Study on Acquiring and Using Linguistic Semantic Information for Search Systems

by

NGUYEN TRI THANH

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor AKIRA SHIMAZU

School of Information Science
Japan Advanced Institute of Science and Technology

March 2008

For my wife and family

Abstract

Semantic search (a content-based search method) is an expected method of information retrieval in the future. Semantic search aims at finding information that satisfies a given question more than the results of keyword-based search (which is the method of current search engines). Despite the improvements of the current semantic search systems, they still return results in pages. In contrast to such page-based approaches, there are researches for directly answering questions in form of concrete answers instead of pages. In this research, we focus our attention on using semantic relations between terms or between text spans in order to appropriately answer some types of questions. Especially, in this study, we take up named-entity-related relations, and Rhetorical Structure Theory (RST) [Mann & Thompson (1988)] relations as semantic relations, since these relations can be clues for answering some typical question types such as *who, list, why, suggestion* and *how to*, because we can see many such relations in sentences, and we relate such relations to the question types.

Questions can be roughly classified into named-entity-related (NE-related) and non-named-entity-related (non-NE-related) types. For NE-related questions, NE-related relations can be clues for extracting answers, and RST relations can be clues for answering non-NE-related questions. The goal of this dissertation is to exploit such relations in texts in order to extract answers to some typical question types. Concretely, the following problems are targeted in this research:

- Since Named Entities (NEs) are important in many Natural Language Processing (NLP) applications including semantic search, where queries related to named entities occupy a relatively large portion of frequently asked queries, our first issue is to extract some named-entity-related relations, then to utilize these relations to answer some named-entity-related question types. The experiments of our relation extraction algorithms on "Wall Street Journal" corpus give very good performance, so the experiment with the utilization of the extracted information for answering questions also provides promising results.

- There are a lot of question types besides named-entity-related questions. The second issue is to exploit RST relations among text spans for extracting answers to some other question types. We use the assumption that, in the relation between two text spans, one text span can be the answer to a question that is related to the other. Our experiments with "RST Treebank" corpus give promising results that are better than baseline programs.

- In order to make a search system understand questions, given a question, the system needs to identify question type to select appropriate relations as clues for extracting answers. Thus, our next problem is question classification (QC). In question classification, increasing the classification performance based on machine learning is a promising approach. Since labeled questions are expensive, while unlabeled questions are abundant and cheap to collect, we originally propose to use labeled and unlabeled questions in semi-supervised approach to improve the performance of

question classification. We also propose to use a hierarchy of classifiers in order to reduce the number of question classes per classifier, since a big number of question classes adversely affects the performance of classifiers. Different learning algorithms for classifiers in the classifier hierarchy are also investigated. Our experiments with the above proposals prove the significant improvements.

- Finally, based on the above issues, we build a prototype of a search system. Our system can accept a question in the form of an English sentence from a user, and classify the given question in order to carry out a suitable answer extraction method. If found, the answer in form of a sentence or paragraph is returned to the user. The results of our experiments show that this is a good direction.

In summary, the thesis focuses on exploiting the semantic relations in documents in order to extract answers to questions. The solutions for the investigated, analyzed problems are provided. The contributions of this thesis include both theoretical and empirical issues. The theoretical issue is the development of algorithms for both extraction of named-entity-related relations and for question classification. The empirical issue is the application of the above algorithms, and the proposed method for extracting answers to questions. Our experiments give promising results.

Key words: Computational Linguistic, Natural Language Processing, Information Extraction, Information Search, Rhetorical Structure Theory, Fine Category Extraction, Question Classification, Semi-supervised Learning.

Acknowledgments

First of all, I wish to express my deepest respect and appreciation to my supervisor: Professor Akira Shimazu, Natural Language Processing Laboratory, School of Information Science, Japan Advanced Institute of Science and Technology (JAIST) for his kindly guidance, warm encouragement, and supports before and during my study. He has given me much invaluable knowledge not only how to formulate a research idea or to write a good paper, but also the vision, and much useful experience in the academic life. I am grateful for his patient supervision, and I am really lucky and proud to be one of his students. Besides academic life, Professor Shimazu also cared about my daily life whenever I have unexpected problems. I am really impressed by his kind consideration.

I wish to say sincere thanks to Professor Tatsunori Mori, Yokohama National University, Professor Satoshi Tojo, Professor Ho Tu Bao, and Associate Professor Kiyoaki Shirai, JAIST for being the members of my dissertation committee, and giving me valuable comments. My dissertation is improved very much by these comments.

I wish to say deep thanks to Associate Professor Kiyoaki Shirai, School of Information Science, JAIST for his a lot of helps during my study period, such as setting server environments for my experiments.

I would also like to say my special thanks to Professor Ho Tu Bao for his valuable discussions and supports. I would also like to express my gratitude to him for all his helps not only for my study but also for my life from my first day till now in JAIST. Professor Ho Tu Bao is also the one who set up the Machine Learning Seminar Group in JAIST, from which I have learnt a lot of new knowledge.

I wish to convey sincere thanks to Professor Yoshiteru Nakamori, and Assistant Professor Huynh Van Nam, Knowledge System Science Laboratory, School of Knowledge Science, JAIST for their help in my sub-theme research. They have given me as good as possible conditions for my work during this time.

I would also like to express my appreciation to Professor Ha Quang Thuy and Professor Ho Sy Dam, College of Technology, Vietnam National University, Hanoi (VNUH), for their kindly recommendations and constant encouragement before and during my research at JAIST. Without their helps, I could not receive the permission to go to JAIST. Professor Ha Quang Thuy was my former adviser who guided me since I was an undergraduate. He was also the leader of a research group in College of Technology, Vietnam National University, where I learned a lot of knowledge through several seminars.

I would like to thank the Lecture Mary Ann Mooradian, and the Technical Communication Reviewer Mark G. Elwell for their help in proof-reading and correcting errors in my papers. I have learnt a lot from her corrections.

I have received a lot of help from colleagues and friends in Shimazu-Lab and Shirai-Lab during last three years. Let me say special thanks to Assistant Professor Makoto Nakamura, and Mr. Kenji Takano for all their helps for my life from the first day I came to Japan. I would like to thank my colleagues in Shimazu-Lab: Mr. Nguyen Phuong

Thai, Mr. Nguyen Van Vinh, Dr. Le Anh Cuong, the post-doctoral fellow Nguyen Le Minh, and others for sharing their research ideas, useful experiences, as well as valuable discussions and comments. Especially, the post-doctoral fellow Nguyen Le Minh has given me a lot of valuable discussions and constant supports in my study since the early days when I came to JAIST.

I also wish to send my deep acknowledgements to "The 21^{st} Century COE Program: Verifiable and Evolvable e-Society" for supporting fund for me during the past three years; and to JAIST staffs for their kind and convenient procedures and services.

I'd like to appreciate the authors of open source tools/packages such as Maximum Entropy Models (MEM), Support Vector Machines (SVM), Sparse Network of Winnows (SNoW), Charniak parser and Tri-training. Without these packages, my experiments were hard to be completed.

Last, but not least, my family is really the biggest motivation behind me. My wife, Tao Thi Thu Phuong, who always gives me encouragements, cares about my daily life. My dear parents, my dear brother and sisters' families, my father-in-law's family together with their unconditional sacrifices, love, and supports are always endless sources of inspiration for me to move forwards, so this thesis is dedicated to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we briefly introduce the research domain, the research motivations, and the contributions of the thesis. We first introduce the need of semantic search. Secondly, we state the research problems which the thesis targets to solve. Finally, we outline the structure of the thesis.

1.1 The need of semantic search

With the rapid growth of the Internet, the volume of information is extremely large. Almost documents are free-text such as text files or HyperText Markup Language (HTML) files, or semi-structured such as eXtensible Markup Language[1] (XML). These formats are suitable for human reading and not suitable for machine understanding. It is difficult for human to find what they need. In order to overcome this obstacle, Information Retrieval (IR), a branch of Natural Language Processing (NLP) was born. The task of IR is to find information within a large unstructured documents (e.g., web pages on the Internet) that satisfies the users' need (which is expressed via a query). Examples of applications resulted from IR are search engines, such as google[2],Yahoo[3], Altavista[4] and MSN[5], which have successfully served an uncountable number of users' queries. The common characteristic of these search engines is keyword-based. Given a query, a search engine finds all documents that contain any words appearing in the query. After having found the list of documents, these documents are ranked so that the documents more related to the query than the others are shown to users first [Brin & Page(1998b)]. Though the results of search engines are ranked, the number of documents containing words of the query is still very large, and users have to read through (up to thousands of) pages to find what they need.

In order to overcome this problem, semantic search (a content-based search method) is proposed [Heflin & Hendler (2000b), Popov, *et al.* (2003), Bonino, *et al.* (2004)] [Chu-Carroll, *et al.* (2006)]. The motivation of semantic search is to find documents that

[1]http://www.w3.org/XML/
[2]http://www.google.com
[3]http://www.yahoo.com
[4]http://www.altavistar.com
[5]http://www.msn.com

related to the query in term of content not simply words, so it helps reduce the number of returned pages while still retaining the quality of the results.

1.2 Research Motivations

Sematic search has been the interest of several studies. One emerged approach, which is applied for World Wide Web (WWW), is Semantic Web. The mainstream of Semantic Web is to add additional information into web pages so that computers can understand these web pages. World Wide Web Consortium[6] (W3C) is an organization that pioneers in Semantic Web and proposes languages as well as tools for applying to Semantic Web.

An example of a search engine that follows Semantic Web approach is Simple HTML Ontology Extension (SHOE) [Heflin & Hendler (2000b)]. In SHOE, terms and concepts in a web page must be annotated according to SHOE specification, and the process of annotating web resource into SHOE specification is done manually. In other words, there must be a group of people to insert semantic information to web pages. This process is tedious and time-consuming.

Another study that follows Semantic Web is Knowledge and Information Management (KIM) platform [Popov, *et al.* (2003)]. This study exploits the advantages of named entities (NE) in information retrieval. There is an ontology which consists of 250 entity classes accompanied with a set of 100 attributes and relations among these classes. All named entities in documents are identified and classified into appropriate classes. Based on the set of named entities and their properties as well as relations, this platform can help retrieve pages that related to queries.

The second approach for semantic search is Ontology driven semantic search which exploits the Ontology in the searching process [Bonino, *et al.* (2004)]. In this search engine, authors used the concept vector model, which is based on the classical vector space model for matching queries with documents.

The third approach that uses statistical method is Latent Semantic Indexing (LSI) [Deerwester, *et al.* (1990)]. The key idea of this approach is to map documents and queries into a lower dimensional space which is composed of a higher level of concepts. The number of concepts is smaller than that of the indexed terms. This method has advantages over keyword-based searches, since in LSI the synonyms were mapped to the same concept, so the results can contain pages that have no keywords occurring in the query.

One more search engine, which aims at data in eXtensible Markup Language (XML) format - an intended version of WWW - is XML search (Xsearch) [Cohen, *et al.* (2003)]. This search engine explores the semantics of tags of XML, so it gives better results. Nonetheless, currently, there is a huge source of information written in HTML and other formats (e.g., pdf, ps), in this domain, XML search engine does not expose its actual power. In addition, user queries must be given in the form of tag:keyword. This may be inconvenient because users may not know exactly the correct tags to use.

In order to exploit the semantics of XML tags, [Chu-Carroll, *et al.* (2006)] proposed to convert documents into XML formats. After that the semantics of the XML tags were exploited in the process of finding answers.

Despite of the improvements in the above approaches, these semantic search systems

[6]http://www.w3c.org

still return results in the form of pages. There are several types of questions for which users prefer to have direct answers (in the form of a sentence, or a paragraph) rather than pages containing the answers. An example of such questions is "Why didn't Mr. Bush have to wait for a law?" From this observation, this study focuses on building a search system which uses methods to extract answers to some question types based on linguistic semantic information as follows:

• Firstly, named entities are now more and more widely used in many Natural Language Processing (NLP) applications [Popov, *et al.* (2003), Li & Roth (2005)] [Al-Onaizan & Knight (2001), Kumaran & Allan (2004), Kadri & Wayne (2003)] [Hassel (2003)], including semantic search, where the number of queries concerning about named entities comprises a significant proportion. Current Named Entity Recognition (NER) systems usually rely on a predefined set of classes. The named entity set presented by "The sixth Message Understanding Conference" (MUC6) for application in business activities consists of 7 classes [Grishman & Sundheim (1996)], while "Conference on Computational Natural Language Learning" (CoNLL) shared task defined only 4 classes of named entities [Sang & Meulder (2003)]. Nonetheless, finer distinctions of named entity are needed in some applications, thus Sekine proposed to extend the named entity hierarchy to about 150 classes (and currently about 200 classes) [Sekine, *et al.* (2000a)], and [Popov, *et al.* (2003)] presented a hierarchy of 250 named entity classes to support semantic search.

Though the named entity class sets of [Sekine, *et al.* (2000a)] and [Popov, *et al.* (2003)] contain relatively large numbers of types, current named entity recognition systems usually assign a unique type to a named entity [Chieu & Tou (2003)]. This approach does not reflect the real world, where a named entity can have more than one type. For example, a person named entity can be both "*executive vice president*" and "*chief financial officer*". In addition, in real applications, such as question-answering (QA) and semantic search systems, users may query the list of even finer categories of named entities, such as "*US presidents*".

In order to support searches of fine-grained categories of named-entities, the first task of the thesis is to develop a new algorithm to extract fine-grained categories of named-entities in the form of "*named-entity* ISA *category*".

With the extracted categories, a named entity is described by its category itself, e.g., "*named-entity* ISA *category*". For improving the performance of the algorithm, a validation method is proposed to check whether or not a fine category of a named entity is valid to be accepted.

A document may contain more information describing a named entity. Obviously, the more information of a named-entity is extracted, the more complete the named entity is described. With this observation, the next task of the thesis is to extend the above algorithm to extract more complete information of a named entity. The extracted information of named entities is used for answering named-entity-related questions, such as *who* and *list* questions.

• Secondly, there are other questions other than named-entity-related ones. In order to extract answers to these questions, we exploit RST relations which hold between adjacent text spans. From our analysis, for some RST relations, one text span can be clues for finding the answer to a question related to the other text span. We

exploit this characteristic of RST relations for extracting answers to some types of questions, such as *how to* and *why* questions.

• Thirdly, corresponding to a question, there are certain semantic relations that are clues for extracting the answer. Thus, given a question, its type must be identified before the answer extraction step is carried out. The next task of this thesis is to deal with question classification. For solving this task, statistical approaches are dominant in comparison with rule-based approaches, in which an expert manually constructs a number of regular expressions and keywords corresponding to each question type. In statistical approach, a dataset of labeled questions are used for training the classification algorithm. In general, a large dataset is required to get high classification precision. Current question classification studies concentrate on supervised learning which relies only on the labeled dataset. Labeled questions, however, are expensive and time-consuming to collect as they need the efforts of experienced annotators. Whereas, unlabeled questions are easily to collect (e.g., from the log file of search engines, or QA systems). Semi-supervised learning is a good choice, in this situation, since it can exploit the unlabeled questions in combination with labeled ones to build better precision models. Our proposal is to use semi-supervised learning in the question classification task to improve the accuracy. Another problem, when the number of question classes is large, the performance of classification algorithm may be affected. In order to reduce the number of question classes per classifier, we propose to use hierarchical classifiers corresponding to question class hierarchy. Different learning methods are investigated in this hierarchy. Also a method to automatically expand the nodes which consists of a large number of question class in the hierarchy is presented.

• Finally, based on the above investigated issues, we build a prototype of a search system. Our system can accept users' questions in the form of complete natural language sentences. Next, the system identifies the question type in order to carry out a suitable answer extraction method. Finally, the concise answer (in the form of a sentence or a paragraph) is returned to the user.

All in all, this thesis focuses on exploiting the semantic relations for building a semantic system with both theoretical and empirical aspects. The theoretical aspect of this thesis concerns about the design of new algorithms for pattern extraction, and a more suitable semi-supervised learning algorithm for question classification based on Tri-training algorithm. The other aspect is the application of these algorithms to solve real problems in some modules of our search system.

1.3 Main Contributions

As stated earlier, the thesis aims at building a search system based on semantic relations. The main contributions of this thesis are summarized as follows:

• Questions concerning about named entities take a relatively large percentage. Though some studies the set of named entity classes up to 250, current named entity recognition system usually assign a unique type to a named entity. This does

not reflect the real world where a named entity can belong to more than one class. In this thesis, we developed a new algorithm to extract named-entity-related relations (e.g., ISA relations) from documents. This work was published in "IEICE Transactions on Information and Systems, Special section on Knowledge, Information and Creativity Support System" [Nguyen & Shimazu (2007b)]. We extended our algorithm to extract more information about a named entity besides ISA relations. And we used the extracted named-entity-related relations for answering some named-entity-related question types (e.g., *list* and *who* questions). This work was presented at "The 21st Pacific Asia Conference on Language, Information and Computation (PACLIC21)", Korea [Nguyen & Shimazu (2007c)].

• The semantic relations between adjacent text spans in documents are exploited to answer some other question types. We exploited the RST relations in text documents for solving this problem. This work was presented at "The 9th International Conference on Text, Speech and Dialog", Czech [Nguyen, *et al.* (2006a)].

• Semi-supervised learning was proposed to apply to Question Classification with the purpose of exploiting the unlabeled questions to improve the performance of classification algorithm. We modified the Tri-training algorithm to make it more suitable for question data. This work was presented at "The 21st International Conference on the Computer Processing of Oriental Languages", Singapore [Nguyen, *et al.* (2006b)]. The extended version of this work will be published in the "Journal of Natural Language Processing" in January 2008 [Nguyen, *et al.* (2008)]. The hierarchical classifier in combination with different learning methods were used to increase the precision of question classification, and the method to automatically expand the nodes consisting a large number of question classes. This work was presented at "The 5th International Conference on Research, Innovation & Vision for the Future: RIVF'07", Vietnam. The proceedings of this conference are also available on "IEEE Explore digital library[7]" [Nguyen, *et al.* (2007a)].

• Finally, we build a prototype of a semantic search system which can extract answers to some typical question types based on previous studies. Our system can accept questions in the form of plain-text. It classifies the given question to identify the type of the question. From the discovered type, it can extract possible answers based on the suitable semantic relations.

1.4 Thesis Structure

The thesis consists of seven chapters. The main purposes and contents of the six remaining chapters are as follows:

Chapter 2 presents the previous studies about semantic search.

Chapter 3 proposes a new nearly-unsupervised algorithm for extracting fine-grained categories of named entities from text documents. The algorithm exploits the fact that the fine-grained category of a named-entity may occurs along with the named entity

[7]http://ieeexplore.ieee.org

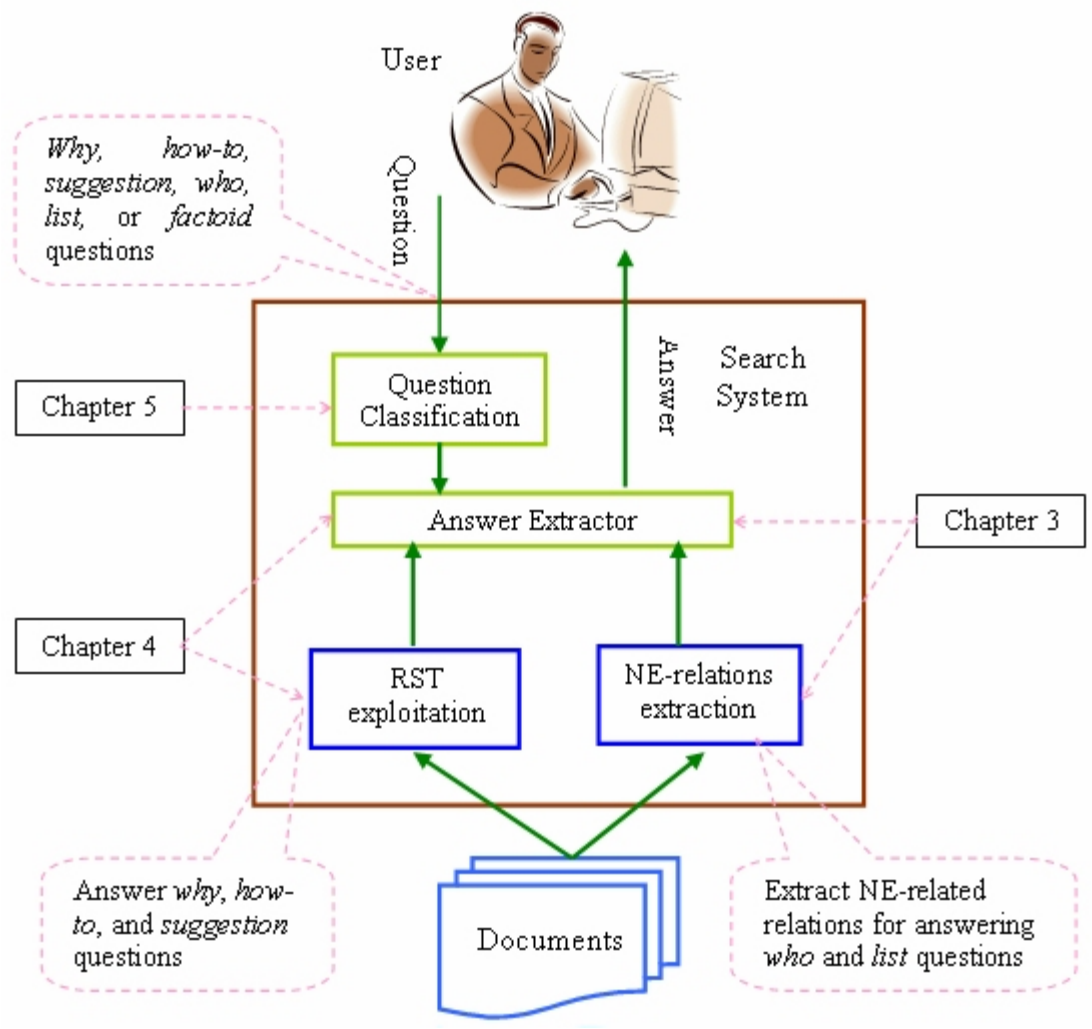

Figure 1.1: The framework of our search system.

itself in frequent templates or expressions. Based on the proposed algorithm, we extend it to extract more information that describes the named entities appeared in documents. The experiments of the two algorithms on a large dataset give very good results. The extracted information of named entities, in turn, is used for generating answer to some named-entity-related questions. The experiments and evaluation with good results are presented in this chapter.

Chapter 4 analyzes the RST relations to find what relations are clues for a certain question type. A new method for indexing documents as well as a method for matching questions with facts in order to improve the performance are proposed. The experiments and evaluation are given with promising results.

Chapter 5 proposes to apply semi-supervised learning for question classification problem. The characteristic of question data and the drawback of Tri-training when it is applied directly to classify questions are investigated and analyzed. The proposal of modifying the Tri-training algorithm to make it more suitable for question data is given. We also propose to use hierarchical classifiers for dataset which contains a

larger number of question classes in order to reduce the number of question classes per classifier. Different learning methods are investigated in this hierarchy. Also a method to automatically expand nodes which consists of a large number of question classes is proposed. The experiments provide significant improvement.

Chapter 6 introduces the prototype of a semantic system which is built based on the above studied issues. The results of experiments are also promising.

Chapter 7 first summarizes the main tasks of the thesis including the main achievements and contributions, as well as the remaining problems. Next, open problems that are interesting to be solved from this thesis will be mentioned as the future research directions.

The framework of our search system and the corresponding tasks of the most important chapters are shown in Figure 1.1.

Chapter 2

Current semantic search approaches

"Semantic search" is studies about methods that can search and discover the relevant parts of texts or information which a user wants to find out. The semantic search methods exploit the properties of words, their relations as well as linguistic information besides the words themselves in order to roughly understand the questions, sentences or documents at an abstract level.

Unlike the typical method of search engines which only search the occurrence of keyword(s) on a Web page, semantic search finds information that satisfies the given query more than pages containing the keywords of the query. In this chapter, we summarized the current status of studies about semantic search. Current semantic search studies can be classified into three main approaches: searching directly on text documents, data annotation, and XML exploitation, which are briefly introduced in the next sections.

2.1 Latent Semantic Indexing

Search engines based on keywords can not solve some of the language related problems such as *synonymy*. Synonymy is a case where an object can be referred in many ways. People use different words to search for the same object, such as 'car' and 'automobile'. Latent Semantic Indexing (LSI) offers a better method which solves the synonymy problem by allowing synonyms can be mapped into the same concept [Deerwester, *et al.* (1990)]. This is an approach that can be applied directly to plain-text. By using a Singular Value Decomposition (SVD) on a term by document matrix of term frequency. The dimension of the transformed space is reduced by selection of the highest singular values, where the most of the variance of the original space is. The major associative patterns are extracted from the document space and the small patterns are ignored. LSI can be briefly introduced as follows [Yates, *et al.* (1999)]:

The main idea in the latent semantic indexing model is to map each document into a lower dimensional space which is associated with concepts. This is accomplished by mapping the index term vectors into this lower dimensional space. Let t be the number of of index terms in the collection, and N be the total number of documents. Define $\vec{M} = (M^{ij})$ as a term-document association matrix with t row and N columns. To each element M^{ij} of this matrix is assigned a weight $w^{i,j}$ associated with the term-document

pair $[k^i, d^j]$. This $w^{i,j}$ weight could be generated using the Term Frequency/Inverse Document Frequency (tf-idf) weighting technique commonly in the classic Vector Space Model [Lewis (1991), Salton, *et al.* (1975)]. LSI proposes to decompose the $\vec{M}$ association matrix in three component using SVD as follows:

$$\vec{M} = \vec{K}\vec{S}\vec{D}^t \tag{2.1}$$

The matrix $\vec{K}$ is the matrix of eigenvectors derived from the term-to-term correlation matrix given by $\vec{M}\vec{M}^t$. The matrix $\vec{D}^t$ is the matrix of eigenvectors derived from the transpose of the document-to-document matrix give by $\vec{M}^t\vec{M}$. The matrix $\vec{S}$ is an $r \times r$ diagonal matrix of singular values where $r = min(t, N)$ is the rank of $\vec{M}$. Consider now that only the s largest singular values of $\vec{S}$ are kept along with their corresponding columns in $\vec{K}$ and $\vec{D}^t$ (i.e., the remaining singular values of $\vec{S}$ are selected). The resultant $\vec{M}_s$ matrix is the matrix of rank s which is closest to the original matrix $\vec{M}$ in the least square sense. This matrix is given by

$$\vec{M}_s = \vec{K}_s\vec{S}_s\vec{D}_s^t \tag{2.2}$$

where s $(s < r)$ is the dimensionality of a reduced concept space. The selection of a value for s attempts to balance two opposing effects. First, s should be large enough to allow fitting all the structure in the real data. Second, s should be small enough to allow filtering out all the non-relevant representational details. In information retrieval, the similarity between two documents can be calculated as follows:

$$\begin{aligned} \vec{M}_s^t\vec{M}_s &= (\vec{K}_s\vec{S}_s\vec{D}_s^t)^t\vec{K}_s\vec{S}_s\vec{D}_s^t \\ &= \vec{D}_s\vec{S}_s\vec{K}_s^t\vec{K}_s\vec{S}_s\vec{D}_s^t \\ &= \vec{D}_s\vec{S}_s\vec{S}_s\vec{D}_s^t \\ &= (\vec{D}_s\vec{S}_s)(\vec{D}_s\vec{S}_s)^t \end{aligned} \tag{2.3}$$

LSI can be applied for different domains including text classification and Information Retrieval [Berry, *et al.* (1995), Letscher & Berry (1997)]. When applying LSI for information retrieval, the query terms are also transform into this subspace to find the closest documents as results. The returned documents, which contain expected information for users, may contain no terms that appear in the given query. This is an advantage over keyword-based search approach. Another advantage of LSI is that it is fully automatic to compute, and does not use language expertise.

2.2 Data Annotation

Data annotation concerning about approaches in which data is represented in certain ways so that it is understandable by computers. In this section, we briefly introduce the Semantic Web, a typical semantic search system following Semantic Web, and another proposal of data annotation to support semantic search as well as a semantic search system

pertaining to this proposal.

2.2.1 Semantic Web

Semantic Web is an emerging trend as an expected next version of HTML. As the name stated, Semantic Web is applied to web pages as an extension of WWW. At present, HTML is the formal language of WWW, which defines how the data should be presented in a browser. However, it does not include the ability to describe the semantic of data. Semantic Web overcomes this drawback by enabling the ability to describe the data. Concretely, in semantic web, *resources* (e.g., web pages, images, audio clips, entities or objects) in web pages are defined and linked in such a way that the web content can be understood by computers. Thus, web content can be used for more effective discovery, automation, integration or reused in various applications. The W3C has introduced the philosophy, a set of design principles, and several languages as well as technologies for enabling semantic web. One of the most important languages that is used for defining resources and the relations among them is Resource Description Framework[1] (RDF). The Semantic Web information can be viewed as directed graphs. A segment of Semantic Web pertaining to a person named "John Smith" is given in Figure 2.1, and the corresponding RDF representation is as follows[2]:

```
<?namespace href="http://docs.r.us.com/bibliography-info" as="bib"?>

<?namespace href="http://www.w3.org/schemas/rdf-schema" as="RDF"?>

<RDF:serialization>

  <RDF:assertions href="http://www.bar.com/some.doc">

    <bib:author href="#John_Smith"/>

  </RDF:assertions>

</RDF:serialization>

<RDF:resource id="John_Smith">

  <bib:name>John Smith</bib:name>

  <bib:email>john@smith.com</bib:email>

  <bib:phone>+1 (555) 123-4567</bib:phone>

</RDF:resource>
```

In companion with RDF, W3C also introduces an additional language - RDF Schema[3] (RDFS) - for defining vocabulary, constraints of resources presented in RDF files, and validating RDF files.

Like Relational Database which is provided a declarative query language (e.g., Structured Query Language (SQL)), there are some proposals of specification of query lan-

[1] http://www.w3c.org/RDF

[2] The example is from http://www.w3.org/TR/WD-rdf-syntax-971002/

[3] http://www.w3.org/TR/rdf-schema/

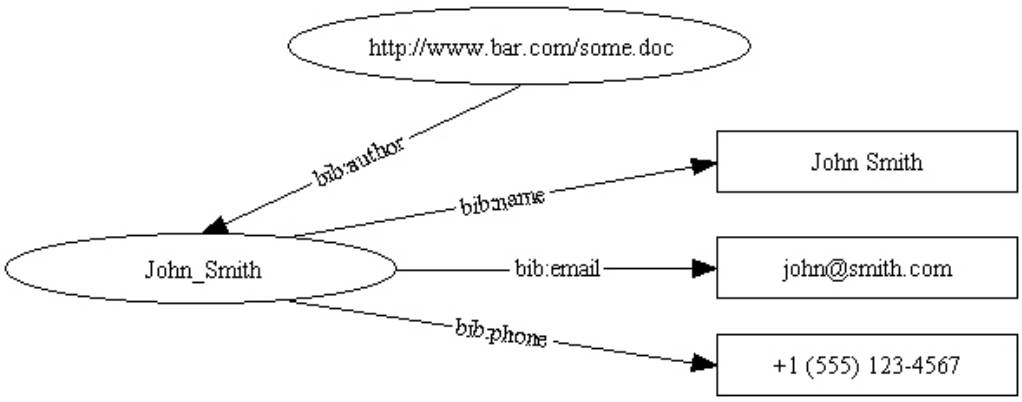

Figure 2.1: A segment of Semantic Web describing John Smith

guages for manipulating RDF files, such as "Simple Protocol and RDF Query Language"[4] (SPARSQL) of W3C, and "RDF Query Language" (RQL) of [Karvounarakis, *et al.* (2002)]. Currently, the implementation of these languages is still an open task, since these languages are rather complicated. Fortunately, there are some studies about organizing RDF as initial steps to realize Semantic Web, such as [Matono, *et al.* (2004), Kim, *et al.* (2006), Matono, *et al.* (2005)].

The task of adding annotations to web pages is tedious and time-consuming, and currently the dominant documents in the WWW are in the form of HTML and text. In order to reduce efforts for annotation, some efforts have been concentrated on this problem, such as [Kogut & Holmes (2001)]. [Kogut & Holmes (2001)] applied Information Extraction (IE) to build a tool called AeroDAML which automatically generate annotations for a number of common domain-independent classes and properties. Another graphical tool which allows authors to create annotation via drag and drop actions is OntoMat[5].

Since the annotation of different sources may use different vocabulary, the integration or interoperability may face problem. Defense Advanced Research Projects Agency (DARPA) and W3C solved this problem by focusing on machine-readable ontologies. Ontologies are explicit semantic models, which include taxonomies of terms and semantic relations among them. Based on the ontologies computers can reason with knowledge. The integration of ontologies is a problem which attracts the interest of researchers, such as [Pinto, *et al.* (2001)]. A proposal of language for representing ontologies is "DARPA Agent Markup Language"[6] (DAML). The practical applications of DAML are discussed in [Kogut & Heflin (2003)]. Similarly to RDF language, a query language named "DAML query language"[7] (DQL) for DAML was proposed.

[Kogut & Heflin (2003)] discussed the potential applications of Semantic Web for Aerospace. A Distributed Open Semantic Elaboration (DOSE) Platform, which includes ontology, annotations, lexical entities and search functions, was proposed [Bonino, *et al.* (2003)]. The platform is still an ongoing task, and needs more efforts to be applied in real world. A typical semantic search study that follows Semantic Web is introduced by

[4]http://www.w3.org/TR/rdf-sparql-query/
[5]http://annotation.semanticweb.org/ontomat/index.html
[6]http://www.daml.org/
[7]http://www.daml.org/dql/

[Guha, *et al.* (2003)] which will be described in the next section.

2.2.2 Semantic Search Systems for Semantic Web

[Guha, *et al.* (2003)] presented two semantic search engines called ABS and "W3C Semantic Search" for different domains. These semantic search systems realize the Semantic Web approach, i.e., the systems use W3C's RDF and the schema vocabulary provided by RDFS for describing resources and their inter-relations.

Users' queries can be classified into two types: *navigational* and *research* searches. Navigational searches concern about the queries which contain a phrase or combination of keywords which users expect to find in the documents. The words in the queries do not denote a concept (e.g., "mp3 open source conversion"), and users use a search engine as a navigation tool to find a particular intended document. Whereas, research searches concern about queries whose words denote an object (e.g., a person "John Smith") about which the user is trying to collect information. Maybe, there is no particular document provides full information about the object. Rather the user is trying to collect a number of documents which together will provide the expected information. Semantic Search systems of [Guha, *et al.* (2003)] focuse on the research searches.

Current documents in WWW do not contain much semantic information, it is required to generate the annotations for these documents. Though there exist some tools for doing this task, such as AeroDAML [Kogut & Holmes (2001)], maybe these tools do not generate the expected annotations, [Guha, *et al.* (2003)] built HTML scrapers to dynamically locate and convert the relevant documents into machine-readable semantic data. For storing and manipulating (e.g., querying) semantic data, they developed a knowledge base called TAP [Guha & McCool (2002)]. TAP provides a set of simple mechanisms for sites to publish data onto the Semantic Web with a minimalist query interface. The interface is provided via an Apache HTTP server[8] module called TAPache. The goal of this module is to make it extremely simple to publish data, not intended to be a high end solution for sites with large amounts of data and traffic, since such sites require flexibility and scalability rather than simplicity. To enable remote machines to query data on a server, the query interface can be accessed via Simple Object Access Protocol[9] (SOAP). SOAP provides a mechanism for performing Remote Procedure Call (RPC), and SOAP is beginning to be widely accepted as a standard protocol.

For ABS semantic search system, the semantic data about people (musicians, athletes, actors, politicians), organizations (companies, music groups, sport teams), places (cities, countries, states) and products is collected (by scrapers) from a large number of sources, such as AllMusic[10], Ebay[11], Amazon[12], AOL shoppong[13], TicketMaster[14], People

[8] http://www.apache.org
[9] http://www.w3.org/TR/soap/
[10] http://www.allmusic.com
[11] http://www.ebay.com
[12] http://www.amazon.com
[13] http://shopping.aol.com
[14] http://www.ticketmaster.com

Magazine, Wheather.com[15], Mapquest[16], Carpoint[17], Digital cities, and Walmart.com[18].

In contrast to ABS, the semantic data for W3C Semantic Search system is collected from a relatively small number of sources, which are all internal to W3C, such as *people* (staff and authors of various documents), *W3C activities* (each is related to people), *Working groups and other committees* (each is related to activities and staff), *documents* (each is related to working groups and activities that produced them) and *news* (RSS[19] news feeds about newsworthy events of W3C).

Both systems are incorporated a basic ontology about people, places, event, organizations, etc.

Recall that the semantic data describing resources can be viewed as directed graphs. Given a query, the systems carry out the following steps to find the answer:

- *Choosing a denotation*: Analyze the query to find whether it contains any concept (to conclude whether the query is a *research search*). If this is a research search, then map the search term to one or more nodes in the graphs (of semantic data). If several nodes corresponding to one are term found, the disambiguation step is carried out to select the most preferred one.

- *Determining what data to show*: From the found node (called *Anchor node*), the subgraph around the anchor node is extracted as the results.

- Formatting the results: This step is trivial in comparison with previous ones. A set of templates is used to display the found results.

2.2.3 Knowledge and Information Management Platform

Similar to the previous approach, [Popov, *et al.* (2003)] proposed a platform called "Knowledge and Information Management" (KIM) which applied Information Extraction to automatically annotate named entities in documents. The architecture of KIM is depicted in Figure 2.2 [Popov, *et al.* (2003)]. KIM Server API, which can be used for remote access, embedding and integration, provides functionality and infrastructure for semantic annotation, indexing, and retrieval, as well as document management, and KB navigation.

With the discussion about the limitation of the set of named-entity classes which does not satisfy the diversity of users' queries, [Popov, *et al.* (2003)] constructed an ontology which consists of about 250 named-entity classes and 100 attributes and relations. The extraction module was customized from GATE [Cunningham, *et al.* (2002)]. Semantic annotations, KIM ontology and Knowledge Base (KB) are stored in semantic repositories which are based on RDFS repositories SESAME[20] and ontology middleware[21].

After semantic annotations, documents can be indexed with respect to the contained named entities for later searching with respect to entities. In searching, users could specify the named entities that are of users' interest, and constraints of these named entities, KIM

[15]http://www.wheather.com
[16]http://www.mapquest.com
[17]http://www.carpoint.com.au
[18]http://www.walmart.com
[19]http://www.purl.org/rss/1.0
[20]http://www.openrdf.org/
[21]http://www.ontotext.com/omm

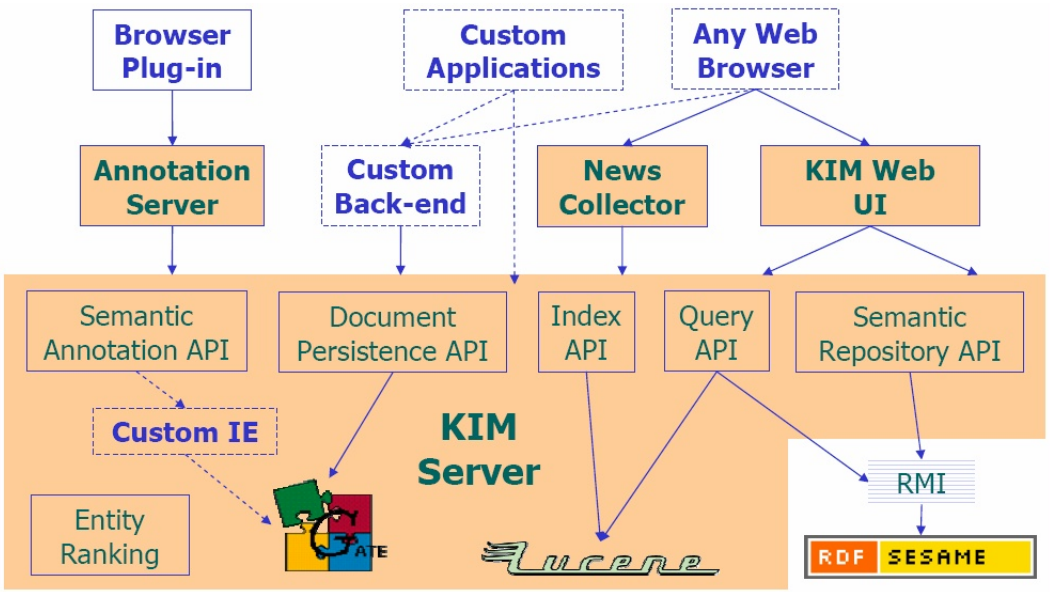

Figure 2.2: The architecture of KIM

returns documents containing named entities satisfying the query. To answer a query, KIM applies semantic restrictions over the named entities in the KB. Then the documents containing (or referring) the found named entities are retrieved and ranked according to named entities. Lucene[22] is adapted to perform full-text indexing and retrieval.

2.2.4 Simple HTML Ontology Extensions

Another approach that follows the data annotation is Simple HTML Ontology Extensions (SHOE) [Heflin & Hendler (2000b)]. For supporting semantic search, terms, concepts and objects in HTML documents must be annotated accordingly to the SHOE specification [Luke & Heflin (1997)]. When annotating a web page, the user must select an appropriate ontology, and then use the ontology's vocabulary to describe the concepts on the page. Below is a snippet of an annotation using SHOE specification:

```
<HTML> ... <BODY>
<ONTOLOGY ID="cs-dept-ontology" VERSION="1.1" BACKWARD-COMPATIBLE-WITH="1.0">
<USE-ONTOLOGY ID="univ-ontology" VERSION="1.0" PREFIX="u"
                                        URL="http://ontlib.org/univ_v1.0.html">
...
<DEF-CATEGORY NAME="ComputerScience" ISA="u.ResearchArea">
...
<DEF-RELATION NAME="writtenIn">
```

[22] http://jakarta.apache.org/lucene/

```
   <DEF-ARG POS=1 TYPE="Program">
   <DEF-ARG POS=2 TYPE="ComputerLanguage">
</DEF-RELATION>
...
<DEF-RENAME FROM="u.Department" TO="Department">
<DEF-RENAME FROM="u.Chair" TO="DepartmentHead">
...
</ONTOLOGY>
</BODY>
</HTML>
```

For assisting users to annotate the data correctly, the *Knowledge Annotator* was built. This is a tool that makes it easy to add annotation by making selection and filling in forms. The tool also has the ability to check possible errors in the annotating process to ensure the correctness, and converts inputs into legal SHOE syntaxes.

When SHOE pages are created, and placed on the Web, the knowledge from the pages is extracted and stored in a Knowledge Base. This process is done by a web-crawler called *Exposé*. Currently, SHOE stores knowledge in a Parka DB [Stoffel, *et al.* (1997)]. Since the all the SHOE-related tools were designed with a generic Application Programming Interface (API), it is possible to use another Knowledge Representation (KR) (or Knowledge Base).

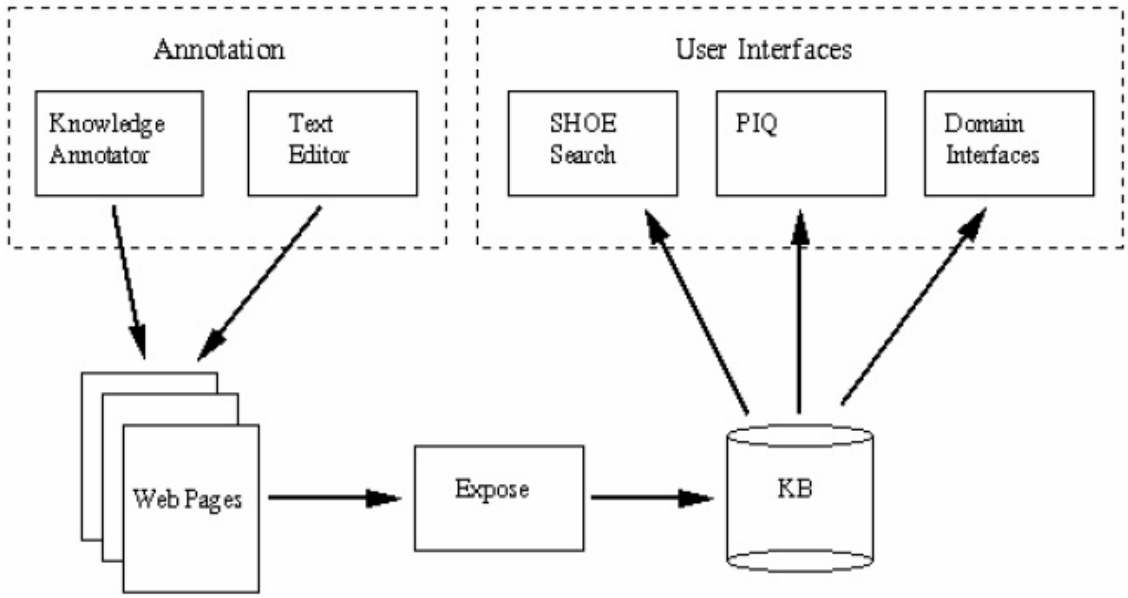

Figure 2.3: The architecture of SHOE

For searching, SHOE provides a graphical tool called "SHOE Search" which allows a user to select a context by choosing an appropriate ontology which the user is interested

in. After choosing an ontology, the system creates a list of categories existing in that ontology. This list is organized so that the user can quickly locate the classes she/he needs. The property list of a class is displayed, and the user can fill the constraints to search. In case the system does not find any relevant document, it helps the user to transform the given query into a query string that is commonly used by current search engines for searching in these search engines.

Since SHOE operates based on ontologies, some studies related to ontologies have been carried out, such as coping with changing ontologies [Heflin, *et al.* (1999b)] [Heflin & Hendler (2000a)].

2.3 XML Exploitation

The next approach in semantic search is to exploit the semantics of XML tags. XML is widely used as a mean for data interchange. In contrast to HTML which is a language for defining how to display data on a browser, XML is a language for describing data.

2.3.1 XSearch

Current search engines do not exploit the advantage of XML, since they do not have the ability to expose the query that refer to meta-data (i.e., the XML tags).

[Cohen, *et al.* (2003)] proposed a semantic search system to work on XML data called *XSearch*. The architecture of XSearch as described in [Cohen, *et al.* (2003)] is shown in Figure 2.4. Similarly to the current search engines, the XML documents are indexed according to a method proposed by [Cohen, *et al.* (2003)] before serving queries. Given a query, the "*Search query processor*" finds the possible answers from the repository. Then the answers are ranked before being presented to users.

There exists a query language for processing XML data: XQuery[23] which is a declarative query language like SQL, and some XQuery-like languages [Chamberlin, *et al.* (2002) , Chinenyanga & Kushmerick (2002), Fuhr & Großjohann (2001)]. Since the syntax of these query languages is rather complex, not suitable for the average users, [Cohen, *et al.* (2003)] proposed a simple query language which can be used by naive users. In XSearch, a query consists of a sequence of required and optional search terms $t_1, t_2, \ldots, t_m$. Each term t_i has the form $l : k$, $l :$ or $: k$, where l is a label (an XML tag) and k is a keyword. If a search term is preceded by a plus sign ('+'), it is required, otherwise it is optional. An example of a query is Q(title:, +author:Kempster) which searches for the list of books whose author is Kempster. The query is matched again the XML trees, where each tree is corresponding to an XML document, and the answers are nodes that satisfies the query.

2.3.2 Semantic Search Via XML Fragments

In another semantic search approach is of [Chu-Carroll, *et al.* (2006)] that annotates documents in form of XML documents, and applied XML searches to exploit the semantics

[23] http://www.w3.org/TR/xquery/

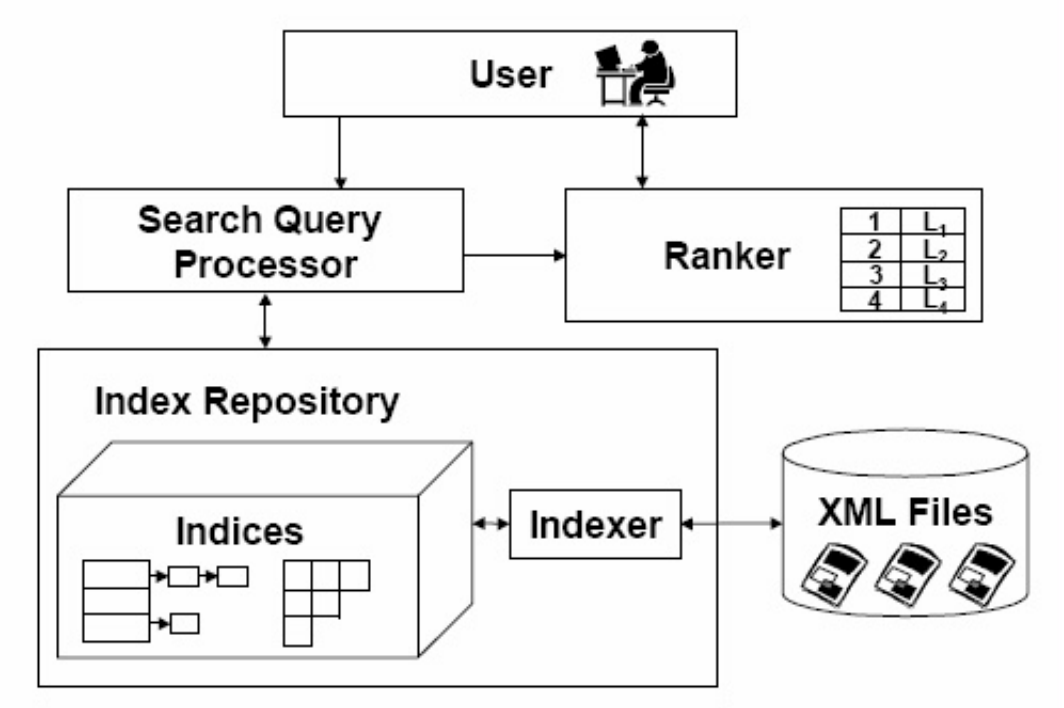

Figure 2.4: The architecture of XSearch

of XML tags. For example, the sentence "*Clinton was graduated from Georgetown University and in 1968 won a Rhodes Scholarship to Oxford University.*" will be annotated as:

```
<AlmaMater> <Person> Clinton
</Person> was graduated from <College> Georgetown University
</College> </AlmaMater> and in <Date> 1968 </Date> won a Rhodes
Scholarship to <College> Oxford University </College>.
```

For querying the data, [Chu-Carroll, *et al.* (2006)] used XML fragment query language [Carmel, *et al.* (2003), Broder, *et al.* (2004)]. An XML fragment has the form of *[[<tag> ... </tag>]]*, where the double square brackets are used to denote the boundaries, and one XML fragment can contain other fragments. An example of an XML fragment is *[[<Book> "Gone with the wind"</Book>]]*.

In order to enrich the query expressiveness, three operations were identified in the process of finding answers [Chu-Carroll, *et al.* (2006)]:

- **Conceptualization** operation: generalizes a lexical string to an appropriate concept in the type system represented by that string. For example, the query *animal* returns documents containing the word 'animal', while the conceptual query *[[<Animal></Animal>]]* returns documents containing the tag *Animal*, which applies to all subtypes of the concept, such as *lion*, *owl*, and *salmon*.

- **Restriction** operation: constraints the XML tags, in which keywords must appear to be considered relevant. For example, *[[<Animal> bass </Animal>]]* returns

documents in which the literal "bass" is used in its fish sense, while *[[<Instrument> bass </Instrument>]]* retrieves those where it represents a musical instrument.

- **Relation** operation: the annotation represents a relation that holds between terms. These relations include syntactic e.g., *[[<SubjectVerb> Unabomber kill </SubjectVerb>]]*, semantic, e.g., *[[<Kill> Unabomber <Person></Person> </Kill>]]*, or pragmatic, e.g., *[[<HasNegativeOpinion> Clinton war on Iraq </HasNegativeOpinion>]]*. In addition, the XML Fragment query syntax allows the nesting of relation and entity annotations, e.g., *[[<Visit><Person> John </Person><Person> Victoria </Person></Visit>]]*. This generally matches documents where John and Victoria visited one another but excludes those where John visited the city of Victoria.

These operations can be used with XML Fragment query syntax with its classical operators which are described in [Broder, *et al.* (2004)]. The three operations have the ability to express four different query-time semantic needs, i.e., to specify target information type, to disambiguate keywords, to specify search term context, and to specify relations between selected terms.

2.4 Discussion

Except the XSearch system which returns results in form of XML nodes, all other methods give results in form of a list of pages. In our approach, we try in extract the answers to questions from documents by exploiting the semantic relations in documents. The comparison of the above mentioned approaches and ours are shown in Table 2.1, where *Query* indicates the characteristics of input queries; *QC* stands for "Question classification".

2.5 Summary

In this chapter, we have classified and summarized the current approaches of semantic search, and some typical semantic systems were described in more detail.

Table 2.1: The characteristics of approaches

Approach	Human effort	Preprocessing	Exploitation	Query form	QC	Output
LSI	No	Compute concept space	Concept	Keywords or phrases	No	Pages
Guha	Both manually and automatically	Annotate objects with their properties and relations	Semantic of annotation	Keywords or phrases	No	Pages with additionally enriched information
KIM	No	Annotate Named entities and their properties and relations according to an ontology	Named entities with their properties and relations	Keywords or phrases	No	Pages
SHOE	Yes	Annotate objects and their properties, relations according to ontologies	Semantic of annotation	Concept and constraints on their properties and relations	No	Pages
Xsearch	No	No	The semantics of XML tags	XML fragment with constraints	No	XML nodes
Chu-Carroll	No	Annotate named entities (with a predefined set of classes), objects, terms and convert documents into XML format	The semantics of XML tags, named entities	Keywords or phrases	No	Pages
Our	No	Extract named-entity-related relations (the set of NE classes is not fixed, but depending on the data), construct RST tree of documents	The semantic relations of RST and named entities	Complete questions	Yes	Direct answers

Chapter 3

Extraction and Utilization of Named-Entity-Related Relations for Information Search

Named entities play important roles in many NLP applications including Information Retrieval. Discovering the relations related to named entities may contribute benefit to these applications. In this chapter, firstly, we present a data-driven approach to extract *named-entity* ISA *category* relations from documents. Secondly, we extend the algorithm to extract the information that describes named-entities more completely. Finally, we utilize the extracted information for answering some types of named-entity-related questions.

3.1 Introduction

Named entities play important roles in many Natural Language Processing (NLP) applications, including Machine Translation [Al-Onaizan & Knight (2001)], Text Summarization [Nobata, *et al.* (2002), Hassel (2003)], Text/question Classification [Li & Roth (2005)] [Kumaran & Allan (2004)], Question Answering [Srihari & Li (2000)], and Information Retrieval [Popov, *et al.* (2003)]. The queries/questions related to named entities take a significant portion as discovered in [Dumais, *et al.* (2003)]: "*The most common query types in our logs were People/Places/Things, Computers/Internet, and Health/Science. In the People/Places/Things category, names were especially prevalent. Their importance is highlighted by the fact that 25% of the queries involved peoples names, which suggests that people are a powerful memory cue for personal content. In contrast, general informational queries are less prevalent.*"

The named entity (NE) set presented by the sixth Message Understanding Conference (MUC6) for application in business activities consists of 7 types: *organization*, *location*, *person*, *date*, *time*, *money* and *percent* [Grishman & Sundheim (1996)]. Nonetheless, finer distinctions (finer-grained classes) of named entities are needed in some applications, thus [Sekine, *et al.* (2000a)] proposed to extend the named entity hierarchy to about 150 types

(and currently about 200 types), and [Popov, *et al.* (2003)] presented a hierarchy of 250 named entity types to support semantic search.

Though the named entity sets of Sekine and Popov contain relatively large numbers of types, current named entity recognition systems usually assign a unique type to a named entity [Chieu & Tou (2003)]. This approach does not reflect the real world, where a named entity can have more than one type. For example, a person named entity can be both "*executive vice president*" and "*chief financial officer*". In addition, in real applications, such as Question-answering (QA) or search systems, users may query the list of even finer categories of named entities, such as "*US presidents*". Fortunately, the actual fine category of a named entity may appear along with itself somewhere in the text, in certain patterns, as in the following example, where "*analyst*" is the actual type of the named entity "*Bette Raptapoulos*":

> *There's a generally more positive attitude toward the economy, said Bette Raptapoulos, analyst for Prudential-Bache Securities Inc. ...*[1]
>
> *He discussed a new strategy for the company ...*

Given which, a user may ask "Which *analyst* discussed a new strategy for the company?" From the first sentence, if we could recognize the actual type of the named entity "*Bette Raptapoulos*", then from the second sentence with co-reference resolution, we can easily answer the above question. This is a possible application of finely categorized named-entities.

In this chapter, we, firstly, proposes to extract the actual fine categories of named entities by exploiting valuable hidden patterns in text documents in a data-driven way based on Brin's model [Brin (1998a)]. We start with seed patterns instead of seed tuples, so that the number of tuples extracted as well as the number of patterns generated in each iteration is consequently large. We explore the generation of different pattern types for further extracting tuples, and propose a method for checking whether a newly extracted tuple (*named_entity*, *category*) is valid to improve the performance of the algorithm.

Secondly, if from the sentence "*There's a generally more positive attitude toward the economy, said Bette Raptapoulos, analyst for Prudential-Bache Securities Inc. ...*", we only extract the tuple ("Bette Raptapoulos", "analyst"), then "analyst" does not completely describe the person "Bette Raptapoulos". From this observation we extend our algorithm to extract (*named_entity, category, related-to, object*) quadruples, each of them describes that the *named_entity* ISA *category*, and the *category* IS-RELATED-TO *object*. We call such relations "*named_entity* ISA *category*", and "*category* related-to *object*" relations. An example of a quadruple, which is extracted from the above sentence, is ("Bette Raptapoulos", "analyst", "for", "Prudential-Bache Securities Inc.").

Thirdly, a quadruple (*named_entity, category, related-to, object*) is a useful, and can be exploited in NLP applications including semantic search or QA systems, for example, for answering *WHO* (e.g., "Who is Bette Raptapoulos?"), *WHICH* (e.g., "Which *analyst* discussed a new strategy for the company?"), and *LIST* (e.g., "Give me the list of analyst for Prudential-Bache Securities Inc.?") questions.

[1] An example from the Wall Street Journal corpus

3.2 Related Work

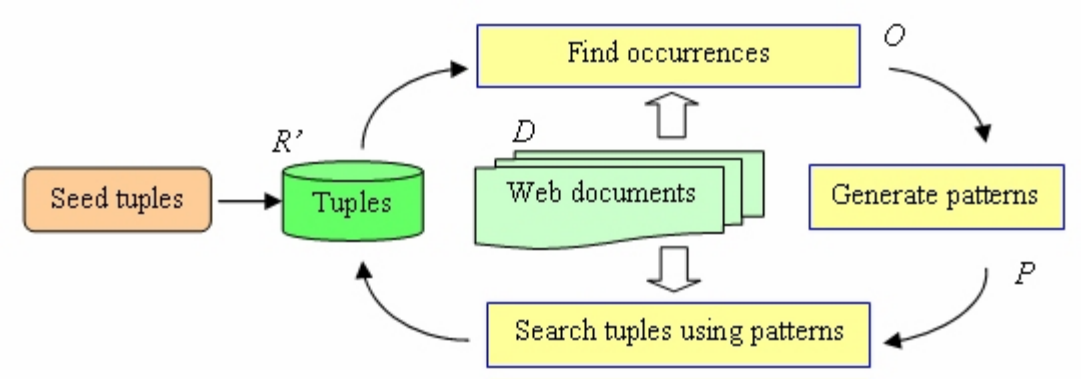

Figure 3.1: Brin's DIPRE model

Pattern extraction was proposed to solve many information extraction problems including relation extraction [Brin (1998a), Agichtein & Gravano (2000), Pasca (2004)]. [Brin (1998a)] proposed an iterative model to extract relations that occur in certain patterns in documents as described in details below.

Brin's model: the model is called "Dual Iterative Pattern Relation Extraction" (DIPRE) which is depicted in Figure 3.1, and detailed description of the algorithm is given in Figure 3.2, where $GenPatterns(O)$ is the procedure for generating new patterns which is given in Figure 3.3.

Brin used DIPRE to extract (*author*, *title*) tuples having the relation: the author of the book *title* is *author*. Starting with a small number of (*author*, *title*) seed tuples, DIPRE finds the occurrences of tuples in order to generate new patterns.

DIPRE's occurrences: An occurrence of a (*author*, *title*) tuple is represented by a 7-tuple:

(*author*, *title*, *order*, *url*, *prefix*, *middle*, *suffix*)

where *url* is the URL of the document in which the occurrence appeared; *order*, a boolean value, indicates the occurring order of the *author* and *title* in the text; if the *author* precedes the *title*, the *order* is *true*, otherwise it is *false*; *prefix* is m characters preceding the *author* (or *title* if the *title* is first); *middle* is the string between the *author* and *title*; and *suffix* is m characters following the *title* (or *author* if the *author* is last).

DIPRE's patterns: A new pattern is a 5-tuple:

(*order*, *urlprefix*, *prefix*, *middle*, *suffix*)

is generated from a group of occurrences O having the same *url* (or prefix of *url*), if all the occurrences in O have the same *middle*, *order*, *prefix* (or some last characters of *prefix*) and *suffix* (or some first characters of *suffix*). New patterns are, then, used to

Input: The large collection D of Web documents;
The set *Sample* of seed tuples.
Output: The set R' of tuples.

1. $R' \leftarrow Sample$
   Start with a small sample, R' of the target relation. This sample is given by the user and can be very small.

2. $O \leftarrow FindOccurrences(R', D)$
   Find all occurrences of tuples of R' in D. Along with the tuple found, keep the context of every occurrence (URL and surrounding text).

3. $P \leftarrow GenPatterns(O)$
   Generate patterns based on the set of occurrences. This is the tricky part of the algorithm. Roughly speaking, this routine must generate patterns for sets of occurrences with similar context. The patterns need to have a low error rate, so it is important that they are not overly general. The higher the coverage of the patterns the better. However, a low coverage can be compensated for with a larger collection D.

4. $R' \leftarrow M_D(P)$
   Search the collection D for tuples matching any of the patterns.

5. **if** R' is large enough **then return**;
   else go to Step 2.

Figure 3.2: The DIPRE algorithm

extract further (*author*, *title*) tuples.

DIPRE's tuple extraction: A new (*author*, *title*) tuple is extracted if there is a document having URL matching $urlprefix*$[2], and the document contains texts matching the regular expression:

$$*prefix, author, middle, title, suffix*$$

which is constructed from the pattern (*order*, *urlprefix*, *prefix*, *middle*, *suffix*), where the *order* (indicating the order of *author* and *title*) is true. Non-empty strings for $prefix$, *middle* and $suffix$ are used to determine the boundary of *author* and *title*.

DIPRE's pattern generation procedure: this is an important step of the DIPRE which is based on some heuristics, in order to generate patterns that extract few false positives (tuples that are non-books). This constraint makes each pattern have a small coverage, however, the total coverage of the patterns can still be substantial since the Web is vast and there are many sources of information. The pattern generation procedure is described in Figure 3.3, where *GenOnePattern* is the procedure for generating one pattern as described in Figure 3.4. In procedure *GenOnePattern*, for guaranteeing the above constraint, DIPRE considers the **specificity** of a pattern. The specificity of a

[2] '*' is a wild card of regular expression, which matches any sequence of characters in this context.

pattern p is measured as:

$$specificity(p) = |middle||p.urlprefix||p.prefix||p.suffix|$$

A pattern is generated iff its *specificity* satisfies the condition $specificity(p)n > t$, where n is the number of books with occurrences supporting the pattern p, and t is a threshold. This ensures that all the strings of a pattern are nonempty (otherwise the *specificity* is zero). Also DIPRE requires $n > 1$, since generating a pattern based on one example is very error-prone.

Input: The list O of occurrences;
Output: The patterns;

1. Group all occurrences o in O by *order* and then *middle*;
   Let the resulting groups be $O_1, \ldots, O_k$;
2. **for** each group O_i

   $p \leftarrow GenOnePattern(O_i)$;
   if (p meets the specificity requirements) **then** output p;
   else
   if (all o in O_i have the same URL) **then** reject O_i;
   else
   Separate the occurrences o in O_i into subgroups grouped by the character in their URLs which is one past $p.urlprefix$;
   Repeat the procedure in step 2 for these subgroups;

end for

Figure 3.3: The GenPatterns procedure of DIPRE

Based on DIPRE, by defining a new type of patterns as well as the constraints for generating patterns, [Agichtein & Gravano (2000)] developed the Snowball system for extracting (*organization*, *location*) tuples expressing the relation: the headquarters of *organization* is in *location*.

Inspired by the hypernym extraction study of [Hearst (1992)], Pasca presented a model based on DIPRE for acquiring (C, N) tuples (where C and N stand for category and named entity, respectively) from web documents by matching sentences with the pattern [Pasca (2004)]:

`[StartOfSent]` C `[such as|including]` N `[and|,|.]`

where C matches a plural noun; N matches consecutive proper nouns. Below are some sentences that match the above template, where C and N are underlined [Pasca (2004)]:

- *That is because software firewalls, including Zone Alarm, offer some semblance of*

Input: The list O of occurrences;
Output: A pattern *outpattern* if possible;

1. Verify that the *order* and *middle* of all the occurrences is the same;
   If not, it is not possible to generate a pattern to match them all;
   Set *outpattern.order* and *outpattern.middle* to *order* and *middle* respectively.

2. Find the longest matching prefix of all the URLs;
   Set $outpattern.urlprefix$ to that $prefix$.

3. Set $outpattern.prefix$ to the longest matching $suffix$ of the $prefix$'s of the occurrences.

4. Set $outpattern.suffix$ to the longest matching $prefix$ of the $suffix$'s of the occurrences.

Figure 3.4: The GenOnePattern procedure of DIPRE

this feature.

- *API Adapter can be written in other programming languages such as C++.*

Based on extracted tuples, new patterns are generated to extract potential tuples. Pasca's model is interesting, because it is nearly an unsupervised approach. However, Pasca's model considers only categories that are expressed using plural nouns. In regular text documents, potential categories can be available using singular nouns. In addition, sentences that match this template do not appear frequently in all text corpora, e.g., the Wall Street Journal (WSJ) corpus which is used in our experiments. In this case, the approach may need a very large number of text documents in order to get a sufficient list of (C, N) tuples for further generation of new templates, as seen in the experiments of Pasca on a large dataset consisting of 500 million of web documents and news articles.

Our study can be seen as a complement to Pasca's. We extract (*named_entity, category, related-to, object*) quadruples from text documents, in which *categories* are expressed by singular nouns.

[Shinzato & Torisawa (2004)] targeted at extracting hyponymy relations, which are presented in lists (or itemization) as exemplified below, from HTML documents.

```
<UL>
    <LI>Car Specification</LI>
    <UL>
        <LI>Toyota</LI>
        <LI>Honda</LI>
        <LI>Nissan</LI>
    </UL>
```

</UL>

[Fleischman, *et al.* (2003)] aimed at extracting concept-instance relations from documents by exploiting two patterns. The first pattern is common noun/proper noun constructions, such as "trainer Victor Valle", in which the concept is "trainer" and the instance is "Victor Valle". The second pattern is appositions, such as "David Werner, a real estate investor", in which the concept is "real estate investor" and the instance is "David Werner". After extracting all concept-instance pairs from documents, they used a method to filter the incorrect ones. However, the concept-instance relations can present in several patterns other then the above two patterns. Thus, for improving the recall, a large corpus (e.g., 15GB of newspaper text) was used in the experiments.

[Sumida, *et al.* (2006)] extracted concept-instance relations, that reside in two patterns consisting of noun sequences, from Japanese documents. The first pattern is sequences in which the boundary between the concept name and the instance name is explicitly marked by quotation marks, for example, "*Monk Story*" *movie*. The second type is sequences in which no evident clues indicate the boundary, such as "*Maruei hotel*". Similarly to [Fleischman, *et al.* (2003)], after extracting the noun sequences satisfying the two patterns, [Sumida, *et al.* (2006)] used some heuristics in combination with a search engine to filter incorrect ones.

3.3 Automatic Extraction of the Fine Category of Named Entities

In this section, we describe the drawbacks of DIPRE algorithm when applying to our problem, and the proposal to modify DIPRE to be more suitable for our problem.

3.3.1 Our Extraction Algorithm

DIPRE starts with a small set of (*author*, *title*) seed tuples. The selection of these tuples must be done carefully, because if we select tuples that do not appear in the target corpus, then no pattern can be generated for extracting new tuples. If the seed tuples do not frequently occur, there may be a small number of new patterns discovered for further extraction, and the algorithm is time-consuming to scan the corpus several times for extracting new (*author*, *title*) tuples. In our model, we start with seed patterns as described in Section 3.3.2. By starting with seed patterns, the number of tuples extracted in the first scan is relatively large, consequently, the number of new patterns discovered for the next scan is large, and the algorithm may need fewer scans on the corpus.

In DIPRE, a new (*author*, *title*) tuple is extracted if there is a document with URL matching $urlprefix*$, and the document contains texts matching the regular expression:

$$*prefix, author, middle, title, suffix*$$

which is constructed from the pattern (*order*, *urlprefix*, *prefix*, *middle*, *suffix*), where the *order* (indicating the order of *author* and *title*) is true. Non-empty strings for $prefix$,

middle and $suffix$ are used to determine the boundary of *author* and *title*. If we use DIPRE to extract (*named_entity*, *category*) tuples, this regular expression fails to extract (*named_entity*, *category*) tuples whose *named_entity* (or *category*) appears at the beginning (or the end) of a sentence. Moreover, all *prefix*, *middle* and *suffix* should not be a space, since a space does not specify a clear boundary of *named_entity* and *category*. Thus, the pattern fails to work in situations where *named_entity* and *category* are separated by a space, such as "*He demanded that Treasury Secretary Nicholas Brady appear before the Senate Banking Committee to explain...*"[3], where the *category* is "*Treasury Secretary*", and the *named_entity* is "*Nicholas Brady*".

To avoid the above problems, we use a named entity recognition (NER) system to determine the boundary of named entities. Because the *category* of a (*named_entity*, *category*) tuple is a noun phrase, we use a shallow parser to identify the boundary of *category*. Thus, our patterns do not need $prefix$ and $suffix$ components, and the method for generating new patterns is different from Brin's as discussed in Section 3.3.3. For improving the performance of the algorithm, we propose to use a function to validate a *category* in the extraction process as discussed in Section 3.3.4. We call our algorithm "*Named Entity, Category Extraction*" (NECE), and it is described in Figure 3.5

Input: A seed pattern set P_1; a text corpus D;
Output: The list L of (*named_entity*, *category*) tuples;

1. Initiation: the pattern set $P \leftarrow P_1$; $L \leftarrow \emptyset$ (empty list)
   Find named entities in every sentence in D;
   $D \leftarrow D$ - {*sentence* | *sentence* contains no named entity};
   Add part-of-speech and chunks tags for every sentence in D;

2. Extract the list L' of $(named\_entity, category)$ tuples from sentences that match any pattern in P; $L \leftarrow L + L'$;
   Let D' be the list of sentences, from which the $(named\_entity, category)$ tuples in the list L' were extracted; $D \leftarrow D$ - D';
   if (D is empty or L' is empty) **then return**;

3. Find the list O of the occurrences of (*named_entity*, *category*) tuples in D;

4. From the list of occurrences O, generate a new pattern set P'; $P \leftarrow P'$;
   if (P is empty) **then return**;
   else go to Step 2;

Figure 3.5: Named Entity, Category Extraction algorithm

In Step 2 of our algorithm, we remove sentences from which one or more tuples were extracted, because if we keep them for later scans, then duplicate tuples can be extracted. In Step 4, we replace P by P' (the set of newly generated patterns), since the old pattern set P in the previous scan cannot extract new tuples in the next scan. The graphical demonstration of our algorithm is depicted in Figure 3.6.

[3]An example taken from the WSJ corpus.

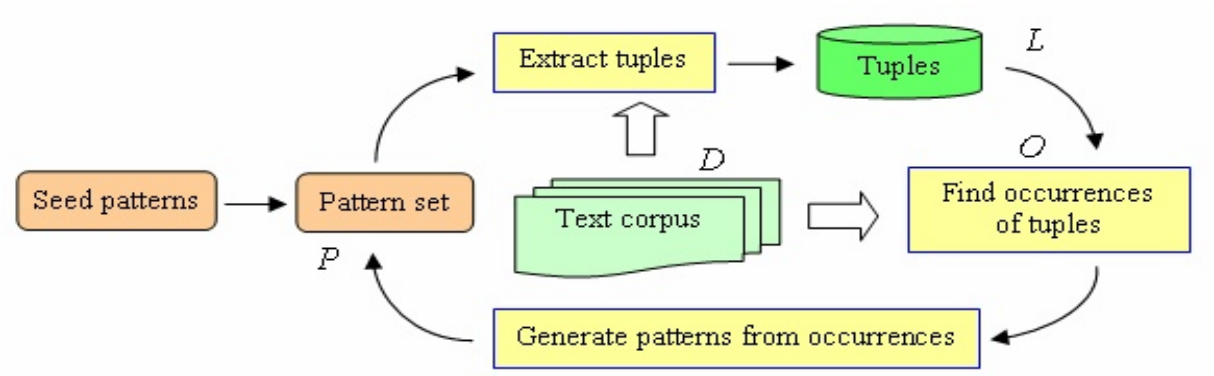

Figure 3.6: Our (*named_entity*, *category*) extraction model

3.3.2 Clue and Seed Patterns

Our seed patterns are based on *appositives*. Grammatically, an appositive is a noun phrase that renames or describes another noun phrase, with no word interposed between the two phrases. For example, in the sentence "*George Bush, the US president, announced* ···", the appositive "*the US president*" describes more concretely the person named entity "*George Bush*", and "*US president*" can be regarded as the actual type of "George Bush". In another complex example of appositive: "*Daniel Akerson, executive vice president and chief financial officer, said MCI's growth is being fueled by* ···"[4], two noun phrases: "*executive vice president*" and "*chief financial officer*" describe "*Daniel Akerson*", so two tuples are expected to be extracted. We only consider appositives whose head is a singular noun (tagged NN or NNP[5]), because a noun that describes a *named_entity* should be in singular form.

Because we work on sentences which have been parsed by a shallow parser, our patterns may contain part-of-speech (POS) and chunk tags. A chunk is a syntactically related non-overlapping group of words. A chunk is assigned a tag, such as NP (noun phrase), and surrounded by a square bracket pair, e.g., "[NP executive/JJ vice/NN president/NN]".

Pattern: We define a *pattern* as a 4-tuple:

(*order*, *named_entity_slot*, *middle*, *category_pattern*),

where *order* indicates the occurrence order of *named_entity* and *category* in a sentence. If *named_entity* is before *category*, the *order* is *named_entity* then *category* (hereafter we call this NEC for short), otherwise *order* is *category* then *named_entity* (hereafter we call this CNE for short). *named_entity_slot* is a slot which will be replaced with a *named_entity* (with POS tags) that appears in the sentence currently being processed. For example, if the current sentence has the named entity "*George Bush*", then *named_entity_slot* is: George/NNP Bush/NNP. Let *simple_noun* be a pattern matching a noun phrase with POS tags which consists of one or zero determiner, adjectives, gerunds

[4]An example taken from the WSJ corpus.
[5]NN and NNP are used to tag singular lowercase and singular proper nouns, respectively.

and nouns, e.g., "a/DT managing/VBG director/NN":

simple_noun:=($*word*/DT)?($*word*/(JJ|VBG))*
($*word*/NNP?S?)*($*word*/NNP?)+[6]

Then *category_pattern* is defined as:

category_pattern:=*simple_noun*$_1$ (and/CC *simple_noun*$_2$)?[7]

The sentence "*George Bush, the US president, announced* ···", after having been parsed, has the form: [NP George/NNP Bush/NNP] ,/, [NP the/DT US/NNP president/NN] ..., so the *middle* component for the first seed pattern is: "] ,/, [NP ", and the first seed pattern is:

(NEC, *named_entity_slot*, "] ,/, [NP " , *category_pattern*)

A named entity may also lie in an appositive, e.g., "*Semi-Tech's president and chief executive officer, James Ting, said it was likely that the Singer board would approve ...*". In this example, the appositive "*James Ting*" renames "*president and chief executive officer*". Thus, the second seed pattern is:

(CNE, *named_entity_slot*, "] ,/, [NP " , *category_pattern*).

If the *order* is NEC, then from a pattern:

(*order*, *named_entity_slot*, *middle*, *category_pattern*),

where *named_entity_slot* is replaced with a named entity (with POS tags) that appeared in a sentence s, the regular expression:

$*$*named_entity_slot*, *middle*, *category_pattern*$*$

is constructed to match s. If the *order* is CNE, then *named_entity_slot* and *category_pattern* are reversed.

Let *named_entity* be a *named_entity_slot* after removing POS tags. Let *category* be a *simple_noun* after removing the possible determiner (tagged DT) and POS tags. If a match is found, the expected tuples are (*named_entity*, *category*$_1$) and possibly (*named_entity*, *category*$_2$). This is an advance from DIPRE, because our method can extract two tuples from a match if there are two.

[6]? stands for "there is zero or one"; | stands for "or"; + stands for "there is one or more"; * stands for "there is zero or more".

[7]This pattern covers the cases where there are two noun phrases in the appositive.

1. Group all occurrences in the list O by *order* and *middle*;
   Let the resulting groups be $O_1, O_2, \ldots, O_N$;

2. For each group O_i, if the *middle* satisfies the two conditions, generate a new pattern:
   (*order*, *named_entity_slot*, *middle*, *category_pattern*);

Figure 3.7: Pattern generation procedure

3.3.3 Pattern Generation

Similar to Brin's model, our extraction model exploits the fact that (*named_entity*, *category*) tuples can be expressed in different lexical forms, which tend to appear in uniform patterns repeated in collections of documents. For example, the tuple ("*George Bush*", "*US president*") can be expressed in different ways as follows:

George Bush, the US president, announced ...
US President George Bush announced ...

Occurrence: Similarly to Brin's model, we define an occurrence of a (*named_entity*, *category*) tuple as a 4-tuple:

$$(order, named\_entity, middle, category)$$

where *order* has the same meaning as that of our patterns; *middle* is the string surrounded by *named_entity* and *category*. Based on the list of occurrences, we explore the method for generating different pattern types in the next subsections.

Exact Patterns

Occurrences of (*named_entity*, *category*) tuples are used to generate new patterns. However, a *middle* of an occurrence is not necessarily reliable, we need a method to retain reliable ones. Our first contraint is based on two criteria: *repetition* and *diversity* as follows:

Repetition of a *middle* (repetition(*middle*)) is the number of times the *middle* appears between the *named_entity* and *category* of (*named_entity*, *category*) tuples which have the same *named_entity*.

Diversity of a *middle* (diversity(*middle*)) is the number of times the *middle* appears between the *named_entity* and *category* of (*named_entity*, *category*) tuples which have different *named_entitys*.

A *middle* that has repetition(*middle*)$>$ $threshold_R$ seems reliable and is kept. A pattern seems specific if it is generated based on tuples of a *named_entity*, so we only keep *middles* that have diversity(*middle*) $>$ $threshold_D$ to make the generated patterns general (Condition 1).

If a *middle* contains a verb phrase, the verb phrase should express the relation *named_entity* ISA *category* (Condition 2).

Some valid verbs are: '*be*', '*assign*', '*elected*', '*take over*', '*name*', '*continue*', '*remain*'. We also care about the tense of these verbs. For some verbs, such as '*be*', '*assign*', and '*elect*', we do not accept their future or future perfect tenses. Because, for example, "a *named_entity* will be a *category*" does not certainly mean the *named_entity* is a *category*. The *middle* that contains a verb phrase is retained if it satisfies this constraint.

These two conditions are used in the pattern generation procedure described in Figure 3.7. We call these patterns *exact patterns*.

Examples of *middle* and sentences that match the corresponding exact patterns are given in Table 3.1, where Order is the *order* of the exact patterns; *named_entities* are underlined and *categories* are in italics.

Table 3.1: Examples of the *middle* of exact patterns

Middle and matched sentences	Order
Middle:] ,/, [NP Apple/NNP] [NP 's/POS Sentence: ... [VP says/VBZ] [NP Randall/NNP Battat/NNP] ,/, [NP Apple/NNP] [NP 's/POS *product-marketing*/JJ *vice*/NN *president*/NN] ./.	NEC
Middle: A space Sentence: [PP that/IN] [NP *Treasury*/NNP *Secretary*/NNP Nicholas/NNP Brady/NNP] [VP appear/VBP] [PP before/IN] [NP the/DT Senate/NNP Banking/NNP Committee/NNP] [VP to/TO explain/VB] ...	CNE

Exact patterns are relatively reliable; however, they have narrow coverage, so we propose other ways for extending coverage, as given in the next subsections.

Sketch Patterns

For the *middle* of an exact pattern having the *order* NEC:

"] ,/, [NP ABC/NNP] [NP 's/POS" (1)

its corresponding exact pattern can match the sentence:

[NP Harvey/NNP Dzodin/NNP] ,/, [NP ABC/NNP] [NP 's/POS vice/NN president/NN] ... (2)

However, this pattern cannot match a similar sentence that describes the "*director*" of another company (organization), e.g., IBM, in the same syntax as (2):

[NP Alan/NNP Baratz/NNP] ,/, [NP IBM/NNP] [NP 's/POS director/NN]... (3)

If we modify the *middle* (1) so that its pattern can match (3), then expected tuples in both (2) and (3) can be extracted. In order to do this, we convert (1) into a template

Table 3.2: Examples of the *middle* of sketch patterns

Middle and matched sentences	Order
Middle:] ,/, [NP who/WP] [VP is/VBZ] [NP $*word*/NNP] [NP 's/POS Sentence: [NP Mr./NNP Petit/NNP] ,/, [NP who/WP] [VP is/VBZ] [NP Healthdyne/NNP] [NP 's/POS *chairman*/NN ...	NEC
Middle:] [PP of/IN] [NP $*word*/DT $*word*/NNP $*word*/NNP $*word*/NNP] ,/, [NP $*word*/NNP Sentence: [NP A/DT *former*/JJ *governor*/NN] [PP of/IN] [NP the/DT Spanish/NNP Central/NNP Bank/NNP] ,/, [NP Mr./NNP Rendueles/NNP] ...	CNE

that can match other sequences having similar structure, except for nouns, adjectives, cardinals or articles. Concretely, we replace nouns, adjectives, cardinals and articles in a *middle* with a variable $*word* that matches a word. Below is the template constructed from the *middle* (1):

"] ,/, [NP $*word*/NNP] [NP 's/POS " (4)

We keep other words in a *middle* intact, such as verbs, prepositions or conjunctions, because their modification can make the context of the *middle* different. We call this template the *sketch* of a *middle*. We produce a new pattern type that we call *sketch patterns*, of which the *middle* component is replaced with a *sketch*.

Examples of *middles* of sketch patterns and matched sentences are given in Table 3.2.

Extended Sketch Patterns

Let's consider the sketch (*middle* component) of a sketch pattern with *order* NEC:

"] ,/, [NP $*word*/NNP $*word*/NNP] [NP 's/POS " (5)

The pattern can match the sentence:

[NP Bill/NNP Gates/NNP] ,/, [NP Microsoft/NNP Corporation/NNP] [NP 's/POS chairman/NN] ... (6)

However, this pattern can not match the sentence:

[NP Alfonso/NNP J./NNP Fanjul/NNP Jr./NNP] ,/, [NP Southeast/NNP Banking/NNP Corp./NNP] [NP 's/POS director/NN] ... (7)

If the sketch (5) could be extended so that its corresponding pattern matches (7), then

expected tuples in (6) and (7) would be extracted. We do this by generalizing the noun phrase of sketch (5) to enable it to match noun phrases which have one or more proper nouns. Concretely, we replace *consecutive* (proper) nouns (or adjectives) template in a sketch with another template that can match one or more consecutive (proper) nouns (or adjectives). For example, (5) is generalized as:

"] ,/, [NP($*word*/NNP)+] [NP 's/POS " (8)

We call a generalized sketch an *extended sketch*, and introduce a new pattern type called *extended sketch pattern*, of which the *middle* component is replaced with an *extended sketch*.

Table 3.3 gives an example of the *middle* of an extended sketch pattern and a sentence that matches the pattern.

Table 3.3: The *middle* of an extended sketch pattern

Middle and matched sentence	Order
Middle:] ,/, [NP $*word*/DT($*word*/NNP)+] ,/, [NP $*word*/NNP] ,/, [NP	
Sentence: [NP James/NNP Bopp/NNP Jr./NNP] ,/, [NP the/DT Terre/NNP Haute/NNP] ,/, [NP Ind./NNP] ,/, [NP *lawyer*/NN] [NP who/WP] [VP filed/VBD] [NP the/DT high-court/NN ...	NEC

Pattern Generation Order

Obviously, the tuples extracted by exact patterns are a subset of those extracted by sketch patterns; the tuples extracted by sketch patterns are a subset of those extracted by extended sketch patterns. Thus, we give exact patterns highest priority and extended sketch pattern the lowest priority, and run lower priority (or larger coverage) patterns only on the remaining dataset, which is obtained after processing by higher priority patterns.

3.3.4 Category Validation

Though our algorithm runs on documents, in which named entities have been tagged by an NER system, the NER system may incorrectly assign a type to a named entity (e.g., assign type *person* to a named entity which is actually of type *organization*). Consequently, the system may extract an incorrect tuple. Also, not all new patterns are 100% reliable, so some extracted $(named\_entity, category)$ tuples are incorrect, and should be discarded.

We propose an additional method for validating whether or not a newly extracted tuple $(named\_entity, category)$ is correct. Since named entities can be classified into some rough (top level) classes, such as *person*, *organization* or *location*, and other fine categories of named entities can be subtype of the above rough classes. When we extract $(named\_entity, category)$ tuples of a rough class $named\_entity\_type$ we can have the constraint: if a $named\_entity$ is a $category$, then the head noun of the noun phrase describing

category must be a sort of *named_entity_type*. For example, if *named_entity_type* is *person*, then *category* must be a sort of *person*. In other words, the *category* must be a subtype (more specific type) of *named_entity_type*. The subtype relation is represented as *hyponym* relation in WordNet [Fellbaum (1998)]. Thus, a *category* is valid if it is a hyponym of *named_entity_type*. The reverse relation of hyponym is *hypernym*, so it is equivalent to say a *category* is valid if its hypernym is a *named_entity_type*. Checking whether *named_entity_type* is a hypernym of a *category* seems faster than checking whether a *category* is a hyponym of *named_entity_type*, because the hyponym list of a *named_entity_type* is relatively large. We used WordNet to check hypernym relations for validation.

The validation function is integrated in Step 2 of the algorithm in Figure 3.5, and is described in Figure 3.8. From each sentence, when the validation function is not used, the algorithm extracts tuples only at the first match, which is not always the expected match.

---

1. $L' \leftarrow \emptyset$; //L' is defined in the algorithm in Figure 3.5
2. **for** every sentence s in D **do**
3. **for** every *named_entity* in s **do**
4. **for** every pattern p in P **do**
5. Construct a regular expression r from *named_entity* and p;
6. **if** (r matches s) **then**
7. Extract possible (*named_entity*, *category*) tuples;
8. **if** (*category* is valid) **then**
9. Let T the extracted tuples; $L' \leftarrow L' + T$;
10. **exit for**; //Skip other patterns for the current *named_entity* named entity
11. **else** Discard the tuples;
12. **end for**
13. **end for**
14. **end for**

Figure 3.8: Tuples extraction with the validation function

3.4 Named-Entity-Related Relations Extraction

The purpose of NECE is to extract (*named_entity*, *category*) tuples, in which *category* is the fine-grained categories of *named_entity*, so the set of named entity classes can be expanded by automatically extracting from texts. When we extract the tuple ("Bette Raptapoulos", "analyst") from the sentence "*There's a generally more positive attitude toward the economy, said Bette Raptapoulos, analyst for Prudential-Bache Securities Inc.,...*", we only have information: "Bette Raptapoulos" ISA "analyst". If we can extract the relation: "analyst" **for** "Prudential-Bache Securities Inc.", we will have complete information about "Bette Raptapoulos". In this section we extend the NECE to extract (*named_entity, category, related-to, object*) quadruples describing the relations: *named_entity* ISA *category* (or ISA relations for short), and category *related-to* object (or related-to relations for

short). From our observations, the related-to relations can be expressed in the following ways[8]:

a) The *category* and *object* are linked by a preposition: "*category preposition object*", e.g., "analyst for Prudential-Bache Securities Inc."

b) The *category* and *object* are connected by a possessive apostrophe: "*object*'s *category*", e.g., "Semi-Tech's chief executive officer". This can be interpreted as "category of object", e.g., "chief executive officer of Semi-Tech".

c) The *object* and *named_entity* are linked by a preposition: "*category named_entity preposition object*", e.g., "... said economist David Littmann of Manufacturers National Bank ...", from which an expected quadruple is ("David Littmann", "economist", "of", "Manufacturers National Bank").

d) The *object* is embedded in *category*, e.g., "IBM president". This can also be interpreted as "*category* of *object*", e.g., "president of IBM".

e) The *related-to* relation is implicitly expressed, e.g., "Mr. Baird, who heads the Manhattan U.S. attorney's securities-fraud unit, denied the quote ...", from which an expected quadruple is ("Baird", "head", "of", "securities-fraud unit").

Since case e) does not have fixed expressions, we do not treat such cases. In case d), because object is already embedded in category, we do not need to extract the object. For cases a), b) and c), we construct regular expressions to extract the *object*. We modify the procedure in Figure 4 to extract (*named_entity, category, related-to, object*) quadruples instead of (*named_entity, category*) tuples. Let *category_str* be the string containing the *category*, the regular expressions corresponding to each case are (we omit POS and chunk tags for readability):

a) * *category_str preposition noun_phrase* *

b) * *noun_phrase*'s *category_str* *

c) * *named_entity preposition noun_phrase* *

After extracting a valid *category* and a *named_entity* (Step 8 of the algorithm in Figure 3.8), if the current processing sentence matches one of the above regular expressions, the *object* is produced by removing POS tags in *noun_phrase*; *related-to* is the preposition after removing POS tags in cases a) and c); *related-to* is "of" in case b), then, (*named_entity, category, related-to, object*) quadruples are returned instead of tuples. If no regular expressions match the current processing sentence, then *object* and *related-to* are null. We call our new algorithm NECOE which stands for "*Named Entity, Category, Object Extraction*".

[8]All the below examples are from the WSJ corpus.

3.5 Utilization of Named-Entity-Related Relations for Information Search

The extracted (*named_entity, category, related-to, object*) quadruples are valuable for NLP applications. In this section, we use them for answering some types of questions. If *named_entity* in a quadruple is a person, the quadruple helps answer the query: "Who is *named_entity*?", e.g., "Who is Bette Raptapoulos?" If *named_entity* is of another type, such as *organization* or *location*, the quadruple helps answer the query: "What is *named_entity*?", e.g., "What is IBM?" For answering the question, we just search for quadruples having the same *named_entity* as that of the question. If a quadruple is found, then the answer is:

named_entity is a *category related-to object*.

The extracted quadruples also help answer list questions, e.g., "Give me the list of analyst for Prudential-Bache Securities Inc." The general form of this question type is "Give me the list of *category* [*related-to object*]", where the part in square brackets is optional. For answering this question type, we search for the list L of quadruples having the same *category*, [*related-to, and object*] as those of the question. The answer is the list of *named_entity* of quadruples in L. If the *related-to* of a question is "of", we also search for quadruples whose object is embedded in category (case (d) as discussed in Section 3.4). For example, if the question is "Give me the list of president of IBM", we also search for quadruples whose category is "IBM president".

The next possible type is *Which* question as discussed in Section 3.1, e.g., "Which *analyst* discussed a new strategy for the company?"

3.6 Experiments and Evaluation

Among named-entity-related questions, ones concerning about person named entities comprise a relative large portion as seen in the question list in Text Retrieval Conference (TREC) 9, Question Answering track[9], so this chapter concentrates on extracting (*name_entity*, *category*) tuples and (*name_entity, category, related-to, object*) quadruples of person named entities from plain-text corpora as a preliminary step, which can also be applied for extracting other named-entity types, such as *organization* or *location*.

In the category validation function, a word may have more than one *sense* (or meaning), and there may be multiple hypernyms for each sense. If all senses of a word are checked whether their hypernyms are persons or not, we may get unexpected results. For example, if all senses of the words: '*study*', '*guide*' or '*computer*' are checked, then one of their hypernyms is a *person*. For this reason, in the experiments, only the first sense of a given word is checked whether its hypernyms be a *person* or not in the validation function.

From our observations, for seed pattern, there may be an adverb which modifies an adjective in the noun phrase. After having been processed by a shallow parser, the adverb is

[9]http://tangra.si.umich.edu/clair/NSIR/cgi-bin/trec-question.cgi?collection=9&script=html/nsir.cgi

separated from the noun phrase as in the example: "... and/CC [NP Edward/NNP A./NN Masi/NNP] ,/, [ADVP formerly/RB] [NP vice/NN president/NN]..." In this case the appositive "formerly vice president" has an adverb "formerly". In the experiments, we consider about this issue for NECE, so seed patterns are treated a little different from other patterns.

3.6.1 Text dataset

We used the Wall Street Journal[10] corpus, which consists of 595 files, as the dataset. After extracting the body part and removing other parts, e.g., the headers, we got a plain text collection with the size of 308 MB consisting of nearly 3 million sentences. In the initiation step of the algorithm in Figure 3.5, we tagged all named entities in this plain text collection by an NER system. Through an investigation over free open source NER systems, we considered two competitive systems: OpenNLP[11] and LingPipe[12]. For NER task, OpenNLP is rather slow in comparison with LingPipe, so we used LingPipe for tagging named entities. After removing sentences that contained no person named entity, 667,981 sentences were collected, we call this the *big dataset* for later reference. Next step was to add POS and chunk tags for each sentence. In this task, OpenNLP was used because it has better performance than LingPipe.

The test set is 1,000 sentences which are randomly selected from the WSJ corpus. From the test set, 385 $(person, category)$ tuples were manually extracted. Among 385 tuples, 199 tuples have additional *related-to* relations. The distribution of *related-to* relations accordingly to the cases discussed in Section 3.4 is given in Table 3.4.

We ran our programs on the big dataset to get patterns, and tested the precision and recall of the patterns on a test set.

Table 3.4: The distribution of *related-to* relations

Case	a)	b)	c)	d)	e)
%	74.87	17.74	2.05	3.59	2.05

3.6.2 Experiments with NECE

We used C++ with the regular expression library *boost*[13] to build the NECE program. We also wrote a similar program called "NECE-Novalidation" which has no category validation function. We calculated precision (P), recall (R) and F-score (F) as follows:

$$P = \frac{T_{correct}}{T_{extract}} * 100\%,\ R = \frac{T_{correct}}{T_{manual}} * 100\%,\ F = \frac{2RP}{R+P}$$

where $T_{correct}$ was the number of tuples correctly extracted; T_{manual} was the number

[10]The dataset is included in TIPSTER corpus and distributed by Linguistic Data Consortium (LDC): http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93T3A

[11]http://opennlp.sourceforge.net/

[12]http://www.alias-i.com/lingpipe/index.html

[13]http://www.boost.org

of tuples manually extracted; and $T_{extract}$ was the number of tuples extracted by the program.

In general, the repetition threshold $threshold_R$ can be different from the diversity threshold $threshold_D$, so there are many combinations of the values of these variables. In our experiments, for the sake of simplicity, we set $threshold_R$ and $threshold_D$ to the same value. Later, we simply called them *threshold* for short. Table 3.5 shows the results of different pattern types (*seed*, *exact*, *sketch* and *extended sketch* patterns) of the NECE-Novalidation and NECE with the threshold of 3. This threshold is based on trying several values. From the results, we can see the important effect of validation function to the performance.

In order to investigate a proper threshold, we ran NECE with the thresholds of 4 and 5. Results in Table 3.6 show that 3 seems to be the proper value of the threshold. Figure 3.9 shows the number of accumulated tuples extracted by NECE with different thresholds and pattern types from the big dataset. Figure 3.9 also shows that the number of tuples extracted by seed patterns is about 41.6% (a relatively large portion) of the total number of extracted tuples. Especially, the precision of the seed patterns is very high (99.26%) proving that the selection of seed patterns is good.

Figure 3.10 shows the growth of distinct categories of NECE with threshold of 3 from the big dataset. The figure shows that the number of potential categories is relatively large (it reaches 40,810 for extended sketch patterns). And though sketch patterns help to extract only 5.5% of the total extracted quadruples, their discovered categories comprise 24% of total distinct categories.

Table 3.7 lists some top and bottom ranked categories along with their frequency that are extracted by our system.

Excluding the processing time of the Initiation step of the algorithm in Figure 3.5, our programs take about 40 hours.

Table 3.5: Results of NECE-Novalidation and the NECE with threshold of 3

Pattern	NECE-Novalidation			NECE 3		
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Seed	89.03	35.84	51.11	99.26	35.06	**51.82**
Exact	63.41	72.47	67.64	94.48	75.58	**83.98**
Sketch	62.88	74.81	68.33	94.50	80.26	**86.80**
E. sketch	62.88	74.81	68.33	94.50	80.26	**86.80**

Table 3.6: Results of the NECE with threshold of 4 and 5

Pattern	NECE 4			NECE 5		
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Seed	99.26	35.06	51.82	99.26	35.06	51.82
Exact	94.46	75.32	83.82	94.46	75.32	83.82
Sketch	94.48	80.00	86.64	94.46	79.74	86.48
E. sketch	94.48	80.00	86.64	94.46	79.74	86.48

From the statistics, the patterns whose middles are listed in Table 3.8 contribute the most of the extracted tuples. And the tuples whose categories are top-ranked categories as in Table 3.7 contribute the most to generate new patterns.

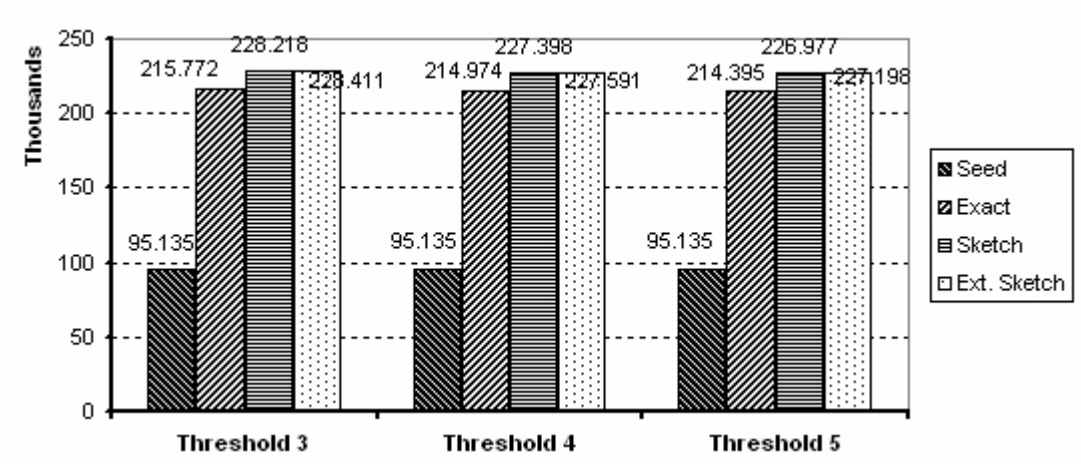

Figure 3.9: Extracted tuples at different thresholds

Table 3.7: Some top, bottom categories with frequency and *related-to* relations

Top categories	President (22679), Chairman (12835), Analyst (6729), Vice President (6011), Director (5821), Chief Executive Officer (5326), Judge (5050), Dr. (4931), Rep. (3479), Senior Vice President (3028), Executive Vice President (2698), Attorney (2537), Managing Director (1904), Chief Executive (1834), Chief Economist (1706), Lawyer (1601), Manager (1586), Economist (1510), Editor (1477)
Bottom categories	part-time CIA employee (1), partnership analyst (1), parliament deputy(1), parts marketing administrator (1), past finance director (1), patent specialist (1), paintings specialist (1), personal translator (1), freight carrier (1)
Related-to relations	managing director of investment bank, vice president for economic research, president of Trans World International Inc., deputy of Japanese equities, manager of sales

We also implemented another version of NECE called RNECE (*Reverse* NECE) which starts with the seed tuples instead of seed patterns. A program similar to RNECE without the category validation function was coded as the corresponding baseline which we called RNECE-Novalidation. The seed set consisted of 20 (*person*, *category*) tuples that are randomly selected from the WSJ corpus. The thresholds of both RNECE and RNECE-Novalidation are 3. And the results of the experiment on the same dataset are given in Table 3.9.

Except for the first scan, which took about 35 minutes, for generating new patterns from seed (*person*, *category*) tuples, the computation time of RNECE was about the same as that of NECE.

Table 3.10 records the number of middles generated by our programs, and this is also the number of patterns. Some NEC middles of exact patterns that are detected by NECE-3 but are not detected by RNECE-3 are listed below:

"] ,/, [NP Falconbridge/NNP] [NP 's/POS"

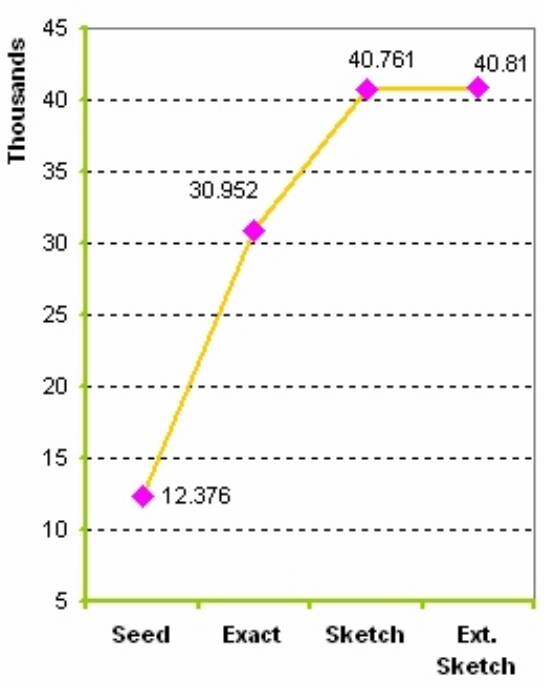

Figure 3.10: The growth of the number of distinct categories

Table 3.8: Middles of important patterns

Pattern	Middle	
Seed	Middle of seed patterns	
Exact	] [VP is/VBZ] [NP	(order NEC)
	] ,/, [NP who/WP] [VP is/VBZ] [NP	(order NEC)
	] [VP was/VBD] [NP	(order NEC)
	A space	(order CNE)
	] ,/, [NP Mr./NNP	(order CNE)
Sketch	] ,/, [NP *$word*/NNP] [NP 's/POS	(order NEC)
	] ,/, [NP *$word*/NNP *$word*/NNP] [NP 's/POS	(order NEC)

"] ,/, [NP Fannie/NNP Mae/NNP] [NP 's/POS"

We also implemented the DIPRE algorithm, and applied for extracting (*person*, *category*) tuples. In our experiments with the big dataset, the precision of DIPRE is about 90%, and the recall is about 4%. The low performance of DIPRE is understandable, since the intention of DIPRE is to focus on the specificity of a pattern in order to increase its reliability, and the recall of the algorithm can be significant with the compensation of a very large dataset as seen in the experiments of [Brin (1998a)], in which a repository consisting of 25 million web pages totalling 147 gigabytes was used. One more thing is about the computation time of DIPRE is big. In our experiments with the big dataset, DIPRE iterated 11 times in about 3.5 weeks.

From our observations, there are some factors that decrease the recall of NECE, as follows:

Table 3.9: Results of the RNECE with threshold of 3

Pattern	RNECE-Novalidation			RNECE 3		
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Exact	61.70	69.87	65.53	94.28	72.73	82.11
Sketch	61.10	72.21	66.19	94.30	77.40	85.02
E. sketch	61.10	72.21	66.19	94.30	77.40	85.02

Table 3.10: Number of generated middles

	Exact		Sketch		E.Sketch		Total
	NEC	CNE	NEC	CNE	NEC	CNE	
NECE 3	198	57	84	31	15	0	385
NECE 4	150	46	73	25	12	0	306
NECE 5	118	44	68	22	12	0	264
RNECE 3	196	54	79	32	8	0	369

- There are some sequences which do not satisfy the criteria for generating a new pattern.
- LingPipe incorrectly assigned some named entities as person named entities, while they were actually of another type, such as *organization*.
- A person may be represented by a pronoun: "*He is a worker*", where the pronoun '*He*' refers to a person named entity in a preceding sentence. This sentence is ignored because there is no person named entity.
- Person named entities that are not recognized by LingPipe can not be extracted, because the sentences containing them are removed in Initiation step.
- The category validation function fails to validate a *category* whose head noun has a meaning different from its first sense.
- Category may be expressed in plural form, such as "*John is one of the well-known leaders*". We do not treat such cases.
- A sentence may contain several tuples, however not all of them are extracted because the pattern set in a certain iteration does not cover all tuples of that sentence, so the sentence was removed in Step 2 of the algorithm in Figure 3.5.

Among the above reasons, the first reason is most often responsible, in comparison with the rest.

The main reason that decreases precision is the generation of incorrect patterns.

3.6.3 Experiments with NECOE

We extended NECE into NECOE, and NECE-Novalidation into NECOE-Novalidation. Since a *related-to* relation was extracted after an ISA relation was extracted, we evaluated the results of the two relations in quadruples separately. In our experiments, we also set

the value of the threshold to 3. The results of our experiments are shown in Table 3.11. Since extended sketch patterns did not increase the coverage much, we do not show their results. Some *related-to* relations are given in Table 3.7.

Table 3.11: Results of quadruple extraction

ISA relations						
Pattern	NECOE-Novalidation			NECOE		
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Seed	89.03	35.84	51.11	99.26	35.06	**51.82**
Exact	63.41	72.47	67.64	94.48	75.58	**83.98**
Sketch	62.88	74.81	68.33	94.50	80.26	**86.80**
Related-to relations						
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Seed	92.13	41.21	**56.94**	97.53	39.7	56.43
Exact	79.34	48.24	60.00	97.03	49.25	**63.97**
Sketch	76.64	52.76	62.50	96.64	57.64	**72.33**

From our observations, the reason that decreases the precision of *related-to* relations is derived from the extraction of incorrect ISA relations. The precision of *related-to* relations is 100%, if it is calculated on correctly extracted ISA relations. Since the *related-to* relations has a relatively large portion in ISA relations that can not be extracted by NECE, this is the reason that decreases the recall of *related-to* relations.

3.6.4 Experiments with Information Search

In this section, we carried out the experiments to consume the extract quadruples for answering some named-entity-related questions, as discussed Section 3.5. Since we concentrated on extracting the relations related to person named-entities, the experiments can answer person named-entity-related questions: *who* and *list*. Because freely existing co-reference resolution tools (e.g., Lingpipe and OpenNLP), when applied to the WSJ corpus, did not give reasonable results, we do not try to answer *which* questions. For the sake of reference, we call the system which consumes the quadruples extracted by NECOE to answer *who* and *list* questions PSearch (standing for Person Search). A similar system that uses quadruples extracted by NECOE-Novalidation to answer questions was also written, and called PSearch-Novalidation.

Answering *who* questions

From the test set, we randomly selected 50 person to construct 50 *who* questions (e.g., "*Who is Jeffrey Feiner*?"). We separated two types of answers:

- Answers without *objects*: when a quadruple (*named_entity, category, related-to, object*) having the same *named_entity* as that of question was found, an answer: *named_entity* is a(n) *category* (e.g., "Jeffrey Feiner is a(n) analyst") was constructed.
- Answers with *objects*: when a quadruple (*named_entity, category, related-to, object*) having the same *named_entity* as that of question was found, an answer:

named_entity is a(n) *category related-to object* (e.g., "Jeffrey Feiner is a(n) analyst at Merrill Lynch Capital Markets") was constructed.

In general case, several people may have the same name, so the answer to a *who* question may be a list of *named_entity* is a(n) *category related-to object*. Since the answer of a question of our system may include some incorrect *named_entity* is a(n) *category related-to object*, we used *cosine* to measure the similarity between the returned answer and the correct answer. Let $A = (r_1, r_2, \ldots, r_n)$ be a returned answer of a question, where r_i is "*named_entity* is a(n) *category* [*related-to object*]". Let $C = (c_1, c_2, \ldots, c_m)$ be the answer of that question, where c_i is "*named_entity* is a(n) *category* [*related-to object*]". The $cosine(A, C)$ is defined as:

$$cosine(A, C) = \frac{\sum_{i=1}^{l} 1}{\sqrt{\sum_{j=1}^{n} 1 \sum_{k=1}^{m} 1}} \tag{3.1}$$

where l is the number of correct r_i (r_i is the same as a certain c_j).

Suppose, for an answer A, if $cosine(A, C) \geq threhold$, then A is an correct answer, the results of the experiments with *who* questions are shown in Table 3.12 and 3.13.

Table 3.12: Answers of *who* questions without objects

Threshold	PSearch			PSearch-Novalidation		
	Precision%	Recall%	F1%	Precision%	Recall%	F1%
0.5	97.44	76.00	**85.39**	87.80	72.00	79.12
0.7	97.44	76.00	**85.39**	80.49	66.00	72.53
0.8	89.74	70.00	**78.65**	73.17	60.00	65.93
1.0	87.18	68.00	**76.40**	73.17	60.00	65.93

Table 3.13: Answers of *who* questions with objects

Threshold	PSearch			PSearch-Novalidation		
	Precision%	Recall%	F1%	Precision%	Recall%	F1%
0.5	92.31	72.00	80.90	92.68	76.00	**83.52**
0.7	92.31	72.00	**80.90**	85.37	70.00	76.92
0.8	89.74	70.00	**78.65**	73.17	60.00	65.93
1.0	87.18	68.00	**76.40**	73.17	60.00	65.93

Answering *list* questions

We separated two types of *list* questions:

- *list* questions without *objects*, e.g., "Give me the list of *analyst*?"

- *list* questions with *objects*, e.g., "Give me the list of *analyst at Merrill Lynch Capital Markets*?"

We also used *cosine* to measure the similarity of the returned answers and correct ones. For *list* questions without *objects*, we randomly selected 30 different *categories* to construct questions. The results of the experiment is shown in Table 3.14.

For *list* questions with *objects*, we randomly constructed 40 questions which have different *category related-to object*. The results of the experiment is shown in Table 3.15. Since all the *cosine* value in this experiment were 1, we did not used the threshold.

Table 3.14: Answers of *list* questions without objects

Threshold	PSearch			PSearch-Novalidation		
	Precision%	Recall%	F1%	Precision%	Recall%	F1%
0.5	100.00	86.67	**92.86**	96.15	83.33	89.29
0.7	88.46	76.67	**82.14**	88.46	76.67	**82.14**
0.8	84.62	73.33	**78.57**	76.92	66.67	71.43
1.0	69.23	60.00	**64.29**	65.38	56.67	60.71

Table 3.15: Answers of *list* questions with objects

PSearch			PSearch-Novalidation		
Precision%	Recall%	F1%	Precision%	Recall%	F1%
100	77.50	**87.32**	100	62.50	76.92

3.7 Summary

In this chapter, we proposed a method for automatically extracting actually categories of named entities from text documents in a data-driven way. We proposed new constraints for generating new patterns, as well as a method for validating whether a new tuple is correct or not.

The extension of the algorithm was applied to extract (*named_entity, category, related-to, object*) quadruples, each of them describes the relations: *named_entity* ISA *category*, and *category related-to object*. These two relations give detail information about the *named_entity*, so they are consumed to answer some named-entity-related question types.

We performed experiments on the Wall Street Journal corpus, and obtained good results.

In the current implementation, we have not taken the priority of patterns in the same pattern set, so that more reliable pattern is selected before less reliable ones with the purpose to improve the precision. This is a possible problem for future study.

Our model can be applied to extract fine categories of other named entity types, such as *organization* and *location*, so that it can help answer *what* questions.

Chapter 4

Application of RST Relations for Information Search

In this chapter, we present a method to extract answers to some non-named-entity-related questions based on the Rhetorical Structure Theory. We exploit the fact that one text span can be the answer to a question related to its adjacent text span.

4.1 Introduction

There are various question types beside named-entity related questions, e.g., "Why didn't Mr. Bush need to wait for a law?" or "How to preserve the integrity of the Arby's system?"

This chapter proposes a method for extracting answers (not pages) to some question types from documents by exploiting the structure of documents. The structure of documents includes the characteristic that one text span can be an answer to a question related to the adjacent text span. The structure of documents can be represented by Rhetorical Structural Theory proposed by [Mann & Thompson (1988)]. In their proposal a document is represented in form of a tree, in which there are relations between adjacent text spans. Each relation has a specified meaning and some relations are clue for extracting answers to some types of questions.

To preserve the local order of words in a sentence, we propose to represent sentences of documents and questions in the form of Ternary expressions. The indexing and matching process is based on *Ternary expressions* [Katz (1991)]. Our experiments prove this method to be better than keyword-based indexing and matching.

4.2 Related work

The Rhetorical Structure Theory has been applied to solved some NLP problems such as query-based summarization [Bosma (2004)], Information Retrieval [Marir & Haouam (2004)], news program generation [Lindley, *et al.* (2001)] or question answering [Fukumoto (2007)].

[Lindley, *et al.* (2001)] exploits the RST to develop a interactive video system in which a linear video presentation is generated dynamically and adaptively from an underlying database having no predetermined linear structure. The system targets at creating a presentation tuned to the needs and interests of a particular viewer. For example, the system can be applied for narrative which, basically, is about a telling a story, and hence involves a system of causally interrelated events, actions and situations.

[Bosma (2004)] studied RST for removing unimportant part of a paragraph in order to generate a concise answer to a query. In other words, [Bosma (2004)] focused on creating an extractive summarization or to extract the most salient sentences from a document.

[Marir & Haouam (2004)] used RST for information retrieval. Since the rhetorical relation between units of text can be identified based on cue phrases, [Marir & Haouam (2004)] saved the discovered rhetorical relations into the database. In the serving mode, for a given query, the system can search the collection of documents not only keywords (as traditional Information Retrieval systems), but also rhetorical relations. The technique gives improvements over keyword-based search system.

[Fukumoto (2007)][1] developed a question answering system (for Japanese) which can answer some non-factoid question types, e.g., *why*, *definition* and *how to*. [Fukumoto (2007)] identified patterns for extracting answer candidates for each question type. For why-type questions, the inter-sentential relations proposed in RST (e.g., the causal and manner relations) were chosen as the clues for extracting answers. If one element of these relations matches the question, the other element will be the answer. If one sentence matches extraction patterns, the sentence is deemed as an answer candidate. The semantic clue words which mean 'reason', 'cause' and 'background' were also used as clues for extracting answers. For definition-type questions, [Fukumoto (2007)] analyzed the question answer data and newspaper articles, and extracted patterns for this question type. These patterns are descriptive patterns which consist of some terms and their definition or descriptions. For how-type questions, some kinds of approach as the definition-type questions were applied. The patterns of these questions are description of procedures. For extracting answers to why-type questions [Fukumoto (2007)] did not identify the rhetorical relations between the units of text. This is different from our approach in which we need to identified the rhetorical relations, and index the documents with respect to these relations for later answer extraction as discussed in the subsequent sections.

4.3 Rhetorical Relation Exploration

This section introduces the Rhetorical Structure Theory (RST), and how RST can help to extract answers to some question types. Rhetorical Structure Theory was proposed by [Mann & Thompson (1988)], which is a method to present the coherence of texts so that the reader can understand the discourse structure. [Mann & Thompson (1988)] defined a set of 23 rhetorical relations. This model represents the structure of a text in the form of a tree (called "*rhetorical tree*", "*discourse tree*" or "*RST tree*") that labels relations between adjacent text spans, such as clauses (lowest level), sentences, or paragraphs. The smallest text spans (leaves of the discourse tree) are called *elementary discourse units* which correspond to clauses or clause-like units with independent functional integrity.

[1]The Fukumoto's study was published in 2007, and our study in this chapter was published in 2006.

Internal nodes of a discourse tree are non-overlapping spans that are larger than clauses, and the root node spans the entire document.

Constraints on N: presents a situation that is not a volitional action.	
Constraints on S: none.	
Constraints on the N+S combination:	
	S presents a situation that, by means other than motivating a volitional action caused the situation presented in N; without the presentation of S, reader might not know the particular cause of the situation; a presentation of N is more central than S to Writer's purpose in putting forth the N-S combination.
The effect:	Reader recognizes the situation presented in S as a cause of the situation presented in N.

Figure 4.1: Definition of the Non-volitional cause relation.

There is a relation (called "*rhetorical relation*") between adjacent spans (spans having the same parent), e.g., solutionhood, elaboration or purpose. This relation can be asymmetric or symmetric. An asymmetric relation, also called a *nuclear-satellite* relation, involves two spans, one of which is more essential to the writers goals than the other. The more important span in a rhetorical relation is called a *nucleus* (N); whereas the less important one is called a *satellite* (S). The nucleus of a rhetorical relation is comprehensive and independent of the satellite, but not viceversa. For example, the sentence "[*Because the car broke down*], [John was late for the meeting.]" can be divided into two text spans: "Because the car broke down"(denoted as SP1), and "John was late for the meeting" (denoted as SP2). The deletion of the clause SP1 does not significantly affect the meaning of the whole text. The clause SP2 is still understandable without the clause SP1. Whereas, the clause SP1 is is not understandable without the clause SP2. Hence, the clause SP1 is more important than the second clause in respect to the writers purpose, so it serves as the nucleus, and the clause SP1 is the satellite. There is an asymmetric rhetorical relation "non-volitional cause" between the two spans according to the definition in Figure 4.1 [Mann & Thompson (1988)]. The rhetorical structure of this sentence is represented in Figure 4.2.

A *symmetric* relation, also called a *multi-nuclear* relation, involves two or more spans, each is equally important in respect to the writers intention in producing texts, such as

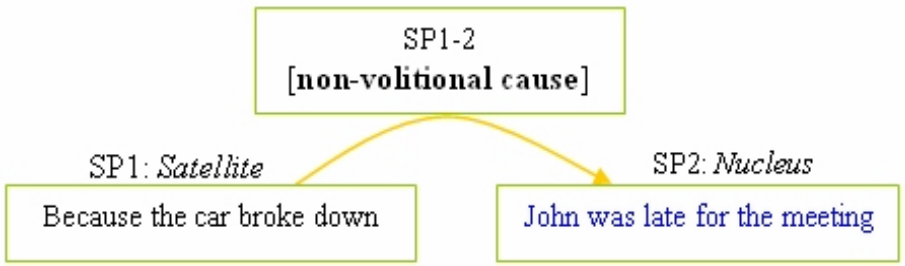

Figure 4.2: An example of a rhetorical relation.

the two clauses "Three seats currently are vacant" and "and three others are likely to be filled within a few years" in the sentence "[Three seats currently are vacant][and three others are likely to be filled within a few years.]". Every node in a symmetric relation is a nucleus.

Each rhetorical relation has a specified meaning and constraints, as the definition of the non-volitional cause rhetorical relation given in Figure 4.1. From the definition, in the rhetorical relation in Figure 4.2, SP1 is the cause of the situation presented in SP2. Therefore, span SP1 is an answer to the question "*Why was John late*?" which relates to the situation stated in SP2. In general, SP2 may consist of smaller text spans. When the question is related to a child node of SP2, how to extract the answer in this case is a problem. Fortunately, according to [Marcu (2000)], "*If a rhetorical relation R holds between two text spans of the tree structure of a text, that relation also holds between the most important units of the constituent spans*". Hence, if the question is related to one of the most important descendants of span SP2, then SP1 is still an answer.

The process of constructing the discourse tree of a document can be automatic as presented in [Marcu (2000), Le (2004), Nomoto (2004)].

4.3.1 Rhetorical Relation Application

RST relations can be applied to extracting answers to a questions of some specific types. As discussed in Section 4.3, the span SP2 has a non-volitional cause relation with span SP2, so if given a question "Why was John late?", then the span SP1 is an answer. This is the idea how to apply rhetorical relations to extract the answers to questions.

The types of question given in the TREC [Voorhees (1999)] are divided into 3 types: *list*, *definition* and *factoid* questions. These are all factual questions and the answers can be directly extracted from the sentences that match the questions. There can be more types of questions which are frequently asked as seen in the Frequently Asked Questions (FAQ)[2]:

- *How to* questions: e.g., "How can I recover space after installing updates?".
- *Suggestion* questions: e.g., "What should I do about compilation error in V6.1.b10?".
- *Why* questions: e.g., "Why is the machine is booting over and over?".
- *Yes/no* questions: e.g., "Are there any ftp sites?"

Constraints on N: none.	
Constraints on S: represents a problem.	
Constraints on the N+S combination:	the situation presented in N is a solution to the problem stated in S.
The effect:	Reader recognizes the situation presented in N as a solution to the problem presented in S.

Figure 4.3: Definition of the Solutionhood relation.

[2]http://www.faqs.org

As defined in Figure 4.3 [Mann & Thompson (1988)], Solutionhood relation indicates the situation presented in nucleus is a solution to the problem stated in satellite. Also as

Constraints on N: presents an activity.	
Constraints on S: presents the situation that is unrealized.	
Constraints on the N+S combination: S presents a situation to be realized through the activity in N.	
The effect:	Reader recognizes that the activity in N is initiated in order to realize S.

Figure 4.4: Definition of the Purpose relation.

defined in Figure 4.4 [Mann & Thompson (1988)], Purpose relation indicates that satellite presents a situation to be realized through the activity in nucleus. Thus, solutionhood and purpose relations are clues for extracting answers to "*How to*" questions.

For "*suggestion*" questions, solutionhood relation gives a possible solution, and is suitable for extracting answers.

As defined in [Mann & Thompson (1988)], Volitional cause (Figure 4.5), non-volitional cause (Figure 4.1), Volitional result (Figure 4.6), and Non-volitional cause (Figure 4.7) all are related to causal relation, therefore they are clues for extracting answers to "*Why*" questions.

For "*yes/no*" questions, we can extract the answers based on the facts, no rhetorical relation is needed.

Constraints on N: presents a volitional action or else a situation that could have arisen from a volitional action.	
Constraints on S: none.	
Constraints on the N+S combination: S presents a situation that could have caused the agent of the volitional action in nuclear to perform that action; without the presentation of S, R might not regard the action as motivated or know the particular motivation; N is more central to writer's purpose in putting forth the N-S combination than S is.	
The effect:	Reader recognizes the situation presented in S as a cause for the volitional action presented in N.

Figure 4.5: Definition of the Volitional cause relation.

Table 4.1: The question types and corresponding rhetorical relations

Question types	Rhetorical relations
How to	Solutionhood, purpose
Suggestion	Solutionhood
Why	Volitional cause, non-volitional cause, volitional result, non-volitional result

Constraints on N: none.	
Constraints on S:	presents a volitional action or situation that could have arisen from a volitional action.
Constraints on the N+S combination:	N presents a situation that could have caused the situation presented in S; the situation presented in N is more central to writer's purposes than is that presented in S.
The effect:	Reader recognizes that the situation presented in N could be a cause for the action or situation presented in S.

Figure 4.6: Definition of the Volitional result relation.

Constraints on N: none.	
Constraints on S: presents a situation that is not a volitional action.	
Constraints on the N+S combination:	N presents a situation that caused the situation presented in S; presentation of N is more central to writer's purposes in putting forth N-S combination than is the presentation of S.
The effect:	Reader recognizes that the situation presented in N could have caused the situation presented in S.

Figure 4.7: Definition of the Non-volitional result relation.

The question types and corresponding helpful rhetorical relations are listed in Table 4.1. Other types irrelevant to rhetorical relations are not listed. In order to use rhetorical relations for extracting answers, we need to construct the rhetorical structure of documents and then index documents in such a way that the rhetorical structure of documents remains. The technique for indexing documents is mentioned in Section 4.4. The method for retaining rhetorical structure of documents is described in Section 4.5. After these steps, we have a knowledge base for answering questions. In answering mode, the system operates as the algorithm described in Figure 2, where $Answers$.add($span$,d) means to add text belonging to *span* in the document d to the answer list. The method for matching the question against the knowledge base is described in Section 4.4.

4.4 Indexing Documents and Matching Questions

This section gives details on how we represent text segments, index documents and match questions. We borrow the sentence representation style used by natural language Question and Answering system "SynTactic Analysis using Reversible Transformation" (START[3]) [Katz (1991)], which successfully answers a series of questions by using this representation style.

[3]START is serving at http://start.csail.mit.edu/

```
Search(Question q)
1.    Identify the type of question q;
2.    Identify a set of rhetorical relations R corresponding to this question type;
3.    if (is_empty(R)) then
4.      return "This type of question is not supported!";
5.    else
6.       Match the question q against the knowledge base;
7.       if (no matches found) then return "Not found!";
8.       else
9.          for (each match m) do
10.            for (each relation r in R)do
11.                Find a span sp2 (in the rhetorical structure of a document d containing
12.                the match m) having the relation r with the span sp1 which contains
13.                the match m (or one of its most important constituents contains the
14.                match m);//As depicted in Fig 4.9.
15.                if (found) then Answers.add(sp2, d);
16.           end for
17.        end for
18.    return Answers;
```

Figure 4.8: The search algorithm

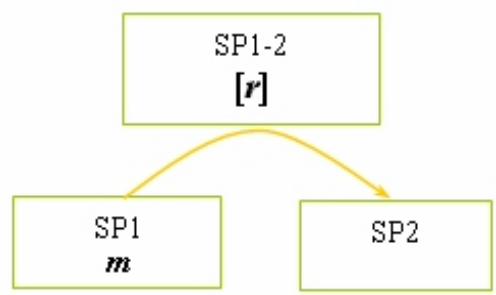

Figure 4.9: Finding an expected span.

4.4.1 Indexing and Answering in START

A sentence, in START, is divided into kernel sentences which usually contain one verb. For example, the sentence "If the orator wants to persuade people, he must speak the things people wish to hear." can be broken up into five smaller units [Katz (1991)], where the subscripts are used to marked the same entities:

(S1) The orator$_i$ wants S2.
(S2) The orator$_i$ persuades people$_j$.
(S3) He must speak thing$_k$.
(S4) People$_k$ wish S5.
(S5) People$_k$ hear the things$_k$.

A kernel sentence is represented by a *Ternary expression* (T-expression) which is a triple of $\langle$*subject relation object*$\rangle$, where *relation* is an infinitive verb, a preposition or some

special words (e.g., describe_relation). Other information of the sentence (such as tense, voice, negation, auxiliary verbs and adverbs) is stored in another place called *history*. For example, the kernel sentence S2 can be represented as ⟨orator speak thing⟩. For complex sentences, START allows any T-expression to take another T-expression as its *subject* or *object*, as seen in the above example, S5 serves as the *object* of the T-expression representing S4.

In answering mode, START converts questions into T-expressions and performs the search against its knowledge base. For example, if given the above fact "John Adams discovered Neptune", START creates the T-expression ⟨"John Adams" discover Neptune⟩ and stores it in its knowledge base. Then if a user asks the question "Who discovered Neptune?", it converts the question into the T-expression ⟨$*who* discover Neptune⟩, where $*who* serves as a matching variable. This T-expression is matched against the knowledge base, in this case, $*who* is matched with "John Adams", then START uses the T-expression ⟨"John Adams" discover Neptune⟩ with its history to restore the sentence "John Adams discovered Neptune" as the answer.

4.4.2 Indexing in Our System

We propose a new way of indexing based on T-expressions. The idea is to *group* T-expressions of a sentence (or a text segment) together so that we can use *cosine* measure to calculate the similarity of a text segment and a question as described in the next section. The method for grouping T-expressions of the same text segment is mentioned in Section 4.5. We use T-expressions for the indexing and matching processes, not for the purpose of generating answers. When a span is found to be an answer, all the text segments belonging to this span are collected and returned.

4.4.3 Matching in Our System

We use *cosine* measure for scoring the similarity between a question and a text segment. Suppose a question q is converted into T-expressions $(tq_1, tq_2, \ldots, tq_n)$ and a text segment s is converted into T-expressions $(ts_1, ts_2, \ldots, ts_l)$. In general cases, n and l are not equal, thus we have to normalize them to be the same size. Let m be the total of unique T-expressions of q and s. Let $(tc_1, tc_2, \ldots, tc_m)$ be the vector of unique T-expressions of q and s. Then s and q can be represented as vectors of size m $(t_1, t_2, \ldots, t_m)$, where $t_i = 1$ if T-expression tc_i is present in its T-expression list, otherwise $t_i = 0$. If one T-expression of the question q contains a variable, we must treat that T-expression differently from ordinary T-expressions in comparison. When two T-expressions are the same except for a variable, they are regarded as being matched. For example, if there is a T-expression ⟨$*who* be worker⟩, this T-expression and T-expressions like ⟨John be worker⟩ and ⟨James be worker⟩ (in the knowledge base) are said to be matched. Finally, the *cosine* between

the question q and the text segment s is defined as follows:

$$cosine(q,s) = \frac{\sum_{k=1}^{m} t_k(q)t_k(s)}{\sqrt{\sum_{k=1}^{m} t_k(q)^2 \sum_{k=1}^{m} t_k(s)^2}} \tag{4.1}$$

where $t_k(s)$, $t_k(q)$ are the presence of the k^{th} element of T-expressions of the text segment s and question q correspondingly.

This method provides the flexibility for matching process, and users have an option to adjust the threshold to filter the result with respect to a degree of similarity in the range (0, 1].

There can be history (tense, negation, auxiliary verbs and adverbs) attached to a T-expression. In the matching process we must consider this issue. For *yes/no* questions, we just measure the similarity between the question and a sentence without considering negation and tense because a match is always an answer. For example, if we have the fact "John bought a new car in June", and the question is any one of "Did John buy a car in June?", "Didn't John buy the car in June" or "Has John bought a car?" that sentence is still the answer. For other types of questions, if the question and a sentence do not match regarding the negation or tense, we will not consider that sentence to be a candidate as an answer. For example, given a question "File access is denied. How to fix this problem?" it is incorrect to match the T-expressions of "File access is denied" with the T-expression of the fact "File access is not denied". We do not take adverbs into account because they do not affect the precision of answers.

4.5 Experiments and Evaluation

We used the RST Discourse Treebank[4] [Carlson, *et al.* (2001)] for testing because the task of building RST trees of documents is outside the scope of this study, and there are some studies on this issue [Marcu (2000), Le (2004), Nomoto (2004)]. This Treebank contains a subset of documents from the Wall Street Journal which are annotated according to RST theory. We used database management system (DBMS) MySQL[5] for storing data. A tree can be represented by parent-child relation. A text span may consist of multiple T-expressions, so this can be represented as a 1-N (one-to-many) relation. Thus, the technique for retaining RST structure, and grouping T-expressions of the same text span is implemented using the relational DBMS MySQL.

The module of converting sentences into T-expressions is a rule-based system. We firstly used the Charniak parser[6] [Charniak (2000)] (whose F-score is 90%) to parse sentences, and secondly the T-expressions were built based on the output of Charniak parser. Starting from a set of basic conversion rules from a sequence of part-of-speech (POS) tags to T-expressions, e.g., "$Noun_1$+($Preposition$+$Noun_2$)" was converted into

[4]The corpus is distributed by LDC:
http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002T07

[5]The software is available at http://www.mysql.com

[6]The source code is available at http://www.cs.brown.edu/people/ec/

$\langle Noun_1\ Preposition\ Noun_2\rangle$, complex sequences of POS tags were recursively processed at the lowest level. For example: for $Noun_1$+($Preposition_1$+($Noun_2$ + ($Preposition_2$ + $Noun_3$))), after having generated $\langle Noun_2\ Preposition_2\ Noun_3\rangle$, the head noun $Noun_2$ was kept for generating $\langle Noun_1\ Preposition_1\ Noun_2\rangle$. The complexity of this module is that of Charniak parser, because the converter of a parsed sentence into T-expressions can run in real time. The precision, recall and F-score of this module are 92%, 90% and 91%, respectively.

We built another similar system, which used keywords instead of T-expressions as a baseline. We created a question set consisting of 91 *how to*, 64 *why*, 25 *suggestion* and 259 *factual* questions. In keyword-based system, we set the threshold for *cosine* measure to 0.30 for factual questions and 0.20 for other questions, while in the T-expression-based system we set the threshold to 0. Non-factual questions have corresponding rhetorical relations to filter the results while factual questions do not. In order to reduce the number of answers returned to factual questions, we set a higher threshold for factual questions in the keyword-based system. We calculated the precision (P), recall (R), and F1-score (F) as follows:

$$P = \frac{\sum A_{correct}}{\sum A_{returned}} * 100\%,\ R = \frac{\sum A_{correct}}{Total\_Questions} * 100\%,\ F = \frac{2RP}{R+P}$$

where $A_{returned}$ is a returned answer; $A_{correct}$ is a correctly returned answer; $Total\_Questions$ is the total number of questions of a certain question class. The results are shown in Table 4.2. Using *cosine* in the matching process, we can not preserve the order of words in the whole sentence, however, the local order of words in a T-expression is still retained. This characteristics made T-expression-based experiments give better results than those of keyword-based ones.

In order to see how effectively the RST contributes to the performance of our system, we implemented another system which extracts answers based on *clue phrases* that signal a rhetorical relation. For example, the clue phrases that usually connect text segments of causal relations are 'because', 'since' or "as the results". Since the causal relations are clue for answering why-questions, the above clue phrases are also the clues for extracting answers to why-questions. Similarly, the clue phrases for extracting answers to how-to-questions are "in order to" or 'to'. For each question, we extract sentences that are close to the question. We do this by calculating the $cosine(q, s)$ between the question q and a sentence s. If $cosine(q, s) > threshold$ and s contains a clue phrase with respect to the type of question q, the sentence s is selected as an answer candidate. If in one document, there are more than one sentence satisfying the above conditions, the sentence having the highest $cosine(q, s)$ is selected. The best result of the experiments were recorded in Table 4.2, and the results of the experiments with different *threshold* values are shown in Table 4.3. With this approach, we can only extract answers in cases both the question and its answer reside in the same sentence. Since in the dataset used in our experiments, the solutionhood relations (that are clue for extracting answers to suggestion questions) hold between two sentences (or paragraphs), not between two clauses of the same sentence, in addition, we did not find any typical clue phrases that signal a solutionhood relation, we can not extract answers to suggestion questions. From our observations of the experiments in Table 4.3, when used a small value of threshold, the keyword-based system without using RST can extract answers to some questions which are not found by the T-expression-based system with using RST. The reason of this problem is not from RST. This happened

Table 4.2: The results of T-expression-based and keyword-based systems, where *Avg* is the average answer per question

Type	#	T-expression with RST				Keyword with RST			
		P(%)	R(%)	F(%)	Avg	P(%)	R(%)	F(%)	Avg
Why	64	83.8	96.9	**89.9**	1.16	68.9	93.8	79.4	1.36
How to	91	80.1	97.8	**88.1**	1.21	73.9	96.7	83.8	1.35
Suggestion	25	95.2	80.0	**86.9**	0.84	77.8	84.0	80.8	1.08
Fact	259	80.6	98.4	**88.6**	1.22	30.6	85.7	45.1	2.79

Type	#	Keyword without RST			
		P(%)	R(%)	F(%)	Avg
Why	64	30.93	32.97	31.91	1.07
How to	91	45.45	31.25	37.04	0.69

Table 4.3: The results of keyword-based system without using RST with different threshold values, where *Avg* is the average answer per question

Threshold	How to				Why			
	P(%)	R(%)	F(%)	Avg	P(%)	R(%)	F(%)	Avg
0.35	0.56	91.21	1.11	164	0.84	70.31	1.66	83.91
0.40	1.99	75.82	3.89	38.03	3.09	51.56	5.84	16.67
0.45	8.33	54.95	14.47	6.59	12.05	42.19	18.75	3.50
0.50	30.93	32.97	31.91	1.07	45.45	31.25	37.04	0.69

when a question that is paraphrased so that its corresponding T-expressions are totally different from those of the related sentence in the database.

The limitations that make the T-expression-based system fail to find answers in some situations are:

- When a question relates to more than one adjacent text segments of a sentence and T-expressions of the question are different from those of related text segments, the matching process will fail.
- The question does not match an important constituent of a span which has the expected relation with the span containing the answer.
- The module for converting a question into T-expressions incorrectly adds a matching variable in some situations.
- Questions contain proper names which are replaced by pronouns in the knowledge base.
- Questions are expressed in different ways from the original sentences used to build the knowledge base.

A possible solution to relieve some limitations is to build T-expressions from the original sentences (by concatenating text segments of the same sentence), and perform co-reference on original documents to resolve pronouns.

From our observations, not all instances of rhetorical relations listed in Table 4.1 can help construct answers. The reason is that from a rhetorical relation between two spans, there hardly be a question related to either spans. For example, in a non-volition-cause relation in the sentence "Earthquakes cause tidal waves" [Girju (2002)], there hardly be a causal question related to this sentence.

By representing text segments in T-expressions, we can support for entailment relation. From one text segment we can entail another one which omits some adjectives. For example, from the text segment "John bought a new car" is converted into ⟨John buy car⟩ and ⟨new describe_relation car⟩, so we can infer the text segment which has the T-expression ⟨John buy car⟩ by omitting the T-expressions having describe_relation relation.

Another possible improvement is to generate finer-grained T-expressions. In the current implementation, the sentence "John and Jane bought a new car" is converted into two T-expressions ⟨"John Jane" buy car⟩ and ⟨new describe_relation car⟩. If we convert the above sentence into three T-expressions: ⟨John buy car⟩, ⟨Jane buy car⟩ and ⟨new describe_relation car⟩, then we can answer a question that is written in a different order, such as "Did Jane and John buy a new car?".

4.6 Summary

In this chapter, we proposed a method for finding and extracting answers to some types of questions based on the rhetorical structure of documents. We exploited the characteristics of document structure in which one span of text can be an answer to a question related to the adjacent text span. According to each question type, we identified related rhetorical relations which help in finding answers. T-expressions were used to index documents and *cosine* measure was used in the matching process. Comparison of the two experimental systems showed that the results of the T-expression-based system were better that those of keyword-based one.

We currently considered only six rhetorical relations, other ones are still valuable for further study.

Though the relation definitions of Mann and Thompson are only one well-known case, we may define any kind of relations that help answer an arbitrary type of question.

In a future study, we intend to apply an ontology, such as WordNet, in the matching process in order to be able to answer questions that have different linguistic expressions but the same meaning as some facts.

Chapter 5

Question classification

For each question type, a certain strategy is used to extract the answer. Given a question the system needs to know its type before continuing the next actions. In this chapter, we study the question classification problem to identify the question type for the system. Though semi-supervised learning has been studied and applied to many NLP problems (e.g., text classification), it is not yet studied for question classification. Our first issue in this chapter is to apply semi-supervised learning for question classification in order to exploit the unlabeled questions to improve the classification precision. In situations where the number of question classes is big, the performance of classification algorithms is adversely affected. Our second issue is to apply hierarchical classifiers for reducing the number of question classes per classifier. We also applied different learning methods in this hierarchy as well as the proposal to automatically expand the hierarchy of question classes for nodes consisting of a large number of classes.

5.1 Introduction

Though in this chapter, question classification is an important step in our final system, we study it generally and thoroughly so that it can also applied to other systems that need such functionality, such as Question Answering. Question classification is the task of identifying the type of a given question among a predefined set of question types. The type of a question can be used as a clue to narrow down the search space to extract the answer, and used for query generation in a QA system [Li & Roth (2002)]. Therefore, it has a significant impact on the overall performance of QA systems.

There have been several studies to solve this problem focusing on supervised learning [Zhang & Lee (2003), Kadri & Wayne (2003), Li & Roth (2002)]. However, the cost of making labeled (training) data is high, and a large training data set is needed to make significant impact on the performance. Also the above methods do not use unlabeled questions, which are readily available to improve the performance of classification. In order to utilize both labeled and unlabeled data, firstly, we propose to use semi-supervised learning. For the semi-supervised learning algorithm, we adopted the Tri-training [Zhou & Li (2005)], since it has a simple but efficient method of deciding how to label an unlabeled instance. Tri-training uses three classifiers of the same algorithm,

and if any two classifiers of the three classifiers predict the same label for an unlabeled instance, while the confidence of the labelling of the classifiers are not needed to be explicitly measured, then that instance is used for further training the other classifier. Such simplicity gives Tri-training advantages over other Co-training algorithms, such as the Co-training algorithm presented by [Goldman & Zhou (2000)], which frequently uses 10-fold cross validation on the labeled set to determine how to label the unlabeled instances and how to produce the final hypothesis. If the original labeled set is rather small, cross validation will give high variance and is not useful for model selection.

The simplicity also makes Tri-training faster than the algorithm of Goldman, in which the frequent use of cross validation makes the learning process time-consuming. At the beginning, Tri-training bootstrap-samples the labeled data to generate different training sets for three classifiers in order to make the three classifiers diverse enough so that the Tri-training algorithm does not degenerate into *self-training* [Nigam & Ghani (2000)] with a single classifier. However, question data is sparse and imbalanced. A question class may include only a few questions in a corpus, so if the bootstrap-sampling procedure duplicates some questions while omitting some questions in the classes with few questions, then classifiers being trained on these bootstrap-sampled sets have higher error rates than those of classifiers being trained on the labeled set. In order to avoid this drawback, while still keeping classifiers diverse, we propose to use more than one classifier with different algorithms. The original training set is initially used by the three classifiers without bootstrap-sampling. Another proposal is to apply more than one *views* (feature spaces) in the learning process. This allows the three classifiers to initially be trained from the labeled set with different feature spaces and still have diversity. In the second proposal, for the sake of simplicity, in the experiments, we used two different classification algorithms: Support Vector Machines [Cortes, *et al.* (1995)] and Maximum Entropy Models [Berger, *et al.* (1996)] in combination with two views: *bag-of-word* and *bag-of-pos&word* features. Two classifiers which use the first algorithm are assigned different views, i.e., the first classifier gets bag-of-word and the other gets bag-of-pos&word features. The third classifier uses the second algorithm with bag-of-word features. With this strategy, three classifiers have initially different hypotheses.

Question answering track in TREC defines six *coarse-grained* classes of questions: *abbreviation*, *description*, *entity*, *human*, *location* and *numeric* which have been used in several studies [Ittycheriah, *et al.* (2001)]. However, six classes of questions for a question answering system in open domain are not sufficient enough. The larger the number of question classes the better performance of a QA system is. The second problem we consider is in cases where the number of question class is big, the performance of classification algorithms is affected. In order to reduce the number of question classes for each classifier, we propose to apply hierarchical classifiers in accordance with question class taxonomy. Each classifier in the hierarchy identifies a finer-grained class from an input question or an output of the preceding classifier. The final classifier in the chain of classifiers determines the fine class (a class at the leaves of the question taxonomy) of a given question. Another proposal, we try to further expand nodes in the question taxonomy that consist of a large number of question classes. Because clustering is a unsupervised method to group classes that are closed (in a certain distance) to each other, we use clustering to solve this problem.

5.2 Related Work

5.2.1 Previous Question Classification Studies

There are two broad classes of approaches to question classification: rule-based and statistical. In rule-based approaches, an expert manually constructs a number of regular expressions and keywords corresponding to each type of question. Meanwhile, in statistical approaches, a model is assumed and trained on a sufficiently large set of labelled questions in order to automatically find out useful patterns for classification.

Statistical approach have advantages over rule-based approach, because they require less expert labor and are easily portable to other domains. Thus, recent work has concentrated on the approach, especially on the supervised learning approach which is a branch of the statistical approach.

[Kadri & Wayne (2003)] employed error correcting codes in combination with support vector machine to improve the results of classification. [Zhang & Lee (2003)] and [Li & Roth (2002)] explored different types of features for improving the classification accuracy. Zhang and Lee considered *bag-of-word*, *bag-of-ngram* (all continuous word sequences in a question) features. Especially, they proposed a kernel function called *tree kernel* to enable support vector machine (SVM) to take advantage of the syntactic structures of questions.

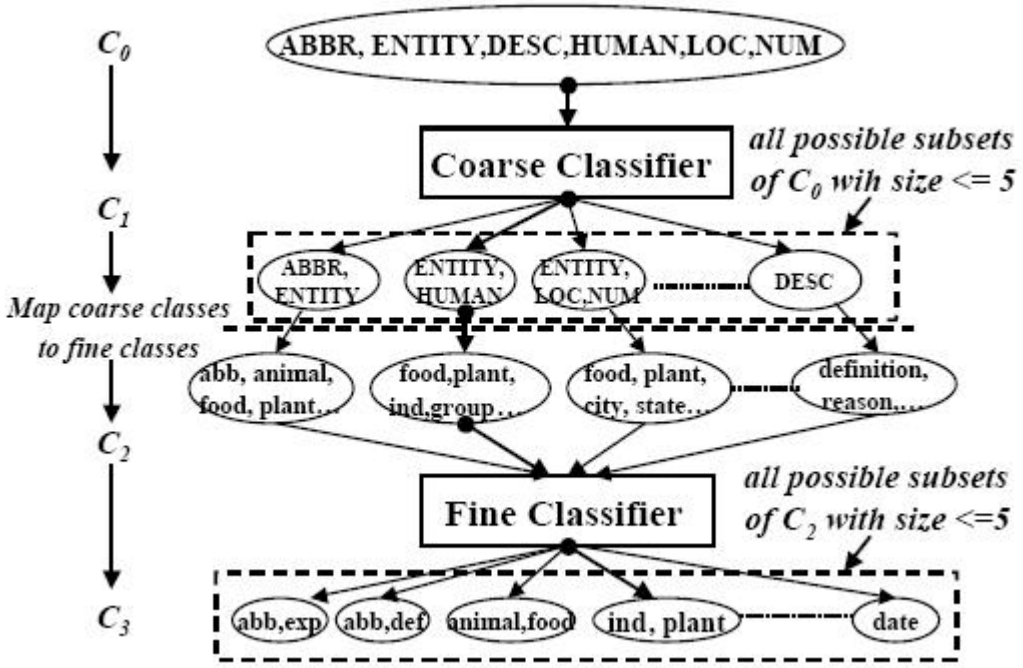

Figure 5.1: Li and Roth's hierarchical classifier

[Li & Roth (2002)] focused on several features: *words*, *pos tags*, *chunks* (non overlapping phrases), *named entities*, *head chunks* (e.g. the first noun chunk in a question) and *semantically related words* (words that often occur in a specific question type). They also applied hierarchical classifiers, in which a question is sequentially classified by two classifiers: *coarse classifier* and *fine classifier*. [Li & Roth (2002)] called the set of all

possible class labels for a given question a *confusion set*. The initial confusion set of any question is $C_0 = \{c_1, c_2, \ldots, c_n\}$, the set of all the coarse classes. The coarse classifier determines a set of preferred labels $C_1 = Coarse\_Classifier(C_0)$, $C_1 \subseteq C_0$ so that $|C_1| \leq 5$. Each coarse class label[1] in C_1 is then expanded to a fixed set of fine classes according to the class hierarchy. Concretely, suppose the coarse class c_i is mapped into the set $c_i = \{f_{i1}, f_{i2}, \ldots, f_{im}\}$ of fine classes, then $C_2 = \bigcup_{c_i \in C_1} c_i$. Finally, the fine classifier identifies a set of preferred labels of fine classes $C_3 = Fine\_Classifier(C_2)$ so that $C_3 \subseteq C_2$ and $|C_3| \leq 5$. The [Li & Roth (2002)]'s of the hierarchical classifier is depicted in Figure 5.1. The main purpose of their approach is to allow one question to be possible to belong to more than one classes. However, when C_1 (with $|C_1| \leq 5$) is mapped into C_2, the number of fine classes in C_2 may be still large. Therefore, their purpose is not the same as ours in which we try to reduce the number of classes for each classifier.

5.2.2 Semi-supervised Learning Algorithms

Co-training Algorithm

The co-training paradigm applies when accurate classification hypotheses for a task can be learned from either of two sets of features of the data, each called a view. The intuition behind [Blum & Mitchell (1998)]'s co-training algorithm is that two views of the data can be used to train two classifiers that can help each other. Each classifier is trained using one view of the labeled data, and then used to predicts labels for unlabeled examples. The most confident predictions of each classifier are selected and adding the corresponding examples with their predicted labels to the other's available training data. According to [Blum & Mitchell (1998)], the condition for co-training algorithm works exactly is that the two views have to satisfy the two assumptions: firstly each view itself is sufficient to label task (or is sufficient to build a good classifier); the second is independent assumption of the two views. In the original definition of co-training, [Blum & Mitchell (1998)] stated conditional independence of the views as a required criterion for co-training to work.

In particularly, suppose that each example is represented by a feature vector x drawn from a set of possible values (an instance space) X. The task is to learn a classification function $f : X \rightarrow Y$ where Y is a set of possible labels. The characteristics of co-training can be described as follows.

- The features can be separated into two types: $X = X_1 \times X_2$ where X_1 and X_2 correspond to two different views of an example. In the named entity task, X_1 might be the instance space for the spelling features, X_2 might be the instance space for the contextual features. By this assumption, each element $x \in X$ can also be represented as $(x_l, x_2) \in X_1 \times X_2$.
- Each view of the example is sufficient for classification. That is, there exist functions f_l and f_2 such that for any example $x = (x_l, x_2)$, $f(x) = f_l(X_l) = f_2(x_2)$. We never see an example $x = (x_l, x_2)$ in training or test data such that $f_l(x_l) \neq f_2(x_2)$.

Thus the method makes the fairly strong assumption that the features can be partitioned into two types such that each type alone is sufficient for classification (x_l and

[1] In this chapter, the terms: question class, question label, class label, class, label are used interchangeably

x_2 are not correlated too tightly). Now assume we have n pairs $(x_{l,i}, x_{2,i})$ drawn from $X_1 \times X_2$, where the first m pairs have labels y_i, whereas for $i = m+1, \ldots, n$ the pairs are unlabeled. In a fully supervised setting, the task is to learn a function f such that for all $i = 1, \ldots, m$, $f(x_{l,i}, x_{2,i}) = y_i$. In the co-training case, [Blum & Mitchell (1998)] argued that the task should be to induce functions f_l and f_2 such that:

1. $f_1(x_{1,i}) = f_2(x_{2,i}) = y_i$, for $i = 1, \ldots, m$
2. $f_1(x_{1,i}) = f_2(x_{2,i})$, for $i = m+1, \ldots, n$

So f_l and f_2 must (1) correctly classify the labeled examples, and (2) must agree with each other on the unlabeled examples. The key point is that the second constraint can be remarkably powerful in reducing the complexity of the learning problem. [Blum & Mitchell (1998)] gave an example that illustrates just how powerful the second constraint can be. Consider the case where $|X_1| = |X_2| = N$ and N is a "medium" sized number so that it is feasible to collect $O(N)$ unlabeled examples. Assume that the two classifiers are "rote learners": that is, f_l and f_2 are defined through look-up tables that list a label for each member of X_1 or X_2. The problem is a binary classification problem. The problem can be represented as a graph with $2N$ vertices corresponding to the members of X_1 and X_2. Each unlabeled pair $(x_{l,i}, x_{2,i})$ is represented as an edge between nodes corresponding to $x_{l,i}$ and $x_{2,i}$ in the graph. An edge indicates that the two features must have the same label. Given a sufficient number of randomly drawn unlabeled examples (i.e., edges), we will induce two completely connected components that together span the entire graph. Each vertex within a connected component must have the same label – in the binary classification case, we need a single labeled example to identify which component should get which label. The original co-training algorithm of [Blum & Mitchell (1998)] is presented in Figure 5.2.

---

co-training(L, U, K)
Input: The labeled set L;
The unlabeled set U;
The number of iteration K;
Output: The augmented labeled set L;
1. Create a pool U' of examples by choosing u examples at random from U;
2. $k \leftarrow 0$;
3. **repeat until** $k > K$
4. $k \leftarrow k + 1$;
5. use L to train a classifier h_1 that considers only the x_1 portion of x;
6. use L to train a classifier h_2 that considers only the x_2 portion of x;
7. allow h_1 to label p positive and n negative examples from U';
8. allow h_2 to label p positive and n negative examples from U';
9. add these self-labeled examples to L;
10. randomly choose $2p + 2n$ examples from U to replenish U';
11. **end repeat**

Figure 5.2: The original Co-training algorithm

Self-training Algorithm

This is also another approach of semi-supervised learning. In self-training, a single classifier is used to label questions in the unlabeled set to augment the labeled set for further training. The pseudo-code of the self-training algorithm is depicted in Figure 5.3 [Nigam & Ghani (2000)], where L, U are the labeled and unlabeled sets, correspondingly; θ is a threshold in the range of [0,1]; m is the number of iterations (m is 20 in our experiments); $Learn$ is a classification algorithm; U' is a subset of unlabeled questions ($U' \subseteq U$); L' is a set of questions that are labeled at each iteration. In the training loop, we select a pool U' of unlabeled questions smaller than U, as suggested by [Blum & Mitchell (1998)].

In each iteration, a subset U' of unlabeled questions is selected, and the set L' is created by selecting questions from U' which are predicted by the hypothesis h with confidence (prediction probability) greater than a threshold θ (θ is 0.9 in our experiments). The union of L' and L is used to train the classifier. Note that L' is not merged with L in each iteration. Instead, it is regarded as unlabeled questions, and put back into the unlabeled set U again. The training process terminates after m iterations.

---

self-training$(L, U, Learn, \theta, m)$
Input: The labeled set L;
The unlabeled set U;
The threshold θ;
The number of iteration m;
The learning algorithm $Learn$;

1. Create a subset U' by randomly selecting examples from U ;
2. $h \leftarrow Learn(L)$;
3. **repeat** m times
4. $L' \leftarrow \emptyset$;
5. **for** every $x \in U'$
6. **if** the prediction $h(x)$ has the confidence greater than θ **then**
7. $L' \leftarrow L' \cup \{(x, h(x))\}$;
8. **end for**
9. $h \leftarrow Learn(L \cup L')$;
10. Re-create the subset U' by randomly selecting examples from U ;
11. **end repeat**
12. **Output**: the learned hypothesis h;

Figure 5.3: The Self-training algorithm

The Tri-training Algorithm

This is also another semi-supervised learning algorithm proposed by [Zhou & Li (2005)]. In the Tri-training algorithm, three classifiers: h_1, h_2 and h_3 are initially trained from a set by bootstrap-sampling the labeled set L. For any classifier, an unlabeled instance can be labeled as long as the other two classifiers predict the same label. For example, if h_1 and h_2 agree on the labelling of an instance x in the unlabeled set U, then x can be labeled for h_3. Obviously, in this scheme, if the prediction of h_1 and h_2 on x is correct,

then h_3 will receive a valid new instance for further training; otherwise, h_3 will get an instance with a noisy label. Nonetheless, as claimed in [Zhou & Li (2005)], even in the worse case, the increase in the classification noise rate can be compensated for, if the number of newly labeled instances is sufficient.

---

tri-training(L,U,$Learn$)
Input: The labeled set L;
The unlabeled set U;
The classification algorithm $Learn$;

1. **for** $i \in \{1..3\}$ **do**
2. $S_i \leftarrow BootstrapSample(L)$;
3. $h_i \leftarrow Learn(S_i)$;
4. $e'_i \leftarrow 0.5$; $l'_i \leftarrow 0$;
5. **end for**
6. **repeat until** none of h_i ($i \in \{1..3\}$) changes
7. **for** $i \in \{1..3\}$ **do**
8. $L_i \leftarrow \emptyset$;$update_i \leftarrow FALSE$;
9. $e_i \leftarrow MeasureError(h_j \& h_k)$ (j,$k \neq i$);
10. **if** ($e_i < e'_i$) **then**
11. **for** every $x \in U$ **do**
12. **if** $h_j(x) = h_k(x)$ ($j, k \neq i$)
13. **then** $L_i \leftarrow L_i \cup \{(x, h_j(x))\}$;
14. **end for**
15. **if** ($l'_i = 0$) **then** $l'_i \leftarrow \lfloor \frac{e_i}{e'_i - e_i} + 1 \rfloor$;
16. **if** ($l'_i < |L_i|$) **then**
17. **if**($e_i |L_i| < e'_i l'_i$) **then** $update_i \leftarrow TRUE$;
18. **else if** $l'_i > \frac{e_i}{e'_i - e_i}$
19. **then** $L_i \leftarrow Subsample(L_i, \lceil \frac{e'_i l'_i}{e_i} - 1 \rceil)$;
20. $update_i \leftarrow TRUE$;
21. **end for**
22. **for** $i \in \{1..3\}$ **do**
23. **if** $update_i = TRUE$ **then**
24. $h_i \leftarrow Learn(L \cup L_i)$;$e'_i \leftarrow e_i$;$l'_i \leftarrow |L_i|$;
25. **end for**
26. **end repeat**
27. **Output**:$h(x) \leftarrow \arg\max_{y \in label} \sum_{i:h_i(x)=y} 1$;

Figure 5.4: The original Tri-training algorithm

Also in the algorithm, each classifier is initially trained from a data set generated by bootstrap-sampling the original labeled set, in order to make classifiers diverse. If all the classifiers are identical, then for any of three classifiers, the unlabeled instances labeled by the other two classifiers will be the same as those labeled by itself, thus, Tri-training becomes *self-training* with a single classifier. The pseudo-code of the algorithm is described in Figure 5.4, where $Learn$ is a classification algorithm; S_i is a labeled set bootstrap-sampled from the labeled set L. e'_i is the error rate of h_i in the $(t\text{-}1)^{th}$ round.

With the assumption that the beginning error rate is less than 0.5, therefore e'_i is initially set to 0.5; e_i is the error rate of h_i in the t^{th} round; L_i is the set of instances that are labeled for h_i in the t^{th} round; l'_i is the size of L_i at (t-1)th round, and in the first round it is estimated by $\lfloor \frac{e_i}{e'_i - e_i} + 1 \rfloor$; $Subsample(L_i, s)$ function randomly removes $|L_i| - s$ number of instances from L_i in order to make current round have better performance than that of the previous round, as proved in [Zhou & Li (2005)]; $MeasureError(h_j \& h_k)$ function attempts to estimate the classification error rate of the hypothesis derived from the combination of h_j and h_k. Because it is difficult to estimate the classification error rate on the unlabeled instances, the algorithm only estimates on the labeled set with the assumption that both the labeled and unlabeled instance sets have the same distribution. In each iteration, L_i is not merged with the original labeled set L. It is put into the unlabeled set U as unlabeled instances.

The interesting point in the Tri-training algorithm is that, in order to ensure that the current round of training has better performance than that of the previous round, the size of each newly labelled set L_i must not be greater than $\lceil \frac{e'_i l'_i}{e_i} - 1 \rceil$. If it is greater than this value, the function $Subsample(L_i, s)$ is used to randomly remove redundant instances. The three classifiers are refined in the training process, and the final hypothesis is produced via *majority voting*. For the sake of saving space, other details can be seen in [Zhou & Li (2005)].

5.3 Modifications of Tri-training Algorithm

In this section, we describe the problem of the original Tri-training algorithm when applying to question classification, and give two proposals to improve it.

Due to its nature, question data type is very sparse and imbalanced as shown in Table 5.2. As stated in [Joachims (1998)], text data type, when represented in the vector space model, is very sparse. For each document, the corresponding document vector contains only a few entries which are non-zero. A question contains quite a few words in comparison with a document, so question data is even more sparse than text data. Because of the imbalance, after bootstrap-sampling, each newly created labeled set misses a number of questions as compared to the original labeled set. If the missed questions are in a class which contains only few questions, then the initial error rate of each classifier increases when being trained from these data sets. The final improvement after learning sometimes does not compensate for this problem. In order to avoid this drawback, we propose to use more than one algorithm for the three classifiers. Each classifier is initially trained on the labeled set. Our experiments showed that, if the performance of one of the three classifiers is much better (or worse) than that of the others, the final result is not improved. For this reason, a constraint on three classifiers is that their performances are similar. The modified version is depicted in Figure 5.5, where $Learn_i$ stands for different algorithms.

Another proposal to avoid bootstrap-sampling is to use more than one views, such as two or three views in the learning process, so that each classifier can be trained from the original labeled set with different feature spaces while still making sure that they are diverse enough. The modified algorithm seems to have the standard Co-training style in the framework of Tri-training. The modified version according to this proposal is given

tri-training(L,U,$Learn_1, Learn_2, Learn_3$)
Input: The labeled set L;
The unlabeled set U;
Different classification algorithms $Learn_1, Learn_2, Learn_3$;

1. **for** $i \in \{1..3\}$ **do**
2.
3. $h_i \leftarrow Learn_i(L)$;
4. $e'_i \leftarrow 0.5$; $l'_i \leftarrow 0$;
5. **end for**
6. **repeat until** none of h_i ($i \in \{1..3\}$) changes
7. **for** $i \in \{1..3\}$ **do**
8. $L_i \leftarrow \emptyset$;$update_i \leftarrow FALSE$;
9. $e_i \leftarrow MeasureError(h_j \& h_k)$ (j,$k \neq i$);
10. **if** ($e_i < e'_i$) **then**
11. **for** every $x \in U$ **do**
12. **if** $h_j(x) = h_k(x)$ $(j, k \neq i)$
13. **then** $L_i \leftarrow L_i \cup \{(x, h_j(x))\}$;
14. **end for**
15. **if** ($l'_i = 0$) **then** $l'_i \leftarrow \lfloor \frac{e_i}{e'_i - e_i} + 1 \rfloor$;
16. **if** ($l'_i < |L_i|$) **then**
17. **if**($e_i |L_i| < e'_i l'_i$) **then** $update_i \leftarrow TRUE$;
18. **else if** $l'_i > \frac{e_i}{e'_i - e_i}$;
19. **then** $L_i \leftarrow Subsample(L_i, \lceil \frac{e'_i l'_i}{e_i} - 1 \rceil)$;
20. $update_i \leftarrow TRUE$;
21. **end for**
22. **for** $i \in \{1..3\}$ **do**
23. **if** $update_i = TRUE$ **then**
24. $h_i \leftarrow Learn_i(L \cup L_i)$; $e'_i \leftarrow e_i$;$l'_i \leftarrow |L_i|$;
25. **end for**
26. **end repeat**
27. **Output**:$h(x) \leftarrow \arg\max_{y \in label} \sum_{i:h_i(x)=y} 1$;

Figure 5.5: Tri-training with multiple learning algorithms

in Figure 5.6, where $view_i(L)$ is the i^{th} view of the data set L. One important aspect of Tri-training algorithm is the needless of redundant views, so it can be applied to problems which have only one view. In this domain, it is easy to get redundant views, that is the reason of this proposal.

5.4 Question Taxonomy and Hierarchical Classifiers

The TREC's question answering track [Voorhees (1999), Voorhees (2000), Voorhees (2001)] defined six coarse classes, namely, *abbreviation*, *description*, *entity*, *human*, *location* and *numeric*. Almost TREC's QA systems rely on this set of question classes. Other QA systems [Hovy, *et al.* (2001), Ittycheriah, *et al.* (2001), Singhal, *et al.* (1999)] rely on the set of question classes whose size is less than 20. For example, [Singhal, *et al.* (1999)]

tri-training(L,U,$Learn_1, Learn_2, Learn_3$)
Input: The labeled set L;
The unlabeled set U;
Different classification algorithms $Learn_1, Learn_2, Learn_3$;

1. **for** $i \in \{1..3\}$ **do**
2.
3. $h_i \leftarrow Learn_i(view_i(L))$;
4. $e'_i \leftarrow 0.5$; $l'_i \leftarrow 0$;
5. **end for**
6. **repeat until** none of h_i ($i \in \{1..3\}$) changes
7. **for** $i \in \{1..3\}$ **do**
8. $L_i \leftarrow \emptyset$;$update_i \leftarrow FALSE$;
9. $e_i \leftarrow MeasureError(h_j \& h_k)$ $(j,k \neq i)$;
10. **if** $(e_i < e'_i)$ **then**
11. **for** every $x \in U$ **do**
12. **if** $h_j(x) = h_k(x)$ $(j, k \neq i)$;
13. **then** $L_i \leftarrow L_i \cup \{(x, h_j(x))\}$;
14. **end for**
15. **if** $(l'_i = 0)$ **then** $l'_i \leftarrow \lfloor \frac{e_i}{e'_i - e_i} + 1 \rfloor$;
16. **if** $(l'_i < |L_i|)$ **then**
17. **if**$(e_i |L_i| < e'_i l'_i)$ **then** $update_i \leftarrow TRUE$;
18. **else if** $l'_i > \frac{e_i}{e'_i - e_i}$
19. **then** $L_i \leftarrow Subsample(L_i, \lceil \frac{e'_i l'_i}{e_i} - 1 \rceil)$;
20. $update_i \leftarrow TRUE$;
21. **end for**
22. **for** $i \in \{1..3\}$ **do**
23. **if** $update_i = TRUE$ **then**
24. $h_i \leftarrow Learn_i(view_i(L \cup L_i))$; $l'_i \leftarrow |L_i|$;
25. **end for**
26. **end repeat**
27. **Output**:$h(x) \leftarrow \arg\max_{y \in label} \sum_{i:h_i(x)=y} 1$;

Figure 5.6: Tri-training with multiple learning algorithms and views

used a set of simple answer entity types consisting of *person*, *location*, *organization*, *date*, *quantity*, *duration* and *linear measure*. Obviously, with a finer-grained set of question classes a system can be more accurate to locate answers.

[Li & Roth (2002)] proposed a two-layer taxonomy in which a coarse-grained class is further expanded into a number of fine-grained classes as shown in Table 5.1. In this taxonomy, the maximum number of fine classes of coarse classes is 22 (in *entity* coarse class).

As stated in [Li & Roth (2002)], there is ambiguity while classifying question. Because there is no completely clear boundary between question classes, thus, a question can be assigned to more than one class. For example, the question "*What do ladybugs eat?*" can be assigned to *food*, *plant* or *animal* fine classes. Nonetheless, each question in the data sets created in TREC [Voorhees (1999), Voorhees (2000), Voorhees (2001)] is assigned a single label, e.g. the label *food* for above question. For this reason, we assume that one

Coarse	Fine class	#
ABBR	abbreviation, expansion	2
DESC	definition, description, manner, reason	4
ENT	animal, body, color, creation, currency, decease/medical, event, food, instrument, language, latter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word	22
HUM	description, group, individual, title	4
LOC	city, country, mountain, other, state	5
NUM	code, count, date, distance, money, order, other, percent, period, speed, temperature, size, weight	13

Table 5.1: Question class taxonomy

question can belong to only one question class. In other words, we select the label with highest probability among possible labels to assign to unlabeled questions.

As briefly described in Section 5.1, we employed a hierarchical architecture (in the form of a tree) of classifiers. According to the question taxonomy listed in Table 5.1, we construct a hierarchical classifiers as depicted in Figure 5.7. The *coarse classifier* lies in the first level while six *fine classifiers*, namely FC_1, FC_2,..., FC_6 lie in the second level of the hierarchy. In this hierarchy, a question is sequentially classified by the coarse classifier and a fine classifier.

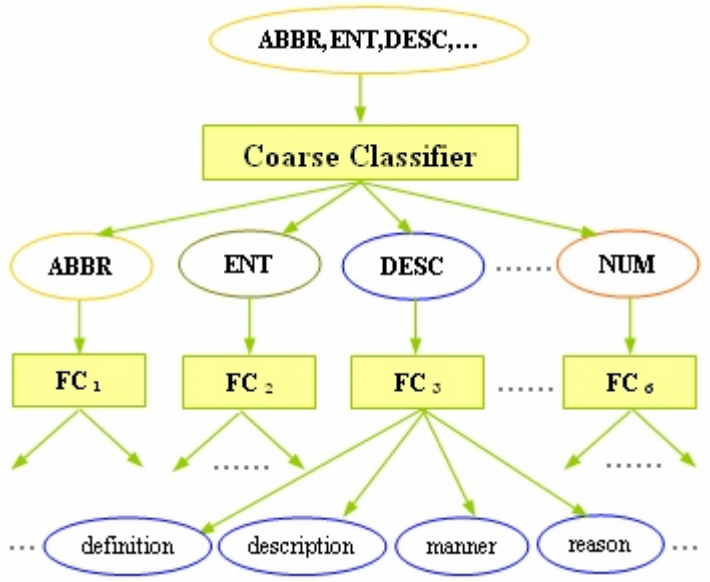

Figure 5.7: System architecture

5.5 Hierarchical Classifiers and Semi-supervised Learning Combination

We use three ways to apply learning methods for classifiers in the system as described in next three subsections.

5.5.1 Supervised Learning Application

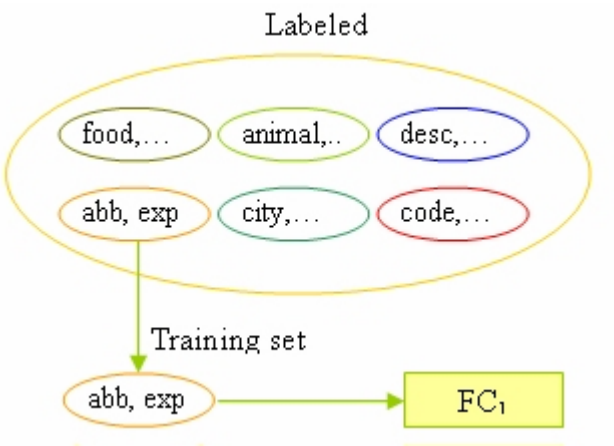

Figure 5.8: Training data for fine classifier FC_1 in supervised learning

The first way of employing classifiers is to use supervised learning for all classifiers. It is easy to create training set for the coarse classifier. Each of the six fine classifiers needs a subset of the labeled set. Thus, we have to extract the corresponding subset for training each fine classifier. For example, for creating the training set for FC_1 classifier, we select questions having the label of either *abb* or *exp* as depicted in Figure 5.8.

5.5.2 Supervised and Semi-supervised Learning Combination Application

With the purpose to consume unlabeled questions to in the learning process, the next proposal is to apply a semi-supervised learning for the first level classifier. Other classifiers still follow supervised learning approach. The method for generating training sets for fine classifiers is the same as that of Section 5.5.1.

5.5.3 Semi-supervised Learning Application

The third approach we try to apply the semi-supervised learning for classifiers at all levels. The method of getting unlabeled questions for fine classifiers must be taken into account. Because each fine classifier FC_i (i=1..6) performs on only a subset of fine classes, while the unlabeled questions can belong to any fine class. We tried two methods for using

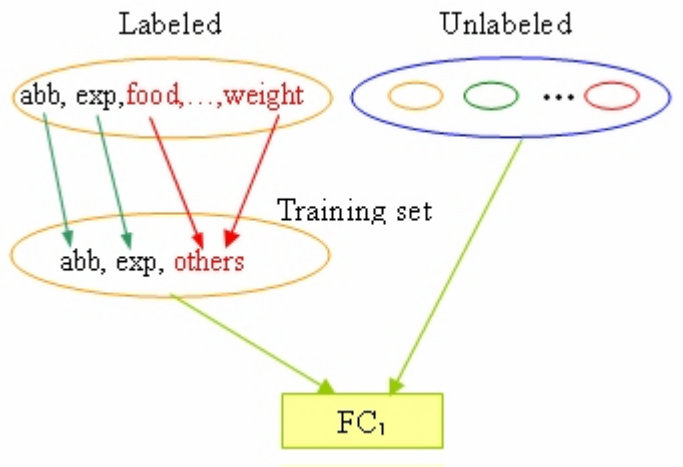

Figure 5.9: Training data for fine classifier FC_1 by adding one additional class label in semi-supervised learning

unlabeled questions. In the first method, for each training set of FC_i we add one additional class label. For example, for FC_1, we add one more class '*others*' to the set of classes $\{abb, exp\}$ to create a new class set $\{abb, exp, others\}$ for training the classifier. Questions having labels different from *abb* and *exp* are mapped into the label '*others*' as depicted in Figure 5.9.

The second method for applying semi-supervised learning for each fine classifier: the training sets for fine classifiers are created in the same way as that of Section 5.5.1; the unlabeled set for FC_i is created by using the coarse classifier to classify the unlabeled set to get questions having the coarse label where the fine classifier is attached. The method for creating these sets for FC_1 is depicted in Figure 5.10.

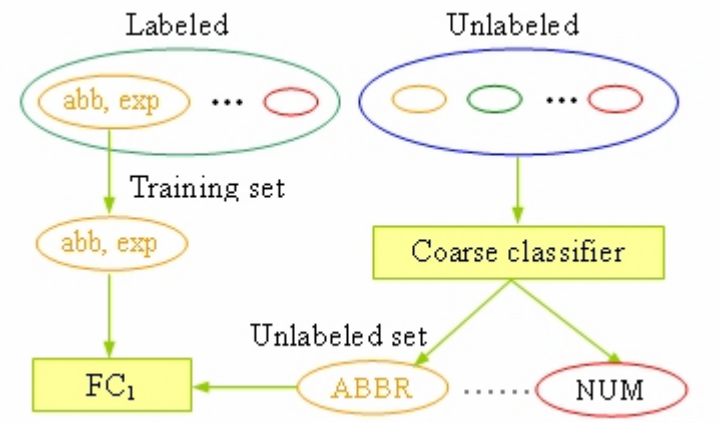

Figure 5.10: Unlabeled data for fine classifier FC_1 by getting the result of the coarse classifier in semi-supervised learning

Because semi-supervised learning for fine classifiers is not so good as shown in Section 5.9, we do not apply semi-supervised learning for fine classifiers and supervised learning for coarse classifier.

5.6 Question Hierarchy Expansion

In this section, we consider nodes consisting of a large number of classes in the taxonomy. It is worth expanding the taxonomy by dividing these nodes into smaller groups in order to gain precision. A possible solution is to group classes that are closed to each other in a certain distance measure. For this reason, we take clustering method into account. Each class label can be seen as a vector of frequency of words appearing in the questions belonging to this class.

5.7 Question Dataset and Feature Selection

This section gives details about the dataset, feature selection for the experiments.

5.7.1 Question Dataset

Table 5.2: Question distribution. #Tr and #Te are the number of labeled and testing questions.

Class	#Tr	#Te	Class	#Tr	#Te	Class	#Tr	#Te
ABBREV.	**86**	**9**	letter	9	0	country	155	3
abb	16	1	other	217	12	mountain	21	3
exp	70	8	plant	13	5	other	464	50
DESC.	**1162**	**138**	product	42	4	state	66	7
definition	421	123	religion	4	0	NUMERIC	**896**	**113**
description	274	7	sport	62	1	code	9	0
manner	276	2	substance	41	15	count	363	9
reason	191	6	symbol	11	0	date	218	47
ENTITY	**1250**	**94**	technique	38	1	distance	34	16
animal	112	16	term	93	7	money	71	3
body	16	2	vehicle	27	4	order	6	0
color	40	10	word	26	0	other	52	12
creative	207	0	HUMAN	**1223**	**65**	period	27	8
currency	4	6	group	47	6	percent	75	3
dis.med.	103	2	individual	189	55	speed	9	6
event	56	2	title	962	1	temp	8	5
food	103	4	description	25	3	size	13	0
instrument	10	1	LOCATION	**835**	**81**	weight	11	4
lang	16	2	city	129	18			

We follow [Li & Roth (2002)]'s proposal to classify questions into 50 fine-grained classes. In the experiments, the data sets were those used in [Li & Roth (2002)] with the total of about 6000 questions (the exact number is 5952), of which 500 questions from TREC 10 [Voorhees (2001)] were the test set, and 4 subsets of size 1000, 2000, 3000 and 4000 were created by randomly selecting from other 5500 questions[2]. We used the 4 sub-

[2]These data sets are all available on http://L2R.cs.uiuc.edu/∼cogcomp/

sets as labeled sets, and created 4 correspondingly unlabeled sets by selecting questions that do not belong to the labeled sets.

The distribution of training and testing data is shown in Table 5.2, where the coarse classes are in capitals, followed by the corresponding fine classes. As listed in the table, some classes consist of few questions, such as 4 questions in the *currency* and *religion* classes.

5.7.2 Feature Selection

In experiments, we used two primitive feature types which were automatically extracted for each question, namely, bag-of-word and bag-of-pos&word.

Question classification is a little different from text classification, because a question contains a small number of words, while a document can have a large number of words. In text classification, common words like 'what', 'is', etc. are considered to be "*stop-words*" and omitted as a dimension reduction step in the process of creating features. This is an important step in improving the performance of classification as proven in [Joachims (1998)]. However, these words are very important for question classification. Also, word frequencies play an important role in document classification, whereas those frequencies are usually equal to 1 in a question, thus, they do not significantly contribute to the classification precision. In order to keep these words while still reducing the dimension space, we used a preprocessing step: all verbs were restored into their infinitive forms. For example, the verb forms 'is', 'were', 'was', 'are' and 'am' were converted to 'be'; plural nouns were changed to their singular forms, such as 'children' was converted to 'child'; words having the CD (*cardinal number*) part-of-speech were made the same value, such as '1998', '2000', '12' were changed into '100'. Given the question:

Who was President of Afghanistan in 1994?

After the reduction step, it becomes:

Who be President of Afghanistan in 100?

After the reduction step, the vector (or vocabulary) V of all distinct words of questions in the corpus was constructed. Let the size of V be N, then each question q was converted into a vector $(q_1, q_2, \ldots, q_N)$, where q_i is 1 if the word w_i in V appears in q, otherwise q_i is 0. These vectors of numbers were the input of classifiers.

Interestingly, this dimension reduction step makes SVM reach the precision of 81.4% training on 5500 questions, while the same features with SVM used in [Zhang & Lee (2003)] gives the precision of 80.2% training on the same data set and with the same *linear* kernel.

For bag-of-pos&word features, each *word* in a question was converted into the form of *POS-word*, where *POS* is the part-of-speech tag of *word*. We also used the preprocessing step similarly to what applied to the process to generate bag-of-word features, for example 'how' was transformed into 'WRB-how', 'who' was converted to 'WP-who', 'are', 'is', 'am', 'were' and 'was' were converted to 'AUX-be', etc. Given the question:

Who was President of Afghanistan in 1994?

After the reduction step, it becomes:

WP-Who AUX-be NN-president IN-of NN-Afghanistan IN-in CD-100?

The process of converting questions into vectors of numbers was similar to that of

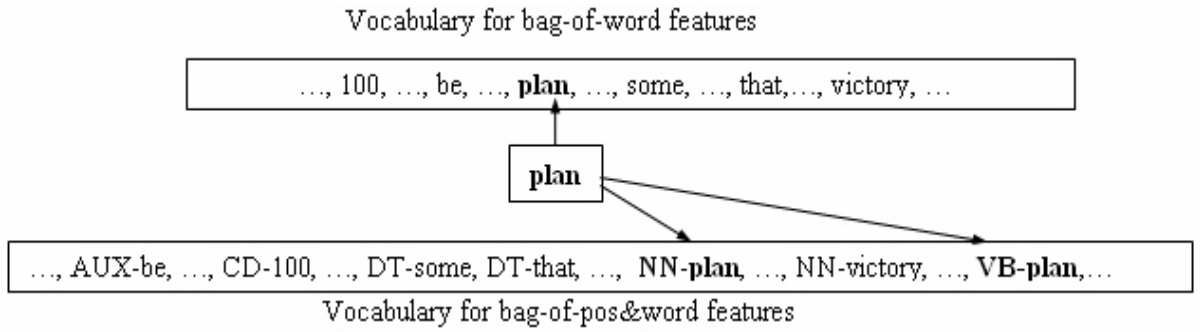

Figure 5.11: The difference between bag-of-word and bag-of-pos&word features

bag-of-word features. There is a difference between bag-of-word and bag-of-pos&word features. A word, such as 'plan' may play different roles in different questions. It can be a verb in this question while being a noun in another one. The role of the word can be distinguished in bag-of-pos&word features, because it is converted into 'VB-plan' (if it is a verb) or 'NN-plan'(if it is a noun) as depicted in Figure 5.11. The bag-of-word features do not have this ability, so the bag-of-pos&word features provide a richer set of features. Concretely, for the dataset used in our experiments, the size of the vocabulary V for bag-of-word and bag-of-pos&word features is 7953 and 9876, respectively. Thus, bag-of-pos&word features may make classification algorithms perform better than bag-of-word features.

We tested the supervised learning with SVM algorithm on the labeled set of size 4000 with bag-of-word and bag-of-pos&word features. The statistics is recorded in Table 5.3, where $\#T$ shows the number of test questions belonging to each question class; $\#W$ and $\#P$, respectively, show the correctly predicted questions of each question class with bag-of-word and bag-of-pos&word; $\%W$ and $\%P$, respectively, are precisions of classification with bag-of-word and bag-of-pos&word. The table shows that SVM fails to classify some question classes, such as *currency*, *event* or *product* with bag-of-word and bag-of-pos&word features. SVM fails to classify the *currency* class because in the labeled set of size 4000, there is only one question belonging to the class *currency*. Another possible reason that make SVM fails to correctly classify other classes is the lack of semantics of bag-of-word and bag-of-pos&word as seen in the three questions from the labeled set:

+ *What is a fear of shadows?* in the class *ENTITY:disease.medicine*.
+ *What is the origin of head lice?* in the class *DESCRIPTION:description*.
+ *What is the nickname for the state of Mississippi?* in the class *LOCATION:state*.

Though these three questions belong to different classes, they have relatively similar forms. This causes ambiguity for classification algorithms. For improving classification

precision, semantic features should be added, such as class-specific *related words* used in [Li & Roth (2002)]. For each question class, class-specific *related words* are a list of words that frequently appear in this class. With this method, a word in a question may have both syntactic and semantic roles, thus the feature is better, and the classification precision is improved.

Table 5.3: Precision of classification of SVM with bag-of-word and bag-of-pos&word features

Class	#T	#W	%W	#P	%P	Class	#T	#W	%W	#P	%P
abb	1	1	100	1	100	term	7	7	100	7	100
exp	8	6	75	6	75	vehicle	4	1	25	1	25
definition	123	123	100	123	100	HUM:desc	3	3	100	3	100
description	7	6	85.7	6	85.7	group	6	3	50	3	50
manner	2	2	100	2	100	individual	55	52	94.5	53	96.7
reason	6	5	83.3	5	83.3	title	1	0	0	0	0
animal	16	8	50	9	56.3	city	18	15	83.3	14	77.8
body	2	1	50	2	100	country	3	3	100	3	100
color	10	10	100	10	100	mountain	3	2	66.7	2	66.7
currency	6	0	0	0	0	LOC:other	50	41	82	41	82
dis.med	2	0	0	1	50	state	7	7	100	7	100
event	2	0	0	0	0	count	9	9	100	9	100
food	4	1	25	1	25	date	47	44	93.6	44	93.6
instrument	1	1	100	1	100	distance	16	9	56.3	8	50
lang	2	2	100	2	100	money	3	0	0	0	0
ENT:other	12	6	50	5	41.7	NUM:other	12	5	41.7	5	41.7
plant	5	1	20	1	20	percent	3	0	0	1	33.3
product	4	0	0	0	0	period	8	7	87.5	7	87.5
sport	1	1	100	1	100	speed	6	3	50	3	50
substance	15	6	40	5	33.3	temp	5	0	0	0	0
technique	1	1	100	1	100	weight	4	1	25	1	25
						TOTAL	500	393	78.6	395	79

5.8 Experiments with Tri-training and Its Modifications for Fine Question Classes

This section gives details about the implementation and evaluation with the Tri-training and our modified Tri-training algorithms[3]. Because the function *Subsample*(.) (in line 20 of Figure 5.4, 5.5 and 5.6) uses randomness to remove redundant questions, so the set L_i generated for each h_i may be different in each run; the final result of each run may be different, and the result of the first run is not always the best one. Thus, in experiments about Tri-training, each algorithm was run 4 times and the best as well as the average results were recorded.

[3]Our implementation of Tri-training algorithms was based on the existing implementation of Tri-training's authors at http://lamda.nju.edu.cn/datacode/tritrain/tritrain.htm

5.8.1 Experiments with Multiple Classifiers

In the first experiment, we developed our programs based on the Sparse Network of Winnows (SNoW) learning architecture[4] [Carlson, *et al.* (1999)], which implements three learning algorithms: Perceptron, Bayes and Winnow. We used these three learning algorithms to apply for the three classifiers of the Tri-training algorithm. Besides, we implemented the original Tri-training algorithm with a single classification algorithm, such as Bayes, Perceptron or Winnow. All the parameters of these algorithms, such as the learning rate α, threshold and the initial weight of Perceptron and Winnow were default values. The bag-of-word features were used in the experiment.

The best and average precision (of 4 runs) of the experiment is listed in Table 5.4, where TB, TP and TW respectively stand for the original Tri-training with a single classification algorithm Bayes, Perceptron and Winnow; TBPW stands for the modified Tri-training with Bayes, Perceptron and Winnow following the algorithm depicted in Figure 5.5. For the original Tri-training with a single classification algorithm Bayes, Perceptron or Winnow, we compare their precision with the baseline produced by the correspondingly supervised learning algorithm being trained on the same labeled set. For example, the baseline of the original Tri-training with Bayesian algorithm is the precision of the supervised learning of Bayes on the same labeled set. For our modified algorithm TBPW, we compared its precision with the best precision of individually supervised learning of the three classifiers (values in *italic*) as the baseline. We also carried out the sign test [Kanji (1994)] for our modified Tri-training algorithm, with a total number of 25 subsets at the 95% significance level (p=0.05), in which the corresponding critical value is 7. The column 'N' shows the number of tests on subsets in which the precision of semi-supervised learning is less than the baseline. According to the sign test theory, a test is significant if the value in the column 'N' is less than or equal the critical value. The sign test shows that our algorithm is significant at the level of 95% for all tests.

The results show that the precision of supervised learning of Bayes, Perceptron and Winnow is not sensitive to the size of labeled sets. Concretely, when the size of the labeled set increases, the corresponding precision does not increase. Maybe, question data type and bag-of-word features are not suitable for these learning algorithms.

In the second experiment, we used two algorithms: Maximum Entropy Model[5](MEM), and SVM[6] which has been proven to perform well for text classification [Joachims (1998)]. The selection of MEM is based on our investigation. It has better performance than Bayes, Winnow and Perceptron. In this domain, SVM classifier has better performance than that of MEM classifier, thus, we used two SVM classifiers and one MEM classifier in the implementation with the expectation of making two SVM classifiers to have high degree of decision on final hypothesis. With SVM classifiers, we used *linear* kernel, and other parameters (e.g., parameter C) were default. In this domain, other kernels of SVM, such as *polynomial*, *radial basic function* or *sigmoid*, give poor performance. For MEM classifier, we used Gaussian smoothing, and all default values of parameters (e.g., L-BFGS parameter estimation). Bag-of-word features were used for all classifiers. In this configuration, the two SVM classifiers are identical at the beginning. In the learning loop, because of the randomness, the *Subsample*(.) procedure (in the line 20 of the algorithm

[4]The software is freely available at http://L2R.cs.uiuc.edu/~cogcomp/software.php

[5]We used a free open source implementation of Maximum Entropy Model available at http://homepages.inf.ed.ac.uk/s0450736/pmwiki/pmwiki.php

[6]We used a free implementation of SVM available at http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/

Table 5.4: The best and average precision (%) of the original Tri-training with single algorithm (TB, TP and TW) and the modified Tri-training with Bayes, Perceptron and Winnow (TBPW)

The best precision								
	Bayes		Perceptron		Winnow		Mod. TriTraining	
#	Base.	TB	Base.	TP	Base.	TW	TBPW	N
1000	59.8	58.0	*60.2*	60.4	58.0	60.4	**65.8**	0
2000	58.4	58.0	*67.2*	67.8	67.0	64.8	**68.8**	1
3000	57.2	56.4	*68.4*	70.0	49.4	65.4	**72.0**	2
4000	51.8	51.8	66.4	65.8	*71.6*	71.4	**72.0**	6

The average precision							
	Bayes		Perceptron		Winnow		Mod. TriTraining
#	Base.	TB	Base.	TP	Base.	TW	TBPW
1000	59.8	55.85	*60.2*	60.15	58.0	59.85	**64.15**
2000	58.4	57.80	*67.2*	66.80	67.0	64.15	**68.60**
3000	57.2	56.30	*68.4*	69.35	49.4	65.00	**70.35**
4000	51.8	51.65	66.4	65.50	*71.6*	69.65	69.70

in Figure 5.5) creates different L_i sets for the two SVM classifiers. As the results, the two SVM classifiers have different hypotheses when they are re-trained (in line 24 of the algorithm in Figure 5.5.

Table 5.5 shows the best and the average precision (of 4 runs) of different algorithms, where TSW and TMW, respectively, stand for the original Tri-training algorithm with SVM and MEM algorithms; TSSM stands for the modified Tri-training with two SVM classifiers and a MEM classifier following the algorithm described in Figure 5.5. We used the precision of supervised learning with MEM and SVM on the same labeled sets as the baseline to compare with the precision of TMW and TSW. For TSSM, we selected the best precision of supervised learning with MEM and SVM on the same labeled sets (values in *italic*) as the baseline. Similar to our first experiment, we carried out the sign test on 25 subsets and at the significance level 95%. The column 'N' records the number of tests on subsets, in which the precision of semi-supervised is less than the baseline. Except for the test on the labeled set size of 1000 which is not improved, our other tests are significant at the level of 95%.

As shown in the table, MEM and SVM are sensitive to the size of the labeled sets. The precision is increased when the size of labeled set increases. This indicates that MEM and SVM are suitable for question data with bag-of-word features.

5.8.2 Experiments with Two Different Algorithms and Two Views

In the third experiment, we implemented the second proposal of using more than one views following the algorithm described in Figure 5.6. In theory, we can use three different algorithms with distinct views, however, our primary purpose is to make the three classifiers diverse at the initial step, so two different algorithms, two views and a suitable assignment of views to classifiers are sufficient. Concretely, among the three classifiers, two of them were SVM classifiers and the third one was a MEM classifier. The first view (feature space) was bag-of-word, and the second view was bag-of-pos&word. We set two

Table 5.5: The best and average precision (%) of the original Tri-training with single MEM, SVM algorithm (TMW and TSW) and the modified Tri-training with both MEM and SVM (TSSM)

The best precision						
	MEM		SVM		Mod. Tritraining	
#	Base.	TMW	Base.	TSW	TSSM	N
1000	67.6	68.0	*68.4*	67.6	68.4	-
2000	74.8	75.2	*75.6*	76.2	**76.4**	4
3000	76.8	76.4	*78.2*	78.4	**78.6**	6
4000	77.2	78.2	*78.6*	78.6	**78.8**	7

The average precision					
	MEM		SVM		Mod. Tritraining
#	Base.	TMW	Base.	TSW	TSSM
1000	67.6	67.40	*68.4*	66.95	68.25
2000	74.8	74.10	*75.6*	75.75	**76.10**
3000	76.8	76.20	*78.2*	78.00	78.20
4000	77.2	77.40	*78.6*	78.50	78.50

SVM classifiers two different views, while the MEM classifier used either of them. Concretely, the first SVM classifier used bag-of-word features, the second SVM classifier used bag-of-pos&word features and the MEM classifier used bag-of-word features.

Table 5.6: The best precision (%) of the original Tri-training with single algorithm (TMW, TMP, TSW and TSP) and the modified Tri-training with MEM, SVM with two views (TSSM2)

	MEM-word		MEM-pos		SVM-word		SVM-pos		Two views	
#	Base.	TMW	Base.	TMP	Base.	TSW	Base.	TSP	TSSM2	N
1000	67.6	68.0	68.8	**69.0**	68.4	67.6	*69.2*	66.6	68.4	-
2000	74.8	75.2	75.4	74.2	*75.6*	76.2	75.2	74.6	**76.0**	5
3000	76.8	76.4	76.8	76.2	*78.2*	78.4	77.0	77.0	**79.0**	3
4000	77.2	78.2	77.8	77.8	78.6	78.6	*79.0*	78.4	**80.4**	2

Let TMW and TMP be the original Tri-training algorithm with MEM using bag-of-word and bag-of-pos&word features, respectively; Let TSW and TSP respectively be the original Tri-training with SVM using bag-of-word and bag-of-pos&word features; Let TSSM2 be the modified Tri-training with two SVM and a MEM classifiers following the algorithm described in Figure 5.6 using two views: bag-of-word and bag-of-pos&word. For TMW, TMP, TSW and TSP, the baseline is the precision of supervised learning with corresponding algorithm and feature space. The best precision of the experiment is given in Table 5.6. The sign test similar to previous experiments is also carried out. Except for the test with the size of 1000, the other tests are significant at the level of 95%.

We recorded the average precision (of 4 runs) of each algorithm of the experiment in Table 5.7. Table 5.8 recorded the number of new questions (Li) added for each classifier in each iteration of the experiment in Table 5.6, where 'Iter.' stands for iteration. The average values (in 4 tests) of these Li are recorded in Table 5.9. In these experiments, TMW, TMP, TSW and TSP took two iterations while TSSM2 took at most two iterations. The initial classifiers were very different because of the use of function $BootstrapSample(.)$

Table 5.7: The average precision (%) of the original Tri-training with single algorithm (TMW, TMP, TSW and TSP) and the modified Tri-training with MEM, SVM with two views (TSSM2)

	MEM-word		MEM-pos		SVM-word		SVM-pos		Two views
#	Base.	TMW	Base.	TMP	Base.	TSW	Base.	TSP	TSSM2
1000	67.6	67.40	68.8	68.55	68.4	66.95	*69.2*	66.30	67.90
2000	74.8	74.10	75.4	73.55	*75.6*	75.75	75.2	74.30	**75.85**
3000	76.8	76.20	76.8	75.90	*78.2*	78.00	77.0	76.85	**78.45**
4000	77.2	77.40	77.8	77.45	78.6	78.50	*79.0*	78.30	**79.65**

Table 5.8: The size of L_i in each round corresponding to the experiment in Table 5.6

#	Iter.	TMW			TMP			TSW		
		L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	L_3
1000	1	30	30	5	42	42	5	26	30	30
	2	4262	4262	4452	4122	4122	4452	489	492	496
2000	1	50	50	6	32	47	30	34	23	28
	2	3199	3199	3452	3198	3216	3309	489	486	493
3000	1	41	36	48	54	81	42	41	33	32
	2	1491	1486	1471	2294	2351	2316	748	732	734
4000	1	40	41	43	65	47	55	46	39	42
	2	648	245	196	990	665	332	391	395	390

#	Iter.	TSP			TSSM2		
		L_1	L_2	L_3	L_1	L_2	L_3
1000	1	32	23	27	3226	499	499
	2	4256	4258	4236	-	-	-
2000	1	29	24	31	999	499	499
	2	984	974	975	-	-	-
3000	1	29	46	32	187	750	187
	2	1497	1497	1480	373	373	0
4000	1	29	39	37	222	399	199
	2	993	997	986	-	-	-

in Line 2 of Figure 5.6, however after having been re-trained in Line 24 of Figure 5.6, the three classifiers became very similar, and took many unlabelled questions in the second iteration, and stopped.

5.8.3 Experiments with Co-training

This section implemented the co-training algorithm as in Figure 5.2 to compare the results with our modified Tri-training algorithm. SVM with *bag-of-word* and *bag-of-word&pos* features were used in the experiments.

The Co-training algorithm in Figure 5.2 is applied for binary classification. In the experiments of [Blum & Mitchell (1998)], both positive and negative examples were added to the labeled set after having been assigned labels, and the parameters were set to $p = 1$, $n = 3$, $k = 30$ and $u = 75$. In our problem, the question data set consists of 50 classes. The questions not belong to a class can be treated as negative examples for it. However,

Table 5.9: The average size of L_i in each round corresponding to the experiments in Table 5.7

#	Iter.	TMW			TMP			TSW		
		L_1	L_2	L_3	L_1	L_2	L_3	L_1	L_2	L_3
1000	1	22	30.5	13.5	4.75	22	22	24	33	28.25
	2	4273	4212.25	4288	4018	3973	3979.5	490	494	488.5
2000	1	32.25	47.25	31.5	34.25	45.5	50	29	33	26.75
	2	3220.75	3203.25	3335.25	3221.5	3240.75	3225	488.25	490.5	488.75
3000	1	43	44	44	48.25	60	42	36	33.75	36.75
	2	1485.5	1480.75	1479.75	2297.75	2325.75	1904.5	740	733.75	741.5
4000	1	46.5	41	46	49.75	45	47	41.25	41.75	36
	2	655.25	554.5	512.25	527.5	641.75	672.25	395.5	395	390.5

#	Iter.	TSP			TSSM2		
		L_1	L_2	L_3	L_1	L_2	L_3
1000	1	30.25	28.75	25.75	3226	499	499
	2	4244.25	4236.75	4233.25	-	-	-
2000	1	26.75	30.25	28	999	499	499
	2	980	980	986.25	-	-	-
3000	1	29.75	31.25	34	187	750	187
	2	1474.25	1485.25	1476	302	283.5	0
4000	1	35.75	36.25	35	222	399	199
	2	983	982.5	988.25	-	-	-

we can not keep the ratio between the negative examples n and positive examples p as in [Blum & Mitchell (1998)]'s experiments. Instead, at each iteration, we add only one unlabeled question for each question class (if there is). The parameters of our experiments were set to: $p = 1$, $k = 30$ and $u = 50 * 5$. We carried out Co-training on labeled sets of different size (1000, 2000, 3000 and 4000), and results are shown in Table 5.10, where *Sup.* and *Final* are, respectively, the precision of supervised learning the co-training of each classifier.

Table 5.10: The precision of co-training with SVM

#	bag-of-word		bag-of-word&pos	
	Sup.	Final	Sup.	Final
1000	68.4	69.2	69.2	69.0
2000	75.6	75.8	75.2	75.6
3000	78.2	78.2	77.0	77.4
4000	78.6	79.0	79.0	78.4

The results in Table 5.10 showed that the classifier with bag-of-word&pos often helped to improve the other classifier. Whereas, the precision of classifier with bag-of-word&pos was rarely improved. We tried to increase the number of iteration k to 100, however, the results were also worse.

We also tried a different method of adding unlabeled questions into the labeled set: the number of unlabeled questions added to a class is directly proportional to its percentage in the training set. However, the results were worse than those in Table 5.10.

5.8.4 Experiments with Self-training

This section implemented the self-training algorithm as in Figure 5.3 to compare the results with our modified Tri-training algorithm.

We carried out self-training on labeled sets of different size (1000, 2000, 3000 and 4000), and the classification algorithm is SVM with bag-of-word features. The results of our experiments are given in Table 5.11, where '*Base.*' is the precision of supervised learning which is used as the baseline; '*Self.*' is the precision of the self-training. The results show that most final precision of self-training is not improved. Though only questions in the unlabeled set U' with high prediction probability are selected to form the labeled set L', it can not guarantee that those questions are correctly predicted as our observation. Thus, in each iteration, the newly created labeled set may contain mislabeled questions, and the error rate may consequently increase. In general, the self-training is not well suitable for question data type with bag-of-word features.

Table 5.11: The precision of self-training with SVM

#	1000		2000		3000		4000	
	Base.	Self.	Base.	Self.	Base.	Self.	Base.	Self.
Precision	68.4	65.8	75.6	73.4	78.2	76.2	78.6	78.4

5.8.5 Discussion

Through experiments we can see that self-training is not suitable for solving this task, because its method to add unlabeled questions for further training the classifier is not good. The co-training algorithm does not work well for this domain, either. The original Tri-training algorithm has a better method of adding unlabeled questions based on the agreement of two classifiers. However, the bootstrap-sampling step may decrease the initial precision of each classifier and the final precision is hard to be improved. Our two proposals remove the bootstrap-sampling while still ensure the three classifiers to have different hypotheses, and the experiments have proved the proposals to be suitable.

5.9 Experiments with Hierarchical Classifiers

In this section, we used bag-of-words features for all experiments.

5.9.1 Supervised Learning Application

Experiments in this section follow the approach discussed in Section 5.5.1. We use supervised learning for all classifiers. The first experiment, MEM is used for all classifiers in the Figure 5.7. The system is tested on 4 pre-divided training sets as stated in Section 5.7.1. Another similar experiment with SVM used for all classifiers is also carried out. The results of two experiments in comparison with flat classifications are given in Table 5.12, where FMEM and HMEM are the precision of flat and hierarchical classification with MEM, respectively; N is the number of tests, in which the hierarchical precision is less

Table 5.12: The precision (%) of flat and hierarchical classification with MEM and SVM on fine classes.

#	FMEM	HMEM	N	FSVM	HSVM	N
1000	67.6	68.8	4	68.4	**69.0**	5
2000	74.8	74.8	8	75.6	**75.8**	5
3000	76.8	77.2	5	78.2	**78.4**	7
4000	77.2	77.4	4	**78.6**	**78.6**	11

Table 5.13: The precision of the original Tri-training with SVM, MEM, and the modified Tri-training on coarse classes.

#	MEM-word			SVM-word			Mod. Tritraining			
	Base.	Ini.	TMW	Base.	Ini.	TSW	Ini. SVM	Ini. MEM	TSSM	N
1000	77.6	76.2	78.4	78.6	75.6	77.6	78.6	77.6	**79.2**	0
2000	84.6	79.6	81.4	83.6	81.6	84.0	83.6	84.6	**85.0**	2
3000	85.2	84.2	85.4	86.2	83.4	86.2	86.2	85.2	**87.0**	3
4000	84.6	85.4	85.8	87.6	85.6	87.8	87.6	84.6	**88.2**	1

than that of flat classification. FSVM and HSVM are flat and hierarchical classification with SVM. Table 5.12 shows that when the size of training set is small, the error rate of flat classification is high and hierarchical classification exposes its advantages.

5.9.2 Experiments with the Original Tri-training and Its Modification for Coarse Classes

Experiments in this section follow the original Tri-training algorithm listed in Figure 5.4, and our modified version listed in Figure 5.4. We used two SVM classifiers and one MEM classifier similar to the experiments in Section 5.8.1.

We tested these algorithms on 4 training sets of coarse classes. We also carried out the sign test at the significance level 95%.

Similar to Section 5.8.1, for the original Tri-training algorithm, we implemented two programs: one with MEM (called TMW) and the other with SVM (called TSW). The precision of the experiments is listed in Table 5.13, where *Super* column is the precision of supervised learning; *Ini.* column is the precision of the worst classifier (among the three classifiers) calculated after having trained on a bootstrap-sampled set at line 4 of the algorithm in Figure 5.4.

The results of our modified Tri-training for coarse classifiers are listed in Table 5.13, where '*Ini. SVM*', '*Ini. MEM*' and *TSSM* are the initial precision of SVM, MEM classifiers and the final output of the algorithm, respectively.

5.9.3 Supervised and Semi-supervised Learning Combination Application

Experiment in this section follows the discussion in Section 5.5.2. We applied semi-supervised learning for the coarse classifier at the first level and supervised learning for

Table 5.14: The precision of flat classification of MEM, SVM and 1 level semi-supervised learning with SVM-MEM.

#	FMEM	Msemi1	N	FSVM	Ssemi1	N
1000	67.6	69.2	0	68.4	**69.6**	1
2000	74.8	75.2	2	75.6	**77.0**	0
3000	76.8	**78.6**	0	78.2	**78.6**	3
4000	77.2	**80.4**	0	78.6	79.0	4

Table 5.15: The precision of flat classification of MEM, SVM and 2 level semi-supervised learning with SVM-MEM.

#	FMEM	FSVM	Semi2-1	Semi2-2
1000	67.6	68.4	63.4	**69.2**
2000	74.8	75.6	71.0	**75.8**
3000	76.8	78.2	74.8	**78.4**
4000	77.2	**78.6**	75.0	**78.6**

fine classifiers in the second level. We used our modified version of Tri-training as discussed in the preceding section.

We tested the system on 4 training sets, for each of them, all questions not belonging to the training set are used as unlabeled ones. In addition, we carried out the sign test for the experiments at the significance level 95%.

The precision of the experiment in comparison with flat classification is shown in Table 5.14, where FMEM and FSVM are flat classification with MEM and SVM; Msemi1 and Ssemi1 are hierarchical classification with semi-supervised learning for coarse classifier and supervised learning for fine classifiers which use MEM and SVM, respectively; N is the number of tests, in which the precision of hierarchical classifier is less than that of flat classification. The results show that the improvement of semi-supervised learning help to improve the final precision. Interestingly, when the size of training sets increase, the performance of fine classifiers with MEM is improved better than that of fine classifiers with SVM. Consequently, the overall performance of the hierarchical classifiers in which fine classifiers using MEM is improved significantly with the training set of the size 4000.

5.9.4 Semi-supervised Learning Application

This section implements the method discussed in Section 5.5.3. We applied semi-supervised learning for all classifiers. We used the same semi-supervised learning as that of Section 5.9.2. The results of two methods for employing semi-supervised learning for fine classifiers are given in Table 5.15, where Semi2-1 is the semi-supervised learning, in which we add one more class label to the class set of each classifier as depicted in Figure 5.9; Semi2-2 is the semi-supervised learning for all classifiers, in which we use coarse classifier to generate unlabeled questions for each fine classifier as depicted in Figure 5.9. The results show that the addition of one more fine class label for each coarse (Semi2-1) class has bad effect on the performance of the system. The second method (Semi2-2) does not improve the precision of fine classifiers. The reason is the erroneous unlabeled questions for each fine classifier generated by the coarse classifier. Because of not promising results,

we did not carry out the significant tests.

Table 5.16: The precision of flat classification of MEM, SVM and hierarchical classification with three levels by expanding the *Entity* coarse class.

#	FMEM	HMEM	N	FSVM	HSVM	N
1000	67.6	**69.6**	0	68.4	**69.6**	2
2000	74.8	75.8	6	75.6	**76.4**	5
3000	76.8	77.4	6	78.2	**79.0**	1
4000	77.2	77.4	8	78.6	**79.0**	6

Table 5.17: The precision of flat classification of MEM, SVM and hierarchical classification with three levels by expanding the *Entity* and *Numeric* coarse classes.

#	FMEM	HMEM	N	FSVM	HSVM	N
1000	67.6	70.0	6	68.4	**70.8**	1
2000	74.8	75.4	4	75.6	**76.4**	8
3000	76.8	77.2	6	78.2	**78.8**	10
4000	77.2	76.8	-	78.6	**79.0**	4

5.9.5 Question Hierarchy Expansion

Experiments in this section follow the discussion in Section 5.6. We expand nodes consisting of a large number of classes. For clustering algorithm, we consider K-mean [Jarvis & Patrick (1973)], and for distance metric, we use *Euclid* measure.

The first experiment, we expanded the *Entity* coarse class, which consists of 22 fine classes. We selected a number of clusters so that the number of classes per cluster is approximate. In this case, the number of cluster is 3, so the *Entity* coarse class is divided into 3 subclasses as follows:

- Subclass 1: *animal*, *creation*, *decease/medical*, *food*, *other*, *term*
- Subclass 2: *body*, *currency*, *instrument*, *language*, *letter*, *plant*, *religion*, *symbol*
- Subclass 3: *color*, *event*, *product*, *sport*, *substance*, *technique*, *vehicle*, *word*

After this step, the question taxonomy is expanded into three levels at *Entity* branch. The results of the experiment on the newly created taxonomy is given in Table 5.16, which show that the precision of each test is better than that of original taxonomy in Table 5.12.

The second experiment, besides *Entity* coarse class, we expanded one more coarse class, that is *Numeric* which consists of 13 fine classes. The number of clusters in this situation is also 3. The three subclasses are as follows:

- Subclass 1: *code*, *order*, *speed*, *temperature*, *size*, *weight*
- Subclass 2: *distance*, *money*, *other*, *percent*, *period*
- Subclass 3: *count*, *date*

The results of the experiment given in Table 5.17 indicate that when the size of training set is small, the hierarchical classifiers with new taxonomy give better results that those of new taxonomy in the first experiment when the size of training set is small.

5.10 Summary

This chapter applied semi-supervised learning to explore unlabeled questions to improve the performance of question classification task and proposed two ways of modifying the Tri-training algorithm presented by [Zhou & Li (2005)] to make it more suitable for question data type. The proposals dealt with a problem at the initial step of Tri-training, where the original labeled set is bootstrap-sampled to generate three different labeled sets, in order to make the three classifiers have different hypotheses, which may make the initial error rate of each classifier increase. With the purpose of using the original labeled set for all classifiers, while ensuring that they are still diverse, in the first proposal, we used more than one learning algorithm for the three classifiers and the second proposal is to use multiple learning algorithms in combination with more than one views. Our experiments indicated that the performance is improved, and our approach was more suitable for this task in comparison with other semi-supervised learning approach, such as Co-training and Self-training.

The second effort, we proposed to applied hierarchical classifiers to reduce the number of question classes per classifier in order to improve the performance. Several learning approaches were investigated in the hierarchical classifiers. And a proposal to automatically expand nodes (in the question hierarchy) that consists of a large number of question classes. The experiments showed promising results.

In the current implementation, we have not considered to select other better feature types, such as those used in [Li & Roth (2002)]. This is one interesting issue to explore in future to achieve higher precision.

Our modified versions of Tri-training algorithm do not have any constraints on data types, therefore, one more issue which is worth studying in the future is to apply these algorithms in other domains, such as text classification.

Chapter 6

Building a Search System Based on Linguistic Semantic Information

In this chapter, we present a prototype of a semantic search system based on semantic relations. Our semantic system was built based on the studies in previous chapters.

6.1 System Architecture

The architecture of our system is depicted in Figure 6.1. In the preprocessing step, documents are processed by "*Relations extractor*" (investigated in Chapter 3) which acquires named-entity-related relations and store in a Knowledge Base. Documents are parsed to construct RST trees (this study does not cover this parser, so it is marked by a dotted rectangle). After that, documents are indexed in the form of T-expressions by "*Document Indexer*" (investigated in Chapter 4) which indexes documents in the form of T-expressions.

In the serving mode, when a user types in a question, it is firstly classified by "*Question classification*" module (investigated in Chapter 5). Next, based on the type of the question given by "*Question classification*", the "*Answer extractor*" (investigated in Chapter 3 and 4) will find the answer in either "*Indexed database*" or "*Relation KB*" with respect to the question type. Finally, the found answer is returned to the user.

6.2 Experiments and Evaluation

6.2.1 Dataset

Since our system supports a different set of question classes in comparison with that of the experiments in Chapter 5, another question dataset is needed for "*Question classification*" module. From the question set in Chapters 3 and 4, we randomly divided into two sets for training and testing. After this step, the dataset consists of 6 of question classes:

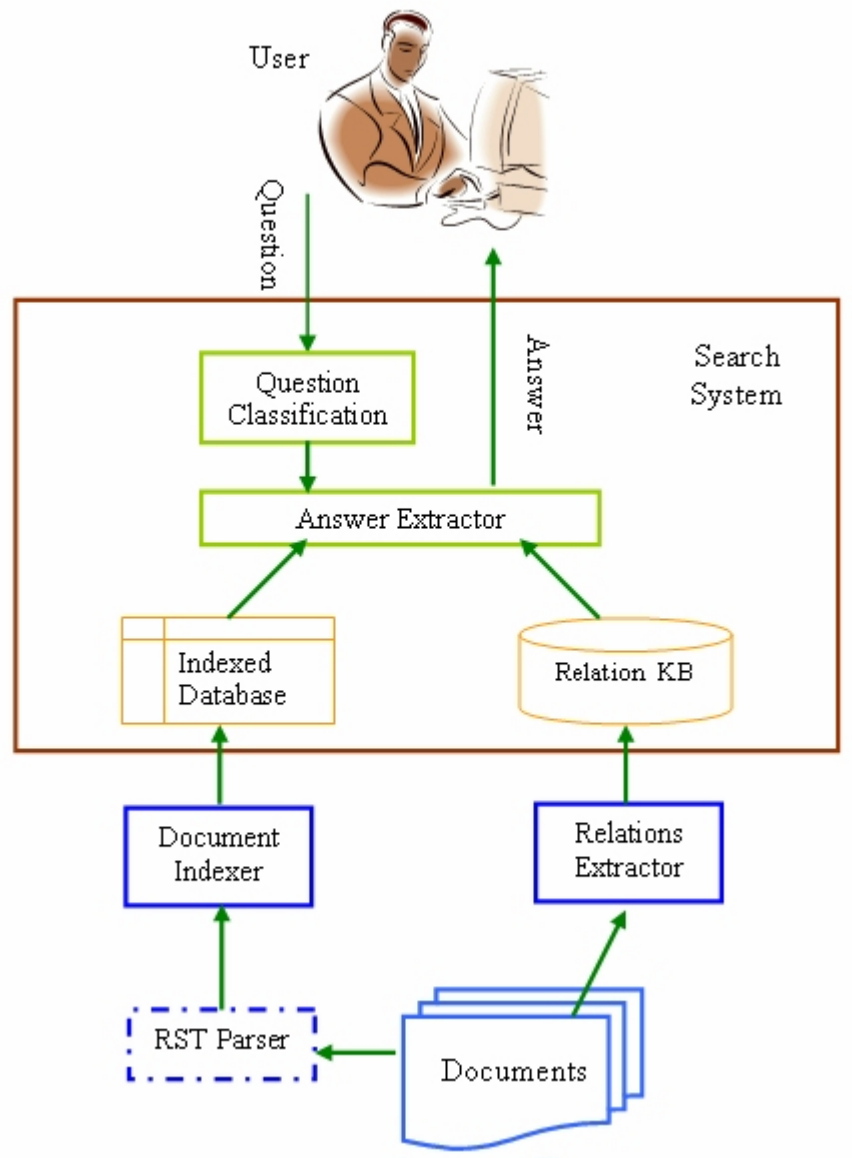

Figure 6.1: Our search system architecture

fact, *how to*, *suggestion*, *why*, *who* and *list*. The training set after the division is rather small, so we manually collected and labeled Frequently Asked Questions (FAQ) from the site http://www.faqs.org. Since the algorithm in Chapter 3 can be applied to extract information of other named entity types, such as *organization* and *location*, our system can support to answer *what* question (e.g., "What is IBM?"). In addition, users may type in questions that are not supported by our system, these questions should be of another question class, so we added two more question classes: *what* and *other*. After this step, we the question dataset consists of 4027 training and 341 testing questions. The training set is randomly divided into two subsets. The first subset consists of 3000 questions was used as the labeled set, while the subset of 1027 questions was used as the unlabeled set. Because of set of question classes is small, we only applied semi-supervised learning for "*Question classification*" module, and the hierarchical classifiers were not suitable for this task. We used the same datasets as those in Chapters 3 and 4 for answering questions.

6.2.2 Experiments

For "*Question classification*" module, we used our modified version of Tri-training as described in Figure 5.6, and followed the experiment in Section 5.8.2. Two SVM classifiers (one with bag-of-word feature and the other with bag-of-word&pos feature) with a MEM classifier (with bag-of-word feature) were used. The results of the experiment with "*Question classification*" module are shown in Table 6.1, where SVMW, SVMP and MEMW are the precision of the supervised learning of classifier SVM with bag-of-word, SVM with bag-of-word&pos and MEM with bag-of-word features, respectively. '*Final*' is the precision of the modified Tri-training algorithm.

The datasets for extracting answers were those of Chapters 3 and 4. Concretely, the RST Discourse Treebank and Wall Street Journal corpus were used in our experiments. The results of our search system are shown in Table 6.2, where the *cosine* thresholds (the *cosine* measures the similarity between a returned answer and the correct answer of a question as discussed in Chapter 3) of *who* and *list* questions were set to 1. We compare the results of our search system with one that used manual question classification (the precision of question classification is 100%).

Table 6.1: The precision (%) of the "*Question classification*" module.

SVMW	MEMW	SVMP	Final
95.3	79.8	*95.6*	**97.1**

Table 6.2: The results of the system.

Type	Our search system			Search with manual QC		
	P(%)	R(%)	F(%)	P(%)	R(%)	F(%)
Who with objects	87.18	68.00	76.40	87.18	68.00	76.40
Who without objects	87.18	68.00	76.40	87.18	68.00	76.40
List with objects	100.00	77.50	87.32	100.00	77.50	87.32
List without objects	69.23	60.00	64.29	69.23	60.00	64.29
Suggestion	93.33	60.00	73.04	95.20	80.0	86.90
How to	80.10	97.80	88.10	80.10	97.80	88.10
Why	81.08	93.75	86.96	83.78	96.88	89.86
Fact	79.43	96.91	87.30	80.60	98.40	88.60

6.3 Summary

In this chapter, we have presented our prototype of a search system based on linguistic semantic information. The components of the system were investigated in Chapters 3, 4 and 5. A new question dataset was created for question classification module. And our experiments showed promising results.

Chapter 7

Conclusions and Future Work

7.1 Summary of the Thesis

In this thesis, we have presented a study of building a search system which can extract answer to some typical question types by exploiting the semantic relations in documents. The study considers the research problems in both theoretical and practical views. The theoretical view concerns about the proposal of a new algorithm for extracting relations in documents, and modification of the semi-supervised Tri-training algorithm for more suitable for imbalanced data. The practical views comes from the effective application of these algorithms in building the final semantic system. Among the seven chapters of the thesis, the main chapters are 3, 4, 5 and 6. And main contributions of the thesis can be summarized as follows:

• Firstly, we analyzed the need of fine-grained categories of named-entities, and proposed a new algorithm to automatically extract the fine-grained categories of named-entities. The fine-grained category of a named-entity expresses the relation "*named entity*" ISA *category*, in which the *category* partially describes the "*named entity*". In documents, there may be more information describing a "*named entity*", so we extended the algorithm to extract more complete information related to named entities. These relations were consumed to answer some question types related to named entities, namely, *who* and *list*. Our experiments gave very good results.

• Secondly, in order to answer some other question types that are not relevant to named entities, such as *why*, *how* and *suggestion*, we analyzed the RST relations in documents, and proposed to exploit their characteristics to extract answers to the above question types. The characteristics of the RST relations are: in some RST relations between two text spans, the content of this span may be the answer to a question related to the other span. Our experiments showed promising results proving this is a good direction.

• Thirdly, one of the most important step in any system (e.g., semantic search or QA system) that needs to know the type of a given question as the clue for extracting/finding the answer. We considered to use statistical machine learning for

question classification. In order to utilize unlabeled questions, which are available with large volumes and and cheap to collect, to improve the performance of question classification, we proposed to apply semi-supervised learning. We adopted the Tri-training algorithm, because it has advantages over other semi-supervised learning algorithms. After analyzing the characteristics of question data, as well as the drawbacks of Tri-training when applying to this domain, we proposed two methods to modify Tri-training to make it more suitable for this problem. Our experiments has proved our proposals are promising. Another effort to improve the classification performance, we noticed that when the number of question classes is large, the performance of classification algorithm is affected. We proposed to apply hierarchical classifiers in order to reduce the number of question classes per classifier. Our experiments showed significant improvement. Different ways of applying learning methods (supervised and semi-supervised learning) were investigated in the hierarchical classifiers. A method to automatic to expand the question hierarchy was also proposed with the motivation to sub-group the nodes consisting of a large number of classes. Our experiments gave promising improvements.

• Finally, a search system based on the semantic relations was built with the methods investigated in the chapters 3, 4 and 5. A new question dataset, which consists of the supported question types of the system, was collected for the experiments with the datasets of other modules. Our experiments showed promising results.

7.2 Future Directions

Each semantic relation can help answer certain question types. Currently, our search system considers only some semantic relations so the number of supported question types is still small. Fortunately, there are various valuable relations in documents, and our system is open, so we can continue to incrementally develop the system by extracting new relations for supporting new question types. For example, the very first issue to do in the future is to used the algorithm in chapter 3 to extract information of other named entity types, such as *organization* and *location*.

Another possible improvement of the system is to apply a good co-reference resolution tool so that the system can help answer the *which* questions.

The method for matching T-expressions in chapter 4 may be not sufficient enough, since it does not have the ability to match questions that are expressed differently from the fact. Another issue for future is to use other method or additional knowledge base (e.g., WordNet) to solve the mentioned problem.

In the current system, we do not have the RST parser, the possible problem for future is to build this module so the system can work in the real world.

An interesting direction in the future is to make the system have the ability to reason by exploiting other semantic relations in documents, such as *entailment* relation.

Bibliography

[Agichtein & Gravano (2000)] Agichtein, E., and Gravano, L., 2000. Snowball: Extracting Relations from Large Plaintext Collections. *Proceedings of the 5th ACM International Conference on Digital Libraries*, pp. 85-94.

[Al-Onaizan & Knight (2001)] Al-Onaizan, Y., and Knight, K., 2001. Translating Named Entities Using Monolingual and Bilingual Resources. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 400-408.

[Berger, *et al.* (1996)] Berger, A., Pietra, S. D., and Pietra, V. D., 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, Vol. 22, No. 1, pp. 39-71.

[Berry, *et al.* (1995)] Berry, M. W., Dumais, S. T., and O'Brien, G. W., 1995. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, Vol. 37, No. 4, pp. 573-595.

[Blum & Mitchell (1998)] Blum, A., and Mitchell, T., 1998. Combining Labeled and Unlabeled Data with Co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory* (COLT), pp. 92 - 100.

[Bonino, *et al.* (2003)] Bonino, D., Corno, F., and Farinetti, L., 2003. DOSE: a Distributed Open Semantic Elaboration Platform. *Proceedings of The 15th IEEE International Conference on Tools with Artificial Intelligence* (ICTAI 2003), pp. 3-5.

[Bonino, *et al.* (2004)] Bonino, D., Corno, F., Farinetti, L., and Bosca, A., 2004. Ontology Driven Semantic Search. *WSEAS Transaction on Information Science and Application*, Vol. 1, Issue 6, pp. 1597-1605.

[Bosma (2004)] Bosma, W., 2004. Query-based Summarization Using Rhetorical Structure Theory. *Proceedings of the 15th Computational Linguistics in the Netherlands*, pp. 29-44.

[Brin (1998a)] Brin, S., 1998. Extracting Patterns and Relations from the World Wide Web. *Proceedings of WebDB Workshop at 6th International Conference on Extending Database Technology* (EDBT'98), pp. 172-183.

[Brin & Page(1998b)] Brin, S., and Page, L., 1998. The Anatomy of a Large-scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, Vol. 30, pp. 107-117.

[Broder, *et al.* (2004)] Broder, A., Maarek, Y., Mandelbrod, M., and Mass, Y., 2004. Using XML to Query XML from Theory to Practice. *Proceedings of Recherche d'Information Assistée par Ordinateur* (RIAO'04).

[Carlson, *et al.* (1999)] Carlson, A., Cumby, C., and Roth, D., 1999. The SNoW Learning Architecture. *Technical Report UIUC-DCS-R-99-2101, UIUC Computer Science Department.*

[Carlson, *et al.* (2001)] Carlson, L., Marcu, D., and Okurowski, M., 2001. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. *Proceedings of the 2nd SIGdial Workshop on Discourse and Dialogue*, Vol. 16, pp. 1-10.

[Carmel, *et al.* (2003)] Carmel, D., Maarek, Y., Mandelbrod, M., Mass, Y., and Soffer, A., 2003. Searching XML Documents via XML Fragments. *Proceedings of 26th SIGIR Conference*, pp. 00-00.

[Chamberlin, *et al.* (2002)] Chamberlin, D., Fankhauser, P., Florescu, D., Marchiori, M., and Robie, J., 2002. XML query use cases. At http://www.w3.org/TR/2002/ WD-xmlquery-use-cases-20020816.

[Charniak (2000)] Charniak, E., 2000. A Maximum-entropy-inspired Parser. *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics*, pp. 132-139.

[Chieu & Tou (2003)] Chieu, H., and Tou, N., 2003. Named Entity Recognition with a Maximum Entropy Approach. *Proceedings of Conference on Natural Language Learning 2003* (CoNLL-2003), pp. 160-163.

[Chinenyanga & Kushmerick (2002)] Chinenyanga, T., and Kushmerick, N., 2002. An Expressive and Efficient Language for XML Information Retrieval. *Journal of the American Society for Information Science and Technology*, Vol. 53, No. 6, pp. 438-453.

[Chu-Carroll, *et al.* (2006)] Chu-Carroll, J., Prager, J., Czuba, K., Ferrucci, D., and Duboue, P., 2006. Semantic Search via XML Fragments: a High-precision Approach to IR. *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 445-452.

[Cohen, *et al.* (2003)] Cohen, S., Mamou, J., Kanza, Y., and Sagiv, Y., 2003. XSearch: A Semantic Search Engine for XML. *Proceedings of the 29th International Conference on Very Large Databases*, pp. 45-56.

[Cortes, *et al.* (1995)] Cortes, C., and Vapnik, V., 1995. Support Vector Networks. *Machine Learning*, Vol. 20, No. 3, pp. 273-297.

[Cunningham, *et al.* (2002)] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V., 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics* (ACL'02), pp. 168-175.

[Deerwester, *et al.* (1990)] Deerwester, S., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A., 1990. Indexing by Latent Semantic Analysis. *Journal of the Society for Information Science*, Vol. 41, No. 6, pp. 391-407.

[Dumais, *et al.* (2003)] Dumais, S., Cutrell, E., Cadiz, J., Jancke, G., Sarin, R., and Robbins, D., 2003. Stuff I've Seen: A System for Personal Information Retrieval and Re-use. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pp. 72-79.

[Fellbaum (1998)] Fellbaum, C., editor, 1998. WordNet: An electronic Lexical Database and Some of Its Applications. *MIT Press*.

[Fleischman, *et al.* (2003)] Fleischman, M., Hovy, E., and Echihabi, A., 2003. Offline Strategies for Online Question Answering: Answering Questions before They are Asked. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pp. 1-7.

[Fuhr & Großjohann (2001)] Fuhr, N., and Großjohann, K., 2001. XIRQL: a Query Language for Information Retrieval in XML Documents. *Proceedings of the 24th International ACM SIGIR Conference on Research and Development in Information Retreival*, pp. 172-180.

[Fukumoto (2007)] Fukumoto, J., 2007. Question Answering System for Non-factoid Type Questions and Automatic Evaluation based on BE Method. *Proceedings of the 4th Question Answering Challenge (QAC-4) at NTCIR Workshop 6*, pp. 441-447.

[Girju (2002)] Girju, C. R., 2002. Text Mining for Semantic Relations. *Doctoral Dissertation*, University of Texas at Dallas.

[Goldman & Zhou (2000)] Goldman, S., and Zhou, Y., 2000. Enhancing Supervised Learning with Unlabeled Data. *Proceedings of the 17th International Conference on Machine Learning*, pp. 327-334.

[Grishman & Sundheim (1996)] Grishman, R., and Sundheim, B., 1996. Message Understanding Conference 6: A Brief History. *Proceedings of COLING-96*, pp. 466-471.

[Guha & McCool (2002)] Guha, R., and McCool, R., 2002. TAP: Towards a Web of Data. At http://tap.stanford.edu Rada Mihalcea.

[Guha, *et al.* (2003)] Guha, R., McCool, R., and Miller, E., 2003. Semantic Search. *Proceedings of the 12th International Conference on World Wide Web*, pp. 700-709 .

[Hearst (1992)] Hearst, M. A., 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. *Proceedings of the 14th Conference on Computational Linguistics*, pp. 539 - 545.

[Hassel (2003)] Hassel, M., 2003. Exploitation of Named Entities in Automatic Text Summarization for Swedish. *Proceedings of the 14th Nordic Conference on Computational Linguistics*.

[Heflin, *et al.* (1999a)] Heflin, J., Hendler, J., and Luke, S., 1999. SHOE: A Knowledge Representation Language for Internet Applications. *Technical Report CS-TR-4078 (UMIACS TR-99-71)*, Dept. of Computer Science, University of Maryland at College Park.

[Heflin, *et al.* (1999b)] Heflin, J., Hendler, J., and Luke, S., 1986. Coping with Changing Ontologies in a Distributed Environment. *Proceedings of the AAAI Workshop*, WS-99-13, AAAI Press, pp. 74-79.

[Heflin & Hendler (2000a)] Heflin, J., and Hendler, J., 2000. Dynamic Ontologies on the Web. *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (AAAI-2000), AAAI/MIT Press, pp. 443-449.

[Heflin & Hendler (2000b)] Heflin, J., and Hendler, J., 2000. Searching the Web with SHOE. *Proceedings of Artificial Intelligence for Web Search*, AAAI Press, pp. 35-40.

[Heflin (2001)] Heflin, J., 2001. Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment. *Doctoral dissertation*, University of Maryland.

[Hovy, *et al.* (2001)] Hovy, E. H., Gerber, L., Hermjakob, U., Lin, C. Y., and Ravichandran, D., 2001. Toward Semantics-based Answer Pinpointing. *Proceedings of the DARPA Human Language Technologies Conference* (HLT), pp. 339-345.

[Ittycheriah, *et al.* (2001)] Ittycheriah, A., Franz, M., Zhu, W. J., Ratnaparkhi, A., Mammone, R. J., 2001. IBM's Statistical Question Answering Systems. *Proceedings of the 9th Text Retrieval Conference*, pp. 229-235.

[Jarvis & Patrick (1973)] Jarvis, R. A., and Patrick, E. A., 1973. Clustering Using a Similarity Measure Based on Shared Near Neighbors, *IEEE Transactions on Computers*, Vol. C22, pp. 1025-1034.

[Joachims (1998)] Joachims, T., 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of ECML-98, the 10th European Conference on Machine Learning*, pp. 137-142.

[Kadri & Wayne (2003)] Kadri, H., and Wayne, W., 2003. Question Classification with Support Vector Machines and Error Correcting Codes. *Proceedings of NAACL/Human Language Technology Conference*, pp. 28-30.

[Kanji (1994)] Kanji, G. K., 1994. 100 Statistical Tests. *SAGE Publications*.

[Karvounarakis, *et al.* (2002)] Karvounarakis, G., Christophides, V., Plexousakis, D., and Scholl, M., 2002 . RQL: a Declarative Query Language for RDF, *Proceedings of the 11th international conference on World Wide Web*, pp. 592-603.

[Katz (1991)] Katz, B., 1991. Using English for Indexing and Retrieving. *Artificial Intelligence at MIT Expanding Frontiers*, MIT Press, pp. 134-165.

[Kim, *et al.* (2006)] Kim, Y., Kim, B., and Lim, H., 2006. The Index Organizations for RDF and RDF Schema. *Proceedings of the 8th International Conference on Advanced Communication Technology* (ICACT 2006), pp. 1871-1874.

[Kogut & Holmes (2001)] Kogut, P., and Holmes, W., 2001. AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. *Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001) Workshop on Knowledge Markup and Semantic Annotation.*

[Kogut & Heflin (2003)] Kogut, P., and Heflin, J., 2003. Semantic Web Technologies for Aerospace. *Proceedings of IEEE Aerospace Conference*, Vol. 6, pp. 2887-2894.

[Kumaran & Allan (2004)] Kumaran, G., Allan, J., 2004. Text Classification and Named Entities for New Event, *Proceedings of the 27th Annual International ACM SIGIR Conference*, Sheffield: ACM Press, pp. 297-304.

[Le (2004)] Le, T. H., 2004. Investigation into an Approach to Automatic Text Summarization. *Doctoral dissertation*, Middlesex University.

[Letscher & Berry (1997)] Letscher, T. A., and Berry, M. W., 1997. Large-scale Information Retrieval with Latent Semantic Indexing. *Information Sciences Applications*, Vol. 100, No. 105, pp. 105-137.

[Lewis (1991)] Lewis, D., 1991. Representation and Learning in Information Retrieval. *Doctoral Dissertation*, University of Massachusetts.

[Li & Roth (2002)] Li, X., and Roth, D., 2002. Learning Question Classifiers. *Proceedings of the 19th International Conference on Computational Linguistics*, pp. 556-562.

[Li & Roth (2005)] Li, X., and Roth, D., 2005. Learning Question Classifiers: The Role of Semantic Information. *Journal of Natural Language Engineering*, Vol. 12, Issue 3, pp. 229-249.

[Lindley, *et al.* (2001)] Lindley, C. A., Davis, J. R., Nack, F., and Rutledge, L. W., 2001. The Application of Rhetorical Structure Theory to Interactive News Program Generation from Digital Archives. *Technical Report: INS-R0101, CWI Centrum voor Wiskunde en Informatica.*

[Luke & Heflin (1997)] Luke, S., and Heflin, J., 1997. SHOE 1.0, Proposed Specification. At http://www.cs.umd.edu/projects/plus/SHOE/spec.html.

[Mann & Thompson (1988)] Mann, C., and Thompson, S. A., 1988. Rhetorical Structure Theory: A Theory of Text Organization. At http://www.sfu.ca/rst/

[Marcu (2000)] Marcu, D., 2000. The Rhetorical Parsing of Unrestricted Texts: A Surface-Based Approach, *Computational Linguistics*, Vol. 26, Issue 3, pp. 395-448.

[Marir & Haouam (2004)] Marir, F., and Haouam, K., 2004. Rhetorical Structure Theory for Content-based Indexing and Retrieval of Web Documents. *Proceedings of the 2nd International Conference on Information Technology: Research and Education*, pp. 160-164.

[Matono, *et al.* (2004)] Matono, A., Amagasa, T., Yoshikawa, M., and Uemura, S., (2004). An Indexing Scheme for RDF and RDF Schema Based on Suffix Array. *Transactions of Information Processing Society of Japan*, Vol. 45, No. SIG4(TOD21), pp. 50-62.

[Matono, *et al.* (2005)] Matono, A., Amagasa, T., Yoshikawa, M., and Uemura, S., 2005. A Path-based Relational RDF Database. *Proceedings of the 16th Australasian Database Conference*, pp. 95-103.

[Mulenbach, *et al.* (2004)] Mulenbach, F., Lallich, S., and Zighed, D. A., 2004. Identifying and Handling Mislabelled Instances. *Journal of Intelligent Information Systems*, Vol. 22, No. 1, pp. 89-109.

[Nguyen, *et al.* (2006a)] Nguyen, T. T., Shimazu, A., Le, A. C., and Nguyen, L. M., 2006. Applying RST Relations to Semantic Search. *Proceedings of the 9th International Conference on Text, Speech and Dialog*, pp. 189-196.

[Nguyen, *et al.* (2006b)] Nguyen, T. T., Nguyen, L. M., and Shimazu, A., 2006. Using Semi-supervised Learning for Question Classification. *Proceedings of the 21st International Conference on the Computer Processing of Oriental Languages*, pp. 31-41.

[Nguyen, *et al.* (2007a)] Nguyen, T. T., Nguyen, L. M., and Shimazu, A., 2007. Improving the Accuracy of Question Classification with Machine Learning. *Proceedings of the 5th International Conference on Research, Innovation & Vision for the Future: RIVF'07, IEEE Explore Digital Library*, pp. 234-241.

[Nguyen & Shimazu (2007b)] Nguyen, T. T., and Shimazu, A., 2007. Automatic Extraction of the Fine Category of Person Named Entities from Text Corpora. *IEICE Transactions on Information and Systems, Special section on Knowledge, Information and Creativity Support System*, Vol. E90-D, No. 10, pp. 1542-1549.

[Nguyen & Shimazu (2007c)] Nguyen, T. T., and Shimazu, A., 2007. Acquisition of Named-Entity-Related Relations for Searching. *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation* (PACLIC21), pp. 349-357.

[Nguyen, *et al.* (2008)] Nguyen, T. T., Nguyen, L. M., and Shimazu, A., 2008. Using Semi-supervised Learning for Question Classification. *Journal of Natural Language Processing*, Vol. 15, No. 1, (to appear).

[Nigam & Ghani (2000)] Nigam, K., and Ghani, R., 2000. Analyzing the Effectiveness and Applicability of Co-training. *Proceedings of the 9th International Conference on Information and Knowledge Management*, pp. 86-93.

[Nomoto (2004)] Nomoto, T., 2004. Machine Learning Approaches to Rhetorical Parsing and Open-domain Text Summarization. *Doctoral Dissertation*, Nara Institute of Science and Technology.

[Nobata, *et al.* (2002)] Nobata, C., Sekine, S., Isahara, H., and Grishman, R., 2002. Summarization System Integrated with Named Entity Tagging and IE Pattern Discovery. *Proceedings of the 3rd International Conference on Language Resources and Evaluation.*

[Pinto, *et al.* (2001)] Pinto, H. S., and Martins, J., 2001. A Methodology for Ontology Integration. *Proceedings of the 1st International Conference on Knowledge Capture*, pp. 131-138.

[Popov, *et al.* (2003)] Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D., and Goranov, M., 2003. KIM - Semantic Annotation Platform, *Proceedings of the International Semantic Web Conference* (ISWC), pp. 834-849.

[Pasca (2004)] Pasca, M., 2004. Acquisition of Categorized Named Entities for Web Search. *Proceedings of the 13th ACM Conference on Information and Knowledge Management* (CIKM-04), pp. 137-145.

[Reitter (2003)] Reitter, D., 2003. Rhetorical Analysis with Rich-Feature Support Vector Models. *Diplomas Thesis in Computation Linguistics*, University of Potsdam.

[Salton, *et al.* (1975)] Salton, G., Wong, A., and Yang, H. S., 1975. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, Vol. 18, No. 11, pp. 613-620.

[Sang & Meulder (2003)] Sang, E. F. T. K., and Meulder, F. D., 2003. Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition. *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003*, Vol. 4, pp. 142-147.

[Sekine, *et al.* (2000a)] Sekine, S., Sudo, K., and Nobata, C., 2000. Extended Named Entity Hierarchy. *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pp. 1818-1824.

[Sekine & Isahara (2000b)] Sekine, S., Isahara, H., 2000. IREX: IR and IE Evaluation Project in Japanese. *Proceedings of the Second International Conference on Language Resources and Evaluation*, Vol. 1, pp. 1475-1480.

[Shinzato & Torisawa (2004)] Shinzato, K., and Torisawa, K., 2004. Acquiring Hyponymy Relations from Web Documents. *Proceedings of HLT-NAACL 2004*, pp. 73-80.

[Singhal, *et al.* (1999)] Singhal, A., Choi, J., Hindle, D., Lewis, D. D., and Pereira, F. C. N., 1999. AT&T at TREC-8. *Proceedings of the 8th Text Retrieval Conference*, pp. 186-198.

[Srihari & Li (2000)] Srihari, R., and Li, W., 2000. A Question Answering System Supported by Information Extraction. *Proceedings of the 6th Applied Natural Language Processing Conference*, pp. 166-172.

[Stoffel, *et al.* (1997)] Stoffel, K., Taylor, M., and Hendler, J., 1997. Efficient Management of Very Large Ontologies. *Proceedings of American Association for Artificial Intelligence Conference* (AAAI-97), pp. 442-447.

[Sumida, *et al.* (2006)] Sumida, A., Torisawa, K., and Shinzato, K., 2006. Concept-Instance Relation Extraction from Simple Noun Sequences Using a Full-text Search Engine. *Proceedings of the Web Content Mining with Human Language Technologies workshop on the fifth International Semantic Web*, pp. 442-447.

[Voorhees (1999)] Voorhees, E., 1999. The TREC-8 Question Answering Track Report. *Proceedings of the 8th Text Retrieval Conference* (TREC-8), pp. 77-82.

[Voorhees (2000)] Voorhees, E., 2000. The TREC-9 Question Answering Track. *Proceedings of the 9th Text Retrieval Conference* (TREC-9), pp. 71-80.

[Voorhees (2001)] Voorhees, E., 2001. Overview of the TREC 2001 Question Answering Track. *Proceedings of the 10th Text Retrieval Conference* (TREC-10), pp. 157-165.

[Yates, *et al.* (1999)] Yates, R. B., and Neto, B. R., 1999. Modern Information Retrieval. *Addison-Wesley.*

[Zhang & Lee (2003)] Zhang, D., and Lee, W. S., 2003. Question Classification Using Support Vector Machines. *Proceedings of the 26th Annual International ACM SIGIR Conference*, pp. 26-32.

[Zhou & Li (2005)] Zhou, Z. H., and Li, M., 2005. Tri-training: Exploiting Unlabeled Data Using Three Classifiers. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 17, No. 11, pp. 1529-1541.

Publications

Journals

[1] Nguyen, T. T., and Shimazu, A., 2007. Automatic Extraction of the Fine Category of Person Named Entities from Text Corpora. *IEICE Transactions on Information and Systems, Special section on Knowledge, Information and Creativity Support System*, Vol. E90-D, No. 10, pp. 1542-1549.

[2] Nguyen, T. T., Nguyen, L. M., and Shimazu, A., 2008. Using Semi-supervised Learning for Question Classification. *Journal of Natural Language Processing*, Vol. 15, No. 1, (to appear).

Refereed International Conferences

[3] Nguyen, T. T., Shimazu, A., Le, A. C., and Nguyen, L. M., 2006. Applying RST Relations to Semantic Search. *Proceedings of the 9th International Conference on Text, Speech and Dialog*, pp. 189-196.

[4] Nguyen, T. T., Nguyen, L. M., and Shimazu, A., 2006. Using Semi-supervised Learning for Question Classification. *Proceedings of the 21st International Conference on the Computer Processing of Oriental Languages*, pp. 31-41.

[5] Nguyen, T. T., Nguyen, L. M., and Shimazu, A., 2007. Improving the Accuracy of Question Classification with Machine Learning. *Proceedings of the 5th International Conference on Research, Innovation & Vision for the Future: RIVF'07, IEEE Explore Digital Library*, pp. 234-241.

[6] Nguyen, T. T. and Shimazu, A., 2007. Acquisition of Named-Entity-Related Relations for Searching. *Proceedings of the 21st Pacific Asia Conference on Language, Information and Computation* (PACLIC21), pp. 349-357.

[7] Nguyen, M. L., Nguyen, T. T., and Shimazu, A., 2007. Subtree mining for question classification problem. *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (IJCAI), pp. 1695-1700.