RELATION-BASED MODELS FOR PASSAGE RETRIEVAL IN OPEN DOMAIN QUESTION ANSWERING

SUN REN XU

NATIONAL UNIVERSITY OF SINGAPORE

2006

RELATION-BASED MODELS FOR PASSAGE RETRIEVAL IN OPEN DOMAIN QUESTION ANSWERING

SUN REN XU

(B. Computing (Hons), National University of Singapore)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgement

I must first thank my supervisor Prof. Chua Tat Seng. I began to know Prof. Chua in my second year of undergraduate study in NUS and that was also the time when I started to work in the area of Information Retrieval and Question Answering. During these four years, we have been working on a lot of projects: StepTalk, TREC QA, my HYP project and finally my master's project. Even though his schedule is packed most of the time, he is very patient during each meeting we had, guiding me throughout the whole course of the project, taking his time to edit my reports and papers. He is able to penetrate my project, update and point out deep idea. I am indebted to him for providing me the opportunity to work on great projects, the freedom to think on my own and the support for achieving excellence.

I also like to thank Cui Hang, Li Keya, Jiang Jing, Tan Yee Fan and Ong Chai Huat. We have worked days and nights for the TREC QA task. Many ideas of this work came from numerous discussions we had in the lab.

Last but not the least; I have to thank all the people in the chime text and multimedia group, who have helped me during the course of the project and all the staff in NUS, who have provided me with such great environment for doing research. Especially, I have to thank Neo ShiYong and Victor Goh. They have shared with me their valuable experience in working in this area.

Table of Contents

Summary

Most current passage retrieval systems for open domain question answering use statistical co-occurrence or lexical distance to model the relations between query terms. However, we know that such statistical measure provide only an approximation to the “real” relations between terms. In this thesis, we propose the use of relation-based models for passage ranking and query expansion. We will propose two models, one using syntactic dependency relation and the other using semantic relation. Experimental results show that our syntactic dependency relation-based models significantly outperform density-based passage ranking with co-occurrence-based query expansion by up to 68% in mean reciprocal rank. Our semantic relation-based model also outperforms the density-based model by 4%. Based on this result, we combine the two models and propose a framework for passage retrieval for open domain question answering using relationship graph.

List of Tables

List of Figures

Chapter 1 Introduction

1.1 Introduction to Passage Retrieval

Passage retrieval has long been studied in information retrieval [17]. It aims to search for more precise and compact text excerpts in response to users' queries, rather than providing whole documents. Recently, passage retrieval has become a crucial component in question answering (QA) systems. Most current QA systems employ a pipeline structure that consists of several modules to get short and precise answers to users' questions. A typical QA system searches for answers at increasingly finer-grained units by: (1) locating the relevant documents, (2) retrieving passages that may contain the answer, and (3) pinpointing the exact answer from candidate passages.

Passage retrieval (Step 2) greatly affects the performance of a 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. Also, a passage can often answer a question sufficiently. Lin et al. [23] showed that users prefer passages to phrase-long answers because passages provide sufficient context for them to understand the answer.

Tellex et al. [29] conducted a thorough quantitative component evaluation for passage retrieval algorithms employed by state-of-the-art 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 1.1.1, where only sentence S1 contains the correct answer. The other three sentences share many question terms (in italics) but are incorrect.

To address this problem, we propose to integrate relation analysis into current passage retrieval system. In this thesis, we propose two approaches of incorporating different relation (dependency relation and semantic relation) analysis for passage retrieval.

<Question> What percent of the nation's cheese does Wisconsin produce?
Correct: <S1> In *Wisconsin*, where farmers *produce* roughly 28 *percent* of the *nation*'s *cheese*, the outrage is palpable.
Incorrect: <S2> … 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*.
Incorrect: <S3> 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*, …
Incorrect: <S4> Awareness of the Real California *Cheese* logo, which appears on about 95 *percent* of California *cheeses*, has also made strides.

Figure. 1.1.1 Sample question and answer candidate passages to illustrate that lexical matching does not lead the correct answer.

1.2 Project Overview

This project is part of NUS TREC 2004 and 2005 [5] QA task effort. The system architecture is illustrated in the figure 1.2.1:

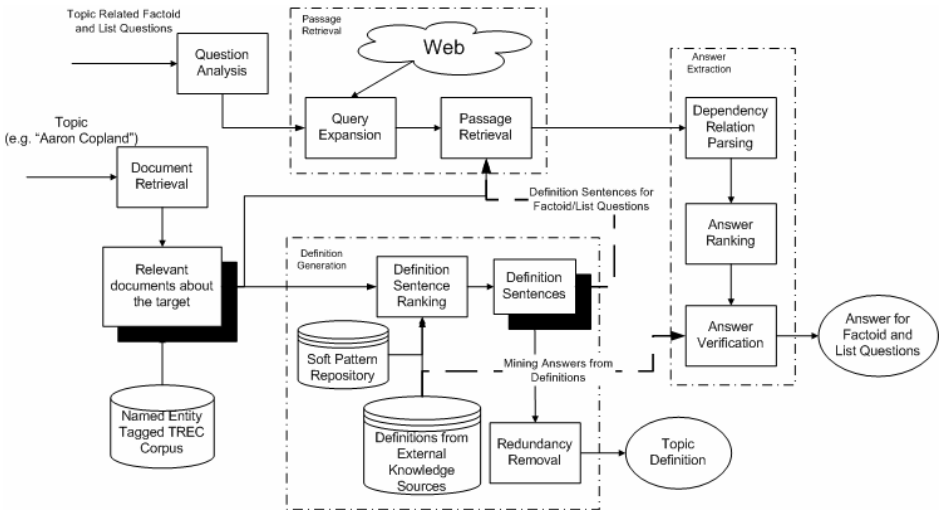

Figure. 1.2.1 TREC QA system (NUS Team) overview

The whole system used for TREC evaluation is very complex and involves many modules. Basically the whole system can be divided into Question Analysis, Query Expansion and Passage Retrieval, NE Analysis, Document Retrieval (topic level), and Answer Extraction, Definition Sentence Retrieval and Redundancy removal etc.

Each of the modules requires considerable description for one to understand thoroughly how the whole system works in detail. However, those are outside the scope of this thesis. In this work, we will focus mainly on the passage retrieval part, which includes query expansion and passage ranking.

1.3 Contributions

The main contribution of the thesis is in employing relation-based models for query expansion to enhance passage retrieval in the QA framework. In particular, I have made the following contributions:

Passage Retrieval and Query Expansion using Dependency Relation Matching based on Minipar [22]: I have incorporated the use of dependency parsing (using Minipar) into the framework of passage retrieval with query expansion. In particular, we employ relation-based model to identify good expansion terms from both the QA corpus and external internet resources.

Passage Retrieval and Query Expansion using Semantic Frame Matching based on ASSERT [27]: I have proposed a semantic relation-based model for passage retrieval with query expansion, which makes use of semantic parsing (based on ASSERT parser) in the framework of QA.

Passage Retrieval and Query Expansion using Relationship Graph: I have analyzed the strengths and weaknesses of the two models and proposed a merged model (Relationship Graph) for passage retrieval in QA.

1.4 Organizing of the Thesis

This report is organized as follow: In Chapter 2, we summarize the related works on passage retrieval systems, focusing on passage ranking and query expansion techniques. In chapter 3, we propose a novel fuzzy relation matching model which examines grammatical dependency relations between question terms to improve the current passage retrieval techniques. We also propose a new query expansion technique to be integrated into the dependency relation matching model for short queries. Chapter 3 will be organized into four sections. In Section 3.1, we will give a brief introduction to dependency grammar and dependency parsing. In Section 3.2, we will propose a passage retrieval model based on fuzzy matching of dependency relations. In Section 3.3, we will introduce a new query expansion technique based on dependency relation matching, and in Section 3.4, we will provide experimental results and result analysis. In Section 3.5, we will summarize the dependency matching model for passage retrieval. In chapter 4, we propose a passage retrieval model that is based on the semantic structures of the question and the answer candidate passages. We use a shallow semantic parser to identify the semantic structures in the sentences. We then score a passage based on the similarity between its semantic structure(s) and the semantic structure(s) contained in the question. Chapter 4 will be organized in the similar manner as chapter 3. In chapter 5, we will combine the two models proposed in chapters 3 and 4 and propose a framework for passage retrieval using relationship graph. Finally, we summarize and conclude the entire work and propose directions of future research in Chapter 6.

Chapter 2 Related Works

2.1 General Framework for Passage Retrieval

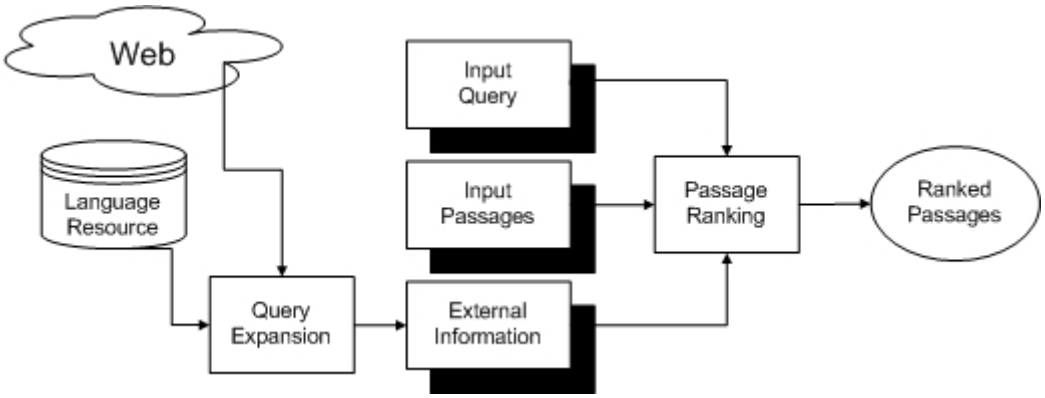

Figure. 2.1.1 General framework for passage retrieval

Figure 2.1 shows a general framework for passage retrieval. The two key components in this framework are passage ranking and query expansion.

Passage ranking function is the key for passage retrieval. The function takes in input passage, input query and external information as its arguments. The input passage usually comes from the result of document retrieval. Its scale usually varies from several hundred passages to several thousand passages. The input query refers to the query entered by the user. The external information refers to the additional query terms expanded from external resources as a result of query expansion. The purpose of the ranking function is to rank passages relevant to the query higher than those irrelevant to the query in a reasonable period of time (usually a few seconds).

Query expansion is another important technique used in passage retrieval. The purpose of query expansion is to overcome the problem of word mismatch which is fundamental to information retrieval. Simply stated, it means that people often use different words to describe concepts in their queries rather than those used in the

documents. The severity of the problem tends to decrease as the queries get longer, since there are more chances of some important words co-occurring in the query and relevant documents. In many applications, however, the queries are very short. For example, applications that provide searching across the World-Wide Web typically record average query lengths of two words [8]. Although this may be one extreme in terms of IR applications, it does indicate that most IR queries are not long and that techniques for dealing with word mismatch are needed.

In the next two sections, we will provide detailed analysis of related works on passage ranking and query expansion. In Section 2.4, we will discuss other issues related to passage retrieval. Finally, in Section 2.5 we will summarize the current frameworks and discuss the challenges of current passage retrieval framework.

2.2 Related Works on Passage Ranking

Word Matching, Language Model, Syntactic Relation Analysis and Semantic Relation Analysis are the four major approaches that have been proposed so far for passage ranking.

The simplest approach for passage ranking employed by MITRE [21] performs only word matching method, i.e. it counts the number of matched question terms in a passage. Other word matching based passage retrieval systems, such as those employed in SiteQ [20] and IBM [15], are called density-based as they take into account the distances between question terms in the candidate passages. The advantage of word matching approach is its efficiency and the method can be easily scaled to very large corpus. However, the underlying assumption of such approach is that each question term is considered an independent token. However, this simplification does not hold in many cases because dependency relations exist

between words.

Therefore, several researchers propose to use language model to capture the relationship between words. For instance, some language modeling approaches capture simple dependency relations by using bigrams (e.g., [28]). But these models only capture dependency relations between adjacent words. Recently, Gao *et al.* [12] 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 co-occurrences of words in the training data. The language model approach has incorporate relationship between words into passage ranking. However, it only examines the statistical co-occurrences between words without considering any language structures.

Syntactic Relation Analysis approach was first introduced in answer sentence ranking for Question Answering by PiQASso [3]. They used Minipar [22], a dependency parser to extract the answer from a candidate sentence if the relations reflected in the question are matched in that sentence. However, the system does not perform well due to low recall resulting from matching relations in only the top ranked sentences. To overcome the recall problem, Katz and Lin [18] 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 use dependency relation analysis to perform sentence ranking. However, their matching method is based on strict matching of dependency relations. Strict matching is problematic when conducted on a large corpus because relations between the same pair of words often differ between questions and answer sentences.

The goal of passage retrieval is to identify passages similar to the question in

semantic content. Based on this assumption, Semantic Relation Analysis is used to perform passage retrieval. Durme *et al.* [10] used a simple semantic representation to represent questions and passages, and used fuzzy unification to measure the similarity between the question and an answer passage. They used the Link Grammar parser, a syntactic parser, to parse passages to identify semantic structures. It is not clear how they assign the semantic roles to each argument based on the syntactic parsing results. There was also no systematic evaluation of their method. Semantics have been introduced into QA systems previously. Narayanan and Harabagiu proposed a more sophisticated question answering system that performs probabilistic inference on the semantic structures obtained from the question and the passages [13],[27]. However, the method needs to first build a topic model that stores the inference rules. Such a knowledge base is constructed from the context of the domain of interest. It is therefore a closed domain QA system rather than an open domain QA system. And their method may not be applicable to information retrieval due to its high computational cost.

2.3 Related Works on Query Expansion

The idea of query expansion is that the query is expanded using words or phrases with similar semantic meaning to those in the query with respect to the query context and the chances of matching words in relevant documents are therefore increased. This is the basic idea behind the use of a thesaurus in query formulation. There is, however, little evidence that a general thesaurus is useful in improving the effectiveness of the search, even if words are selected by the searchers [32]. Instead, it has been proposed that by automatically analyzing the text of the corpus being searched, a more effective thesaurus or query expansion technique can be produced.

On the other hand, query expansion can be considered as a special kind of term retrieval. In query expansion, we are ranking terms that are relevant to the query instead of passages. However, as query expansion is usually performed at the document retrieval level the complexity of the expansion technique is greatly restricted. Also due to efficiency and scalability reasons, word co-occurrence model and language model are the only general frameworks proposed so far to tackle this problem. Syntactic and semantic relation analysis for query expansion have not been proposed.

The word co-occurrence model is the first model proposed to perform query expansion. One of the earliest studies of this type was carried out by Sparck Jones [16] who clustered words based on co-occurrence in documents and used those clusters to expand the queries. A number of similar studies followed but it was not until 1994 that consistently positive results have been obtained. The techniques that have been used can be described as being based on either global or local analysis of the documents in the corpus being searched. The global techniques examine word occurrences and relationships in the corpus as a whole, and use this information to expand any particular query. Given their focus on analyzing the corpus, these techniques can be considered as extensions of Sparck Jones' original approach.

Local analysis, on the other hand, involves only the top ranked documents retrieved by the original query. We have called it local because the techniques are variations of the original work on local feedback [2],[9]. This work treated local feedback as a special case of relevance feedback where the top ranked documents were assumed to be relevant. Queries were both re-weighted and expanded based on this information.

Both global and local analysis have the advantage of expanding the query based on all the words in the query. This is in contrast to a thesaurus-based approach where

individual words and phrases in the query are expanded and word ambiguity is a problem. Global analysis is inherently more expensive than local analysis [35]. On the other hand, global analysis provides a thesaurus-like resource that can be used for browsing without searching, and retrieval results with local feedback on small test collections were not promising.

The most recent approach for query expansion based on word co-occurrence model is called local context analysis proposed by Xu *et al.* in 1996 [35]. The technique borrows ideas from global analysis, such as the use of context and phrase structure, but applies them to the local document set. The key idea of such analysis is based on the co-occurrence between query terms and expanded terms in the local context.

In 2004, Cronen-Townsend *et al.* [30] proposed to use language model approach for selective query expansion. They built a language model to select queries that are expected to have good expanded terms.

All the approaches we have described use statistical co-occurrence to infer the semantic relatedness between query terms and expanded terms. Mainly due to efficiency and scalability reasons none of the approaches uses any language feature for query expansion on passage retrieval task.

2.4 Other Issues Related to Passage Retrieval

As the work of this thesis is closely related to the TREC QA project, we have to address other issues that are related to both passage retrieval and question answering.

a) Size of each Passage

Most passage retrieval system use fixed sized passage size (e.g 1000 bytes) or fixed sentence window frames (e.g 3 sentence window frame) for passage retrieval. Considering that the syntactic or semantic parser needs a complete sentence as input,

we use fixed sentence window in our case. Because of the high accuracy demand for answer extraction in the final stage, we need to restrict the window size to 1. Therefore, in this thesis we assume each passage is one complete sentence.

b) Number of Passages after Document Retrieval

A typical input size for passage retrieval is using top N documents from document retrieval, in which N varies from 100 to 1000 [29]. In our passage retrieval system, we use N=200.

c) Resource for Query Expansion

In our passage retrieval system, query expansion is performed on two kinds of resources, including General language resources such as WordNet and web resources, in particular Google snippets.

d) Format of the Query

As the passage retrieval system is developed under the framework of QA system. And most of our algorithms require certain linguistic parsing. Therefore we assume that the query is in natural language format and most of our evaluations are done by using natural language queries rather than key word based queries. However, query expansion is developed under a general framework. Hence, the query expansion module can expand both natural language queries and web queries, which is a bag of keywords.

2.5 Summary

Tellex et al. [29] conducted a thorough quantitative component evaluation for passage retrieval algorithms. 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.

Currently, we can see that most efforts have been put on using statistical co-occurrence models to approximate the dependency or semantic relations between query terms without integrating linguistic features. There are several systems [3] that integrate linguistic features such as dependency relation analysis into passage retrieval. However, their performance is not comparable to density-based approach as they cannot handle the variation of linguistic features. Although semantic relation match has been proven to be effective for QA, the method may not be applicable to passage retrieval due to its high computational cost.

Therefore, the main challenge for passage retrieval today is to propose an effective and efficient model that utilizes linguistic features to perform matching based on syntactic and semantic relationships between query terms rather then simply count word co-occurrences.

Chapter 3 Dependency Relation-Based Model for Passage Retrieval

In this chapter, we propose a novel fuzzy relation matching model which examines grammatical dependency relations between query terms to improve current passage retrieval models for question answering. We then develop a new query expansion algorithm based on dependency relation analysis so that we can make use of external resources to provide additional contextual information for short queries.

In Section 3.1, we will introduce dependency grammar and Minipar [22]. In Section 3.2, we will describe in detail the dependency matching algorithm as described in [7] for passage retrieval including the training algorithm and matching algorithm. Section 3.3 describes our main contribution in a novel query expansion algorithm based on dependency relation analysis and how it can be integrated into the framework of dependency relation matching. Section 3.4 provides experimental results analysis and error analysis. Finally in Section 3.5, we will summarize the work.

3.1 Dependency Grammar and Minipar

We make use of the dependency tree generated by Minipar for our passage retrieval algorithm. A dependency tree explains dependency relations between elements of a sentence. Figure 3.1.1 and 3.1.2 is an example of a parsed sentence for “What book did Rachel Carson write in 1962?” (one of them in the output form and the other in tree for illustration purpose). Each relation (a path between two adjacent nodes in the tree) is an asymmetric relation between two words (a head and a modifier). Every node has exactly one parent. That means one node can modify only one other node.

But a node can be modified by many other nodes. There is a root node for each sentence. Usually it's not a word from the sentence. Each path in the tree is labeled by a type of relationships to describe the relations between the two words. There are in all 42 types of relations with 27 commonly used shown in Appendix I together with their meanings. There will be some empty nodes in the parse tree. For example, in the above sentence node E0, E1, E3 and E4 are all empty nodes. Usually the root is an empty node. E3 is inserted because "book" is the object of "write".

Minipar [22] has reported about 80% accuracy in dependency parsing. But it also has drawbacks. According to PiQASso [3] the selection of dependency relations is somewhat arbitrary, so that two similar word pairs may have quite different relations, although both are correct. This makes the parsing rather unpredictable.

```
(E1      (()      ~ Q      *          )
1        (What    ~ Det    2          det       (gov book))
2        (book    ~ N      E1         whn       (gov ~))
E0       (()      ~ YNQ    E1         head      (gov ~))
3        (did     do Aux   E0         inv-aux   (gov ~))
4        (Rachel  ~ U      5          lex-mod   (gov "Rachel Carson"))
5        (Carson  "Rachel Carson" N   6         s         (gov write))
6        (write   ~ V      E0         head      (gov ~))
E3       (()      book N   6          obj       (gov write)        (antecedent 2))
E4       (()      "Rachel Carson" N  6         subj      (gov write)        (antecedent 5))
7        (in      ~ Prep   6          mod       (gov write))
8        (1962    ~ N      7          pcomp-n   (gov in))
)
```

Figure. 3.1.1 The parsing output from MiniPar for the sample sentence

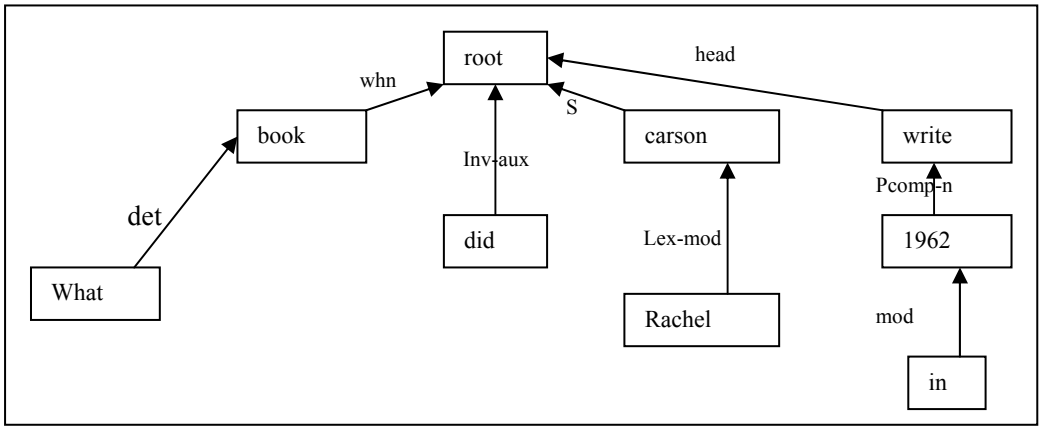

Figure. 3.1.2 The parsing tree from MiniPar for the sample sentence

3.2 Fuzzy Relation Matching for Passage Ranking

In this section, we discuss how fuzzy relation matching is performed in detail. The technique we employed is based on the technique described in [7]. For completeness, we summarize the approach by [7] here. Throughout this section, we use the term relation-based model or system to refer to the system described in [7].

3.2.1 Extracting and Pairing Relation Paths

The relation-based model first extracts the relation paths between words from dependency trees for sentences generated by Minipar. Figure 3.2.1.1 shows part of the dependency trees for the sample question and the answer sentence S1 presented in Figure 1.1.1.

Question:

Path_ID	Node1	Path	Node2
$<P_{Q1}>$	Wisconsin	*<subj>*	produce
$<P_{Q2}>$	produce	*<head, whn, prep, pcomp-n>*	cheese
$<P_{Q3}>$	nation	<gen>	cheese
S1:			
$<P_{S1}>$	Wisconsin	*<pcomp-n, mod, i>*	produce
$<P_{S2}>$	produce	*<obj, mod, pcomp-n>*	cheese
$<P_{S3}>$	nation	*<gen>*	cheese

Figure. 3.2.1.1 Partial output for question and sentence S1 shown in Figure 1.1.1 generated by Minipar.

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, we 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 dependency relation between the two nodes. Examples of relation labels (or relations for short) are *subj* (subjective), *mod* (modifying) and *pcomp-n* (nominal complement of a preposition).

In order to handle long dependency relation, the system further defines a relationship path (or simply path) between nodes n1 and n2 as the series of edges that traverse from n_1 to n_2, as in [24]. For simplicity, a path is considered as a vector $P <Rel_i>$, where Rel_i denotes single relations. Figure 3.2.1.2 illustrates several paths extracted from two parse trees.

The system imposes two constraints when extracting paths:

(a) 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 our configuration, the threshold is set to 7 based on our experiments on a small validation dataset. The purpose is to exclude exceptionally long paths as Minipar only resolves nearby dependencies reliably.

(b) It ignores the 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. For instance, we ignore the relation between “28” and “percent” in “28 percent” because they belong to the same NP chunk as parsed by Minipar. A similar example is “New” and “York” in “New York”.

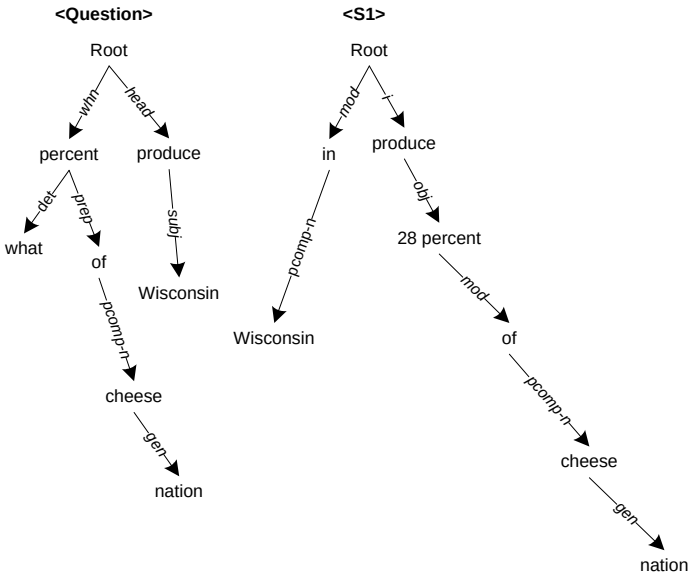

Figure. 3.2.1.2 Illustration of relation paths extracted from the dependency trees in Figure 3.2.1.1

To determine the relevance of a sentence given another sentence in terms of dependency relations, one need to examine how similar all the corresponding paths embedded in these two sentences are. To achieve this, the system determines such paired corresponding paths from both sentences by matching their nodes at both ends. For instance, P_{Q1} and P_{S1} are paired corresponding paths with the matched nodes "Wisconsin" and "produce". Note that the system matches only the root forms of open class words (or phrases), such as nouns, verbs and adjectives, when pairing corresponding paths.

3.2.2 Measuring Path Matching Score

After extracting and pairing relation paths from both a question and a candidate sentence, we need to measure the matching score of the paths extracted from the sentence according to those from the question. For instance, in Figure 3.2.1.1, we 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. We conjecture that such variations in relations hinder existing techniques (e.g., [3],[18]) that attempt to use strict matching to achieve significant improvements over lexical matching methods. To handle variations in relations, the system tackles the problem by employing a fuzzy method to achieve approximate relation matching.

The system derives the matching score between paths by extending IBM statistical translation model 1. It treats 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's 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 $Prob(P_S|P_Q)$ is the sum over all possible alignments:

$$\Pr ob(P_S \mid P_Q) = \frac{\varepsilon}{m^n} \sum_{\alpha_1=1}^{m} \cdots \sum_{\alpha_n=1}^{m} \prod_{i=1}^{n} P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_{\alpha_i}^{(Q)}) \tag{3.2.1}$$

where $\mathrm{Re}l_i^{(S)}$ stands for the ith relation in path PS and $\mathrm{Re}l_{\alpha_i}^{(Q)}$ is the corresponding relation in path PQ. The alignments of relations are given by the values of α_i which indicates the corresponding relation in the question given relation $\mathrm{Re}l_i^{(S)}$. ε stands for a small constant. $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$ 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 in the original application of machine translation, [7] assumes that every relation can be translated to another; thus, it does not include a NULL relation in position 0. Note that $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$ 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, the system proposed in [7] considers 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. We 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:

$$\Pr ob(P_S \mid P_Q) = \frac{\varepsilon}{m^n} \prod_{i=1}^{n} P_t(\mathrm{Re}\, l_i^{(S)} \mid \mathrm{Re}\, l_{A_i}^{(Q)}) \tag{3.2.2}$$

where A_i denotes the most probable alignment. The system in [7] uses only the length n of the path P_S in normalizing Equation (3.2.2). As the system ranks 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. It takes the log-likelihood of Equation (3.2.2) and remove all constants. The matching score of P_S is as follows:

$$MatchScore\ (P_S) = \Pr ob(P_S \mid P_Q) = \frac{\varepsilon'}{n} \sum_{i=1}^{n} \log P_t(\mathrm{Re}\, l_i^{(S)} \mid \mathrm{Re}\, l_{A_i}^{(Q)}) \tag{3.2.3}$$

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

Finally, it sums 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.

3.2.3 Model Training

We have described in the above section 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., $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$ in Equation (3.2.3). In this subsection, we show how the mapping model is acquired in [7] 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. The Minipar is used 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 3.2.1.

The system in [7] first employs a variation of mutual information 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, we account for path length in its calculation. Specifically, it discounts the co-occurrence of two relations in long paths. The mutual information based score of mapping relation $\mathrm{Re}l_j^{(Q)}$ to relation $\mathrm{Re}l_i^{(S)}$ is calculated as:

$$P_t^{(MI)}(\mathrm{Re}\,l_i^{(S)} \mid \mathrm{Re}\,l_j^{(Q)}) = \log \frac{\sum \gamma \times \delta(\mathrm{Re}\,l_j^{(Q)}, \mathrm{Re}\,l_i^{(S)})}{|\,\mathrm{Re}\,l_j^{(Q)}\,| \times |\,\mathrm{Re}\,l_i^{(S)}\,|} \tag{3.2.4}$$

where $\delta(\mathrm{Re}l_j^{(Q)}, \mathrm{Re}l_i^{(S)})$ is an indicator function which returns 1 when $\mathrm{Re}l_j^{(Q)}$ and $\mathrm{Re}\,l_i^{(S)}$ 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, [7] employs GIZA [1], 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 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 $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$. 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. It initializes the training process by setting translation probability between identical relations to 1 and a small uniform value for all other cases, and then run EM to convergence.

3.3 Query Expansion using Dependency Relation Analysis

In this section, we will discuss in detail our main contribution in this research, i.e. how we perform query expansion using dependency relations. We first present the extraction of relation paths from parse trees in web snippets. We then describe in detail the two query expansion methods, namely: (a) dependency relation-based term expansion (DRQET), which is to be employed in a density-based passage retrieval system [6],[15], and (b) dependency relation-based path expansion (DRQER), which is to be employed in a relation-based passage retrieval system [7]. Finally, we will present details on how we train our relation language model for query expansion.

In our framework for query expansion, we adopt a variation of local context method by applying language modeling techniques on relations to select the expanded terms and relation paths. Figure 3.3.1 illustrates the general framework for relation-based query expansion. The framework compromises two major phases: the training phase and the query expansion phase. During the training phase, we use the training QA pairs to derive the weights of relations between query terms and expansion terms. The relation weights are stored in the relation score table.

The query expansion phase implements two separate query expansion methods DRQET and DRQER. Both methods make use of the trained relation score table to measure the relevance of the expanded terms and relation paths from Web resources.

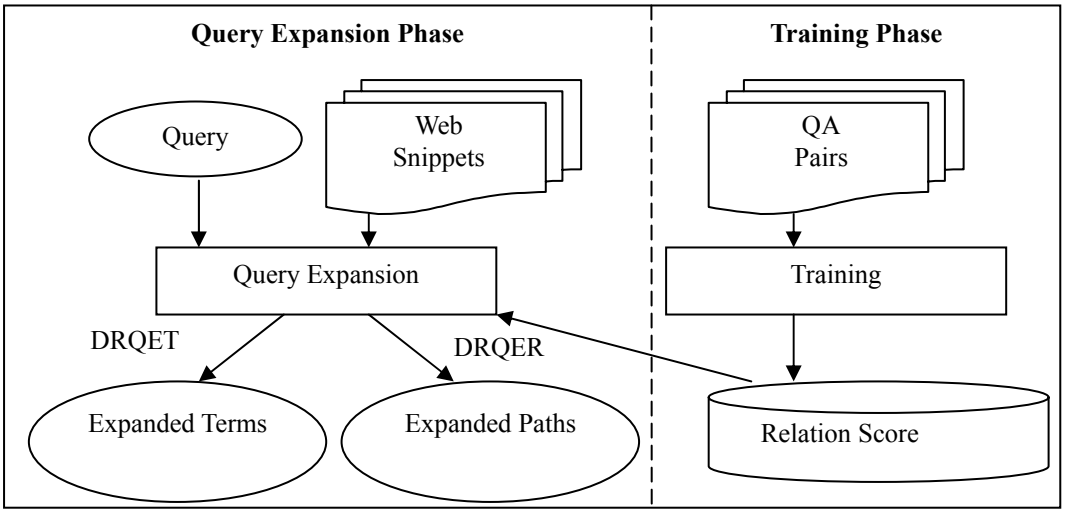

Figure. 3.3.1 Framework of Relation-Based Query Expansion

3.3.1 Dependency Relation Paths from Web Snippets

There are two sources of information corpus from which query expansion may be carried out. They are: (a) the original corpus from which information is to be found, and (b) the parallel corpus from which relevant contextual information may be mined. Original corpus is the most obvious collection and is used by most query expansion techniques based on relevance feedback [2],[9],[35]. Parallel corpus, which means

another corpus of the same time or topic domain, is another choice and it is often adopted by news video retrieval system using ASR (Automatic Speech Recognition). With the wide spread adoption of World Wide Web, web-based query expansion is adopted by most IR and Question-Answering (QA) ([31],[33],[34]) systems. There are two major reasons for using the web as a parallel corpus for IR and open domain QA: (1) the content of the web is more complete than any other existing corpus, and (2) the content of the web is always up-to-date.

In our experiment, we use Google snippets as the basis for query expansion. We first send the queries to Google and collect the top k snippets. We adopt a similar approach as that of the local context analysis (LCA) method. In LCA method presented in [35], top 200 passages are found to be the ideal size for query expansion for query expansion based on relations. Since we are performing sentence based matching, each sentence is considered to be a passage while each snippet on average contains two complete sentences.[1] Therefore there are on average 2k passages contained in the top k snippets, and thus we set k to 100 in our experiment. There are two reasons for using snippets rather than complete web pages for passage retrieval systems. First, complete html pages can be very long and about multiple topics, it's often the case that a term at the beginning and a term at the end of a long document do not have any dependency relations. Second, it is more efficient to use snippets because we can eliminate the cost of processing the unnecessary parts of the web pages.

After we have collected the snippets, we use a sentence splitter to split the sentences within these snippets. We then parse the snippets using Minipar [22], a dependency grammar parser. We denote the question or query as Q and the set of snippets corresponding to the query as $S:=\{s_1,s_2...s_m\}$.We denote the resulting set of

[1] If the sentence is not complete, then we will locate the original complete sentence from the source html page and use it as a complete sentence.

passages (or sentences) derived from S as P:={p_1,p_2...p_n} with the expected value of n to be 200. We denote the set of parse trees of passages in P as T:={t_1,t_2...t_n}.

Figure 3.3.1.1 illustrates an example of dependency parsing. It shows the parse tree of a sample question (Q: When is Alaska purchased?) in Figure 3.3.1.1(a) and the parse tree of a sample answer snippet (s_i: Alaska was purchased from Russia in 1876) retrieved from Google in Figure 3.3.1.1(b). In a dependency tree, each node represents a word or a chunked phrase, and is attached with a link or edge representing the relation pointing from this node (the governor) to its modifier node. In this paper, we define each node in the dependency tree as a term and each edge in the dependency tree as a dependency relation. Although dependency relations are directed links, we ignore the directions of the 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 dependency relation between two nodes. Some examples of relation labels (or relations for short) as shown in Figure 3.3.1.1 are obj (objective), from (relation that indicates the direction of the action) and in (relation that indicates the time information of the action). There are 42 commonly used relations defined in Minipar [22].

We further define a relation path (or simply path) between nodes n_1 and n_2 as the series of edges that traverse from n_1 to n_2. In this way, our system is able to capture long dependency relations. For simplicity, we consider a path as a vector path: <Start_Term, Rel_1,Rel_2…Rel_m, End_Term>, where Start_Term is the starting node of the path, End_Term is the ending node and Rel_i denotes single relations. For example, the relation path between “When” and “purchased” as shown in Figure 3.3.1.1(a) can be defined as Path <When, wha, head, purchased>.

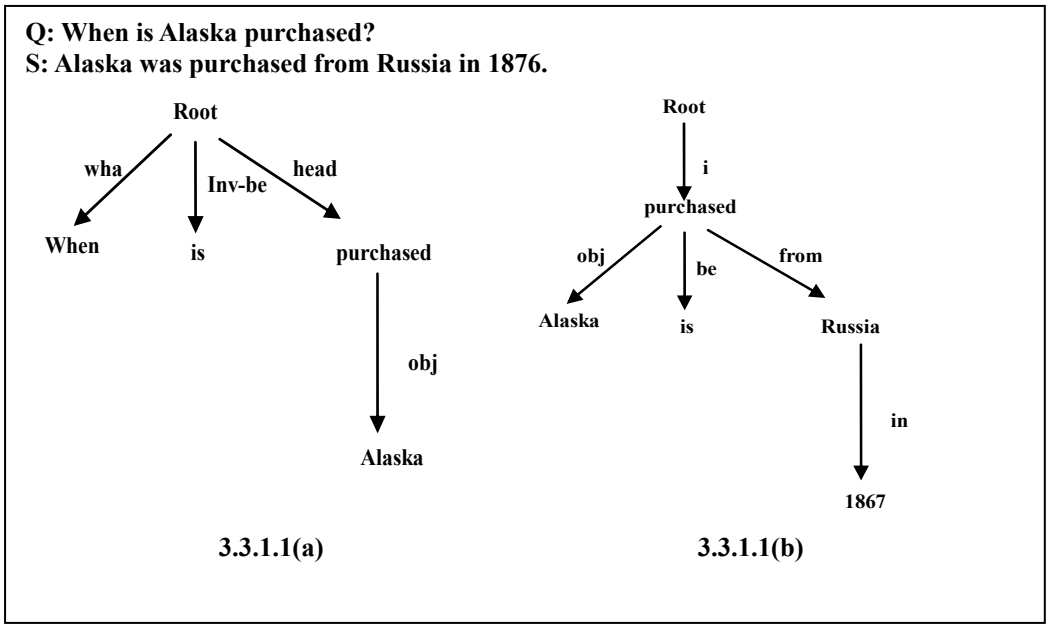

Figure. 3.3.1.1. The parse trees of the sample question and sentence

3.3.2 Term Expansion for Density-Based Passage Retrieval System

The goal of term expansion is to employ relation-based technique to select addition terms from the parse trees of the retrieved passages. Statistical co-occurrence-based methods such as LCA only perform term selection based on its co-occurrence with the query terms without considering their relationships. These techniques are unable to differentiate high quality contextual terms from noise. Moreover, studies [30] show that if we were to perform query expansion based on the original corpus only, the results is highly dependent on the quality of the initial retrieval. To make the technique more robust, we employ relation-based models to select high quality terms from external information source such as the Web. A ranked list of expanded terms, with a weight associated with each term indicating its relatedness to the original query, is derived for query expansion.

In the relation-based model for term expansion, the importance of a query term is determined by two factors: global importance and local importance. After extracting the relation paths from web snippets, we have n dependency parsing trees in T which corresponds to n passages in P (see Section 3.1). The global importance is measured by the inverse document frequency (idf) of the expanded term, while the local importance of an expanded term is measured by its relation path linking to the query term. The overall importance of the relation path is a function of the importance of each individual relation, which is obtained through training. The assumption is that certain paths are more likely to infer a relevant expanded terms than other paths and these useful relation paths are obtained by training. The non-stop terms denoted as *Tk* in the snippet set *S* are ranked according to the formula:

$$Score(T_k, Q) = \prod_{t_i \in Q} \left(\frac{\delta + \log_{10}\left(\sum_{j=1}^{n} path\_score(T_k, t_i, j)\right) \times idf_{Tk}}{\log_{10} N} \right)^{idf_{t_i}} \quad (3.3.1)$$

where

$$path\_score(T_k, t, j) = \prod_{\substack{Tk \in p_j \wedge t \in p_j \wedge \\ \text{Re}\, l_i \in path(Tk, t, j)}} score(\text{Re}\, l_i)$$

$$idf_{Tk} = \log_{10}(N / N_{Tk}) / \log_{10} N$$

$$idf_{t_i} = \log_{10}(N / N_{t_i}) / \log_{10} N$$

T_k is the term to be ranked,

p_j is the *j*th passage in the passage set *P*,

$path(T_k,t,j)$ is the relation path in the dependency parsing tree of

p_j with start node T_k and ending node *t*,

$path\_score(T_k,t,j)$ is the score of $path(T_k,t,j)$,

N is the number of passages in the snippet set *S*,

N_{Tk} is the number of passages in P that contains term T_k,

N_{ti} is the number of passages in P that contains term t_i,

$score(Rel_i)$ is the score of individual relation which is obtained through training, and δ is set to 0.1 to avoid zero values.

The above formula is a variant of the term ranking formula in local context analysis [35]. In our modified model, the global importance is still modeled using the same way as in LCA method. However, we use the importance of relation path to model the local importance instead of term co-occurrence.

We treat a relation path, Path <Start_Node, Rel_1… Rel_k…Rel_m, End_Node>, as a sequence of independent relation labels. We only consider the paths where the End_Node is a query term, and the Start_Node is the term that we want to rank. Therefore by considering only the relation labels in the path, we can treat the sequence just in the same way as a word sequence and apply language model on the relation sequence, where the vocabulary of the relational language model is all the 42 possible relation labels. Thus the score of a path is proportional to the probability by which a "useful" expansion term is inferred; and this probability is calculated using the formula $\mathbf{\Pi} score(Rel_i)$ under the assumption that each relation in the sequence is independent of eath other.

Finally, we formulate our new query Q'_t by adding the top k terms denoted as $\{T_1, T_2 ... T_k\}$ with k set to 10 derived from term expansion to the original query. We set the weight of original query terms to be 1.0 and the weight of i^{th} expanded token to be (1-0.9*i/k). The new query Q'_t is a bag of terms, that can be written as <($word_1$,$weight_1$),…($word_i$,$weight_i$)>. We then issue the new query to any density-based method for passage retrieval to rank the set of passages.

3.3.3 Relation Path Expansion for Relation-Based Passage Retrieval System

A new framework for passage retrieval based on dependency relations has been proposed in [7], in which they found that the use of dependency relations among the query terms can significantly improve the performance of passage retrieval. The framework, however, did not incorporate query expansion and it does not work well for short queries. As traditional query expansion methods only derive expanded terms without considering their relationship to the query terms, they cannot be applied directly to the fuzzy relation-based framework. Here we present a technique for extracting additional relation paths from the Web, to be used on top of the relation-based framework ([7], [18]) for passage retrieval. The path expansion technique extracts additional relation paths linking the expanded terms with the original query terms. By performing path expansion, the dependencies between query terms and the expanded terms are captured.

We now describe the main stages in performing relation path expansion from external resources. First, after performing term expansion (see Section 3.2), we name the path with starting node T_k as the path associated with T_k, and we index such paths according to T_k. In Section 3.2, we have already ranked all the T_k's in S. For each Tk, we select the path associated with T_k that has the maximum *path_score(T_k,t,j)* to be the expanded path of T_k denoted as *path_ex(T_k)*. The selection formula is given in Equation (2):

$$path\_ex(Tk) = \{path(T_k, t, j) \mid path\_score(T_k, t, j) = \max_{\substack{t \in Q \\ 1 \le j \le n}} \{path\_score(T_k, t, j\}\} \qquad (3.3.2)$$

Second, we formulate the expanded query Q'_r as comprising the relation paths derived from the original query Q, if any, and those extracted from the external resources. We

simply append the top k paths with weights (1-0.9*i/k) to the set of paths derived from the original query.

Third, we use Q'_r in a relation-based method as presented in [7] for passage retrieval. Essentially, we use the relation-based method proposed in [7] to perform passage re-ranking based on the initial set of passages obtained by the density-based method. For each answer candidate passage, *S*, we employ MiniPar to generate its dependency relation parse tree T_s. We then compute the similarity between T_s and Q'_r, by first finding all possible relation path pairs from T_s and Q'_r that have the same starting and ending nodes. We then treat the matching score (between 0 and 1) of a relation path from the candidate sentence as the probability of translating it to its corresponding path in the question. We denote the paired paths from the parsed query Q'_r and sentence T_s respectively as P_Q and P_S, whose lengths are represented as *m* and *n*. The translation probability $Prob(P_S|P_Q)$ is the sum over all possible alignments:

$$\Pr ob(P_S \mid P_Q) = \frac{\varepsilon}{m^n} \sum_{\alpha_1=1}^{m} \sum_{\alpha_n=1}^{m} \prod_{i=1}^{n} P_t(\mathrm{Re}\, l_i^{(S)} \mid \mathrm{Re}\, l_{\alpha_i}^{(Q)}) \tag{3.3.3}$$

where $\mathrm{Re}\, l_i^{(S)}$ stands for the i^{th} relation in path P_S and $\mathrm{Re} l_{\alpha_i}^{(Q)}$ 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 $\mathrm{Re}\, l_i^{(S)}$. ε stands for a small constant. $P_t(\mathrm{Re}\, l_i^{(S)} \mid \mathrm{Re}\, l_j^{(Q)})$ denotes the relation translation probability, i.e., relation mapping scores, which are given by a translation model learned during training.

3.3.4 Model Training

As explained in the previous section, the relevance of the expanded term Tk is inferred by its relation paths linking it to the query terms. To avoid the sparse data

problem in training, we assume that each relation appears independently of the other relations in the same path. Hence, we have:

$$path\_score(T_k, t) = \prod_{\text{Re}\, l_{i \in path(Tk,t)}} score(\text{Re}\, l_i) \tag{3.3.4}$$

Therefore for each type of relation in the dependency parsing tree, we need to estimate *score(Rel$_i$)* from the training corpus.

To perform training, we use the TREC 8 and TREC 9 QA question-answer pairs as the training set. We denote each QA pair as (Qi,Ai). We retrieve the top 100 snippets from Google for each question, perform sentence splitting and dependency parsing, and select the "relevant" paths from the set of parsing trees of the snippets. A path p in the snippets corresponding to Qi (denoted as <Start_Node, Rel$_1$...Rel$_k$...Rel$_m$, End_Node>) is relevant if $Start\_Node \in A_i$ and $End\_Node \in Q_i$. In other words, the relevant paths are those inferring a useful term to the question. After collecting all the relevant paths, we employ a unigram language model to train the weight of individual relations. Relation labels are treated as vocabularies in a language model. Therefore, the score of individual relation should be proportional to the probability of such a relation appearing in the training data set as shown in Equation (3.3.5). We use the smoothed probability to avoid zero values and take the log of frequency count to reduce the variance of the score. The eventual formula used to calculate the final score is given in Equation (3.3.4.3).

$$P(\text{Re}\, l_i) = C_{\text{Re}\, l_i \in relevant\_path} + 1 / ((\sum_{1 \le i \le N} C_{\text{Re}\, l_i \in relevant\_path}) + N) \tag{3.3.5}$$

$$score(\text{Re}\, l_i) = \log(C_{\text{Re}\, l_i \in relevant\_path} + 1) / \log((\sum_{1 \le i \le N} C_{\text{Re}\, l_i \in relevant\_path}) + N) \tag{3.3.6}$$

where $C_{Rel_i \in relevant\_path}$ is the number of Rel_i in relevant paths, and

N is the total number of relation types.

3.4 Evaluation

In this section, we present empirical evaluation results to assess our relation matching technique for passage ranking and query expansion. In Section 3.4.1, we will evaluate our passage ranking algorithm using dependency relations. In Section 3.4.2, we will evaluate our query expansion algorithm. In Section 3.4.3, we will provide a summary of evaluation results and error analysis.

To evaluate the performance of passage ranking, we employ three performance metrics: mean reciprocal rank (MRR)[29], percentage of questions that have no correct answers within top 20 passages, and precision at the top one passage. The former two metrics are calculated on the returned 20 passages by each system. However, there is no separate evaluation metric for query expansion. Hence, we have to integrate query expansion into some passage ranking systems and perform evaluation based on the above three metrics.

3.4.1 Evaluation of Dependency-Based Passage Ranking

We will test on the three hypotheses:

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

(2) Long questions are more suitable for relation matching. We 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. Therefore, we show that query expansion is still necessary for relation matching.

(3) It is computationally feasible to implement a relation matching algorithm for passage retrieval. The response time of a typical passage retrieval system varies from several seconds up to one minute. Therefore based on this time constraint we want to determine the input size and the efficiency performance curve based on the size of input passages.

- **Experimental Setup**

We use the factoid questions from the TREC-12 QA task [31] as test data and the AQUAINT corpus to search for answers. We use TREC-12 test data because the questions are long enough to obtain corresponding relation paths to perform relation matching. We 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.

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 we intend to evaluate passage retrieval techniques, we create the gold standard based on the official judgment list for the passage retrieval task provided by TREC. For each question, we generate a list of passages that are judged to be correct and supported by the corpus in the judgment list as standard answer passages. We cannot create the gold standard for 59 of the questions because no correct passages for them were judged by TREC evaluators. This leaves us with a final test set of 324 QA pairs, on which all evaluations in this paper are based. While Tellex et al. [29] made use of TREC-supplied exact answer

patterns to assess returned passages, we observe that common answer patterns can be matched in incorrect passages as answer patterns are usually very short. We 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. [29], we 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, we add the supporting documents that have the answer into the document set. We 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 [29]. In our evaluations for relation matching techniques, we take one sentence as a passage, as Minipar can only resolve intrasentential dependency relations. But for SiteQ, we still use the three-sentence window to define a passage.

We 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 fuzzy 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. We follow the adaptation described in [29] in our implementation.

NUS [6]: Another top-performing factoid question answering system. We utilize its

passage retrieval module, 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.

- ## Testing Hypothesis I

Hypothesis:

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

Testing Result (shown in Table. 3.4.1.1) and Result Analysis:

Applying relation matching over lexical matching methods boosts system performance dramatically. Applied on top of the MITRE and NUS systems, both strict and fuzzy 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 fuzzy 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

Table. 3.4.1.1 Overall performance comparison between passage retrieval systems 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. Fuzzy 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

judged by using paired t-test [14] ($p < 0.001$). We believe that the improvement 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.

Fuzzy 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 fuzzy 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 fuzzy 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 fuzzy matching methods. In this way, our statistical model is able to accommodate the variation in natural language texts.

Using MI and iterative EM to train relation mapping scores does not make any obvious difference in our tests. However, we 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. We 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, we cannot show the scalability of the two training methods given our limited test and training data.

- **Testing Hypothesis II**

Hypothesis:

Long questions are more suitable for relation matching. We 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. Therefore, we show that query expansion is still necessary for relation matching.

Testing Result (shown in Figure. 3.4.1.1) and Result Analysis:

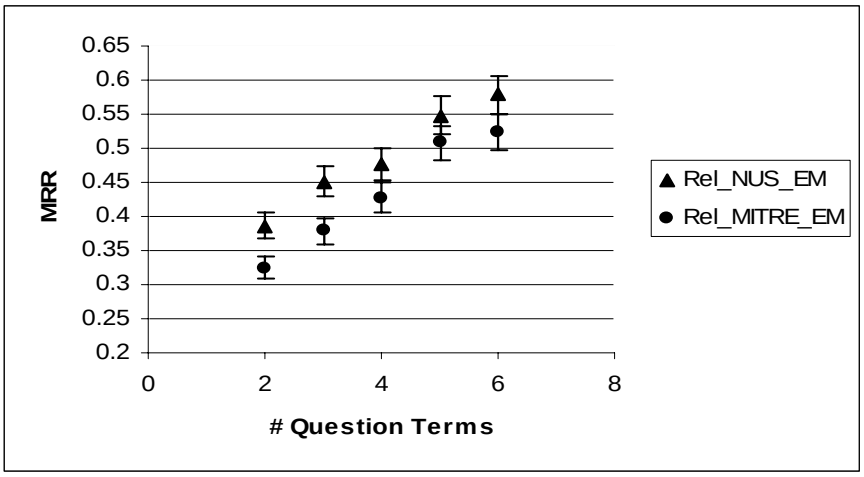

Figure. 3.4.1.1 Illustration of MRR variation to change in number of question terms.

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, we examine the effect of varying the number of non-trivial question terms on MRR.

Among the 324 questions in our test set, the number of question terms varies from one to 13, after removing trivial stop words such as “what”. In Figure 3.4.1.1, we plot the MRR values along with 95% error bars of the systems that apply fuzzy relation

matching with EM training on top of the MITRE and NUS systems when question length is varied. We 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 our test set.

From Figure 3.4.1.1, we can see 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. We surmise that with more paired corresponding paths, relation matching based ranking would be of higher precision. The result also indicates that query expansion technique is still need for short queries based on dependency relation matching.

- **Test Hypothesis III**

Hypothesis:

It is computationally feasible to implement a relation matching algorithm for passage retrieval. The response time of a typical passage retrieval system varies from several seconds up to one minute. Therefore based on this time constraint we want to determine the input size and the efficiency performance curve based on the number of input passages.

Testing Result (shown in Figure. 3.4.1.2) and Result Analysis:

We compared the performance curve of three passage retrieval algorithms in Figure 3.4.1.2. The curve with diamond shaped dot is the NUS_EM passage ranking algorithm using dependency matching. The curve with square shaped dot is SiteQ passage ranking, a density-based passage ranking algorithm. The curve with triangle shaped dot is NUS_EM_OFFLINE, which uses the same ranking algorithm as NUS_EM but it pre-computes the dependency parsing tree. We vary the number of

input passage from 100 to 2000 to test the efficiency of these algorithms. The test is performed on a PC with CPU 2.8MHz and 1G main memory.

We have the following observations:

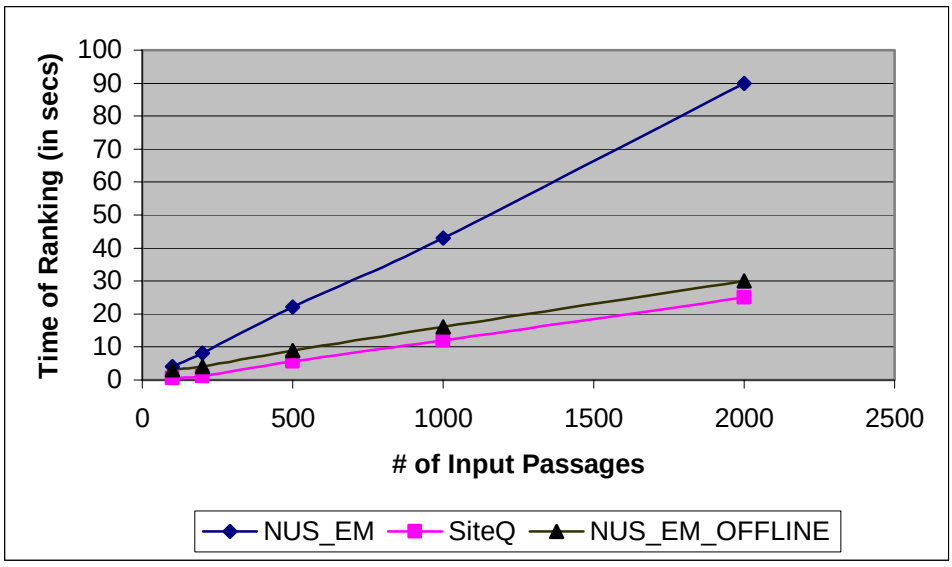

Figure. 3.4.1.2 Efficiency comparison between density-based algorithm and dependency matching algorithm for passage ranking.

(1) Density-based algorithms are more efficient than dependency matching algorithm. The most computational cost of dependency matching is dependency parsing. And if we pre-compute the parse tree the difference between the two algorithms are not significant.

(2) The time complexity of both density-based algorithm and dependency-based algorithm increase linearly with respect to the number of input passage.

(3) Even we set the number input passage to 2000, which is a large input size for passage retrieval, it is still computationally feasible to use dependency relation matching to perform passage ranking. However, in real applications, we suggest input size to be set between 500 and 1000 (for single CPU) in order to balance between efficiency and accuracy. Another possible strategy is to perform parsing offline, which requires more storage space.

3.4.2 Evaluation of Dependency-Based Query Expansion

We will test on the four hypotheses:

(1) Dependency-based query expansion technique further improves the precision of passage ranking when integrated with fuzzy relation matching technique.

(2) Short queries are more suitable for query expansion. We hypothesize that the effectiveness of query expansion is affected by query length. Short queries with fewer key terms are more like to have word mismatch problems when performing passage retrieval. Therefore, short queries will benefit more from dependency-based query expansion.

(3) (3) It is still possible to bring dependency-based term expansion (DRQET) technique into a density-based passage ranking framework. As normal web queries do not have any language constructs, we cannot use fuzzy relation matching technique, which requires the query to be in natural language format. And we hope that our query expansion technique outperforms the co-occurrence based query expansion technique.

(4) It is computationally feasible to implement the query expansion algorithm even for an interactive application, which usually requires a response time within a few seconds.

- **Experimental Setup**

We accumulate 10,255 factoid question-answer pairs from the TREC-8 and 9 QA tasks. For each question, we collect the top 100 snippets from Google, from which we extract 8,892 relevant paths used for training of individual relation weights using unigram language model.

We prepare two data sets D1 and D2 for experiment.

D1: It is the same data set from TREC-12 [31] QA task as described in Section 3.4.1.

D2: We prepare another data set D2, which only consists of short questions with less than three non-trivial question terms to simulate web queries. D2, which have 356 short questions, is obtained from the factoid questions in TREC-11 [33] and TREC-12 QA task [31] after filtering out questions with more than three non-trivial terms.

We create the gold standard for dataset D1 and D2 based on the official judgment list for the passage retrieval task provided by TREC. For each question, we generate a list of passages that are judged to be correct and supported by the corpus in the judgment list as standard answer passages. We use the following 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. [29], we 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, we add the supporting documents that have the answer into the document set. We conduct different passage retrieval algorithms on the document set to return the top 20 ranked passages.

We implement five systems and test them on both datasets D1 and D2:

NUS [6]: One of the top-performing factoid question answering system. We utilize its passage retrieval module, 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.

NUS+LCA: NUS system integrated with LCA (local context analysis method) for query expansion [35] based on top 100 Google snippets into the NUS system described above.

NUS+DRQET: NUS system integrated with dependency relation-based term expansion (DRQET) for query expansion based on top 100 Google snippets into the NUS system described above.

Rel_EM: A passage ranking algorithm based on fuzzy relation matching as described in the experimental setup of Section 3.4.1.

Rel_EM+DRQER: Rel_EM integrated with dependency relation-based path expansion (DRQER).

- **Testing Hypothesis I**

Hypothesis:

Dependency-based query expansion technique further improves the precision of passage ranking when integrated with fuzzy relation matching technique.

Testing Result (shown in Table. 3.4.2.1) and Result Analysis:

Table. 3.4.2.1. Overall performance comparison of query expansion systems of MRR, percentage of incorrectly answered questions (% Incorrect) and precision at top one passage of the five systems tested on dataset D1. All improvements obtained by relation matching techniques are statistically significant (p<0.001).

Passage retrieval systems	NUS	NUS+LCA	NUS+DRQET	Rel_EM	Rel_EM+DRQER
MRR	0.2677	0.3293	0.3616	0.4761	0.5541
% MRR improvement over					
NUS	N/A	+23.01	+35.08	+77.83	+106.99
NUS+LCA	N/A	N/A	+9.81	+44.58	+68.27
Rel_EM	N/A	N/A	N/A	N/A	+17.49
% Incorrect	33.02%	28.40%	27.16%	24.07%	21.60%
Precision at top one passage	0.1759	0.2315	0.2840	0.3889	0.4228

We observe from Table 3.4.2.1 that dependency-based query expansion (DRQER) technique further brings a 17.49% of improvement over fuzzy relation matching without integrated with DRQER. The improvement is statistically significant based on p=0.01. Obviously, the passage retrieval framework using relation analysis (Rel_EM+DRQER) significantly outperforms the density-based framework (NUS+LCA) by 68.27%.

- **Testing Hypothesis II**

Hypothesis:

Short queries are more suitable for query expansion. We hypothesize that the effectiveness of query expansion is affected by query length. Short queries with fewer key terms are more like to have word mismatch problems when performing passage retrieval. Therefore, short queries will benefit more from dependency-based query expansion.

Testing Result (shown in Figure. 3.4.2.1) and Result Analysis:

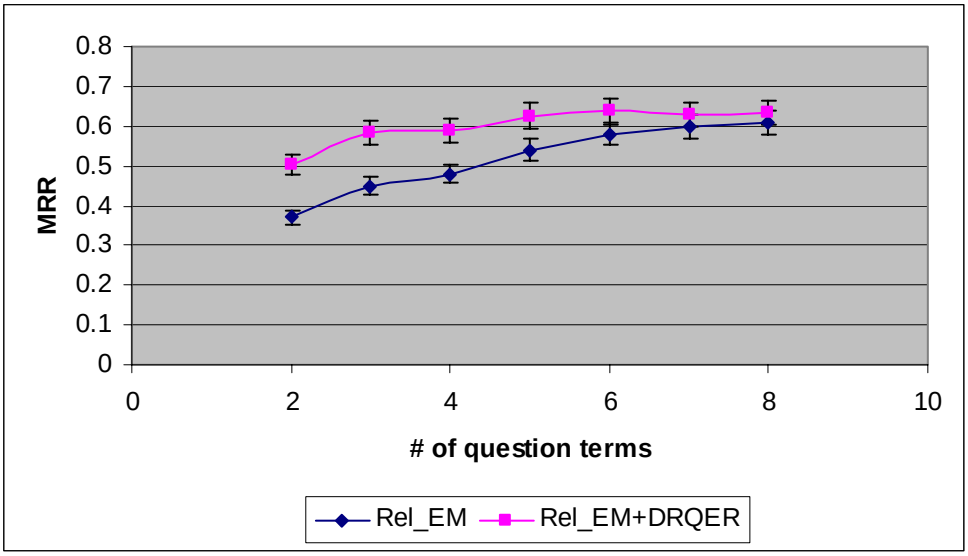

Figure. 3.4.2.1 Change of MRR after query expansion vs # of non-trivial question terms. (Test is performed on dataset D1)

In local context analysis framework, [35] research shows that query expansion is most useful for short queries. Therefore we want to test whether such statement still holds in the relation-based query expansion. From Figure. 3.4.2.1, we observe that such statement still holds. For example, query expansion can bring more than 30% of improvement for queries with less than three terms. However, for queries with more than five terms it can only bring less than 10% of improvement. As the number of question terms increases, the gap between two curves reduces, which means less improvement is brought by query expansion.

The reason is that the main idea of relation expansion is based on expanded term's relationship with query terms, which is still a term ranking technique. Such relation-based expansion is used to overcome the problem of word mismatch in passage retrieval. As we know, the severity of the problem tends to decrease as queries get longer, since there is more chance of some important words co-occurring in the query and relevant documents. So query expansion may be less useful in these cases.

- **Testing Hypothesis III**

Hypothesis:

It is still possible to bring dependency-based query expansion (term ranking) technique into a density-based passage ranking framework. Because normal web queries do not have any language constructs. Therefore, we cannot use fuzzy relation matching technique, which requires the query to be in natural langue format. And we hope that our query expansion technique outperforms the co-occurrence-based query expansion technique.

Testing Result (shown in Table. 3.4.2.1 and 3.4.2.2) and Result Analysis:

From Table 3.4.2.1, we can see that even relation-based query expansion (DRQET) is integrated with a density-based passage ranking system it still significantly outperforms the density-based system with only co-occurrence query expansion. A 9.81% of improvement is shown by comparing NUS+LCA and NUS+DRQET. We have known that short queries can benefit more from query expansion. Therefore we perform the same test on a different dataset D2, which only contains short queries with less than four terms to emphasize on the effect of relation-based query expansion on short queries. Table 3.4.2.2 shows the experiment result on dataset D2.

Table. 3.4.2.2 Overall performance comparison between query expansion systems of MRR, percentage of incorrectly answered questions (% Incorrect) and precision at top one passage of the five systems tested on dataset D2. All improvements NOT IN BOLD obtained by relation matching techniques are statistically significant (p<0.001).

* The improvement of +6.50 is not statistically significant with p=0.001.

Passage retrieval systems	NUS	NUS+LCA	NUS+DRQET	Rel_EM	Rel_EM+DRQER
MRR	0.1812	0.2413	0.2816	0.2570	0.3314
% MRR improvement over					
NUS	N/A	+33.17	+55.40	+41.83	+82.89
NUS+LCA	N/A	N/A	+16.70	**+6.50***	+37.34
Rel_EM	N/A	N/A	N/A	N/A	+28.94
% Incorrect	48.31%	41.85%	33.70%	38.48%	29.78%
Precision at top one passage	0.1096	0.1657	0.2022	0.1741	0.2612

The improvement is even more obvious on short queries. A 16.7% of improvement is shown by comparing NUS+LCA and NUS+DRQER, which indicates that even without considering language constructs in the question, relation based query expansion can still perform better than co-occurrence based query expansion. This result is especially useful for short keyword based web queries which are not in natural language format. As for these short keyword based queries, we cannot perform dependency relation matching. Therefore, an effective query expansion technique is especially useful in this case as it is one of effective ways to improve the performance on these short queries.

Interestingly, we find that relation matching without query expansion performs even worse than a density-based passage ranking plus a dependency-based query expansion module. This shows that when processing short queries improving the recall of term matching through query expansion should be given higher priority than resolving word dependencies within the original query. However, Rel_EM+DRQER performs the best among all the five system for both short queries and long queries. This is because Rel_EM+DRQER expand the query terms first and then resolve the dependency relations between query terms and expansion term. This result indicates that incorporating dependency relation analysis into both query expansion and passage ranking will boost the performance for passage retrieval.

- **Testing Hypothesis IV**

Hypothesis:

It is computationally feasible to implement the query expansion algorithm even for an interactive application, which usually requires a response time within a few seconds.

Testing Result (shown in Figure. 3.4.2.2) and Result Analysis:

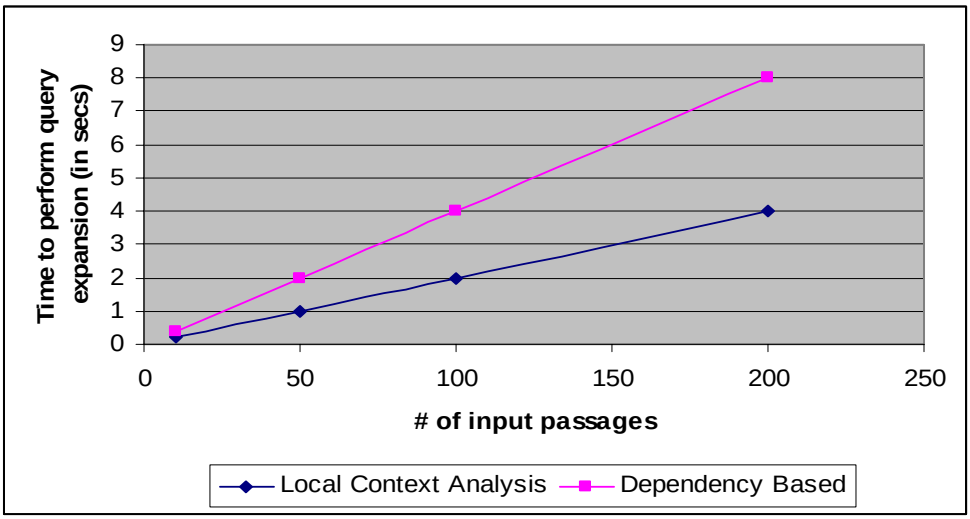

Figure. 3.4.2.2 Efficiency comparison between LCA and DRQER

The test is performed on a PC with CPU 2.8MHz and 1G main memory. Figure 3.4.2.2 shows that with an optimal [29] number of input passages equal to 200, the dependency relation-based query expansion only need 8 seconds to perform query expansion, which is within the acceptance range of an interactive application. And the major computational cost still comes from dependency parsing.

3.4.3 Error Analysis

Although we have shown that relation matching greatly improves passage retrieval, there is still plenty of room for improvement. A key question is whether we can further characterize the types of questions that are adversely affected by relationship matching. Based on the above two experiments, we perform micro-level error analysis on those questions for which relation matching degrades performance. We find that

fuzzy relation matching sometimes fails with incorrectly paired relation paths mainly for the following reasons:

(1) Mismatch of question terms: Since we do not consider the intermediate nodes in the relation path in some cases, we will mismatch the sentence with the different semantic meaning but the same syntactic dependency. In particular, we would encounter verb mismatch problem. For example, considering the questions "How many products of A is shipped to country B?" The following two answer candidates will have almost the same score: (1) 1000 As are imported from A. (2)1000 As are exported to A. although their semantic meaning is totally different. The solution to this problem is making use of "relationship graph" which is proposed in chapter 5.

(2) 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 [24] 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.

(3) Long-term dependency parsing error. As we mention earlier in this chapter, MiniPar usually fails to capture dependencies between terms with large lexical distance. We try to avoid such problem by imposing the path length restriction. However, such parsing error still exists for long-term dependencies.

(4) Noise data in external resources. We have proposed the use of dependency relation analysis in query expansion based on web snippets. During the expansion

process, we also observe some inconsistency between relation paths in the corpus and on the web. Our testing corpus is from news domain and the expressions in the documents are very limited. However, due to the noisy environment on web, the variation of the language expression is much more than those in the corpus, which will cause inconsistency in relation matching. We try to use the mutual information between relations to overcome the problem, but the training instances are not enough to cover all the cases. We also observed that some questions require domain dependent knowledge. However, we do not have a trained corpus of the domain as we do not employ a domain classifier into our framework.

3.5 Summary

In this chapter, we have presented a framework for passage retrieval using dependency relation analysis. The framework includes a novel fuzzy relation matching technique for passage ranking and a query expansion technique using dependency relation analysis. Past work has incorporated strict relation matching into QA system and claimed that the recall of such matching is low. However, we have shown that this conclusion does not generalize to passage retrieval. We show 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. Our evaluation results show that our technique produces significant improvements in retrieval performance in current systems: a vast 80% improvement in MRR over the density-based framework with co-occurrence-based query expansion. We have also demonstrated that our query expansion technique can be integrated into a density-based passage ranking system which can bring 16.7% of improvement for short queries compared to a co-occurrence-based query expansion module.

Chapter 4 Semantic Relation-Based Model for Passage Retrieval

In the previous chapter, we have successfully proposed a model for passage retrieval using dependency relation analysis [7], which makes us believe that relation analysis does help to improve the accuracy of passage retrieval. However, we realized that the goal of passage retrieval is to identify passages similar to the question in semantic content. Dependency relation, which is a syntactic relation, is only an approximation of semantic context just as co-occurrence relation is an approximation of dependency relation. In an attempt to improve the performance of passage retrieval, we want to explore further whether it is possible to employ a semantic matching model for passage retrieval and whether it is computationally feasible to implement such a model for real applications. In Section 4.1, we will propose a framework for semantic-based passage retrieval, which includes semantic parsing, verb expansion and semantic matching. In Section 4.2, we will provide evaluation results and error analysis. In Section 4.3, we will give a summary based on our semantic matching framework.

4.1 Framework for Semantic Passage Retrieval

Our goal is to match the semantic structure contained in the question in the passages. There are two major challenges to perfom semantic matching. One is that there are variations in syntactic representations of the same semantic structure, while the other is the variation in lexical terms (or synonyms) used to refer to the same concept. To tackle these problems, we use a shallow semantic parser to unify different

syntactic representations into the same semantic representation, and we use WordNet and eXtendedWordNet to find synonyms and semantically related terms.

In Section 4.1.1, we show how we use a shallow semantic parser to identify the semantic structures contained in the question and in the passages. Then in Section 4.1.2, we show how WordNet and eXtendedWordNet can be used to find semantically related verbs. Finally, in Section 4.1.3, we describe how we score a passage based on its semantic similarity with the question.

4.1.1 Shallow Semantic Parsing

To capture the semantic structures contained in a sentence, we need to identify verbs and their arguments. We also need to label the arguments with their semantic roles. This goal is achieved by shallow semantic parsing. The shallow semantic parser we use is the ASSERT [27] parser, which is trained on the PropBank [19] corpus and uses support vector machine classifiers.

PropBank was manually annotated with verb-argument structures. Following the annotation rules in PropBank, the ASSERT parser tags the arguments of a verb with labels from ARG0 up to ARG5. Although such semantic roles are verb-specific, some labels such as ARG0 and ARG1 tend to be general to all verb classes. For example, for any transitive verb, ARG0 is always the subject, and ARG1 is always the direct object. Besides these core arguments, ASSERT also tags adjunctive arguments. Examples are ARGM-LOC for locatives and ARGM-TMP for temporal. Figure.4.1.1 shows a sample output of ASSERT containing the parsing result of the question and several answer candidate sentences.

Q: [ARG2-FROM Where] was [ARG1 the first Burger King restaurant] [TARGET opened]

P1: [ARG1 The first Burger King] [TARGET opens] [ARGM-LOC in Miami]

P2: [ARG0 Burger King Corp.] [TARGET announced] [ARGM-TMP Thursday] [ARG1 it has canceled the rights of an Israeli company to operate a controversial franchise in the West Bank and ordered the restaurant to remove the Burger King brand from the site immediately]

P2: Burger King Corp. announced Thursday [ARG0 it] has [TARGET cancelled] the rights of an Israeli company [ARGM-PNC to operate a controversial franchise in the West Bank] and ordered the restaurant to remove the Burger King brand from the site immediately

P2: Burger King Corp. announced Thursday it has canceled the rights of an Israeli company to [TARGET operate] [ARG1 a controversial franchise in the West Bank] and ordered the restaurant to remove the Burger King brand from the site immediately

P2: Burger King Corp. announced Thursday [ARG0 it] has cancelled the rights of an Israeli company to operate a controversial franchise in the West Bank and [TARGET ordered] [ARG1 the restaurant] [ARG2 to remove the Burger King brand from the site immediately]

P2: Burger King Corp. announced Thursday it has canceled the rights of an Israeli company to operate a controversial franchise in the West Bank and ordered [ARG0 the restaurant] to [TARGET remove] [ARG1 the Burger King brand] [ARG2 from the site immediately]

Figure. 4.1.1.1 Sample Output By ASSERT Parser

To represent sentences in terms of the semantic structures, we define a semantic frame (or frame for short) as a verb-argument structure obtained from a sentence by the ASSERT parser. A frame consists of a verb, which we call the predicate, and a set of arguments. The arguments include both core arguments and adjunctive arguments. Each argument is associated with a label such as ARG0 and ARG1 to indicate the semantic role of the argument. Therefore, a frame F can be represented as $F=(v,A)$, where v is the predicate and A is the set of arguments. Each element a in A is a pair consisting the argument label and the argument text, represented by $a=(l,T)$, where l is the label and T is the set of terms the argument contains. Because a sentence may contain more than one semantic structure, a sentence is represented by a set of frames.

4.1.2 Verb Expansion using WordNet

Before we show our similarity scoring function, we first look at verb similarity scores. An answer passage can express the same semantic structure of the question with a different verb that is either of the same meaning as the verb in the question or semantically related to the verb in the question. Therefore, when matching semantic

frames, we need to consider the semantic similarity between two verbs. However, co-occurrence-based or relation-based query expansion techniques only expand terms without considering whether the expanded term is a verb or noun and how it is semantically related to the verb in the question. Therefore, instead of performing corpus or web-based query expansion, we decide to use specific langue resources. In this thesis, we use WordNet and eXtendedWordNet to expand the verb and measure their similarity.

Our verb similarity function is very similar to the weighting function used Moldovan *et al. 2002*. [25] Suppose that we want to measure the similarity between two verbs v_1 and v_2. We start from one of the verbs, say, v_1, which is assigned a score of W_0 (we set W_0=1). We select the synset that corresponds to the first sense of v_1 in WordNet. All words in this synset get the same score as the original word. From this synset, we follow the links to other synsets with relations such as hyponyms and entailment. We also follow the gloss links and reverse gloss links provided by the eXtendedWordNet. A gloss link from synset S_1 to synset S_2 means S_2 appears in the gloss of S_1, and a reverse gloss link from synset S_1 to synset S_2 means S_1 appears in the gloss of S_2.

If from v_1 we follow the relation R to a synset which contains the word w, then the score for the word w is $W_0 \times W_R$, where W_R is the weight for the relation R. If we follow the link further from w to another synset which contains the word u by relation S, then the score of the word u is $W_0 \times W_R \times W_s$. Such expansion continues until we reach a certain depth. In our experiments, we set the depth to 2 because our preliminary experiments show that a deeper expansion does not improve the performance. The weights of the relations are the same as that used in Moldovan *et al. 2002*. [25] We also penalize synsets that are more commonly used. Each synset gets a

generality score *G* which is defined as

$$G_s = \frac{C}{C + N_{r-gloss}} \tag{4.1.1}$$

where *C* is a constant, and $N_{r-gloss}$ is the number of glosses in which the synset *S* appears. We set *C*=500. After taking into account this penalizing weight, the score of the word w is therefore $s_v \times R_{v,w} \times Gs_w$, where s_v is the score of the previous word *v*, $R_{v,w}$ is the relation connecting v and w, and s_w is the synset containing *w*.

After we expand the verb v_1, we check if v_2 appears in the expanded set. If it does, then it is assigned the score as explained above. If not, v_2 is assigned a score of 0. We denote this similarity score between v_1 and v_2 as $Sim_V(v_1, v_2)$.

4.1.3 Semantic Matching

First, we define the similarity scores between two frames. Let *F1= (v1, A1)*, *F2= (v2, A2)*. We divide the similarity score into two components, one indicating the similarity between the verbs, and the other indicating the similarity between the arguments.

$$Sim(F_1, F_2) = \alpha \times Sim_V(v_1, v_2) + (1-\alpha) \times Sim_A(A_1, A_2) \tag{4.1.2}$$

where $Sim_A(A_1, A_2)$ denotes the similarity score between two argument sets, and *v* is a weighting parameter that can be tuned. In this thesis, we simply fix α to be 0.5.

Similarity between the two sets of arguments is measured at the lexical level. We do not use WordNet to expand the terms in the arguments because many of the arguments are named entities such as persons and organizations, for which finding similar terms is not so meaningful.

To precisely match the argument sets of two frames, we should do pairwise matching of the arguments, that is, matching ARG0 in the first frame with ARG0 in

the second frame, matching ARG1 in the first frame with ARG1 in the second frame, etc. However, we choose to do a fuzzy matching by considering all arguments in a frame together. There are two reasons for doing fuzzy matching: (1) ASSERT can make mistakes and therefore does not tag the arguments consistently, especially for adjunctive arguments. (2) Because we consider semantically related verbs, the semantic roles of the arguments may be different in different frames. Our preliminary experimental results also showed that considering all arguments together is better than considering them separately.

We use Jaccard coefficient to measure the similarity between the two sets of arguments. Suppose that we are to compute $Sim_A(A_1, A_2)$, where A_1 and A_2 are the two argument sets:

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

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

$l_{i,j}$ is the argument label of the jth argument of Ai, and $T_{i,j}$ is the set of terms in the jth argument of Ai. Let

$$T_1 = \bigcup_{i=1}^{m} T_{1,i}$$

$$T_2 = \bigcup_{i=1}^{n} T_{2,i}$$

We then remove the stop words from T_1 and T_2. Let the sets of terms after stop word removal be T_1' and T_2'. We then define the similarity between A_1 and A_2 as

$$Sim_A(A_1, A_2) = \frac{|T_1' \bigcap T_2'|}{|T_1' \bigcup T_2'|} \tag{4.1.3}$$

Both the question and the answer passages can contain more than one semantic frame. We compute pairwise frame similarity scores between the question and an

answer passage, and pick the maximum score as the semantic similarity between the question and the answer passage. We then use the semantic similarity scores to rank passages.

As you may notice that we don't propose any sophisticated model for semantic passage ranking. The reason is shown in our later experiment that even such a simple model is computationally expensive and not feasible for a passage retrieval system.

4.2 Evaluation

In this section, we present empirical evaluation results to assess our semantic matching technique for passage retrieval.

4.2.1 Evaluation of Semantic-based Passage Retrieval

We have three hypotheses to test:

(1) Semantic-based passage retrieval can outperform density-based and dependency-based passage retrieval.

(2) Semantic passage retrieval based on verb similarity can outperform the strict matching of verbs.

(3) It is computationally feasible to implement a semantic-based framework for passage retrieval.

- **Experimental Setup**

We use the same experimental setup and evaluation metric as described in Section 3.4.2 to evaluate our system. During the experiment setup we noticed that the ASSERT parser can only handle simple semantic structures in the question. Some complex questions will result in empty output by the parser. Hence for many questions we are not able to perform the matching based on semantic structures.

Therefore we use only a sub-set of short questions with simple semantic structures, which is referred as D2 in Section 3.4.2 in our experiments.

Dataset D2 only consists of short questions with less than three non-trivial question terms to simulate web queries. D2, which have 356 short questions, is obtained from the factoid questions in TREC-11 [33] and TREC-12 QA task [31] after filtering out questions with more than three non-trivial terms.

Other settings remain unchanged as in Section 3.4.2. Apart from the five systems implemented in Section 3.4.2, which are listed below. We implement two systems using semantic matching, known as Sem_Strict, Sem_Sim. Sem_Strict perform semantic frame matching using strict matching of verb, while Sem_Sim uses verb similarity to measure the similarity between verbs.

Below are the seven systems that we have implemented and the test is performed on dataset D2:

1) NUS [6]: A top-performing factoid question answering system. We utilize its passage retrieval module, 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.

2) NUS+LCA: NUS system integrated with LCA (local context method) for query expansion based on the top 100 Google snippets.

3) NUS+DRQET: NUS system integrated with dependency relation-based query expansion (DRQET) for query expansion based on the top 100 Google snippets.

4) Rel_EM: A passage ranking algorithm based on fuzzy relation matching as described in the experimental setup of Section 3.4.1.

5) Rel_EM+DRQER: Rel_EM integrated with dependency relation-based query expansion (DRQER).

6) Sem_Strict: A passage retrieval system based on semantic frame matching using strict matching of verbs.

7) Sem_Sim: Another passage retrieval system based on semantic frame matching. However, it adopts verb similarity defined in Section 4.1.2 to measure the similarity between verbs.

- **Testing Hypothesis I**

Hypothesis:

Semantic-based passage retrieval can outperform density-based and dependency-based passage retrieval.

Testing Result (shown in Table. 4.2.1.1) and Result Analysis:

Table. 4.2.1.1 Overall performance comparison between passage retrieval systems of MRR, percentage of incorrectly answered questions (% Incorrect) and precision at top one passage of the seven systems tested on dataset D2. All improvements WITHOUT * obtained by relation matching techniques are statistically significant (p<0.001).

Passage retrieval systems	NUS	NUS+LCA	NUS+ DRQET	Rel_EM	Rel_EM+ DRQER	Sem_ Strict	Sem_ Sim
MRR	0.1812	0.2413	0.2816	0.2570	0.3314	0.2274	0.2520
% MRR improvement over							
NUS	N/A	+33.17	+55.40	+41.83	+82.89	+25.50	+39.07
NUS+LCA	N/A	N/A	+16.70	+6.50*	+37.34	-5.76*	+4.43*
Rel_EM	N/A	N/A	N/A	N/A	+28.94	-11.51*	-1.95*
Rel_EM+DRQER	N/A	N/A	N/A	N/A	N/A	-31.38	-23.96
% Incorrect	48.31%	41.85%	33.70%	38.48%	29.78%	42.98%	38.76%
Precision at top one passage	0.1096	0.1657	0.2022	0.1741	0.2612	0.1404	0.1713

We observe from Table 4.2.1.1 that semantic relation matching does not perform as well as expected. Sem_Strict performs even worse than density-based system with query expansion (NUS+LCA). Sem_Sim only improves the performance by 4.43% as compared to NUS+LCA, which is not significant. Both Sem_Strict and Sem_Sim perform worse than passage retrieval system based on dependency relation matching. There are two major reasons for this: (1) The ASSERT parser cannot correctly identify the semantic structure in many of the questions and the candidate passages. (2) Our matching model is too simple to handle semantic variations of natural language.

However, our later experiment shows that even such a simple model is computationally very expensive due to the complexity of semantic parsing.

- **Testing Hypothesis II**

Hypothesis:

Semantic passage retrieval based on verb similarity can outperform the strict matching of verbs.

Testing Result (shown in Table. 4.2.1.1) and Result Analysis:

From Table 4.2.1.1 , we can see the Sem_Sim outperforms Sem_Strict by 10.81%. This indicates that if we still want to improve based on the existing semantic matching model, similarity matching approach is preferred than strict matching approach.

- **Testing Hypothesis III**

Hypothesis:

It is computationally feasible to implement a semantic-based framework for passage retrieval.

Testing Result (shown in Figure. 4.2.1.1) and Result Analysis:

From Figure 4.2.1.1, we can see that it is computationally very expansive to perform semantic matching as compared to density-based approach and relation matching. The major cost comes from semantic parsing. For a typical input size of 1000 passages it takes almost 10 minutes to parse all the sentences. Therefore we would like to conclude that efficiency-wise such a matching model is more appropriate for answer extraction in QA rather than for passage retrieval.

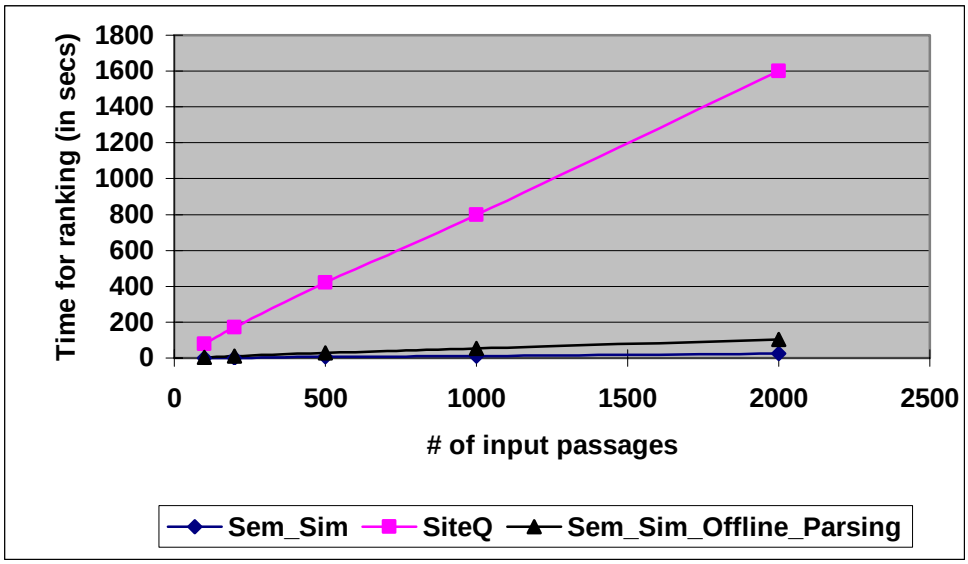

Figure. 4.2.1.1 Efficiency comparison between density-based passage ranking with semantic passage ranking

4.2.2 Error Analysis

Due to computational complexity reasons, we only propose a simple model for semantic matching. Therefore our model can only handle simple semantic structures such as subject-verb-object triples. In order to handle more complex semantic language structures, a more sophisticated matching model should be proposed. And we believe that such model will be more useful for answer selection than passage retrieval in the QA system due to computational complexity.

The following are the major errors occurred during the semantic matching process:

First, currently, when finding semantically related verbs, we take the first sense of a verb in WordNet as its meaning, for both the verbs in the questions and the verbs in the passages. This assumption certainly does not hold all the time. To address this problem, some word sense disambiguation may be performed.

Second, ASSERT does not recognize phrasal verbs. For example, in the question "What does the name stand for?" we cannot isolate the verb stand from the particle for. We will rely on the improvement of shallow semantic parsers to solve this problem.

Finally, the most difficult problem is to handle semantic variations. We can see that

there are hidden semantic structures that a shallow semantic parser may not recognize. For example, “William Ding, the founder of Netease" is equivalent to “William Ding founded Neteast", and “his discovery of prions" is equivalent to “he discovered prions". If there are no such examples in the training data, a shallow semantic parser will not be able to recognize these semantic structures as equvalent.

4.3 Summary

In this work, we proposed a semantic structure based passage retrieval method. The semantic structures of the question and of the answer passages are identified using a shallow semantic parser. Similarity between two semantic structures is measured by considering the similarity of the verbs and the similarity between the arguments. Passages are ranked according to these semantic similarity scores. Our experimental results show that for questions that contain relatively simple verb-argument semantic structures, this semantic structures based method outperforms a density-based method. However, it does not perform as well as our previous dependency relation matching model.

Our experimental result shows that a simple (compare to the complexity of such model in a QA system [13], [27]) semantic matching model does not help much in passage retrieval. Therefore a more sophisticated model, which should make use of logic reasoning among different semantic frames, should be proposed. However, such a model is beyond the complexity of a typical practical passage retrieval framework.

During these experiments we noticed the fact that some methods that work well in one application may not work well in another. For example, [3] claimed that strict matching of dependency relations does not perform well on answer extraction. However, we have shown that such statement is not true for passage retrieval. Another

example is that semantic reasoning is a very useful technique in QA. But we have shown that such framework is not suitable for passage retrieval. All these examples indicate there are some differences between QA and passage retrieval despite the fact that they are closely related to each other. Therefore in the final chapter, we will address such differences and propose future research directions of passage retrieval.

Chapter 5 Passage Retrieval using Relationship Graph

In the previous two chapters, we have proposed two different passage retrieval models based on relationship analysis. We observed from experimental results that both models can outperform the classical density-based model. However, we also identified some advantages and weaknesses of each model. The dependency relation-based model has a high flexibility in matching relations and therefore result in a high recall but low precision. In particular, it usually encounters errors when performing specific verb matching. On the other hand, semantic relation-based model has high precision but low recall because we enforce exact match of the SVO (subject, verb, object) frames. Therefore combining the two models will be an obvious approach if we want to further improve the performance of passage retrieval. In this chapter, we will propose the framework of passage retrieval using relationship graph. In Section 5.1, we will describe in detail the framework. In Section 5.2, we will provide experimental results and result analysis.

5.1 Model of Relationship Graph

In this section, we will propose the framework of passage retrieval using relationship graph, which consists of three major steps: building relationship graph from the query, expanding relationship graph and mapping relationship graph from query to answer candidate. We will cover each of the steps in detail in the following three sections.

5.1.1 Building Relationship Graph from Query

As mentioned in the beginning of this chapter, we are going to combine the dependency-based model and semantic-based model to produce the model of relationship graph. Our goal is to propose a matching model that has high precision as semantic-based model while without losing much of the flexibility of the dependency relation matching. The construction of a relationship graph consists of two steps: (1) Building dependency parsing tree of the query. (2) Adding semantic edges in the dependency parsing tree. (3) Tagging expected answer NE (Named Entity) type. We will illustrate the process in detail using a concrete sample query: "Arms ship to Colombian insurgents; specifically, the different routes used for arms enter Colombia and the entities involved."

(1) Building dependency parsing tree

The query is submitted to the MiniPar and the parsing tree is shown in Figure 5.1.1.1. The parsing tree can be seen as the initial form of the relationship graph.

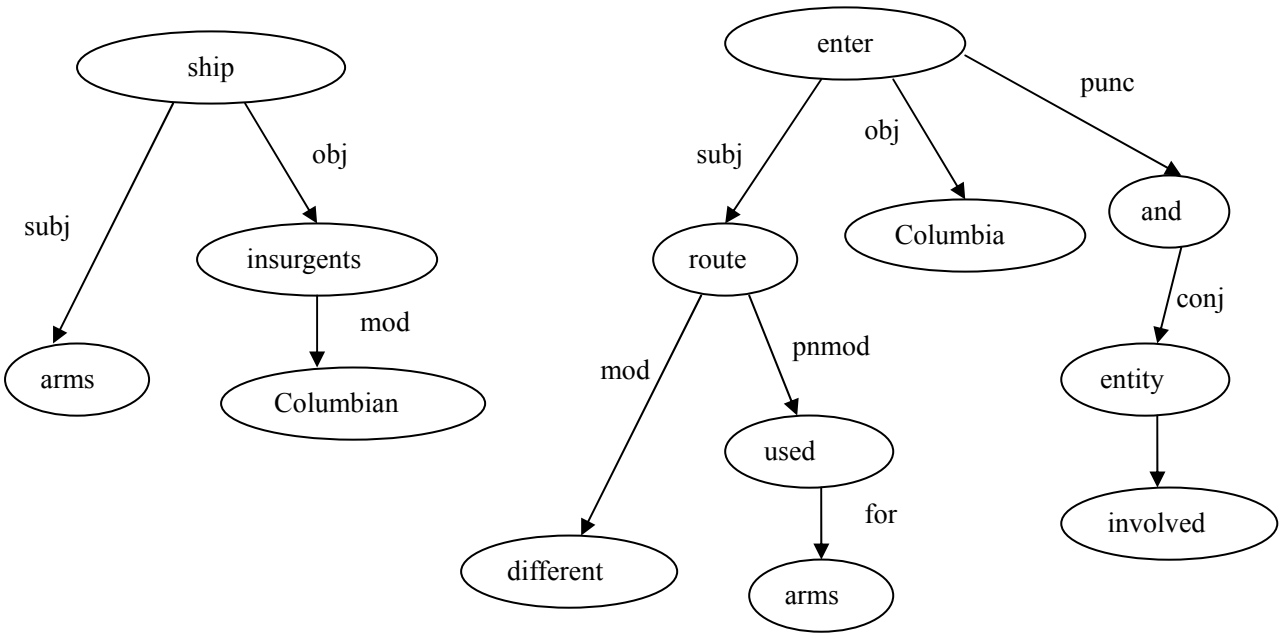

Figure. 5.1.1.1 Dependency parsing of the sample query

(2) Adding semantic edges

After performing step 1, the query is submitted to ASSERT semantic parser where the semantic frames of the query is constructed. There are two semantic frames in the sample query: ship(arms, insurgents) and enter(arms, Columbia). We add semantic edges to the parsing tree to form the relationship graph. A semantic frame is denoted as $P(A_0,A_1)$ and a syntactic dependency edge between two nodes is denoted as $E_{syn}(N_0,N_1)$. A semantic edge E_{sem} is labeled by its predicate P and it is added to the tree T if $subj \in E_{syn}(A_0, A_1) \wedge obj \in E_{syn}(A_0, A_1)$. Figure. 5.1.1.2 shows the result of adding semantic edges to the parsing tree in Figure 5.1.1.1.

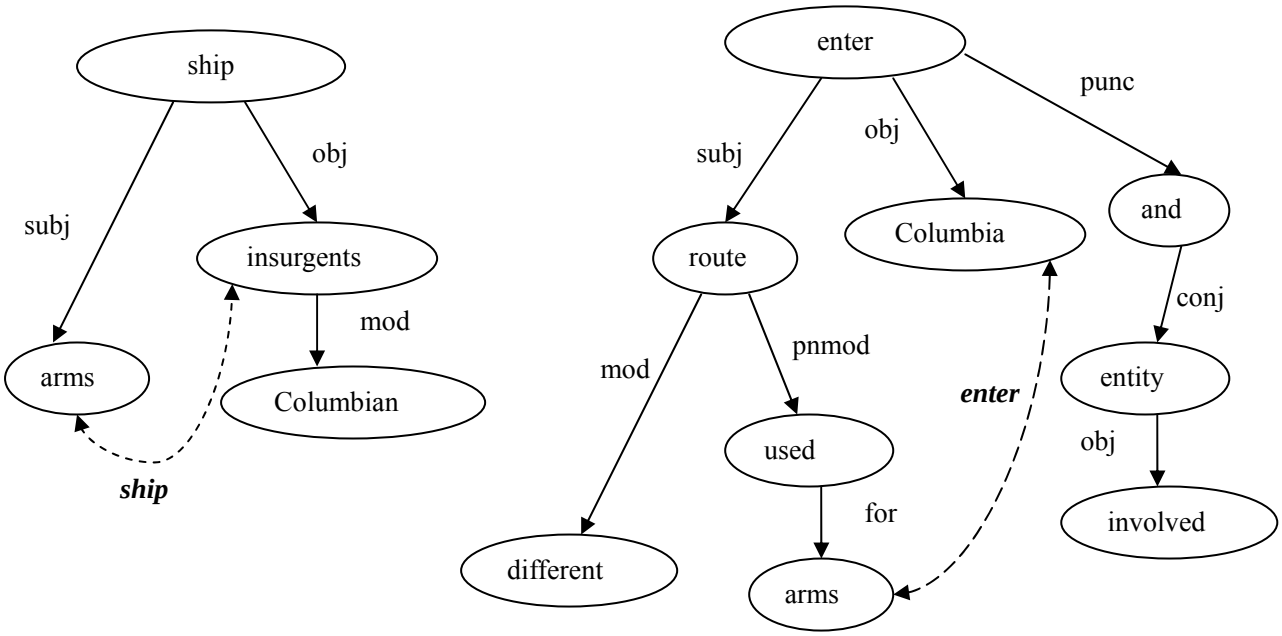

Figure. 5.1.1.2 Adding semantic edges to Figure 5.1.1. The dotted line is the semantic edge.

(3) Tagging expected answer NE (Named Entity) type

Finally, general entities will be substituted by their respective NE types for flexible matching. For example, county is replace by the NE type LOC_COUNTRY. Figure. 5.1.1.3 shows the relationship graph after step 3. In the sample query, entity is replace by NE type ORG.

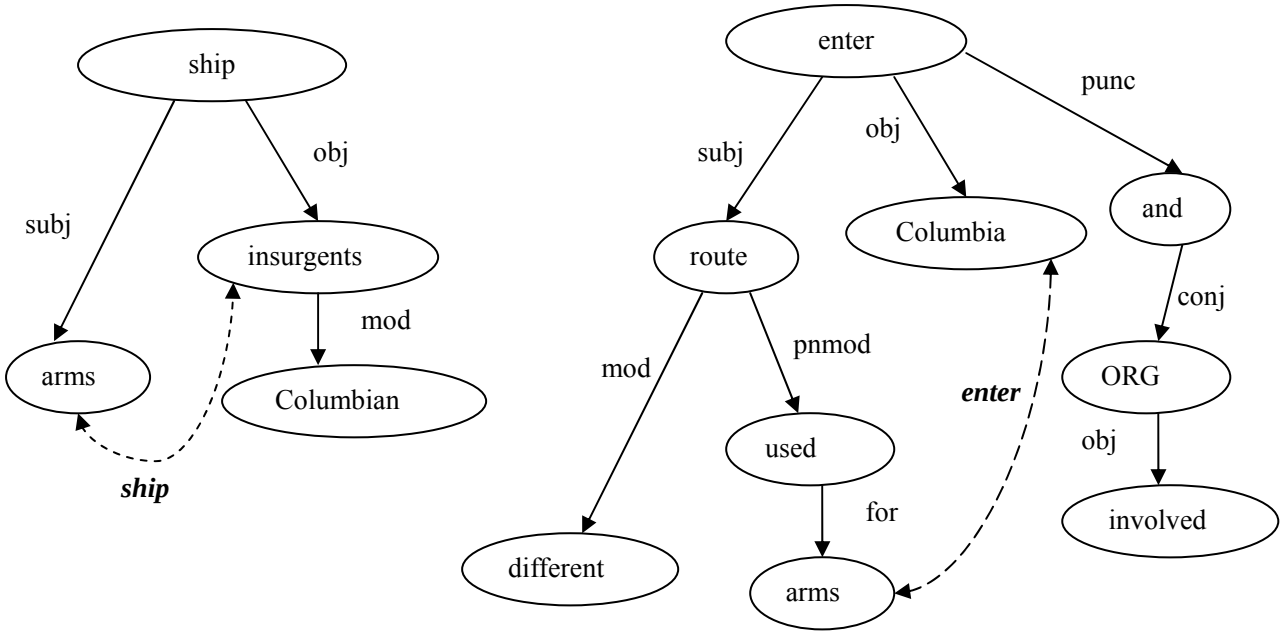

Figure. 5.1.1.3 Relationship graph built from the query after substituting the NE types

5.1.2 Expanding Relationship Graph

The expansion consists of two steps: (1) Dependency path expansion (2) Semantic verb expansion. We submit the query to Google and use top 100 Google snippets to perfrom dependency path expansion. And we use WordNet and eXtendedWordNet for semantic verb expansion. The details of the methods have been described in Sections 3.3 and 4.1.2 respectively. Figure. 5.1.2.1 shows partial result of the relationship graph after expansion.

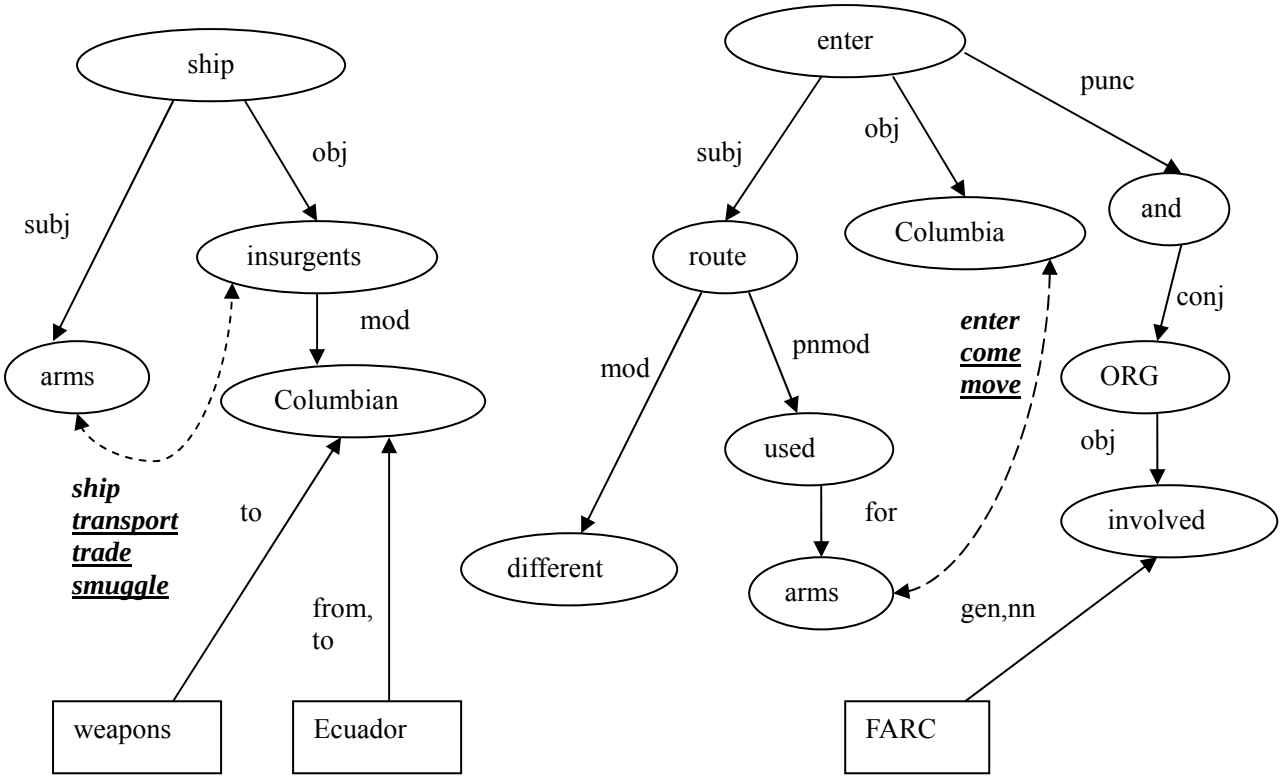

Figure. 5.1.2.1 Relationship graph after expansion. Nodes in square shape are the expanded nodes and the underlined verbs are the expanded verbs.

5.1.3 Matching Relationship Graph

After building and expanding the relation graph, we perform graph matching against answer candidate sentences. For each answer candidate, we parse the sentence using MiniPar and ASSERT and perform NE tagging. We then perform dependency matching and semantic matching separately using the method described in Section 3.2 and Section 4.1.3. Now each edge in the relationship graph has a matching score between zero and one. If the edge is not matched it will get a score of zero. Finally, we calculate the overall matching score of the candidate. In the model of relationship graph, two types of edges may appear between two vertices in the graph syntactic dependency edge E_{syn} and semantic edge E_{sem}. As our earlier experimental results in Section 4.2 showed that semantic matching has a higher precision, we therefore trust more on the results of semantic matching. Thus we impose the following edge

selection criteria: (1) If both E_{syn} and E_{sem} appear between N_0 and N_1 then we only calculate the score of E_{sem}. (2) If there are only E_{syn} edges or E_{sem} edges appear between N_0 and N_1, we pick the edge with the maximum matching score. Figue 5.1.5 shows the matching of the sample answer candidate sentence "FARC receives arms coming into Columbia from Ecuador and the Pacific and Atlantic coasts." to the sample query.

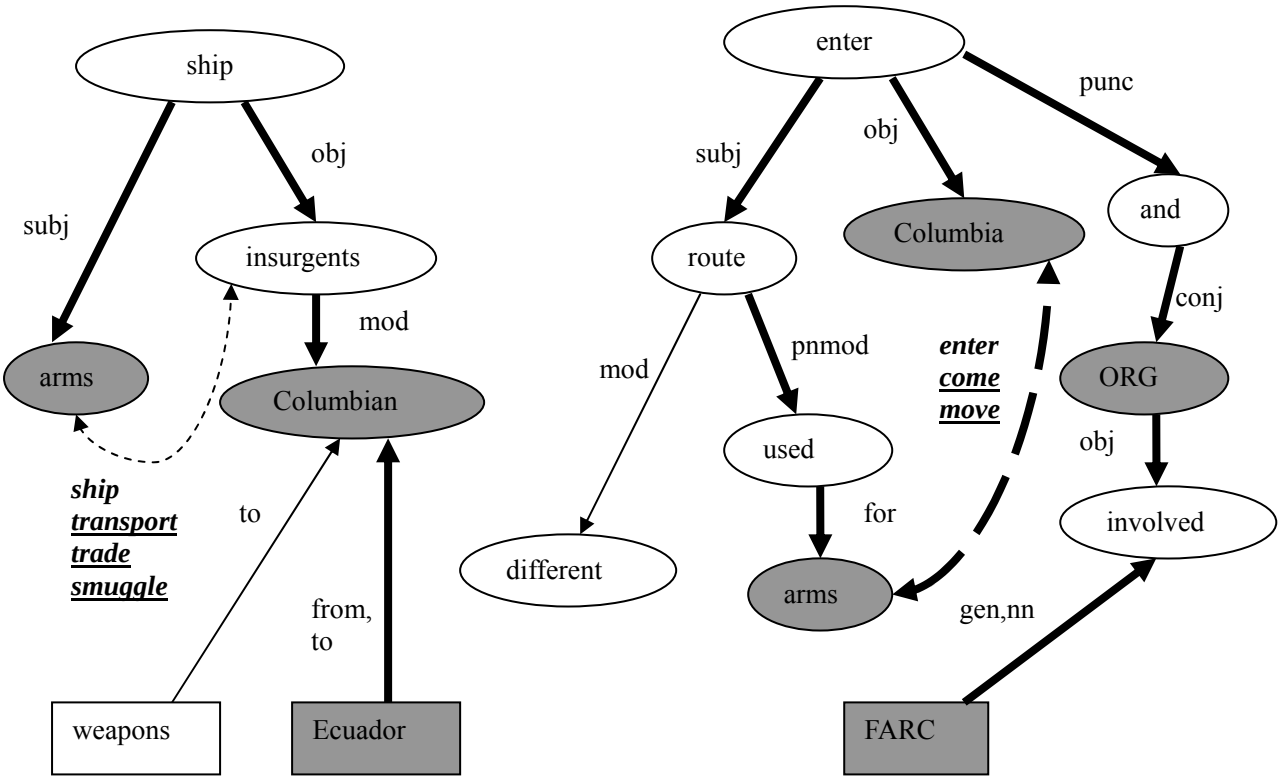

Figure 5.1.2.2 Matching of relationship graph against sample answer candidate sentence. Nodes in grey shows the matched node and arrows in bold shows the matched path.

We can see that there are both semantic edge and syntactic edge between "arms" and "Columbia" in the graph. However, according to rule only the score of semantic matching is counted. As "FARC" is tagged as ORG in the answer candidate we match the word "FARC" as well as the NE type "ORG" against the query.

5.2 Evaluations

In this section, we will provide evaluation results of passage retrieval using relationship graph. In particular, we have two testing hypothesis: (1) Relationship graph is more suitable for complex questions, which involves multiple entities and relationships. (2) Relationship graph matching can perform better than single relation matching using either dependency or semantic relation.

- **Experimental Setup**

We conduct two experiments in the evaluation process. In this first experiment, we study the performance of relationship graph matching on short queries. We use the data set D2 as described in Section 3.4.2 with the same evaluation metric. In the second experiment, we will try to tackle the more complex questions that involve more entities and relations. For this, we use the testing data set from TREC 2005 relationship task [34]. For each query, the TREC assessor manually creates a set of answer nuggets or passages with label either "vital" or "ok", which is used as gold standard answer passage set in our evaluation. For each question, a maximum of 20 ranked nuggets are returned as answers. Given the nugget list and the set of nuggets matched in a system's response, the nugget recall of the response is the ratio of the number of matched nuggets to the total number of vital nuggets in the list. Nugget precision is much more difficult to compute since there is no effective way of enumerating all the concepts in a response. Instead, a measure based on length (in non-white space characters) is used as an approximation to nugget precision. The length-based measure starts with an initial allowance of 100 characters for each (vital or non-vital) nugget matched. If the total system response is less than this number of characters, the value of the measure is 1.0. Otherwise, the measure's value decreases

as the length increases using the function $1 - \frac{length - allowance}{length}$. The final score for a question is computed as the F measure with nugget recall three times as important as nugget precision: $F(\beta = 3) = \frac{10 \times precision \times recall}{9 \times precision + recall}$. In both experiments, we build four comparison systems:

a) NUS+LCA: NUS system [6] integrated with LCA (local context analysis method) for query expansion [35] based on top 100 Google snippets.

b) Rel_EM+DRQER: Rel_EM [7] integrated with dependency relation-based query expansion (DRQER).

c) Sem_Sim: A passage retrieval system based on semantic frame matching. However, it adopts verb similarity defined in Section 4.1.2 to measure the similarity between verbs.

d) RG: A passage retrieval system using relationship graph matching.

- **Experimental Result**

The experiment result is shown in Tables 5.2.1 and 5.2.2 respectively.

Table. 5.2.1 Overall performance comparison of MRR, percentage of incorrectly answered questions (% Incorrect) and precision at top one passage of the four systems tested on dataset D2.

Passage retrieval systems	NUS+LCA	Sem_ Sim	Rel_EM+ DRQER	RG
MRR	0.2413	0.2520	0.3314	0.3410
% MRR improvement over				
NUS+LCA	N/A	+4.43	+37.34	+41.32
Sem_Sim	N/A	N/A	+28.94	+35.32
Rel_EM+DRQER	N/A	N/A	N/A	+2.90
% Incorrect	41.85%	38.76%	29.78%	29.49%
Precision at top one passage	0.1657	0.1713	0.2612	0.2669

Table. 5.2.2 Overall performance comparison of precision, recall and F3 measure of the four systems tested on TREC 2005 relationship task data set.

Passage retrieval systems	NUS+LCA	Sem_ Sim	Rel_EM+ DRQER	RG
F3	0.1537	0.0903	0.2783	0.3002
% F3 improvement over				
NUS+LCA	N/A	-41.24	+81.06	+95.32
Sem_Sim	N/A	N/A	+208.19	+232.44
Rel_EM+DRQER	N/A	N/A	N/A	+7.87
Precision	0.1782	0.3471	0.2413	0.2917
Recall	0.1514	0.0834	0.2831	0.3012

We may draw the following conclusions from the above results:

(1) Relationship graph is more useful when answering complex questions with multiple entities and relationships. We can compare the improvement between RG and Rel_EM+DRQER in simple queries and complex queries. We observe that the improvement is merely 2.90% for simple queries. However, a 7.87% of improvement is observed for complex queries. The main reason is because the main difference between relationship graph and dependency parsing tree is the semantic edges. As we know that semantic edges tend to improve precision, the more semantic edge we have the more precision it is likely to gain. However, for short queries there is usually only one semantic edge in the tree. Thus the performance gain is not significant.

(2) Semantic relation matching improves more on precision and dependency relation matching improves more on recall, which again justify the assumption we made at the beginning of this chapter. From Table 5.2.2, we see that the precision of semantic matching is more than four times higher than its recall. Since we are using F3 measure the final score is severely punished for semantic matching, and the final F3 score of semantic matching is even worse than the density-based method.

5.3 Summary

In this chapter, we have proposed the model of relationship graph, which is a combination of the previous models. Experimental results show that the model is useful to handle complex queries, which is in natural language format. We know that the relationship graph model is a combination of dependency relation and semantic relation. In this thesis, we just experiment on a simple binary merging technique, which simply picks matching scores from one model (either dependency or semantic) based on different cases. We also believe that employing other model fusion techniques such as instance classification might further improve the performance. However, due to time constraint and lack of experimental data, we do not try other alternatives of model fusion. But we do believe this will be a potential area to explore in our future work.

Chapter 6 Conclusions and Future Work

In this thesis, we proposed three frameworks for passage retrieval, all of which are based on relation analysis. Our experimental results showed that all of the models can outperform the density-based method for passage retrieval, thus demonstrating the effectiveness of using relationship analysis in passage retrieval.

However, our result also showed that the use of semantic matching does not perform as well as syntactic relation matching in passage retrieval both in terms of efficiency and accuracy; although semantic matching is usually better than syntactic matching in answer selection module of QA systems. Also in our experiment, we observed that semantic relation matching improves more on precision while dependency relation matching improves more on recall. To explain this, we will first examine the difference between passage retrieval and answer selection in Section 6.1. In Section 6.2, we summarize the success factors of a good passage retrieval model. Finally, in Section 6.3 we will address the major challenges that we are facing to further improve the performance of passage retrieval and question answering.

6.1 Passage Retrieval vs Answer Selection

The major difference between passage retrieval and answer selection is the different approach we use to model such task. In other words, the fundamental model we applied for the two tasks are very different although we usually think that QA is an extension of passage retrieval. Most open domain passage retrieval algorithms are based on statistical modeling of language. However, in answer selection rule-based modeling is more useful.

Answer selection is a question specific task, which means for each type of question

(one way of defining such type is using the named entity type of answer target), the way to select answer string is different from the other types. Even within the set of same typed questions, answers may also appear in different language structures. Therefore it is very difficult for statistical models to handle such specific task as they are usually used to learn rules that occur frequently. Hence, people propose using rule-based models to perform answer selection [13], [27]. The rules can be as specific as one question or as general as a set of questions; and when new instances come in we just update the rules and make sure that existing rules do not conflict with each other. The size of such a rule base can be much larger than the set of rules that you can obtain from statistical learning and therefore it is more precise. For example, in [27] researchers performs answer selection based on reasoning rules over semantic frames. Harabagiu *et al.* [13] applied a theorem prover that conducts rule-based reasoning over WordNet to derive semantic relationship between words.

Unfortunately, rule-based models are not suitable for open domain passage retrieval due to its complexity because it is very inefficient to apply the set of rules for each of passage even if they are totally irrelevant to the query. Therefore in the QA system pipeline [29], we first use statistical modeling approach to calculate such similarity between the query and candidate passage, and then apply rule-based reasoning to select answers from top ranked passages, which statistical models cannot further differentiate their relatedness to the query.

In conclusion, a statistical model is more suitable for passage retrieval. Therefore, in the next section we shall discuss what constitutes a good statistical model for passage retrieval.

6.2 Statistical PM Modeling for Passage Retrieval

In Section 2.1, we have already described the framework for passage retrieval from the technical view. In this section, we will examine the framework from theoretical view. In the previous section we conclude that a passage retrieval system is a statistical model. We can show that such a model can be further decomposed into two sub-models, a parsing model P and a mapping model M, and M can also be further decomposed into word mapping model M_{word} and relation mapping model M_{Rel}. P is used for parse the query Q and candidate passages Pas into some language representations (usually a parse tree) and let's call them T_Q and T_{Pas}. The mapping model M is used to map the T_Q to T_{Pas}. The quality of such mapping is defined as similarity. We can use PM to denote the all the current frameworks for passage retrieval, with query expansion is a special case under PM. Query expansion just performs a partial mapping from T_Q to the vertex or nodes in T_{Pas}, which is only a sub-model of the full mapping model. The quality of this sub-mapping is defined as term ranking function.

In Table 6.2.1 we list down all the existing frameworks for passage retrieval and show their model decomposition under the PM framework.

Table. 6.2.1 Passage retrieval models under PM representation

Framework	Parsing Model (P)	Mapping Model(M)	
		Word Mapping	Relation Mapping
Word Matching [21]	A bag of independent words	Lexical Match	None
Density-Based [20]	A bag of independent words	Lexical Match	Dependence is defined as the lexical distance between words
Language Model (ngram) [28]	A bag of words, which the current word depends on the previous n words	Lexical Match	Dependence is defined as the co-occurrence probability between words
Dependence Language Model [12]	Sentences, with dependency between words within the sentence.	Lexical Match	Dependence is defined as the co-occurrence probability between words
Dependency Relation Matching [7]	Sentences, with dependency between words within the sentence.	Lexical Match	Dependence is defined as the dependency relation paths between words
Semantic Matching	Semantic frames with a predicate and arguments	Semantic similarity based on WordNet	Dependence is defined as the semantic relationship between the predicate and arguments
Relationship Graph	Sentences are represented as graphs with semantic edges over SVO triples and syntactic dependency edges elsewhere	Lexical Match, Syntactic Query Expansion and Semantic Verb Expansion	Dependence is defined as general syntactic dependency with specific semantic dependency over SVO semantic frames

Under the PM framework, a good parsing model should be able to identify the relationship between words correctly. We can also see from Table 6.2.1 that a mapping model is usually derived naturally from a parsing model. There are two major reasons why dependency matching can perform well in passage retrieval:

First, before our proposed dependency matching framework, all the previous model only use density-based or co-occurrence-based method to approximate the relationship between words. The main research contribution by our work is that we have shown that such relationship between words should be modeled using relation path, which is a vector in higher dimensional space rather than just approximating the relation using a number of co-occurrences. Obviously, the mapping model derived from dependency parsing should perform path mapping instead of comparing the number of co-occurrence counts.

Second, we should still remember that our model is a statistical model, for which an important criterion is its convergence. Although in the parsing model we can represent the relationship between words in high dimensional vectors we have to train a mapping model between vectors. We have about 10,000 training instances where there are only 27 common types of relations, which results in 729 mapping pairs. As on average there a more than 10 instances to cover a mapping, our mapping model is able to converge. Therefore our passage retrieval model can perform well in passage retrieval.

So far, we have proposed two important criteria to evaluate a statistical-based passage retrieval model: (1) Parsing model should correctly model the relation path between words. (2) Mapping model should converge based on the parsing model after training. However, it must not be over-trained as well. And if we apply these two criteria on our semantic model, we shall be able to tackle the problem.

However, the parsing model cannot handle complex semantic structures, which means it cannot handle correctly the relationship between frames. Meanwhile, it only classifies the relationship between words using less than five types. Therefore it is easy to see that such parsing model will result in over-training the mapping model. In our framework we have already tried to avoid such problem by simply using Jaccard coefficient to measure the similarity between two sets of arguments without matching their relationships. However, such a mapping is not naturally derived from the parsing model and thus does not make use of all the linguistic features obtained by the parser. As a result, the framework may not perform well.

However, our semantic matching model does improve the word mapping model as it uses semantic similarity between verbs while all the previous models only perform lexical level of word matching. The major advantage of the semantic model is its ability to perform precise matching on semantic frames and its flexibility to match paraphrasing structures only based on verbs.

In conclusion, the dependency relation-based model has high flexibility in matching relations and therefore results in high recall but low precision. It tens to make errors when performing specific verb matching since it does not consider the word but only the dependency relations. On the other hand, semantic relation-based model has high precision but low recall because we enforce exact match of the SVO (subject, verb, object) frames. Therefore a combination of the two models, between which the synergies are maximized, is an obvious approach. Therefore in chapter 5, we proposed the model of relationship graph (RG). The RG model has a flexible matching over syntactic relations while it has a precise matching over semantic frames and their verb variations. Since it satisfies both of the criteria we proposed it has achieved the best performance among all the models we proposed so far.

6.3 Challenges for Passage Retrieval in QA

We have successfully demonstrated the effectiveness of relationship analysis in passage retrieval for open domain question answering. However, there is still much room for improvement.

First, the main problem is the paraphrasing problem between verb phrase and noun phrase. For example: “Ship A to B” is the same as “shipment from A to B”. The semantic parsing can only recognize the first frame but not the second. This will cause vertex misalignment problem between queries and answer candidates.

Second, both dependency parsing and semantic parsing cannot handle long term relations. For example, in the sentence “Organization A distributed leaflets claiming responsibility of murdering X”. Both dependency parser and semantic parser misrecognize the dependency of “A murdered Y.” as “Leaflets murdered Y”.

Third, resolving the inconsistency between external data and corpus data is very crucial. As the data are from two different sources, the expression they use to describe the same concept might be different, which poses a great challenge in resolving the data inconsistency.

Finally, cross sentence relations still remain unsolved as no existing parsers can handle multiple sentences well.

Bibliography

[1] Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. Och, D. Purdy, N. Smith, and D. Yarowsky, *Statistical machine translation,* Final Report, JHU Summer Workshop, 1999.

[2] R. Attar, A.S. Fraenkel, (1977). *Local Feedback in Full-Text Retrieval Systems,* Journal of the Association for Computing Machinery, 24(3), 397-417.

[3] G. Attardi, A. Cisternino, F. Formica, M. Simi and A. Tommasi, *PiQASso: Pisa Question Answering System*, Proc. of TREC-2001, 2001, pp. 599-607.

[4] A. Berger and J. Lafferty, *Information retrieval as statistical translation*, Proc. of SIGIR '99, 1999, pp. 222-229.

[5] P. Brown, S. Della, V. Della Pietra and R. Mercer, *The mathematics of statistical machine translation: Parameter estimation*, Computational Linguistics, 19(2), 1993, pp. 263-311.

[6] H. Cui, K. Li, R. Sun, T.-S. Chua and M.-Y. Kan, *National University of Singapore at the TREC-13 Question Answering Main Task*, Proc. of TREC-13, 2004.

[7] H. Cui, R. Sun, K. Li, M.-Y. Kan and T.-S. Chua. *Questiong Answering Passage Retrieval Using Dependency Relations,* Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, Salvador, Brazil , Aug 15-19, pp. 400 - 407.

[8] W.B. Croft, R. Cook, D. Wilder, *Providing Government Information on The Internet: Experiences with THOMAS,* In Digital Libraries Conference DL'95, pp. 19-24.

[9] W. B. Croft, D. J. Harper, (1979). *Using probabilistic models of document retrieval without relevance information,* Journal of Documentation, 35, 285-295.

[10] B. V. Durme, Y. Huang, A. Kupsc and E. Nyberg. *Towords Light Semantic Processing for Question Answering,* HLT-NAACL 03 Work-shop: Text meaning, pp. 54-61, 2003.

[11] A. Echihabi and D. Marcu, *A noisy-channel approach for question answering,* Proc. of ACL ’03, 2003.

[12] J. Gao, J.-Y. Nie, G. Wu and G. Cao, *Dependency language model for information retrieval,* Proc. of SIGIR ’04, Sheffield, UK, 2004, pp. 170-177.

[13] S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams and J. Bensley, *Answer Mining by Combining Extraction Techniques with Abductive Reasoning*, Proc. of TREC-12, 2003, pp. 375-382.

[14] D. Hull, *Using statistical testing in the evaluation of retrieval experiments*, Proc. of SIGIR ’93, 1993.

[15] A. Ittycheriah, M. Franz, and S. Roukos, *IBM’s statistical question answering system - TREC-10*, Proc. of TREC-10, 2001.

[16] K. Sparck Jones, (1971). *Automatic Keyword Classification for Information Retrieval.* Butterworth, London.

[17] M. Kaszkeil and J. Zobel, *Passage retrieval revisited*, Proc. of SIGIR ’97, Philadelphia, PA, USA, 1997, pp. 178-185.

[18] B. Katz and J. Lin, *Selectively Using Relations to Improve Precision in Question Answering*, Proc. of the EACL-2003 Workshop on Natural Language Processing for Question Answering, April 2003.

[19] P. Kingsbury, M. Palmer and M. Marcus. *Adding Semantic Annotation to the Penn TreeBank.* Proc. of the Human Language Technology Conference, San Diego, California, 2002.

[20] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim, *SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP*, Proc. of TREC-10, 2001, pp. 442-451.

[21] M. Light, G. S. Mann, E. Riloff, and E. Breck, *Analyses for elucidating current question answering technology*, Journal of Natural Language Engineering, Special Issue on Question Answering, Fall–Winter, 2001.

[22] D. Lin, *Dependency-based Evaluation of MINIPAR,* Proc. of Workshop on the Evaluation of Parsing Systems, Granada, Spain, May, 1998.

[23] J. Lin, D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz and D. R. Karger, *What makes a good answer? The role of context in question answering*, Proc. of the ninth IFIP TC13 International Conference on Human-Computer Interaction, 2003.

[24] D. Lin and P. Pantel, *Discovery of Inference Rules for Question Answering,* Natural Language Engineering, 2001, 7(4): pp. 343-360.

[25] D. Moldovan and A. Novischi. *Lexical Chains for Question Answering.* Proc. Of COLING '02, pp. 674-680, 2002.

[26] S. Narayanan and S. Harabagiu. *Question Answering Based on Semantic Structures,* Proc. of COLING '04, Geneva, Switzerland, 2004.

[27] S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin and D. Jurafsky. *Shallow Semantic Parsing using Support Vector Machines,* Proc. of HLT/NAACL '04, Boston, MA, 2004.

[28] F. Song and B. Croft, *A general language model for information retrieval*, Proc. of CIKM'99, 1999, pp. 316-321.

[29] S.Tellex, B.Katz, J.Lin, A.Fernandes and G.Marton, *Quantitative evaluation of passage retrieval algorithms for question answering*, Proc. of SIGIR '03, 2003, Toronto, Canada, pp. 41-47.

[30] S.C. Townsend, Y. Zhou W. Bruce Croft *A framework for selective query expansion,* Proceedings of the thirteenth ACM conference on Information and knowledge management,Washington, D.C., USA pp.236 - 237

[31] E.M. Voorhees, *Overview of the TREC 2003 Question Answering Track,* Proc. of TREC-12, pp. 54-68.

[32] E. Voorhees, *Query expansion using lexical-semantic relations*, In Proceedings of ACM 57-GIR International Conference on Research and Development in Information Retrieval, pp. 61-69.

[33] E.M. Voorhees, *Overview of the TREC 2002 Question Answering Track,* Proc. of TREC-11, pp. 60-71.

[34] E.M. Voorhees, *Overview of the TREC 2005 Question Answering Track,* Proc. of TREC-14

[35] J. Xu, W. Bruce Croft, *Query expansion using local and global document analysis,* Proceedings of the 19th annual international ACM SIGIR 1996 conference on Research and development in information retrieval , Zurich, Switzerland, pp. 4 - 11

Appendices

Appendix I

Relation types in Dependency Trees from MiniPar

Relation Name	Description or Example
appo	"ACME president, --appo-> P.W. Buckman"
aux	"should <-aux-- resign"
be	"is <-be-- sleeping"
c	"that <-c-- John loves Mary"
comp1	first complement
det	"the <-det `-- hat"
gen	"Jane's <-gen-- uncle"
have	"have <-have-- disappeared"
i	the relationship between a C clause and its I clause
inv-aux	inverted auxiliary: "Will <-inv-aux-- you stop it?
inv-be	inverted be: "Is <-inv-be-- she sleeping"
inv-have	inverted have: "Have <-inv-have-- you slept"
mod	the relationship between a word and its adjunct modifier
pnmod	post nominal modifier
p-spec	specifier of prepositional phrases
pcomp-c	clausal complement of prepositions
pcomp-n	nominal complement of prepositions
post	post determiner
pre	pre determiner
pred	predicate of a clause
rel	relative clause
vrel	passive verb modifier of nouns
wha, whn, whp	wh-elements at C-spec positions
obj	object of verbs
obj2	second object of ditransitive verbs
subj	subject of verbs
s	surface subject