigram model is simply a product of probabilities of all bigrams in a sequence, I apply linear interpolation (Manning and Schütze, 1999) of unigrams and bigrams to represent probability of bigrams. The reason is two-fold: (1) to smooth probability distribution in order to generate more accurate statistics for unseen data, and (2) to incorporate conditional probability of individual tokens appearing in specific slots. In particular, I model a sequence of pattern tokens as:

$$\begin{aligned}\Pr(t_1 \ldots t_L) &= \Pr(t_1|\mu)\prod_{i=2}^{L}(\lambda \Pr(t_i|t_{i-1}|\mu) + (1-\lambda)\Pr(t_i|\mu)) \\ &= \Pr(t_1, S_1)\prod_{i=2}^{L}(\lambda \Pr(t_i|t_{i-1}) + (1-\lambda)\Pr(t_i, S_i)) \qquad (5.1)\end{aligned}$$

where μ stands for the bigram model and $\Pr(t_i, S_i)$ stands for the probability of token t_i appearing in slot S_i. λ is the mixture weight combining the unigram and bigram probabilities. Note that I use the conditional probability of a unigram being in a slot to represent unigram probability. This is because the position of a token is important in modeling: for instance, a comma always appears in the first slot right of the target in an appositive expression. Incorporating individual slots' probabilities enables the bigram model to allow partial matching, which is a characteristic of soft pattern matching. In other words, even if some slots cannot be matched, the bigram model can still yield a high match score by combining the matched slots' unigram probabilities.

As test instances are often different in length, I normalize the log-likelihood of Equation 5.1 by the length l of the test instance:

$$P_{norm}(t_1 \ldots t_l) = \frac{1}{l}(\log \Pr(t_1, S_1) + \sum_{i=2}^{l}\log(\lambda \Pr(t_i|t_{i-1}) + (1-\lambda)\Pr(t_i, S_i)) \quad (5.2)$$

where l denotes the number of tokens in the test instance, which could be smaller than the model length L.

Next, unigram and bigram probabilities are estimated by their maximum likelihood (ML) estimates:

$$P_{ML}(t_i, S_i) = \frac{|t_i(S_i)|}{\sum_k |t_k(S_k)|} \qquad (5.3)$$

$$P_{ML}(t_i|t_{i-1}) = \frac{|t_i(S_i)t_{i-1}(S_{i-1})|}{|t_{i-1}(S_{i-1})|} \qquad (5.4)$$

where $t_i(S_i)$ denotes that token t_i appears in slot S_i and $|t|$ denotes the frequency of the token t. In language modeling, ML estimates often suffer from the sparse data problem. This is exacerbated in my scenario as I count tokens with respect to slot positions, which makes the training data more sparse. As such, I employ smoothing to counter this problem. For simplicity, I use Laplace smoothing on unigram probabilities (recall that bigram probabilities have already been smoothed by interpolation):

$$\Pr(t_i, S_i) = \frac{|t_i(S_i)| + \delta}{\sum_k |t_k(S_k)| + \delta |N(t)|} \tag{5.5}$$

where $|N(t)|$ gives the total number of unique tokens in my training data and δ is a constant, which is 2 in my experiments. Note that as described in the basic soft matching model, I count the general syntactic tags and specific words separately according to their types in order to avoid general tags skewing the distribution over specific words due to overwhelming frequency counts.

5.1.1 Estimating the Mixture Weight λ

I use Expectation Maximization (EM) (Dempster, Laird, and Rubin, 1977) to find optimal settings of λ. Specifically, I estimate λ by maximizing the likelihood of all training instances (the training set is denoted as INS) given the bigram model. The estimation process is as follows:

$$\begin{aligned}\lambda &= argmax_\lambda \sum_{j=1}^{|INS|} \log \Pr(t_1^{(j)} \dots t_{l(j)}^{(j)} | \mu) \\ &= argmax_\lambda \sum_{j=1}^{|INS|} \frac{1}{l_{(j)} - 1} \sum_{i=2}^{l_{(j)}} \log\left(\lambda \Pr(t_i^{(j)} | t_{i-1}^{(j)}) + (1-\lambda) \Pr(t_i^{(j)}, S_i^{(j)})\right) \end{aligned} \tag{5.6}$$

where $t_i^{(j)}$ stands for the i^{th} token in the j^{th} training instance. $\Pr(t_1, S_1)$ is ignored because it does not affect the estimation of λ. λ can be estimated using the EM iterative procedure:

1. Initialize λ to a random estimate between 0 and 1, say 0.5.

2. Update λ using:

$$\lambda' = \frac{1}{|INS|} \times \sum_{j=1}^{|INS|} \frac{1}{l_{(j)} - 1} \sum_{i=2}^{l_{(j)}} \frac{\lambda \Pr(t_i^{(j)}|t_{i-1}^{(j)})}{\lambda \Pr(t_i^{(j)}|t_{i-1}^{(j)}) + (1-\lambda)\Pr(t_i^{(j)}, S_i^{(j)})} \tag{5.7}$$

   where $|INS|$ is the number of training instances, which is used as a normalization factor.

3. Repeat Step 2 until λ converges.

I set λ to 0.3 according to the experimental results.

Recall that when combining the pattern matching scores of the left and the right sequences to the search target previously discussed in Equation 4.5, I set α to 0.7 based on my observations. I can now apply EM to find an optimal value of α.

Note that the bigram model is actually a generalization of the basic soft matching model described before. The bigram model also captures the two metrics of soft matching – namely, the individual slot match degree and sequential fidelity – by using unigram probability and bigram probability. More importantly, the bigram model provides a theoretically sound way to present the process of lexico-syntactic pattern matching, as well as a set of systematic techniques to optimize parameter estimations.

5.2 Profile Hidden Markov Model

Although the bigram model allows partial matching, it lacks the ability to deal with gaps in test instances. For instance, given training instances such as "`<TARGET> which is known for ...`", the trained bigram model cannot give reasonable match scores to test instances such as "`<TARGET> which is best known for ...`" or

"`<TARGET> , whose xxx is known for ...`" even though they are simple variants of the training instances in which insertions or deletions occur. The gaps can be captured by Profile HMMs (Durbin et al., 1999), which allow insertion and deletion editing operations in the matching process. Figure 5.1 shows the topology of a PHMM.

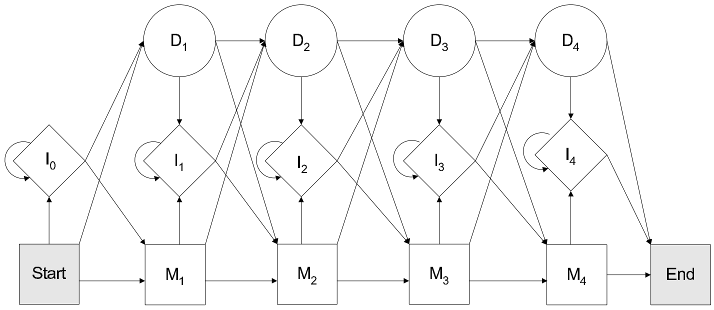


Figure 5.1: Illustration of Topology of the PHMM Model

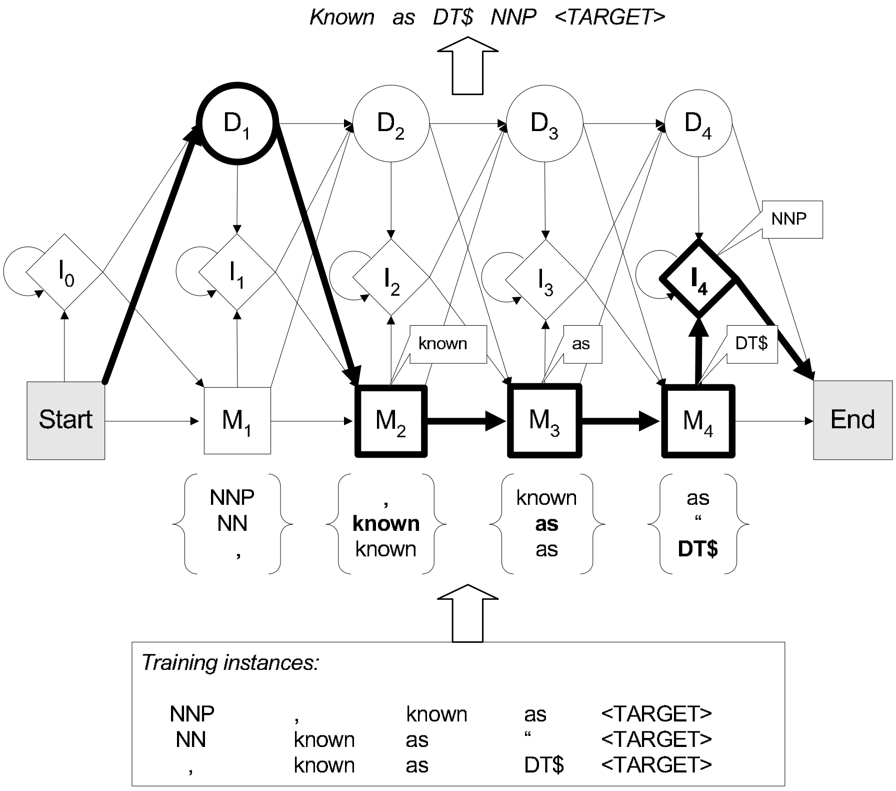
The PHMM contains a sequence of match states, which are denoted by M_i ($i = 1 \ldots L$). These match states correspond to slots in pattern instances and determine model length L. Each match state can emit a token t from all tokens in the training instances with the emission probability $\Pr(t|M_i)$. For each match state, there is a deletion state, denoted by D_i, which does not emit a token and is used to skip the corresponding match state. Insertion states can emit any token t with the emission probability $\Pr(t|I_i)$. Insertion states insert tokens after match or deletion states, as with the word *best* in the earlier example. While transitions from match states and deletion states always move forward in the model, insertion states allow self-loops, corresponding to multiple insertions. A token sequence representing a pattern instance can be generated by moving through this model with state transition probabilities $\Pr(S_i|S_j)$. The deletion and insertion states allow the PHMM to model missing or unobserved words in training. Specifically, the probability of a sequence of tokens $t_1 \ldots t_N$ that are generated by moving through the states $S_0 \ldots S_{L+1}$ (the start and end states are S_0 and S_{L+1}) is represented by the following joint probability of the token sequence $t_1 \dots t_N$ and the state sequence $S_0 \dots S_{L+1}$:

$$\Pr(t_1 \dots t_N,\ S_0 \dots S_{L+1}, \mu) = T(S_{L+1}|S_L) \prod_{i=1}^{L} \Pr(t_{n(i)}|S_i) T(S_i|S_{i-1}) \tag{5.8}$$

where μ stands for the model. $\Pr(t_{n(i)}|S_i)$ is set to 1 when S_i is a deletion state. To recognize a definition pattern, I choose the most probable state path. Equation 5.8 can be efficiently calculated by the forward-backward algorithm (Manning and Schütze, 1999). I employ the Viterbi algorithm (Manning and Schütze, 1999) to find the most probable state path. In Figure 5.2, I show an example to illustrate how the PHMM finds the optimal path to account for the "gaps" between training instances and the test instance. Although the training data does not contain any instance that has "known" in slot 1 and "NNP" in slot 4, the PHMM automatically selects the path that goes through a deletion state to skip slot 1 and uses an insertion state to emit "NNP". Thus, the tokens are re-aligned with their most probable occurring slots such that the unseen test instance can still obtain a reasonable generative probability.

5.2.1 Estimation of the Model

During training, I need to estimate transition and emission probabilities for the PHMM. The training process, also called the estimation process, can be accomplished by employing the standard Baum-Welch algorithm (Manning and Schütze, 1999). Corresponding to my adaptation to the calculation of sequence probability, I use the Viterbi algorithm to determine the path with the highest probability during the re-estimation process, unlike the standard Baum-Welch algorithm which considers all possible paths which are weighted by their probabilities.



Training instances:

NNP	,	known	as	<TARGET>
NN	known	as	“	<TARGET>
,	known	as	DT$	<TARGET>

Figure 5.2: Illustration of Generating a Test Instance with Gaps Using the PHMM. Optimal path in bold; words or tags emitted shown in callouts.

5.2.2 Initialization of the Model

Although probabilities in a PHMM can be estimated automatically using an iterative EM algorithm starting with random or uniform probabilities, the re-estimation process can only guarantee that the model reaches local maxima. In addition, in capturing definition patterns, definition expressions are diverse and sparse in terms of both lexical tokens and POS tags. If I start with random or uniform setting of the model, it is likely to end up with an unsatisfactory model that gives close estimates of different possibilities. To make training manageable given my small training set, I assume that the most probable state path for a sequence should go through as many match states as possible. The reason is that although insertion and deletion states add flexibility, they may hurt generalization of underlying definition patterns if the model gives high probabilities to them. Specifically, I set the emission probabilities for each match and insertion state using the smoothed maximum likelihood estimate of the emission probabilities (Equation 5.5). I apply a discounting parameter, which is 0.2 obtained through a small-scale experiment, to lower the initial value of $\Pr(t|I_i)$ such that the probability of emitting a token from match states is always higher than that from insertion states. I set the initial state transition probabilities to the inverse proportion of the number of transition links from a state.

5.3 Evaluations

I have evaluated the proposed methods in an extensive series of experiments using the TREC question answering datasets (Cui, Kan, and Chua, 2004; Cui, Kan, and Chua, 2005), and extend my experiments to test for soft pattern models' robustness to scalability and to report soft pattern matching results as measured by recent automated metrics. More specifically, my evaluation goals are: (1) to re-affirm the conclusion of soft pattern (SP) models outperform hard pattern matching on a larger test set which includes the latest TREC-14 data; (2) to test if the formal soft matching models are superior to the basic SP model; (3) to verify my hypothesis that the PHMM is able to yield better performance over the bigram model, given more training data; and (4) to assess the performance of the soft pattern models using the automatic metrics ROUGE and POURPRE. The latter was recently proposed and specifically designed for automatic evaluation of definitional QA systems.

I first discuss the experimental setup, then report on the evaluation results. The evaluations come in two parts: I first show preliminary evaluations on the basic characteristics of the models – to examine both models' sensitivity to the setting of model length and to compare them with the basic soft matching model;

I then present the main experiment which evaluates the two formal SP models on two TREC datasets. I complete the section with a discussion on the performance differences between the two SP models.

5.3.1 Evaluation Setup

5.3.1.1 Data Set

I still employ the TREC Question Answering Task dataset for my experiments. Here, I employ over 200 definition questions, which include questions from recent TREC-13 and TREC-14. For training, I use the TREC-12 and TREC-13 data, consisting of 114 definition question-answer pairs. Based on the answer nuggets (ground truth, manually edited data) provided by TREC for these questions, I manually label 1,769 sentences that cover the nuggets from the corpus as training definition sentences for estimating the soft matching models.

5.3.1.2 Evaluation Metrics

I adopt three metrics to evaluate definitional QA performance: F_3, POURPRE and ROUGE. The F_3 measure is based on a manual examination while the latter two metrics are automatically evaluated. Automatic scores are a good supplement to manual evaluations for two reasons. First, as Lin and Demner-Fushman (2005a), and Xu et al. (2004) suggested, POURPRE and ROUGE are highly correlated with manual counting of nuggets. Second, manual evaluation can often be inconsistent across runs (Voorhees, 2003b).

F_3 **Measure:** I first adopt the evaluation metrics used in the TREC definitional question answering task (Voorhees, 2004). Along with each topic, TREC provides a list of answer keys (nuggets) to evaluate system responses. Answer nuggets are labeled as either *vital* or *okay* (see Figure 2.2). As described in Section 4.5.3, vital nuggets represent the most important facts about the target and should be included in a definition. Okay nuggets contribute to relevant information but are not essential. Here, I make changes to the construction of gold standards as the recent guidelines from TREC are different from that used in TREC-12. From TREC-13, the definitional nuggets are designed for answering questions not covered by answers to the factoid or list questions about the target. I use these nuggets plus the nuggets reflected by factoid and list questions to assess the definitional QA systems in my evaluation. The nuggets entailed by factoid and list questions are deemed "vital" (see Section 5.3.1.3). In the manual assessment used in official TREC evaluations, an assessor examines how many vital and okay nuggets are covered in the returned answer. Each definition is scored using nugget recall (NR) and an approximation to nugget precision (NP) based on answer length. These scores are combined using the F_3 measure with recall being weighted three times as important as precision. I listed the official definition of NR, NP and F_3 measures previously in Table 4.3.

POURPRE: Lin and Demner-Fushman (2005a) proposed the POURPRE metric specifically for evaluating definitional QA systems. POURPRE simulates the process of manual checking of answer nuggets. It counts the answer nuggets that are covered by the system response by examining non-trivial unigrams shared between them. It calculates F_3-like scores by using the automatically determined nuggets. To ensure integrity of the answers, POURPRE counts only the words appearing within the same answer string. POURPRE counts a nugget matched if the system response covers a certain percentage of all the non-trivial unigrams in the gold standard. I set the covering percentage to 25% in my evaluations.

ROUGE: ROUGE (Lin and Hovy, 2003) is a metric originally designed for summarization evaluation and has previously been adapted for definitional QA evaluation by Xu et al. (2004). I use the metric ROUGE-3, which was adopted by Xu et al. and counts the trigrams shared between the official answer and the system answer.

5.3.1.3 Gold Standard for Automatic Scoring

To perform automatic scoring by POURPRE and ROUGE, I construct a gold standard inventory of sentences that contain answer nuggets provided by TREC.

For each search target, I construct two groups of gold standard answers, ALL and VITAL, which are analogous to the TREC answer nuggets but are at the sentence level. As stated in Section 5.3.1.2, TREC currently restricts definitional QA evaluation to "other" questions. As my purpose is to evaluate definitional QA systems independently of factoid and list QA systems, I include such factoid and list questions answers in the gold standard definitional answers. The VITAL group consists of answer nuggets to the factoid and list questions, as well as *vital* nuggets to the "other" questions. The ALL group is a superset of VITAL group, which adds the *okay* nuggets to the "other" questions. Here, factoid and list answers are deemed *vital* nuggets, as the TREC guidelines state that factoid and list questions should embody essential information about the target.

For each answer nugget, I retrieve up to five sentences that reflect that nugget as the gold standard. This is because the answer nugget may be embedded in different sentences, possibly realized with different vocabulary. Accordingly, I construct five groups of sentences as gold standard answers for each target.

The final scores are the average scores obtained by running the evaluation tools over the five groups of gold standard sentences. Sample gold standard sentences for TREC topic #72 are given in Table 5.1.

Table 5.1: Gold Standard Sentences for the Topic 72 “Bollywood”. This is one of the five groups of gold standard sentences. The third column indicates from what kind of question the nugget is constructed.

1	VITAL/ALL	Factoid	Around 800 movies a year come out of India. The center of the film industry is in Bombay, from which the name Bollywood is derived.
2	VITAL/ALL	Factoid	Organizers said they hoped to gain international recognition for Bollywood, the nickname given to Bombay, which boasts the world’s second largest film industry after Hollywood.
3	VITAL/ALL	List	Shabana Azmi, Amitabh Bachchan, Bobby Deal, Madhuri Dixit, Sanjay Dutt, Sunil Dutt, Kajol, Anil Kapoor, Aamir Khan, Salmon Khan, Shah Rukh Khan, Amisha Patel, Aishwarya Rai, Lisa Roy, Hrithik Roshan, Sushmita Sen, Sunil Shetty
4	VITAL/ALL	Other	television production houses, such as Sony entertainment and star TV, pay huge sums to buy the rights of Bollywood favorites.
5	VITAL/ALL	Other	There was no shortage of glitz and glamor Saturday as Bombay’s biggest films stars were honored at the International India Film Awards, their cinematic equivalent of the Oscars.
6	VITAL/ALL	Other	Any taxi driver picked at random can probably give you a detailed tour of the movie-star homes in Juhu Beach and Malabar Hill _ Bombay’s Malibu and Beverly Hills.
7	ALL	Other	Few Americans have even heard of Bollywood.
8	ALL	Other	A new 30-screen cineplex is dedicating six screens to Bollywood. Three Bollywood movies, known as extravagant productions of epic lengths and lavish musical interludes, entered the United Kingdom’s top 10 list this year.
9	ALL	Other	Hollywood star Richard Gere was honored by Bollywood at an awards ceremony for some of the top stars in India.
10	ALL	Other	A Bollywood version of JANE AUSTEN’S PRIDE & PREJUDICE received its world premiere on Mon.
11	ALL	Other	For Hollywood , poaching Indian film talent and learning from Bollywood ’s efficient , low - cost production techniques may become an economic necessity , as American movie - making costs soar .

5.3.1.4 System Settings

In my experiments, the base definitional QA system used is illustrated in Figure 3.2. Many other factors may affect the performance of a definitional QA system, such as the answer length and how external knowledge is utilized. I focus on the effectiveness of pattern matching for definitional QA, and thus I fix the configuration of the systems as described in Section 3.1, varying only the module of definition pattern matching. For comparison, I apply a set of manually constructed hard matching definition patterns which have demonstrated state-of-the-art performance as the baseline. The definition patterns used here are a combination of recent ones from Cui et al. (2004) and Hildebrandt et al. (2004), which comprise the most complete published list of patterns to my knowledge. This set of definition patterns are more complete than those used in the previous chapter, listed in Table 4.2. I list the hard patterns in Table 5.2.

I set answer length N to 14 sentences for all systems to approximate the desirable answer length used in other successful TREC systems.

Since most of the parameters are estimated during the training process for soft pattern models, I only need to set model length for both models. In the next section, I will show the evaluation results of the models' sensitivity to model length.

5.3.2 Analysis of Sensitivity to Model Length

In this section, I vary the only arbitrarily set factor - model length - for the two models. For simplicity, I employ only automatic ROUGE scores in this evaluation. I list the ROUGE scores for both models when varying their model length (number of slots) from 2 to 6 in Table 5.3.

In Table 5.3, we see that the bigram model obtains the best performance with the model length of 3 while the PHMM achieves the highest performance with the model length of 4. Both models slacken in their performance when more slots

Table 5.2: Hard Definition Patterns Used in the Baseline System

```
<TARGET> , (a|an|the)
<TARGET> (is|are|was|were) (a|an|the)
<TARGET> , (also)* (known as|called)
<TARGET> (is|are) (usually|generally|normally)*
(called|known as|defined as)
<TARGET> (refer to|refers to|satisfies|satisfy)
known as <TARGET>
<TARGET> (becomes|become|became)
<TARGET> (.{1,40})
<TARGET> , or
<TARGET> (is|are) (usually|generally|normally)*
(being used to|used to|referred to|employed to|
defined as|formalized as|described as|
concerned with|called)
<TARGET> (-|: )
```

are used.

I also compare percentage change in performance against the highest score for each scoring metric. The performance of the bigram model fluctuates more over different model lengths compared to the PHMM. This is evidence that the PHMM may be more stable amid changes in model typology.

Another observation is that with model lengths of 5 and 6, the PHMM performs better than the bigram model. I believe that the PHMM model may be more capable of dealing with longer contexts.

5.3.3 Comparison to the Basic Soft Matching Model

In this evaluation, I assess the performance obtained by the two soft matching models against the basic soft matching model. I set the model length L to optimal

Table 5.3: ROUGE-3 with Different Model Lengths. The percentage values in parentheses are difference measures compared to the maximum. Note that PHMM SP's minimum length for training is 3.

Model Length (# Slots)	2	3	4	5	6
PHMM (ALL)	N/A	0.2139 (-4.25%)	0.2234	0.2190 (-1.97%)	0.2125 (-4.88%)
PHMM (VITAL)	N/A	0.2369 (-5.09%)	0.2496	0.2422 (-2.97%)	0.2367 (-5.17%)
BIGRAM (ALL)	0.2128 (-7.60%)	0.2303	0.2165 (-6.00%)	0.2152 (-6.56%)	0.2086 (-9.42%)
BIGRAM (VITAL)	0.2340 (-8.34%)	0.2553	0.2363 (-7.44%)	0.2354 (-7.80%)	0.2346 (-8.11%)

values based on experiment which I have presented in the previous subsection. I use the system that uses the basic soft matching method as the baseline. This evaluation is based on the test data of TREC-13 and the soft matching models are trained on TREC-12 data. I list the evaluation results based on F_3 and automatic $ROUGE$ scores in Table 5.4.

Table 5.4: F_3 and ROUGE Performance Comparison (percentage improvement shown in brackets).

Configurations	Basic SP (Baseline)	Bigram SP	PHMM SP
NR	0.5376	0.5519 (+2.66%)	0.5420 (+0.82%)
NP	0.3238	0.3403 (+5.10%)	0.3264 (+0.80%)
F_3	0.4937	0.5088 (+3.06%)	0.4971 (+%0.69)
ROUGE (ALL)	0.2233	0.2303 (+3.13%)	0.2234 (+0.05%)
ROUGE (VITAL)	0.2378	0.2553 (+7.36%)	0.2496 (+5.00%)

From Table 5.4, we can see both the bigram model and the PHMM outperform the Basic SP in all scores. The bigram model outperforms the basic SP by

7.36% ($p = 0.09$) and 3.06% ($p = 0.1$) in ROUGE (using *vital* nuggets) and F_3 scores, respectively. The PHMM achieves 5.00% improvement over the basic SP in ROUGE score (using *vital* nuggets). These results show that the basic soft matching method is not optimized in parameter setting. Finding best parameters is often tedious and difficult for such ad hoc systems. In contrast, the bigram model and the PHMM provide a sound framework for parameter estimation. This should facilitate the migration of the two generic soft matching models to other applications.

In this evaluation, one may see that the PHMM does not perform as well as the bigram model. This is because I employ only the TREC-12 data set for training. Using more training data may make the PHMM perform better than the bigram model, which I will present in the main evaluation described in the next section.

5.3.4 Main Evaluation Results and Discussion

In this section, I show the results for the main evaluation of the two soft pattern matching models. I conducted two experiments on two data configurations: one is trained on TREC-12 data and tested on TREC-13 data; the other is trained on the data sets of TREC-12 and TREC-13 and tested on recent TREC-14 data. I use the system that employs the manually constructed definition patterns listed in Table 5.2 as the baseline system. I list the scores of F_3, POURPRE and ROUGE by the systems in Table 5.5.

I conjectured that the PHMM could perform better as it has a more complex topology that could potentially capture more language variations. However, in the evaluation described in the previous section, limited by the small size of training data, I could not demonstrate that the PHMM is able to perform better than the bigram model. Given the additional training data in the second data configuration, I hope to verify this hypothesis.

Table 5.5: Performance Comparison of F_3, POURPRE and ROUGE Scores on TREC-14 Data Set (trained on TREC-13 and 12 data) - percentage of improvement over the baseline is shown in the brackets; ** and * represent different significance levels, $p < 0.01$ and $p < 0.05$, respectively.

System Setting	Hard Pattern (Baseline)	Bigram SP	PHMM SP
NR	0.3042	0.3348 (+10.06%)	0.3527** (+15.97%)
NP	0.1391	0.1509 (+8.48%)	0.1508 (+8.41%)
F_3	0.2628	0.2911 (+10.73%)	0.3035** (+15.45%)
POURPRE (ALL)	0.2400	0.2539 (+5.77%)	0.2556* (+6.49%)
POURPRE (VITAL)	0.3410	0.3668* (+7.57%)	0.3730** (+9.38%)
ROUGE-3 (ALL)	0.0984	0.1004 (+2.05%)	0.1096 (+11.42%)
ROUGE-3 (VITAL)	0.1022	0.1080 (+5.69%)	0.1095 (+7.15%)

Since the results in Table 5.6 are partially shown in Table 5.4 except newly added POURPRE scores, I mainly discuss the results in Table 5.5. I make the following observations from the results:

1. Soft pattern models outperform hard pattern matching. As Tables 5.5 and 5.6 show, both the bigram model and the PHMM perform significantly better than the baseline system using hard patterns in F_3 scores and automatic POURPRE scores. This re-affirms my conclusion drawn earlier that soft pattern models are more capable of identifying definition sentences and boost the performance of definitional QA systems.

   The significance levels of the improvements over the baseline vary across the automatic scores of POURPRE and ROUGE. This may be caused by mismatches in which automatic evaluation methods incorrectly credit system

Table 5.6: Performance Comparison of F_3, POURPRE and ROUGE Scores on TREC-13 Data Set (trained on TREC-12 data) - percentage of improvement over the baseline is shown in the brackets; ** and * represent different significance levels, $p < 0.01$ and $p < 0.05$, respectively.

System Setting	Hard Pattern (Baseline)	Bigram SP	PHMM SP
NR	0.5027	0.5519* (+9.79%)	0.5420* (+7.82%)
NP	0.3159	0.3403 (+7.72%)	0.3264 (+3.32%)
F_3	0.4633	0.5088** (+9.83%)	0.4971** (+7.30%)
POURPRE (ALL)	0.2785	0.2921 (+4.88%)	0.2896 (+3.99%)
POURPRE (VITAL)	0.4238	0.4580** (+8.07%)	0.4528* (+6.84%)
ROUGE-3 (ALL)	0.2106	0.2303 (+9.37%)	0.2234 (+6.08%)
ROUGE-3 (VITAL)	0.2286	0.2553* (+11.67%)	0.2496 (+9.18%)

responses. For instance, the following sentence for the target "DePauw University":

> *. . . will provide* ***scholarships*** *to* ***DePauw University students*** *from* ***Indiana****, Illinois, Michigan and Ohio.*

is mis-matched to the *vital* nugget:

> *Some institutions, like Rhodes College in Tennessee,* ***DePauw University*** *in* ***Indiana*** *and Bucknell University in Pennsylvania, say they allow* ***students*** *to keep 100 percent of outside* ***scholarships****.*

due to the match of non-trivial words in bold. As such, automatic scores are unable to discern between verbose answers which overlap with the gold-standard sentences. I feel that automatic checking of answer nuggets is a good supplement, but not a substitute, for manual checking.

2. Given more training data, the PHMM outperforms the bigram model. I use 1,769 training sentences by combining the labeled definition sentences from TREC 12 and 13, as compared to only 761 training sentences from using only TREC-12 in my previous work. I further complete the experimentation on the hypothesis that the PHMM can achieve better performance than the bigram model given more training data. As seen in Table 5.5, the PHMM outperforms the bigram model by 4.26% in F_3 measure and by 9.18% in ROUGE score based on All nuggets.

3. Evaluation results are dependent on the determination of *vital* and *okay* nuggets. The evaluation scores by both manual and automatic checking on TREC-14 data are lower across the board compared with those from TREC-13 data. In addition, the statistical significance test values (p-values) on the difference between the evaluation scores obtained by systems using the TREC-14 data are less significant than those using the TREC-13 data. I conjecture that this is due to that there are more targets in TREC-14 that have only a few *vital* nuggets. Lin and Demner-Fushman (2005b) studied the gold standard answer nuggets in TREC-12, 13 and 14. They found that 5 targets (out of 75 targets) have only one *vital* nugget and 16 targets have two *vital* nuggets in TREC-14, whereas the corresponding numbers are 2 and 15 in TREC-13 (out of 64 targets). As only *vital* nuggets count for NR, missing any of the few *vital* nuggets causes scores to drop to zero for some targets. As such, I see significant drop in the evaluation scores and a lower level of statistical significance for the results that are obtained from using the TREC-14 test data.

How Much Can the PHMM Help?

While I have shown that given more training data, the PHMM performs better than both hard patterns and the bigram model, I have yet to quantitatively measure how much PHMM can improve over other matching models. Is the PHMM's more flexible matching mechanism actually responsible for its improvement over the bigram model?

To answer this question, I analyze the sentences that are retrieved only by the PHMM and not by the bigram model. In particular, I rank the sentences for each topic using both soft pattern models alone (without centroid ranking). I take the top ranked 50 sentences per topic (which I deem definitional) and get the differing sentences that are in the PHMM's resulting set but not in the bigram model's. In total, I obtain 1,187 unique sentences.

In particular, I want to find the proportion of these sentences that were retrieved by the PHMM specifically by its edit operations. This can be done by checking whether a sentence retrieved by the PHMM was retrieved due to a non-trivial insertion/deletion operation. Here, insertion immediately followed by deletion (or vice versa) is an alternative to matching, and is considered a trivial use of the PHMM states, because that can be simulated in the bigram model. Use of isolated and repeated insertion or deletion states (*i.e.*, non-trivial uses) in the PHMM cannot be represented in the bigram model. I generate the optimal state transition paths calculated by the Viterbi algorithm for the left and the right sequences for each sentence. I count the number of sentences whose optimal state sequences include such non-trivial insertion and deletion operations.

I obtain 194 sentences (or 16.34% out of 1,187 sentences) that are exclusively retrieved by the PHMM by virtue of its edit operations. Such a small percentage of non-trivial state sequences partially explains why there is not a large margin of difference in the performance between the PHMM and the bigram model on my data set. Given a more noisy data set, such as web pages, the PHMM may perform even better because more sentences that cannot be matched within the training data would benefit from insertion and deletion operations.

Furthermore, I break down the 194 sentences into left and right sequences. I observe 160 left sequences and 38 right sequences. It shows that the left context to the search target is more diversified than the right one, and thus it needs to be combined with the use of insertion and deletion states to find the best matching state sequence. In addition, I have 1,218 left pattern instances vs. 1,563 right pattern instances in the training data. As such, with less training data, it is more likely for a left sequence not to be matched by any training instances.

5.4 Conclusion

I have proposed two generic soft pattern models – one based on a bigram language model and the other on the PHMM – to identify definition sentences in a definitional question answering system. Both models provide formal probabilistic methods to model lexico-syntactic patterns represented by token sequences. The experimental results show that both models significantly outperform the system that uses carefully constructed hard matching patterns. In particular, I have shown that the PHMM is more capable of dealing with gaps in pattern matching caused by language variations by performing insertion and deletion editing operations. In order to show the effectiveness of the PHMM, I employ more manually labeled data for estimating the models. The evaluation results show that given more training data, the PHMM can actually perform better than the bigram model due to the diversity of definition patterns. Moreover, I quantitatively analyze how many definition sentences that are retrieved by the PHMM but missed by the bigram model really benefit from the PHMM's special operations of insertion and deletion. However, in my data set, I find only a small amount of sentences that have gaps with the training patterns, which partially explains why the PHMM does not have a large margin over the bigram model.

Chapter 6

Soft Matching of Dependency Relations

In the previous two chapters, I have discussed soft matching of lexico-syntactic patterns, *i.e.*, soft patterns, in definitional question answering. As described in Chapter 3, in my question answering framework, producing a summary of definition for the search target is only the first step. The system is then able to answer specific factoid questions around the target in the subsequent processes. I present a precise sentence retrieval module for answering factoid questions in this chapter.

Most current factoid question answering (QA) systems employ term-density ranking to retrieve answer passages. Such methods often retrieve incorrect passages as relationships among question terms are not considered. Previous studies attempted to address this problem by matching dependency relations between questions and answers. They used strict matching, which fails when semantically equivalent relationships are phrased differently. I implement soft matching of dependency relations on statistical models. I will present two methods for learning relation mapping scores from past QA pairs: one based on mutual information and the other on expectation maximization.

The soft matching of dependency relations embodies the two-anchor soft matching scheme as described in Section 1.1. It takes the matched question terms as anchors and the dependency relations between them as matching units. I do not apply the formal statistical models developed for the one-anchor soft matching scheme presented in the previous chapter for this task due to two reasons: (1) In definition sentence retrieval, my goal is to generalize generic patterns from the training instances. Pattern generation and matching is to learn to classify token sequences. In contrast, relation matching deals with a more ad-hoc scenario as it computes the similarity between two sets of relations regardless of their sequences. We only need to learn pair-wise individual relation similarities from the training data. (2) The length of matching units varies for relation matching. Different from pattern matching for which I fix the model length for capturing contexts within the text window, the numbers of relations between matched words may vary greatly. As such, it is difficult to employ the bigram model or the PHMM to model relation similarities.

This chapter flows as follows. I first discuss how to extract and pair correspondent relation paths (*i.e.*, multiple relations), from a dependency tree. I then present how to calculate the soft matching scores between the correspondent relation paths. I show evaluation results and discussions to finish the chapter. Note that through out the chapter, I mainly discuss and evaluate the soft matching of dependency relations. Soft matching of relations is built on top of a lexical matching system because the relations are anchored by matched words in the sentences. While lexical matching of words also give credits to determining the relevance of sentences, I will not explicitly disucss it in this chapter.

6.1 Soft Relation Matching for Passage Retrieval

To gain more flexibility in incorporating relation matching, I propose a novel soft relation matching method which examines grammatical dependency relations between question terms to improve current passage retrieval techniques for question answering. I employ Minipar to accomplish dependency parsing. I present a statistical technique for measuring the degree of match of pertinent relations in candidate sentences with their corresponding relations in the question. Sentences that have similar relations between question terms are preferred. Specifically, for non-trivial question terms, I collate all single relations between any two terms (or nodes) in the parse tree and I term that as a *relation path*. The overall likelihood of a candidate sentence in terms of dependency relations is the combination of the soft matching scores of all relation paths between the matched question terms.

I also conduct a series of extrinsic experiments to demonstrate the effectiveness of soft relation matching for passage retrieval on the TREC-12 QA task data. When applied on top of standard density-based lexical matching systems, the relation matching method significantly improves these systems by 50 to 78 percent in mean reciprocal rank (MRR). I also examine how two other QA parameters interact with relation matching in passage retrieval: query length and query expansion. A key finding is that longer queries benefit more from utilizing relations. To show better performance, I apply the soft relation matching method to a QA system that is reinforced by query expansion and obtain a further 50% performance enhancement. In addition, I show that a full QA system employing relationship matching reaches the top performance in TREC, without parameter tuning.

Next, I present how relation paths are extracted and paired from parse trees, and then adopt a variation of IBM translation model 1 (Brown et al., 1993) to calculate the matching score of a relation path given another, which combines the mapping scores of single relations in both paths. I present two methods to learn a pair-wise relation mapping model from training data: one is based on a variation of mutual information (MI) that captures the bipartite co-occurrences of two relations in the training data, and the other is based on the iterative training process presented in (Brown et al., 1993) using expectation maximization (EM).

6.1.1 Extracting and Pairing Relation Paths

Relation paths between nodes from dependency trees for sentences generated by Minipar are extracted for comparison. Figure 6.1 illustrates the dependency trees for the sample question and the answer sentence S1 presented in Figure 2.3.

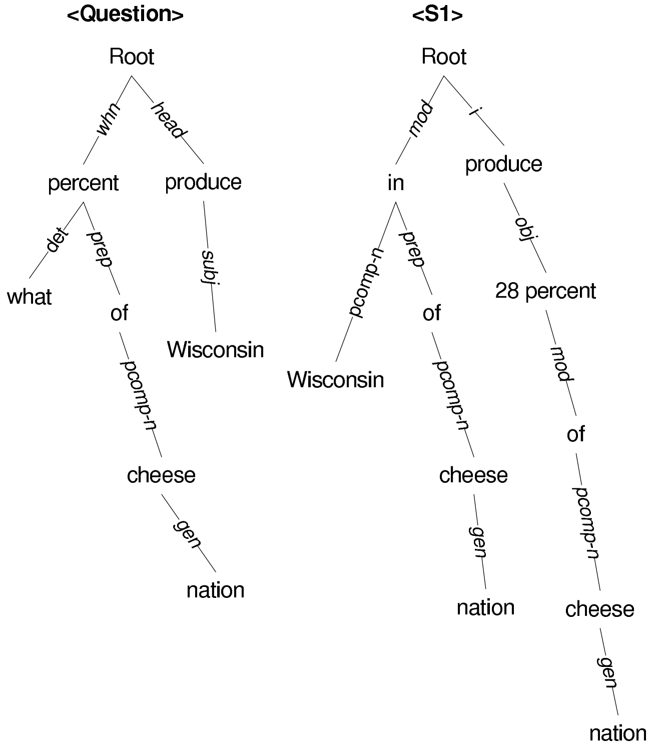


Figure 6.1: Dependency Trees for the Sample Question and Sentence S1 in Figure 2.3. Some nodes are omitted due to lack of space.

In a dependency tree, each node represents a word or a chunked phrase, and is attached with a link representing the relation pointing from this node (the governor) to its modifier node. Although dependency relations are directed links, I ignore the directions of relations. This is because the roles of terms as governor and modifier often change in questions and answers. The label associated with the link is the type of the dependency relation between two nodes. Examples of relation labels (or relations for short) are `subj` (subjective), `mod` (modifying) and `pcomp-n` (nominal complement of a preposition). There are 42 such relation labels defined in Minipar.

I further define a *relationship path* (or simply *path*) between nodes n_1 and n_2 as the series of edges that traverse from n_1 to n_2, as in (Lin and Pantel, 2001). In this way, my system is able to capture long dependency relations. For simplicity, I consider a path as a vector `P` <Rel_i>, where Rel_i denotes single relations. In Figure 6.2, I illustrate several paths extracted from two parse trees.

Question:			
Path_ID	**Node1**	**Path**	**Node2**
<PQ1>	Wisconsin	$< subj >$	produce
<PQ2>	produce	$< head, whn, prep, pcomp - n >$	cheese
<PQ3>	nation	$< gen >$	cheese
S1:			
<PS1>	Wisconsin	$< pcomp - n, mod, i >$	produce
<PS2>	produce	$< obj, mod, pcomp - n >$	cheese
<PS3>	nation	$< gen >$	cheese

Figure 6.2: Relation Paths Extracted from the Dependency Trees in Figure 6.1.

Two constraints are imposed when extracting paths:

1. The path length cannot exceed a pre-defined threshold. The length of a path is defined as the number of relations in the path. In my system, the threshold is set to 7 based on my experiments on a small validation dataset. The purpose is to exclude exceptionally long paths as Minipar only resolves nearby dependencies reliably.

2. Relation paths between two words if they belong to the same chunk (which is usually a noun phrase or a verb phrase, as determined by Minipar) are ignored. For instance, the relation between "28" and "percent" in "28 percent" is ignored because they belong to the same NP chunk as parsed by Minipar. A similar example is "New" and "York" in "New York".

To determine the relevance of a sentence given another sentence in terms of dependency relations, I need to examine how similar all the corresponding paths embedded in these two sentences are. The corresponding paths are determined by anchors which are matched nodes in the dependency trees. The nodes represent words or phrases in the original sentences. As such, I determine such *paired corresponding paths* from both sentences by matching their nodes at both ends. For instance, the paths PQ1 and PS1 in Figure 6.2 are paired corresponding paths with the matched nodes "Wisconsin" and "produce" in Figure 6.2. Note that I match only the root forms of open class words (or phrases), such as nouns, verbs and adjectives, when pairing corresponding paths.

6.1.2 Measuring Path Matching Scores by Translation Model

After extracting and pairing relation paths from both a question and a candidate sentence, I need to measure the soft matching score of the paths extracted from the sentence according to those from the question. For instance, in Figure 6.2, I calculate and combine the matching scores of the paths `<pcomp-n, mod, i>`, `<obj, mod, pcomp-n>` and `<gen>` based on their corresponding counterparts from the question: `<subj>`, `<head, whn, prep, pcomp-n>` and `<gen>` respectively. This example also illustrates that in real corpora, the same relationship between two words is often represented by different combinations of relations. I conjecture that such variations in relations hinder existing techniques (*e.g.*, (Attardi et al., 2001; Katz and Lin, 2003)) that attempt to use strict matching to achieve significant improvements over lexical matching methods. In contrast, I approach this problem by employing a fuzzy method to achieve soft relation matching.

I derive the matching score between paths by extending IBM statistical translation model 1. While statistical translation model has been applied in information retrieval (Berger and Lafferty, 1999) and answer extraction (Echihabi and Marcu, 2003), my use of it for the task of matching dependency relation paths is new. I treat the matching score of a relation path from a candidate sentence as the probability of translating to it from its corresponding path in the question. Let us denote two paired corresponding paths from question Q and sentence S respectively as P_Q and P_S, whose lengths are represented as m and n. The translation probability $\Pr(P_S|P_Q)$ is the sum over all possible alignments:

$$\Pr(P_S|P_Q) = \frac{\epsilon}{m^n} \sum_{\alpha_1=1}^{m} \cdots \sum_{\alpha_n}^{m} \prod_{i=1}^{n} P_t(Rel_i^{(S)}|Rel_{\alpha_i}^{(Q)}) \tag{6.1}$$

where $Rel_i^{(S)}$ stands for the i^{th} relation in path P_S and is the corresponding relation in path P_Q. The alignments of relations are given by the values of α_i which indicates the corresponding relation in the question given relation $Rel_i^{(S)}$. ϵ stands for a small constant. $P_t(Rel_i^{(S)}|Rel_j^{(Q)})$ denotes the relation translation probability, *i.e.*, relation mapping scores, which are given by a translation model learned during training and will be described in the next subsection. Unlike machine translation, I assume that every relation can be translated to another; thus, I do not include a NULL relation in position 0. Note that $P_t(Rel_i^{(S)}|Rel_j^{(Q)})$ is 1 when Rel_i and Rel_j are identical because the translation probability is maximized when a relation is translated to itself.

While IBM model 1 considers all alignments equally likely, I consider only the most probable alignment. The reason is that, unlike text translation that works with long sentences, relation paths are short. Most often, the most probable alignment gives much higher probability than any other alignments. I calculate the alignment by finding the most probable mapped relation in the path from the question for each relation in the path from the sentence based on relation translation probability. As such, the path translation probability is simplified as:

$$\Pr(P_S|P_Q) = \frac{\epsilon}{m^n}\prod_{i=1}^{n} P_t(Rel_i^{(S)}|Rel_{A_i}^{(Q)}) \tag{6.2}$$

where A_i denotes the most probable alignment. Moreover, I use only the length n of the path P_S in normalizing Equation 6.2. Since I rank all candidate sentences according to the same question, the length of each path extracted from the question is constant, and does not affect the calculation of the translation probability. I take the log-likelihood of Equation 6.2 and remove all constants. The matching score of P_S finally is as follows:

$$\begin{aligned} SoftMatchScore(P_S) &= \Pr(P_S|P_Q) \\ &= \frac{\epsilon'}{n}\sum_{i=1}^{n}\log(P_t(Rel_i^{(S)}|Rel_{A_i}^{(Q)})) \end{aligned} \tag{6.3}$$

where n is used as a normalization factor and ϵ' is a small constant.

Finally, I sum up the matching scores of each path from the sentence which has a corresponding path in the question to be the *relation matching score* of the candidate sentence given the question. This score reflects how well the candidate sentence's relations match those of the question: a high score indicates that the question terms are likely to be used with the same semantics as in the question, and that the sentence is more likely to contain a correct answer.

6.1.3 Relation Match Model Training

I have described in the above subsection how to obtain a relation matching score between a sentence and the question, and that this process requires a relation mapping model as input, *i.e.*, $P_t(Rel_i^{(S)}|Rel_j^{(Q)})$ in Equation 6.3. In this subsection, I show how the mapping model can be acquired by two statistical methods from training question-answer pairs: one based on mutual information (MI) and the other based on expectation maximization (EM).

The assumption is that paired corresponding paths extracted from training QA pairs are semantically equivalent. Thus, the relation mapping between such training answer sentences and questions can be used as a model for unseen questions and potential answers as well. I use Minipar to parse all the training questions and corresponding answer sentences. Relation paths extracted from the question are paired with those from answer sentences, as described in Section 6.1.1.

I first employ a variation of mutual information[1] to calculate relation mapping scores. The relatedness of two relations is measured by their bipartite co-occurrences in the training path pairs. Different from standard mutual information, I account for path length in my calculation. Specifically, I discount the co-occurrence of two relations in long paths. The mutual information based score of mapping relation to relation is calculated as:

$$P_t^{(MI)}(Rel_i^{(S)}|Rel_j^{(Q)}) = \log \frac{\sum \gamma \times \delta(Rel_j^{(Q)}, Rel_i^{(S)})}{\left|Rel_j^{(Q)}\right| \times \left|Rel_i^{(S)}\right|} \tag{6.4}$$

where $\delta(Rel_j^{(Q)}, Rel_i^{(S)})$ is an indicator function which returns 1 when $Rel_j^{(Q)}$ and $Rel_i^{(S)}$ appear together in a training path pair, and 0 otherwise. γ is the inverse proportion of the sum of the lengths of the two paths. $|Rel^{(Q)}|$ stands for the number of paths extracted from all questions in which relation Rel occurs. Likewise, $|Rel^{(S)}|$ gives the number of paths extracted from all answer sentences that contain relation Rel.

In the second configuration, I employ GIZA (Al-Onaizan et al., 1999), a publicly available statistical translation package, to implement IBM translation model 1 training over the paired training paths. Each relation is considered a word and each corresponding path pair is treated as a translation sentence pair, in which

[1] I use frequencies instead of probabilities in Equation 6.4 to approximate mutual information and use the logarithm to scale the result.

the path from a question is the source sentence and the path from the answer sentence is the destination sentence. The resulting word translation probability table is used to define relation mapping score $P_t(Rel_i^{(S)}|Rel_j^{(Q)})$. GIZA performs an iterative training process using EM to learn pairwise translation probabilities. In every iteration, the model automatically improves the probabilities by aligning relations based on current parameters. The training process is initialized by setting translation probability between identical relations to 1 and a small uniform value for all other cases, and then runs EM to convergence.

6.2 Evaluation

In this section, I present empirical evaluation results to assess my soft relation matching technique for passage retrieval systems. I have three hypotheses to test in the experiments:

1. The relation matching technique improves the precision of current lexical matching methods. Moreover, the proposed soft relation matching method outperforms the strict matching methods proposed in previous work.

2. Long questions are more suitable for relation matching. I hypothesize that the effectiveness of relation matching is affected by question length. Long questions, with more question terms, have more relation paths than short questions, and benefit more from relation matching.

3. Relation matching also brings further improvement to a system that is already enhanced with query expansion because of the high precision it allows. I test whether the soft relation matching technique brings further improvement to a passage retrieval system that uses query expansion.

6.2.1 Evaluation Setup

I use the factoid questions from the TREC-12 factoid QA task (Voorhees, 2003b) as test data and the AQUAINT corpus to search for answers. The reason to choose TREC-12 test data is that the questions are long enough to obtain corresponding relation paths to perform relation matching. I accumulate 10,255 factoid question-answer pairs from the TREC-8 and 9 QA tasks for use as training data, which results in 3,026 unique corresponding path pairs for model construction using both MI and EM based training methods. As in Model 1, identical relations already have their translation probability set to the maximum value, I did not include relation path pairs with identical relation paths in the training data.

There are 413 factoid questions in the TREC-12 task, from which 30 NIL-answer questions are excluded because they do not have answers in the corpus. TREC-12 had a passage retrieval task which used the same factoid questions as the main task except it accepted longer answers (250 bytes). Since I intend to evaluate passage retrieval techniques, I create the gold standard based on the official judgment list for the passage retrieval task provided by TREC. For each question, I generate a list of passages that are judged to be correct and supported by the corpus in the judgment list as standard answer passages. I cannot create the gold standard for 59 of the questions because no correct passages for them were judged by TREC evaluators. This leaves me with a final test set of 324 QA pairs, on which all evaluations in this paper are based. While Tellex et al. (2003) made use of TREC-supplied exact answer patterns to assess returned passages, I observe that common answer patterns can be matched in incorrect passages as answer patterns are usually very short. I therefore use a stricter criterion when judging whether a passage is correct: it must be matched by the exact answer pattern, and additionally, it must have a cosine similarity equal to or above 0.75 with any standard answer passage.

Similar to the configuration used by Tellex et al. (2003), I use the top 200 documents for each question according to the relevant document list provided by TREC as the basis to construct the relevant document set for the questions. If the 200 documents do not contain the correct answer, I add the supporting documents that have the answer into the document set. I conduct different passage retrieval algorithms on the document set to return the top 20 ranked passages. Note that the optimal passage length varies across different retrieval algorithms. For instance, SiteQ is optimized to use a passage length of three sentences (Tellex et al., 2003). In my evaluations for relation matching techniques, I take one sentence as a passage, as Minipar can only resolve intrasentential dependency relations. But for SiteQ, I still use the three-sentence window to define a passage. I use four systems for comparison:

MITRE (baseline): This approach simply matches stemmed words between question and answer.

Strict Matching of Relations: A system that uses strict matching of relations to rank sentences. It employs the same technique as soft matching to extract and pair relation paths, but it counts the number of exact path matches as its ranking score.

SiteQ: One of the top performing density-based systems in previous work. I follow the adaptation described in (Tellex et al., 2003) in my implementation.

NUS (Cui et al., 2004b): I utilize our factoid QA system at NUS for participating TREC in 2004. It is another top-performing system, which is similar to SiteQ except that it uses single sentences as passages and calculates sentence ranking scores by iteratively boosting a sentence's score with adjacent sentence scores.

I employ three performance metrics: mean reciprocal rank (MRR), percentage of questions that have no correct answers, and precision at the top one passage.

The former two metrics are calculated on the returned 20 passages by each system. MRR is defined as $average(\frac{1}{R_i})$, where R_i is the rank of the first correct answer for question Q_i.

6.2.2 Performance Evaluation

In the first experiment, I evaluate the overall performance of my relation matching technique compared to other passage retrieval systems.

I apply both strict and soft matching of relations in my experiments. I perform relation matching on the MITRE and NUS systems but not on SiteQ as it retrieves multiple-sentence passages, in which cross-sentence dependencies cannot be modeled by my system. For simplicity, I linearly combine the normalized lexical matching score obtained by MITRE or NUS and the relation matching score to obtain the overall ranking score of a sentence. In calculating soft relation matching scores, I utilize the two relation mapping score models generated by both the MI-based and EM-based training methods. I illustrate the evaluation results in Table 6.1.

From the table, I draw the following observations:

1. Applying relation matching over lexical matching methods boosts system performance dramatically. Applied on top of the MITRE and NUS systems, both strict and soft relation matchings augment performance in all metrics significantly. When integrating strict relation matching with the NUS system, MRR improves by 35% and 31% over the results obtained by the standard NUS and SiteQ systems respectively. Relation matching also yields better precision in the top one passage task. When soft relation matching is applied on top of NUS, the system achieves even better results. Here, all improvements obtained by relation matching are statistically significant as judged by using paired t-test (Hull, 1993) ($p < 0.001$). I believe that the improve-

Table 6.1: Overall Performance Comparison of MRR, Percentage of Incorrectly Answered Questions (% Incorrect) and Precision at Top One Passage. Strict relation matching is denoted by Rel_Strict, with the base system in parentheses. Soft relation matching is denoted by Rel_MI or Rel_EM for both training methods. All improvements obtained by relation matching techniques are statistically significant ($p < 0.001$)

Passage retrieval systems	MITRE	SiteQ	NUS	Rel_Strict (MITRE)	Rel_Strict (NUS)	Rel_MI (MITRE)	Rel_EM (MITRE)	Rel_MI (NUS)	Rel_EM (NUS)
MRR	0.2000	0.2765	0.2677	0.2990	0.3625	0.4161	0.4218	0.4756	0.4761
% MRR improvement over:									
MITRE	N/A	+38.26	+33.88	+49.50	+81.25	+108.09	+110.94	+137.85	+138.08
SiteQ	N/A	N/A	N/A	+8.14	+31.10	+50.50	+52.57	+72.03	+72.19
NUS	N/A	N/A	N/A	+11.69	+35.41	+55.43	+57.56	+77.66	+77.83
% Incorrect	45.68%	37.65%	33.02%	41.96%	32.41%	29.63%	29.32%	24.69%	24.07%
Precision at top one passage	0.1235	0.1975	0.1759	0.2253	0.2716	0.3364	0.3457	0.3889	0.3889

ment stems from the ability of the relation matching technique to model dependency relationships between matched question terms. Thus, many false positive sentences that would be favored by normal bag-of-word approaches are subsequently eliminated as they often do not contain the correct relations between question terms.

Interestingly, even strict matching of relations significantly improves the performance of a passage retrieval system while work in answer extraction (*e.g.* (Attardi et al., 2001)) seems to be hindered by strict matching. I conjecture that the passage retrieval task is less constraining than answer extraction as the latter has to match relations of the identified target for the question. As such, I feel passage retrieval is more likely to benefit from relation matching.

2. Soft relation matching outperforms strict matching significantly. When integrated with the NUS system, it gains a statistically significant improvement of 31% in MRR and 43% in precision at top one passage when using soft matching of relations over strict matching. Note that while strict matching does not bring large improvements in terms of percentage of incorrect questions compared to lexical matching methods, the soft relation matching method decreases such errors by 34% in comparison to NUS and by 56% compared to MITRE. Strict matching often fails due to variations in representing the same relationship because of parsing inconsistency and the flexibility exhibited in natural language. Such interchangeability between relations is captured by soft matching methods. In this way, my statistical model is able to accommodate the variation in natural language texts.

3. Using MI and iterative EM to train relation mapping scores does not make any obvious difference in my tests. However, I present both training methods because they differ in complexity and scalability. The MI method has lower complexity compared to the EM method because it does not perform any alignment of relations during training, as it uses relation co-occurrences as approximations to relation mapping. The EM training process does alignment by improving the probability of alignment iteratively. I conjecture that the EM training method could outperform the MI method if a larger amount of training data is available. MI-based mapping scores are likely to be more susceptible to noise when scaling up. The EM training method is unlikely to suffer due to its gradual improvement mechanism. However, I cannot show the scalability of the two training methods given my limited test and training data.

6.2.3 Performance Variation to Question Length

It seems intuitive that longer questions are likely to benefit more from relation matching than shorter questions. The rationale is that more relation paths in longer sentences lead to more reliable relation ranking scores. In this experiment, I examine the effect of varying the number of non-trivial question terms on MRR.

Among the 324 questions in my test set, the number of question terms varies from 1 to 13, after removing trivial stop words such as “what”. In Figure 6.3, I plot the MRR values along with 95% error bars of the systems that apply soft relation matching with EM training on top of the MITRE and NUS systems when question length is varied. I consider only questions with two to six non-trivial question terms because there are less than 10% of questions with fewer than two or more than six question terms in my test set.

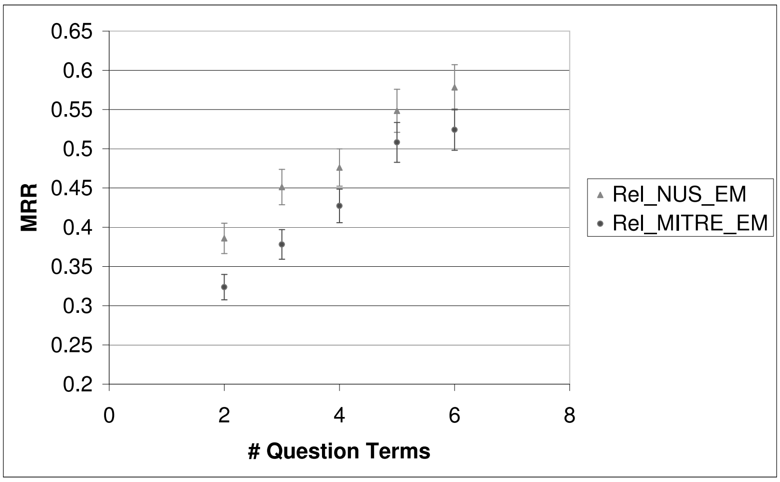


Figure 6.3: MRR Variation w.r.t. Number of Question Terms.

From Figure 6.3, it can be seen that as indicated by little overlap of the error bars, MRR nearly monotonically increases when more terms are present in the question. This is evidence that longer questions are more likely to improve with relation matching. I surmise that with more paired corresponding paths, relation matching based ranking would be of higher precision.

Note that number of question terms is only an approximation of number of actual paired corresponding relation paths. However, as the number of relation paths extracted for each question varies more than the number of question terms does, my small test data prevents us from conducting thorough experiments to examine the effect of number of relation paths on matching. Future work on a larger dataset can be done to reinforce the results shown here.

6.2.4 Performance with Query Expansion

As discussed above, short questions are obstacles in enhancing performance using relation matching. State-of-the-art QA systems adopt query expansion (QE) to alleviate such problems (Cui et al., 2004b; Harabagiu et al., 2003; Ittycheriah, Franz, and Roukos, 2001). Here, I show how performance varies when the relation matching technique is reinforced by query expansion.

I conduct simple query expansion as described in (Cui et al., 2004b), which submits the question to Google and selects expansion terms based on their co-occurrences with question terms in result snippets. I use the same method as described in the first two experiments to linearly combine the lexical matching score with query expansion and the relation matching score. I list the evaluation results in Table 6.2.

With query expansion, the performance of NUS (the lexical matching based system) again improves greatly. Specifically, query expansion reduces the percentage of incorrect answers from 33% to 28.4%. This is close to the figures obtained by relation matching methods without query expansion as listed in Table 6.1. This shows that query expansion boosts recall using expansion terms, allowing the system to answer more questions correctly.

When relation matching is incorporated into the NUS system along with query expansion, MRR values are boosted by 49%, which is statistically significant.

Table 6.2: Performance Comparison with Query Expansion. All the improvements shown are statistically significant ($p - value < 0.001$).

Passage Retrieval Systems	NUS (baseline)	NUS+QE	Rel_MI (NUS+QE)	Rel_EM (NUS+QE)
MRR (% improvement over baseline)	0.2677	0.3293 (+23.00%)	0.4924 (+83.94%)	0.4935 (+84.35%)
% MRR improvement over NUS+QE	N/A	N/A	+49.54%	+49.86%
% Incorrect	33.02%	28.40%	22.22%	22.22%
Precision at top one passage	0.1759	0.2315	0.4074	0.4074

This demonstrates that my relation matching technique can help re-rank passages to allow higher precision when the system is equipped with query expansion.

However, query expansion does not boost the performance of systems with relation matching as significantly as compared to the improvement over the baseline lexical based system without query expansion. Comparing Tables 6.1 and 6.2, the improvement in performance for a system with query expansion is about 2% in MRR (from 0.4756 to 0.4924 when using MI training and from 0.4761 to 0.4935 when using EM training). I believe that this is caused by the simple strategy I use to integrate lexical matching with relation matching. Since I just sum up matching scores, my relation matching model does not take full advantage of query expansion because external expansion terms do not have relation paths with the original question terms in the question. As such, expansion terms do not improve the relation path pairing process in my current system.

6.2.5 Case Study: Constructing a Simple System for TREC QA Passage Task

In the above experiments, I conducted component evaluations for passage retrieval for factoid questions. A natural question is whether the incorporation of relation matching into a standard QA system can yield good performance. Such a fully-fledged QA system adds query expansion, question typing and named entity extraction on top of simple passage similarity. In this case study, I construct a simple QA system on top of the NUS passage retrieval module reinforced by soft relation matching and query expansion. Both question typing and NE extraction modules are rule-based, as employed in a TREC QA system (Cui et al., 2004b). I return the first top-ranked sentence that contains the expected named entity as the answer passage. The average length of the returned passages is 181 bytes.

I evaluate the QA system in the context of the QA passage task of TREC-12 (Voorhees, 2003b). The system answers 175 questions correctly out of the total 324 questions, resulting in an accuracy of 0.540. When averaging over all 383 questions that do not have NIL answers, the accuracy is 0.457, which is still better than the second ranked system in the official TREC-2003 evaluations (Voorhees, 2003b).

6.2.6 Error Analysis and Discussions

Although I have shown that relation matching greatly improves passage retrieval, there is still plenty of room for improvement. A key question is whether I can further characterize the types of questions that are adversely affected by relationship matching. Based on the above two experiments, I perform micro-level error analysis on those questions for which relation matching degrades performance. I find that soft relation matching sometimes fails with incorrectly paired relation paths mainly for the following two reasons:

Mismatch of question terms: In some cases, the paths are incorrectly paired due to the mismatch of question terms. For instance, given the question #1912 "*In which city is the River Seine?*", the correct answer should be "Paris". Without question analysis and typing, the relation matching algorithm mistakenly takes "city" as a question term, instead of recognizing it as the question target. Thus, sentences containing all three question terms, *i.e.*, "city", "river" and "Seine", are ranked high while the correct answer does not contain "city". To overcome this problem, question analysis in the passage retrieval system needs to be incorporated such that the question target and the answer candidate of the expected type can be matched when corresponding relation paths are paired.

Paraphrasing between question and answer sentences: Some correct sentences are paraphrases of the given question. In this case, both lexical matching and relation matching are likely to fail. Consider the question: "`What company manufactures X?`" The correct sentence is: "... `C, the manufacturer of X` ...". The system needs to resolve such a paraphrase as "`C is the manufacturer of X` → `C manufactures X`" to answer this kind of questions. Lin and Pantel (2001) attempted to find paraphrases (also by examining paths in Minipar's output parse trees) by looking at common content between the two nodes at both ends of relations. However, their method is limited as it relies on abundant training data to find inference rules between specific relations.

6.3 Conclusion

In this chapter, I have presented a novel soft relation matching technique for factoid QA passage retrieval. My evaluation results show that the technique produces significant improvements in retrieval performance in current systems: a vast 50–138% improvement in MRR, and over 95% in precision at top one passage. Soft matching of dependency relations is calculated based on the degree of match between relation paths in candidate sentences and the question. To learn a model of relationship matching from training data, I have presented two methods based on mutual information and EM. While these two methods do not make an obvious difference given my test data, I believe that EM scales better and may perform better when given a larger amount of training data. Furthermore, the relation matching technique has shown itself capable of bringing significant improvement in retrieval performance across all the architectures I have tested, regardless of whether or not query expansion is used.

Past work has shown that strict matching does not perform well in answer extraction. I have shown that this conclusion does not generalize to all QA modules. A contribution of my work is the demonstration that even strict matching of relations significantly augments the performance of current passage retrieval modules. This may be explained by the fact that passage retrieval imposes less constraint in matching relations than answer extraction. Future work is expected to improve answer extraction by using relations effectively.

The empirical evaluation results and qualitative error analysis reveal that the relation matching method can be improved by better alignment of relation paths. Relation paths often cannot be paired due to few matched question terms or paraphrasing, both of which could be alleviated by query expansion. While I have benchmarked the performance of relation matching with query expansion, my experiment has not fully integrated the modules in the sense that I have not taken advantage of expanded terms in relation matching. Seamless integration of query expansion with relation matching is likely to produce further gains in performances and is a logical next step in future research.

Chapter 7

Conclusion

In this thesis, I have identified the weaknesses in matching syntactic and semantic features in current question answering systems. I provided two soft matching schemes to theoretically and practically combat the problems. The two soft matching schemes have been implemented using statistical models, which could be utilized to build a question answering system. Given a search target, the QA system first gives a definition for the target and then answers a series of specific factoid questions about it. In the process of writing this thesis, I have touched upon related disciplines of information retrieval and natural language processing in order to provide usable solutions to the problems identified.

In this chapter, I will recap the contributions of the research and summarize them in the next subsections. I will then discuss the limitations of this work. I conclude the whole thesis with the outline of future work.

7.1 Contributions

This thesis makes the following main contributions to the studies in both information retrieval and natural language processing:

1. Soft matching models for lexico-syntactic patterns.
2. Soft matching of dependency relations for passage retrieval.
3. Two key components for an integrated question answering system.

I summarize the contributions individually.

7.1.1 Soft Matching Models for Lexico-Syntactic Patterns

I developed three soft pattern models – one basic model and two formal ones based on the bigram model and the PHMM, respectively – to achieve flexible matching of lexico-syntactic patterns. Soft patterns contribute to the field of pattern learning and matching in natural language processing. Pattern matching has been widely applied to text extraction, such as information extraction and question answering. I have reviewed the literature on rule induction and pattern learning. I found that most existing work focuses on the learning phase of rules and patterns. Often, the pattern rules are represented in regular expressions with constraints in each slot to facilitate matching.

In contrast, the two formal soft pattern models are generic in the sense that they model the pattern matching process as the process of generating token sequences. Soft pattern models a generalization of the conventional hard pattern matching process. By enforcing all individual slots to emit the exact tokens and setting sequential probabilities to one, the soft matching models revert to regular expression based hard matching. In contrast, equipped with the learned probabilities based on training data, the soft pattern models can better model variations in natural language texts. As such, the soft pattern models can be easily extended to other applications where textual pattern matching plays an important role.

In addition to the models of matching, I also developed strategies for generalizing sentences into abstract pattern instances. The generalization procedure produces more generic instances such that the pattern learning and matching are conducted on a more general basis.

A key contribution of this work is that it greatly improves definition sentence retrieval for existing definitional QA systems. I have experimentally shown that the system using soft pattern matching significantly outperforms that using state-of-the-art manually constructed hard matching patterns by 10% - 15% in F_3 measure.

7.1.2 Soft Matching of Dependency Relations for Passage Retrieval

This thesis also contributes to the state-of-the-art in passage retrieval. I developed statistical model for soft matching of dependency relations between matched question terms. Combining lexical matching and similarity measure between grammatical relations, the passage retrieval for factoid QA has been significantly improved. I demonstrated how to adapt the IBM translation model 1 to measure the similarity between relation paths in a parsing tree. In addition, in order to learn similarity between individual relations, I developed two training methods based on past QA pairs – one is based on the co-occurrences of relations measured by mutual information; the other is based on EM to iteratively learn the mapping probabilities of relations.

This work contributes to information retrieval. The two-anchor soft matching scheme employed in this work utilizes dependency relations as the features for soft matching. The features may be replaced by other syntactic and semantic features. As such, one may substitute the matching features to further enhance the performance of passage retrieval.

Based on the evaluation results, I showed that passage retrieval based on simple lexical matching in a factoid QA system could be improved by over 70% when applying soft matching of dependency relations. I believe this has contributed significantly to research in question answering because passage retrieval is a crucial step in modern QA systems.

7.1.3 Two Components for an Integrated Question Answering System

Another contribution of this thesis is that I have developed the two components which can be utilized in an integrated QA system. I successfully applied the two soft matching schemes in the components such that the modules of definition sentence retrieval and passage retrieval for factoid questions achieve the similar level of performance or outperform the state-of-the-art QA systems.

This QA system embodies the information search process for advanced users. To complete the system, I also contributed to other modules, such as definition summarization. It has been employed as a testbed for experimenting with other advanced features for question answering.

7.2 Limitations of this Work

This thesis has made contributions in fulfilling an advanced user's information need. However, it has several limitations that are hopefully addressed in the future work.

- **Lack of approximate lexical match in soft matching of feature sequences** – While I have presented two soft matching schemes to realize flexible match of feature sequences, I noticed that there lacks semantic approximation for lexical terms in my system. Approximate match of lexical terms often relies on external resources, such as WordNet or other thesauri, to semantically relate two words. For instance, “kill” and “murder” should be considered matched if other constraints are satisfied. Such techniques have been employed in current question answering systems, *e.g.*, (Harabagiu et al., 2000; Chu-Carroll et al., 2004). However, in my soft matching methods, I mainly focus on solving the problem brought by editing processes on token sequences, *i.e.*, the gaps between training and test sequences. I employed only simple morphological forms (after stemming) of lexical words in both soft pattern matching and soft relation matching. This simplification incurs problems – *e.g.* the soft pattern match degree is degraded when there are unmatched words in some of the slots due to alternative in wording; similarly, the anchor nodes cannot be matched such that soft relation matching can by no means be conducted.

We have tried to remedy this problem in soft relation matching by adopting WordNet to find similar words when matching anchor nodes (Sun et al., 2005). However, we did not obtain improved results by employing WordNet to expand the words. I conjecture that the main reason is that WordNet is designed for general linguistic use and may not be suitable for accomplishing such real search applications, where certain term usages may be domain dependent.

- **Lack of ability in answering other types of questions** – To date, my question answering system can only answer definition and factoid questions about the search target. I have not applied the soft matching techniques in answering other types of questions in the system. For instance, I have not touched upon list questions. I believe soft matching of dependency relations may help improve the performance of list question answering because list questions can be answered by aggregating factoid answers from different documents. In addition, answering opinion related questions (Yu and Hatzivassiloglou, 2003) should be implemented in such a QA system. For advanced users, they may ask questions similar to “what is the opinions from some parties about the target?” To answer such questions, the system first has to identify opinion sentences about the target. This task is similar to definition sentence retrieval. There is work that learns patterns for subjective expressions to identify the sentences that may bear opinions (Riloff and Wiebe, 2003). I believe that soft pattern matching models can contribute to this field as subjective expressions are even more diversified than objective ones, and thus soft matching patterns may gain more improvements. To synthesize these opinion sentences, the summarization technique introduced in Chapter 3 can be employed along with other specific processes.

7.3 Future Work

I summarize the routes for future research.

- **Experimenting with other features in soft matching.** In soft matching evaluation, I employed syntactic features such as noun phrase chunks and part-of-speech tags, as well as grammatical features like dependency relations. There is still much room to experiment with more semantic features such as named entities. While noun phrase chunks are approximation of named entities, precision can be augmented if named entity classes can be employed as general tags in pattern instance generalization and in anchor matching for relation matching. However, to better make use of named entities, the problem of name normalization should be considered, *e.g.*, a person's name can be written in different forms.

  Another interesting feature in soft matching is sentence syntactic structure. Echihabi and Marcu (2003) showed that integrating parse tree features could greatly facilitate the matching of two similar sentence structures. Introducing parse tree features could enhance the soft matching process because it can directly match a whole substructure, such as a clause. However, this depends heavily on the parsing quality of the parser.

- **Extending the soft matching schemes to other related applications.** As stated before, while I experiment on definitional QA with the soft pattern models, the models are generic and can be applied to the following applications:

    1. Information extraction (IE) – The pattern matching problem in IE tasks is formally the same as definition sentence retrieval. When conducted on free text, an IE system may also suffer from various unseen instances not being matched by trained patterns. Xiao et al. (2004) have demonstrated that soft pattern matching greatly improves recall in an IE system. Although some HMM topologies have been employed for IE tasks, the soft pattern models are more generic and require less configuration and parameter tuning when changing domains.
    2. Factoid question answering – Pattern matching is also utilized to improve precision in factoid QA (Ravichandran and Hovy, 2002; Harabagiu et al., 2005). Soft pattern models should be trained on each kind of questions along with the question taxonomy.

    Soft relation matching can also find its applications in passage retrieval for different retrieval systems. Soft matching of relations between words is supposed to improve all types of passage retrieval systems. It would also be helpful to examine if other types of relations, such as semantic relations based on predicates, help in passage retrieval.

- **Integrating the two soft matching schemes.** I have demonstrated adopting different statistical models to implement both the one-anchor and the two-anchor soft matching schemes. Determined by the question answering applications, I adopted different soft matching schemes for matching definition patterns and dependency relations. In the future work, it is worthwhile investigating the unification of the two soft matching schemes. As pointed out in Chapter 6, the main difficulty in unifying the two schemes in this thesis is that dependency relation matching does not require generalization. However, in other information extraction tasks, very often, we can fix the length of contextual windows and need generalized representations of the contexts. In this case, the two soft matching schemes can be integrated using the soft pattern matching models.

- **More user studies in evaluating the integrated QA system.** I have demonstrated the component-wise performance of the QA modules experimentally. However, the performance measurement is based on the effectiveness of the sub-systems, instead of user satisfactions for the whole integrated QA system. As the QA system is designed to fulfill the advanced requirement of users, it is imperative to conduct user studies to analyze how much the QA system can help users improve quality of their search results. It would be interesting to see how much the definition of the search target can help the user improve search quality for the follow-up factoid questions.

- **Completing the components of the QA system.** The QA system lacks components in dealing with other types of questions. Completing these modules is a reasonable step to take. In addition, it is also helpful to design and build a useful user interface, which can lead the user through the search process.

References

Agichtein, Eugene, Luis Gravano, Jeff Pavel, Viktoriya Sokolova, and Aleksandr Voskoboynik. 2001. Snowball: A prototype system for extracting relations from large text collections. In *SIGMOD Conference*.

Ahn, David, Valentin Jijkoun, Gilad Mishne, Karin M̈uller, Maarten de Rijke, and Stefan Schlobach. 2004. Using Wikipedia at the TREC QA track. In *TREC*.

Al-Onaizan, Yaser, Jan Curin, Michael Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz Josef Och, David Purdy, Noah A. Smith, and David Yarowsky. 1999. Statistical machine translation: Final report. Technical report, Johns Hopkins University 1999 Summer Workshop on Language Engineering.

Attardi, Giuseppe, Antonio Cisternino, Francesco Formica, Maria Simi, and Alessandro Tommasi. 2001. PiQASso: Pisa Question Answering System. In *TREC*.

Battelle, John. 2005. *The Search: How Google and Its Rivals Rewrote the Rules of Business and Transformed Our Culture*. Penguin Group.

Berger, Adam and John Lafferty. 1999. Information retrieval as statistical translation. In *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229, New York, NY, USA. ACM Press.

Blair-Goldensohn, Sasha, Kathleen McKeown, and Andrew Hazen Schlaikjer. 2003. A hybrid approach for qa track definitional questions. In *TREC*, pages 185–192.

Blair-Goldensohn, Sasha, Kathleen McKeown, and Andrew Hazen Schlaikjer. 2004. Answering definitional questions: A hybrid approach. In *New Directions in Question Answering*, pages 47–58.

Borkar, Vinayak, Kaustubh Deshmukh, and Sunita Sarawagi. 2001. Automatic segmentation of text into structured records. *SIGMOD Rec.*, 30(2):175–186.

Brill, Eric, Jimmy J. Lin, Michele Banko, Susan T. Dumais, and Andrew Y. Ng. 2001. Data-intensive question answering. In *TREC*.

Brown, Peter F., Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Comput. Linguist.*, 19(2):263–311.

Buckley, Chris, Gerard Salton, James Allan, and Amit Singhal. 1994. Automatic query expansion using SMART: TREC 3. In *TREC*.

Carbonell, Jaime G. and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 24-28 1998, Melbourne, Australia*, pages 335–336. ACM.

Chu-Carroll, Jennifer, Krzysztof Czuba, John Prager, Abraham Ittycheriah, and Sasha Blair-Goldensohn. 2004. IBM's PIQUANT II in TREC 2004. In *TREC*.

Ciravegna, Fabio. 2001. Adaptive information extraction from text by rule induction and generalisation. In *IJCAI*, pages 1251–1256.

Cui, Hang, Min-Yen Kan, and Tat-Seng Chua. 2004. Unsupervised learning of soft patterns for generating definitions from online news. In *WWW*, pages 90–99.

Cui, Hang, Min-Yen Kan, and Tat-Seng Chua. 2005. Generic soft pattern models for definitional question answering. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 384–391, New York, NY, USA. ACM Press.

Cui, Hang, Min-Yen Kan, Tat-Seng Chua, and Jing Xiao. 2004a. A comparative study on sentence retrieval for definitional question answering. In *SIGIR 2004 Workshop IR4QA: Information Retrieval for Question Answering.*

Cui, Hang, Keya Li, Renxu Sun, Tat-Seng Chua, and Min-Yen Kan. 2004b. National University of Singapore at the TREC-13 question answering main task. In *Proceedings of TREC-13.*

Cui, Hang, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 400–407, New York, NY, USA. ACM Press.

Dempster, A.P., N.M. Laird, and D.B. Rubin. 1977. Maximum Likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society*, 39:1–38.

Durbin, Richard, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1999. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press.

Echihabi, Abdessamad and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 16–23, Morristown, NJ, USA. Association for Computational Linguistics.

Fellbaum, Christiane. 1999. *WordNet: an electronic lexical database.* Cambridge, Mass : MIT Press.

Gaizauskas, Robert, Mark A. Greenwood, Mark Hepple, Ian Roberts, and Horacio Saggion. 2004. The University of Sheffields TREC 2004 Q&A experiments. In *TREC.*

Gao, Jianfeng, Jian-Yun Nie, Guangyuan Wu, and Guihong Cao. 2004. Dependence language model for information retrieval. In *SIGIR '04: Proceedings* *of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 170–177, New York, NY, USA. ACM Press.

Goldstein, Jade, Mark Kantrowitz, Vibhu O. Mittal, and Jaime G. Carbonell. 1999. Summarizing text documents: Sentence selection and evaluation metrics. In *SIGIR*, pages 121–128.

Han, Kyoung-Soo, Hoojung Chung, Sang-Bum Kim, Young-In Song, Joo-Young Lee, , and Hae-Chang Rim. 2004. Korea University question answering system at TREC 2004. In *TREC*.

Harabagiu, Sanda M., Steven J. Maiorano, and Marius A. Pasca. 2003. Open-domain textual question answering techniques. *Nat. Lang. Eng.*, 9(3):231–267.

Harabagiu, Sanda M., Dan I. Moldovan, Christine Clark, Mitchell Bowden, Andrew Hickl, and Patrick Wang. 2005. Employing two question answering systems in TREC-2005. In *TREC*.

Harabagiu, Sanda M., Dan I. Moldovan, Christine Clark, Mitchell Bowden, John Williams, and Jeremy Bensley. 2003. Answer mining by combining extraction techniques with abductive reasoning. In *TREC*, pages 375–382.

Harabagiu, Sanda M., Dan I. Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan C. Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2000. FALCON: Boosting knowledge for answer engines. In *TREC*.

Hildebrandt, Wesley, Boris Katz, and Jimmy J. Lin. 2004. Answering definition questions with multiple knowledge sources. In *HLT-NAACL*, pages 49–56.

Hovy, Eduard, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *HLT '01: Proceedings of the First International Conference on Human Language*

Technology Research, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Hovy, Eduard H., Ulf Hermjakob, and Chin-Yew Lin. 2001. The use of external knowledge of factoid QA. In *TREC*.

Hull, David A. 1993. Using statistical testing in the evaluation of retrieval experiments. In *SIGIR*, pages 329–338.

Ittycheriah, Abraham, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system - TREC-10. In *TREC*.

Kaszkiel, Marcin and Justin Zobel. 1997. Passage retrieval revisited. In *SIGIR '97: Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, New York, NY, USA. ACM Press.

Katz, Boris, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Wesley Hildebrandt, Roni Katzir, Jimmy Lin, Daniel Loreto, Gregory Marton, Federico Mora, and Ozlem Uzuner. 2004. Answering multiple questions on a topic from heterogeneous resources. In *TREC*.

Katz, Boris and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering*.

Klavans, Judith and Smaranda Muresan. 2001. Evaluation of definder: a system to mine definitions from consumer-oriented medical text. In *JCDL*, pages 201–202.

Lee, Gary Geunbae, Jungyun Seo, Seungwoo Lee, Hanmin Jung, Bong-Hyun Cho, Chanhki Lee, Byung-Kwan Kwak, Jeongwon Cha, Dongseok Kim, JooHui An, Harksoo Kim, and Kyungsun Kim. 2001. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *Proceedings of TREC-10*.

Light, Marc, Gideon S. Mann, Ellen Riloff, and Eric Breck. 2001. Analyses for elucidating current question answering technology. *Journal for Natural Language Engineering*, 7(4):325–342.

Lin, Chin-Yew and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *HLT-NAACL*.

Lin, Dekang. 1998. Dependency-based evaluation of MINIPAR. In *Proceedings of Workshop on the Evaluation of Parsing Systems*.

Lin, Dekang and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Journal for Natural Language Engineering*, 7(4):343–360.

Lin, Jimmy and Dina Demner-Fushman. 2005a. Automatically evaluating answers to definition questions. In *HLT/EMNLP*, pages 931–938.

Lin, Jimmy and Dina Demner-Fushman. 2005b. Will pyramids built of nuggets topple over? Technical Report LAMP-TR-127/CS-TR-4771/UMIACS-TR-2005-71, University of Maryland, College Park, December.

Lin, Jimmy, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. 2003. What makes a good answer? the role of context in question answering. In *Proceedings of the Ninth IFIP TC13 International Conference on Human-Computer Interaction*.

Liu, Bing, Chee Wee Chin, and Hwee Tou Ng. 2003. Mining topic-specific concepts and definitions on the web. In *WWW*, pages 251–260.

Manning, Christopher D. and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

McCallum, Andrew, Dayne Freitag, and Fernando C. N. Pereira. 2000. Maximum Entropy Markov Models for information extraction and segmentation. In *ICML*, pages 591–598.

Moldovan, Dan I., Marius Pasca, Sanda M. Harabagiu, and Mihai Surdeanu. 2003.

Performance issues and error analysis in an open-domain question answering system. *ACM Trans. Inf. Syst.*, 21(2):133–154.

Muresan, Smaranda, Samuel D. Popper, Peter T. Davis, and Judith L. Klavans. 2003. Building a terminological database from heterogeneous definitional sources. In *DG.O*.

Muslea, Ion. 1999. Extraction patterns for information extraction tasks: A survey. In *Proceedings of AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 1–6.

Nahm, Un Yong and Raymond J. Mooney. 2001. Mining soft-matching rules from textual data. In *IJCAI*, pages 979–986.

Peng, Fuchun, Ralph Weischedel, Ana Licuanan, and Jinxi Xu. 2005. Combining deep linguistics analysis and surface pattern learning: A hybrid approach to chinese definitional question answering. In *HLT/EMNLP*, pages 307–314.

Prager, John, Dragomir Radev, and Krzysztof Czuba. 2001. Answering what-is questions by virtual annotation. In *HLT '01: Proceedings of the First International Conference on Human Language Technology Research*, pages 1–5, Morristown, NJ, USA. Association for Computational Linguistics.

Prager, John M., Jennifer Chu-Carroll, Krzysztof Czuba, Christopher A. Welty, Abraham Ittycheriah, and Ruchi Mahindru. 2003. IBM's PIQUANT in TREC2003. In *TREC*, pages 283–292.

Radev, Dragomir R., Hongyan Jing, Magorzata Sty, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Inf. Process. Manage.*, 40(6):919–938.

Radev, Dragomir R. and Kathleen McKeown. 1998. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):469–500.

Ravichandran, Deepak and Eduard H. Hovy. 2002. Learning surface text patterns for a question answering system. In *ACL*, pages 41–47.

Riloff, Ellen. 1993. Automatically constructing a dictionary for information extraction tasks. In *AAAI*, pages 811–816.

Riloff, Ellen. 1996. Automatically generating extraction patterns from untagged text. In *AAAI/IAAI, Vol. 2*, pages 1044–1049.

Riloff, Ellen and Janyce Wiebe. 2003. Learning extraction patterns for subjective expressions. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 105–112.

Rosenfeld, Ronald. 2000. Two decades of statistical language modeling: Where do we go from here. *Proceedings of the IEEE*, 88(8).

Salton, Gerard and Michael McGill. 1984. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Company.

Sarner, Margaret H. and Sandra Carberry. 1988. A new strategy for providing definitions in task-oriented dialogues. In *Proceedings of the 12th Conference on Computational linguistics*, pages 567–572, Morristown, NJ, USA. Association for Computational Linguistics.

Schiffman, Barry, Inderjeet Mani, and Kristian J. Concepcion. 2001. Producing biographical summaries: Combining linguistic knowledge with corpus statistics. In *ACL*, pages 450–457.

Schwartz, Ariel S. and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Pacific Symposium on Biocomputing*, pages 451–462.

Skounakis, Marios, Mark Craven, and Soumya Ray. 2003. Hierarchical Hidden Markov Models for information extraction. In *IJCAI*, pages 427–433.

Soderland, Stephen. 1999. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272.

Song, Fei and W. Bruce Croft. 1999. A general language model for information retrieval. In *CIKM*, pages 316–321.

Sudo, Kiyoshi, Satoshi Sekine, and Ralph Grishman. 2001. Automatic pattern acquisition for Japanese information extraction. In *HLT '01: Proceedings of the First International Conference on Human Language Technology Research*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Sun, Renxu, Hang Cui, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Dependency relation matching for answer selection. In *SIGIR '05: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 651–652, New York, NY, USA. ACM Press.

Tellex, Stefanie, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *SIGIR '03: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–47, New York, NY, USA. ACM Press.

Tombros, Anastasios and Mark Sanderson. 1998. Advantages of query biased summaries in information retrieval. In *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2–10, New York, NY, USA. ACM Press.

Voorhees, Ellen M. 2000. Overview of the TREC-9 question answering track. In *TREC*.

Voorhees, Ellen M. 2003a. Evaluating answers to definition questions. In *HLT-NAACL*.

Voorhees, Ellen M. 2003b. Overview of the TREC 2003 question answering track. In *TREC*, pages 54–68.

Voorhees, Ellen M. 2004. Overview of the TREC 2004 question answering track. In *TREC*.

Voorhees, Ellen M. and Hoa Trang Dang. 2005. Overview of the TREC 2005 question answering track. In *TREC*.

White, Michael, Tanya Korelsky, Claire Cardie, Vincent Ng, David Pierce, and Kiri Wagstaff. 2001. Multidocument summarization via information extraction. In *HLT '01: Proceedings of the First International Conference on Human Language Technology Research*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Xiao, Jing, Tat-Seng Chua, and Hang Cui. 2004. Cascading use of soft and hard matching pattern rules for weakly supervised information extraction. In *Proceedings of COLING 2004*, pages 542–548, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Xiao, Jing, Tat-Seng Chua, and Jimin Liu. 2003. A global rule induction approach to information extraction. In *ICTAI*, pages 530–536.

Xu, Jinxi, Ana Licuanan, and Ralph M. Weischedel. 2003. TREC 2003 QA at BBN: Answering definitional questions. In *TREC*, pages 98–106.

Xu, Jinxi, Ralph M. Weischedel, and Ana Licuanan. 2004. Evaluation of an extraction-based approach to answering definitional questions. In *SIGIR*, pages 418–424.

Yang, Hui, Hang Cui, Mstislav Maslennikov, Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2003. QUALIFIER in TREC-12 QA main task. In *TREC*, pages 480–488.

Yu, Hong and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 129–136.

Zahariev, Manuel. 2003. Efficient acronym-expansion matching for automatic acronym acquisition. In *Proc. of IKE*, pages 32–37.

Appendix A

Table A.1: Techniques Employed by Recent TREC Systems to Answer Definition Questions

TREC Systems	**Linguistic Constructs**	**Bag-of-Words Ranking**	**Mining External Knowledge**
Amsterdam (Ahn et al., 2004)	Nugget extraction based on dependency parsing trees.	Rank the sentences from the corpus by measuring their lexical and semantic similarity with the facts mined from the external reference web site. The semantic similarity is measured by the distance of words in WordNet or word co-occurrence statistics in large corpus.	Rely heavily on the external reference database - an online encyclopedia. Mine the facts about the targets from the web site.

BBN (Xu, Licuanan, and Weischedel, 2003; Xu, Weischedel, and Licuanan, 2004)	• Patterns identifying appositive and copulas constructs. • Propositions extracted from parsing trees. • 40+ manual rules for structured definition patterns. • Special relations extracted by a specialized information extraction module. Assign weights to linguistic constructs according to the extraction types.	Ranking linguistic constructs by their similarity with the question profile. The question profile is constructed by 3 options: 1. Centroid of extracted definitions from online dictionaries, encyclopedias and biographical sites. 2. Centroid of 17,000 short biographies for the profile of a person. 3. Centroid of extracted linguistic constructs from the corpus for the target.	Constructing profiles for targets by mining external definitional resources.

Columbia (Blair-Goldensohn, McKeown, and Schlaikjer, 2003)	Extract definitional predicates, which include three types of genus, species and non-specific definitional, based on 23 manual patterns on parsing trees.	Construct a centroid vector for the target by selecting frequent non-trivial words from all extracted constructs. The centroid vector is used to rank the constructs by measuring their similarity with the centroid vector.	
MIT (Katz et al., 2004)	16 classes of regular expression based patterns. These patterns are used to construct a database of definitions offline.	Retrieve sentences that contain the target and rank the sentences by the overlap of keywords in the sentences and the dictionary definitions.	Dictionary look-up on an online dictionary and use the dictionary definitions to score sentences.

IBM PIQUANT (Chu-Carroll et al., 2004; Prager et al., 2003)	• Appositions and relative clauses. • Surface patterns similar to those by Ravichandran and Hovy (2002).	Establish a profile for each target. The profile is constructed by concepts represented by nouns that occur more with the target than random occurrences. Passages are ranked by the number of concepts they contained.	• Pre-defined auxiliary questions for different types of targets. • Hypernyms from WordNet to define the terms. • Biographical data from particular website.

Korea University (Han et al., 2004)	Extract pre-defined constructs from syntactic parsing trees of sentences. Such constructs include modifying phrases of the target, relative pronoun phrases, copulas, general verb phrases, *etc.*	Statistical ranking of extracted constructs based on: • Count of the head word of the target being as the head word in the answer constructs. • Count of terms in extracted constructs. • Trained statistics of biographical terms from an encyclopedia, applying only to persons.	Biographies from external encyclopedia for training term statistics.
LCC (Harabagiu et al., 2003)	Utilized 38 definition patterns, out of which 23 find match in TREC questions.		

Table A.2: The 26 Questions for the Evaluation on the Web Corpus.

Question ID	Questions
1	Who is Brooke Burke?
2	Who is Clay Aiken?
3	Who is Jennifer Lopez?
4	What is Lord of the Rings?
5	Who is Pamela Anderson?
6	What is Hurricane Isabel?
7	What is Final Fantasy?
8	Who is Harry Potter?
9	Who is Carmen Electra?
10	What is Napster?
11	What is Xbox?
12	Who is Martha Stewart?
13	Who is Osama bin Laden?
14	What is Cloning?
15	What is NASA?
16	Who is Halle Berry?
17	What is Enron?
18	What is West Nile Virus?
19	What is SARS?
20	Who is Daniel Pearl?
21	Who is Nostradamus?
22	Who is James Bond?
23	Who is Arnold Schwarzenegger?
24	Who is Mohammed Saeed al-Sahaf?
25	Who is Uday Hussein?
26	What is stem cell?

Appendix B

Evaluation on the Use of External Knowledge

In this section, I briefly discuss the evaluations on the use of external knowledge in definitional QA. The system settings are discussed in Section 4.5. I conduct two experiments – one for the impact of external knowledge on the baseline system that uses manually constructed definition patterns; and the other for the effects on unsupervised learning of soft patterns through group pseudo-relevance feedback (GPRF).

B.1 Impact of External Knowledge on the Baseline System

I construct the baseline system by employing centroid-based ranking with a set of manually constructed rules as listed in Table 4.2. I vary the use of task-independent (general) and task-specific external knowledge and assess their impact on the baseline system. Note that I denote general resources as Google snippets and WordNet definitions, and task-specific resources as existing definitions from Answers.com. In cases where both the general and the specific resources cover the same search term, I use the specific resources. I do not include a configuration that includes both task-specific Web knowledge and the use of WordNet. This is due to that WordNet provides only short definitions to the question terms and the definitions are mostly covered by the task-specific Web resources. The results are shown in Table B.1.

Table B.1: Impact of External Knowledge on the Baseline System.

Configurations	NR	NP	F_5 (% improvement)
Baseline	51.00	19.53	46.69
Baseline + WordNet	56.13	19.72	50.88 (+8.97%)
Baseline + Google	51.45	20.69	47.27 (+1.24%)
Baseline + Task-specific	58.05	21.71	53.37 (+14.32%)
Baseline + Google and Task-specific	58.55	21.59	53.86 (+15.37%)

B.2 Impact of External Knowledge on GPRF

In this evaluation, I use the centroid-based ranking and soft patterns learned from unsupervised labeled definition sentences determined by GPRF as the baseline. I apply combinations of task-independent and task-specific resources to boost the retrieval performance of centroid-based weighting. I also include an experiment that leverages more offline learned patterns, in the form of additional supervised soft patterns learned over the Web corpus (see Section 4.5.1). I present the results in Table B.2.

Table B.2: Impact of External Knowledge on GPRF.

Configurations	NR	NP	F_5 Measure (% improvement)
Centroid + GPRF SP (Baseline)	60.11	22.19	53.91
Baseline + Google	61.89	22.09	55.56 (+3.06%)
Baseline + Task-specific	65.08	24.56	58.74 (+8.96%)
Baseline + Google + Task-specific	65.24	23.49	58.76 (+9.00%)
Baseline + Supervised SP + Google + Task-specific	65.48	23.36	58.96 (+9.36%)