Robust Knowledge Extraction over Large Text Collections

A Thesis

Submitted to the Faculty

of

Drexel University

by

Min Song

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

May 2005

ACKNOWLEDGEMENTS

To finish my PhD in College of Information Science and Technology, I have great debts to others. First of all, I should acknowledge supports from my family. My wife, YoungJoo, always encouraged me to move forward in the journey of my PhD program. My lovely daughters, Joy and Sophia, gave me comfort and rest. Without their sacrifice, I wouldn't have accomplished my dream.

I also thank Professor Il-Yeol Song, who is my advisor. He shaped me to be a good researcher and taught me invaluable assets for a scholar. I also thank my committee members, Tony Hu, Robert Allen, Xia Lin, Mark Lechner, and Eugene Garfield for their input and advices.

I thank my parents who gave me the reasons to complete the degree. My father, Oh-Young Song, provided me with financial and psychological supports for all years of my degree. My mother, Soon-bok Lee, was my mentor who gave me strengths to overcome any problems in this journey.

Last, I thank Jesus who is my savior. I praise him to give me everything needed to complete my degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Robust Knowledge Extraction over Large Text Collections

Min Song

Il-Yeol Song, Ph.D.

Automatic knowledge extraction over large text collections has been a challenging task due to many constraints such as needs of large annotated training data, requirement of extensive manual processing of data, and huge amount of domain-specific terms. In order to address these constraints, this study proposes and develops a complete solution for extracting knowledge from large text collections with minimum human intervention. As a testbed system, a novel robust and quality knowledge extraction system, called RIKE (Robust Iterative Knowledge Extraction), has been developed. RIKE consists of two major components: DocSpotter and HiMMIE. DocSpotter queries and retrieves promising documents for extraction. HiMMIE extracts target entities based on a Mixture Hidden Markov Model from the selected documents from DocSpotter. The following three research questions are examined to evaluate RIKE: 1) How accurately does RIKE retrieve the promising documents for information extraction from huge text collections such as MEDLINE or TREC? 2) Does ontology enhance extraction accuracy of RIKE in retrieving the promising documents? 3) How well does RIKE extract the target entities from a huge

medical text collection, MEDLINE?

The major contributions of this study are1) an automatic unsupervised query generation for effective retrieval from text databases is proposed and evaluated, 2) Mixture Hidden Markov models for automatic instances extraction are proposed and tested, 3) Three Ontology-driven query expansion algorithms are proposed and evaluated, and 4) Object-oriented methodologies for knowledge extraction system are adopted.

Through extensive experiments, RIKE is proved to be a robust and quality knowledge extraction technique. DocSpotter outperforms other leading techniques for retrieving promising documents for extraction from 15.5% to 35.34% in P@20. HiMMIE improves extraction accuracy from 9.43% to 24.67% in F-measures.

CHAPTER 1: INTRODUCTION

1.1 Goal of the Research

Knowledge Extraction (KE) is a relatively new research area at the intersection of Data Mining (DM), Information Extraction (IE), and Information Retrieval (IR). The goal of knowledge extraction is to discover knowledge in natural language text. In terms of knowledge extraction, a variety of knowledge can be extracted from textual data, such as linguistic knowledge and domain-specific lexical and semantic information hidden in unstructured text corpus (Alani et al., 2003). IE, most pertinent to knowledge extraction, locates specific data from corpora of natural language texts and populates a relational table with these data (Feldman et al., 1999). Since the start of the Message Understanding Conferences (MUCs) (Hirschman, 1998), IE addresses the issue of transforming unstructured data into more structured and relational databases. The transformed text corpus can be then mined by various DM techniques whose the major task is to apply statistical and machine-learning methods to discover novel relationships in large relational databases.

Although developing an IE system is a challenging task, there has been significant recent progress in using data mining methods to help automate the construction of IE systems (Cardie, 1997; Freitag, 1998). The IE techniques typically

involve several steps such as name-entity tagging, syntactic parsing, and rule matching. The steps required to transform text are relatively expensive. As the recent flux of electronic texts, processing gigabytes size of text database becomes a difficult challenge for IE to leverage extracted information with relational databases. The IE techniques proposed so far are not feasible for large databases or for the web, since it is not realistic to tag and parse every available documents. In addition, IE requires a dictionary of entities and relation terms, or well-defined rules for identifying them. This requirement posed by IE is hard to satisfy because unstructured text corpus tends to be 1) non-uniform and incomplete, 2) synonymy and alias, and 3) polysemy (Shatkay et al., 2002). These factors are attributed to the low recall of IE systems reported in the literature.

Another major problem with the current IE techniques is that it is labor intensive in terms of processing text collections and also requires extensive knowledge on the target domain. Due to this issue, IE is not applicable to some domains where minimum human intervention is necessary or domain experts are not available.

Although several papers have suggested development of a scalable knowledge extraction system (Shatkay and Feldman, 2003; Agichtein and Gravano, 2003), no specific research has proposed the KE system such as the system proposed in this thesis. In terms of the general principle, to the best of my knowledge, the only similar technique is found in Agichetein and Gravano's paper (2003). This dissertation is differentiated from Agichetein and Gravano's (2003) in the following ways. First, this dissertation introduces a novel query expansion technique based on the keyphrases, while their techniques is based on a single term and combined with Ripper, a rule-based learning system (Cohen, 1995). Second, our system is based on unsupervised query training whereas their system is based on supervised querying learning. Third, we introduce a Mixture Hidden Markov models-based extraction technique for knowledge extraction whereas their approach is based on mutual reinforcement technique extended from Brin's DIPRE algorithm (1998).

The goal of this thesis is to investigate whether 1) keyphrase-based query expansion improves the retrieval performance, 2) Mixture Hidden Markov Models-based information extraction enhances the extraction accuracy, and 3) ontologies have the positive impact on the retrieval performance. To this end, we develop a novel, robust, and high quality knowledge extraction system, called RIKE (Robust

Iterative Knowledge Extraction), based on the integration of IR, IE, and Statistical Machine Learning. RIKE adapts a novel IR technique, a keyphrase extraction based query, to collect promising documents and uses a statistical machine learning technique, Hidden Markov Model (HMM) to transform collection of documents into more structured data which is in turn mined for interesting relationships. As illustrated in Figure 1, RIKE consists of two major components: DocSpotter, querying and retrieving promising documents for extraction, and HMMIE, a statistical generative model based IE. RIKE queries a search engine with ranked keyphrases, discovers prediction rules from the retrieved unstructured documents, and then the rules are used to predict additional information to extract from future documents, thereby improving the recall of IE. The details of RIKE are provided in the Chapter 3.

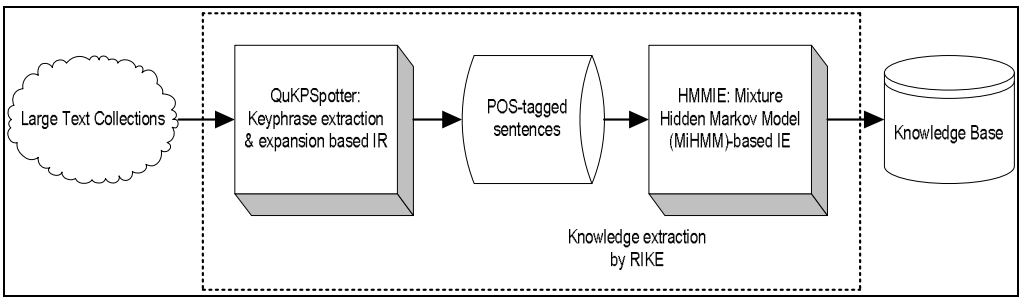

Figure 1: Overview of Knowledge Extraction Framework Proposed by RIKE

1.2 Research Questions

This thesis develops and evaluates a novel knowledge extraction system to automatically extract knowledge from huge unstructured text corpora in a principled and general manner. To this end, huge text collections are required to extract knowledge, and we chose three different sets of text collections. First, we chose MEDLINE queried by the PubMed search engine. The reason for using MEDLINE is that it is the largest biomedical bibliographic database with more than 12 million abstracts. MEDLINE is a knowledge base manually built to expedite the progress of functional bioinformatics. The increasing interest in and importance of bioinformatics also motivates this thesis to emphasize on automatically extracting knowledge in biomedical text corpus. Second, we chose to build our own collections that are downloaded from PubMed. Due to the dynamic nature of the PubMed search engine, we could not guarantee that MEDLINE indexed and searched by PubMed is consistent across time. In addition, PubMed is based on the exact match retrieval model. It could skew experiments. Therefore, we built our own collections which consist of about 260,000 MEDLINE records. These collections are indexed and searched by the Lemur search engine. Third, we chose TREC data to investigate

whether our query expansion algorithm can be equally useful to general topics other than biomedical domain-centric tasks. TREC data was indexed and searched by Zettair.

Since the KE systems proposed in the literature so far are not applicable to large text collections (Agichtein and Gravano, 2003), we need a scalable KE system suited for large collections. In this thesis, we propose a novel KE system, RIKE. The primary goal of this thesis is how accurately and completely RIKE extracts the target entities from MEDLINE with only a handful set of biomedical named-entities interactions such as protein-protein and gene-protein. Specifically, this dissertation concentrates on the how accurately RIKE spots the promising documents to extract the target entities from. In addition, this dissertation investigates how accurately RIKE extracts the target entities from unstructured text collections.

This dissertation also investigates whether ontologies improve the extraction accuracy of RIKE. Several studies report that incorporating an ontology into Query Expansion (QE) improves the retrieval performance (Hersh et al., 2000; Pizzato & de Lima, 2003). Expanding keyphrases with assistance of ontology such as WorldNet (Miller, 1990) for general domains or Unified Medical Language System (UMLS) for a specific domain like medicine is a novel approach to integrating ontology to QE.

This thesis seeks to apply ontology to associate key phrases with synonymously and hierarchically relevant concepts. The succinct description of how key concepts are expanded is as follows: Key phrases or concepts extracted by our keyphrase extraction technique are ranked based on the importance of concepts in relation to the given queries. The top N keyphrases are then mapped to either synonymous or broader/narrower terms by looking up collection-independent knowledge structures.

This thesis investigates ways in which existing sources of domain knowledge or ontologies can be leveraged to learn more accurate queries. Understanding a text can ultimately be possible only if the system can refer to an ontology (or controlled vocabulary), i.e., the association of words to meanings, by including hierarchical relations between them. They can be general such as WorldNet, or specific to a domain of knowledge, e.g., to medicine as MeSH or UMLS.

The following three specific research questions are addressed in this thesis:

R1: How accurately does RIKE retrieve the promising documents for information extraction from huge text collections such as MEDLINE or TREC?

R2: Does an ontology enhance extraction accuracy of RIKE to retrieve the promising documents?

R3: How well does RIKE extract the target entities from huge medical text collections, MEDLINE?

This dissertation first proposes a novel knowledge extraction technique that consists of a keyphrase-based query expansion algorithm and Mixture Hidden Markov model-based information extraction algorithm. With the proposed model, we approach these three questions more in-depth and comprehensively in the subsequent chapters.

The rest of dissertation consists of as follows: Chapter 2 identifies and summarizes the works related to this dissertation. Chapter 3 proposes and describes a novel knowledge extraction technique. Chapter 4 explains the experimental settings and evaluation methodologies. Chapter 5 reports and analyzes the experimental results. Chapter 6 concludes the dissertation and suggests the future work.

CHAPTER 2: RELATED WORK

In this chapter, we cover the research areas relevant to knowledge extraction. The following three research fields are reviewed: 1) Information Retrieval, 2) Information Extraction, and 3) Biomedical Text Mining. In each related field, we focus on the works related to the research topic of this dissertation.

2.1 Information Retrieval

Information Retrieval is one of the focal research field in Information Science. We are particularly interested in how retrieval performance can be improved by automatic query expansion.

2.1.1 Query Expansion

A typical goal of an information retrieval (IR) system is to find a set of documents containing information needed by searchers in the given indexed database(s). In processing a query that searchers formulate, conventional IR query languages require searchers to state precisely what they want. Searchers need to be able to express their needs in terms of precise queries (either in Boolean form or natural languages).

However, due to searchers' lack of knowledge in the search domain (anomalous state of knowledge), query syntax formulated by searcher often does not meet the searchers information need (Belkin, 1982). In addition, a single-term based query that a normal user formulates retrieves many irrelevant articles as well as fails to find hidden knowledge or relationship buried in content of the articles.

To overcome this issue with query formulation, many IR systems provide facilities for relevance feedback, with which searchers can identify documents of interest to them. IR systems can then use the thesaurus terms assigned to these desired documents to find other potentially relevant documents. However, these IR systems fail to distinguish among the attributes of the desired documents for their relative importance to the searchers' needs.

Relevance feedback is the most popular and widely accepted query reformulation strategy. The main idea consists of selecting important terms, or expressions, attached to the documents that have been identified as relevant by the user, and of enhancing the importance of these terms in a new query formulation. The new query should be moved towards the relevant documents and away from the non-relevant ones.

There are a number of formal ways of describing relevance feedback, beginning with the notion of an "optimal query" used in the SMART system (Salton, 1968). The optimal query is defined as a vector obtained by taking the difference between the relevant and non-relevant sets of documents, also represented as vectors. This query vector can be shown to be the "best", under some assumptions, for distinguishing relevant and non-relevant documents. This relevance feedback approach evolved to a query modification process where the old query is modified by a weighted average of the identified relevant documents and, in some versions of the algorithm, the identified non-relevant documents. The revised query is then used to produce a new document ranking. It has been used for the vector space model and found to significantly improve performance (Rocchio, 1971). Another approach to modeling adaptation uses genetic algorithms (GA). For instance, Yang and Korfhage (1993) evolve a population of query individuals to optimize the search. Gordon (1988), on the other hand, uses a method where competing representations are associated with documents. The representations are then altered over time using GA.

Another early approach to relevance feedback is a Bayesian classification model of retrieval (Van Rijsbergen, 1979). In this approach, identified relevant documents are used to estimate the characteristics (probabilities of occurrences of

words) of the relevant class of documents for a query. The corpus is used to estimate probabilities for the non-relevant class. The revised estimates are used to produce a new document ranking based on the probability of belonging to the relevant class. Both of these approaches can be viewed as applications of different machine learning techniques to the problem of identifying relevant documents based on training data. There have been a number of other recent experiments with machine learning techniques, but these have not in general shown significant improvements over the approaches already described (Schapire et al., 1998).

Despite rigorous studies on relevance feedback and its long history in IR, systems adopting relevance feedback shows only limited success (Croft et al., 2001). There are a number of reasons for the apparent failure of relevance feedback in current systems. The primary reason is that relevance feedback provided by systems fails to satisfy searcher needs with revised retrieval results that relevance feedback suggests. In this work, we tackle the issue of ineffectiveness of relevance feedback by applying an emerging data mining technique, relational data mining, to relevance feedback.

The problem of retrieving documents that are "relevant" to a user's information needs has been the focus of the information retrieval (IR) field (Rocchi,

1971; Robertson & Sparck, 1976; Salton, 1989). In IR, it is common to identify phrasal terms from queries and generate their variant expansions. It has been shown that such query expansion promotes effective retrieval (Agichtein et al., 2001; Callan & Connell, 2001; Robertson, 1990; Ro, 1998). The current approaches to Query Expansion (QE) can be classified by both the source of providing the terms for the expansion and the method adapted to select terms to be used in the expansion (Efthimiadis, 1996; Robertson, 1990). For the former case, one source of QE is based on the search results, and it relates to the relevance feedback process. Documents that have been identified as relevant in the earlier iteration of the search become the source for the query expansion terms (Spink, 1994). The other source is some form of a knowledge structure (taking the phrase broadly) that is independent of the search process. The form of the knowledge structure can depend on the collection, that is, it can be corpus based, or it can be independent of it (Crouch, 1988).

On the other hand, the methods used for selecting the terms for QE requires that the system either chooses the terms by itself based on some criteria or makes it present the query expansion terms to the user for selection. In both automatic and interactive approaches, the ranked order of terms is of primary importance. The terms should be ordered in such a way that those terms that are most likely to be useful are

placed close to the top of the list. In addition, heuristic decisions can also be applied during this stage. For example, poor terms are excluded from the term list instead of being given low weights. Term weighting and ranking are important for term selection for query expansion. The literature shows a plethora of ranking algorithms reported (Ro, 1988; Sager & Lockemann, 1976). These algorithms attempt to quantify the value or usefulness of a query term in retrieval.

One subtask of the MUC evaluation text filtering, is relevant to our work [MUC-7]. For example, Diderot (Cowie et al., 1992) classified input documents based on single words and n-gram words prior to doing any further processing, while others filtered documents by using manually constructed keywords filtered to discard documents. The classification-based approach required manually labeled documents for training the classifiers. Other systems developed filters from the extraction patterns devised to extract the target relation. All of these systems are tightly integrated with some domain-specific component, which make it very difficult to port from one application domain to another.

Several techniques use supervised learning to devise queries that match documents about a specific category of interest. Cohen and Singer (1996) constructed topic-specific directories on the web by training a classifier with a labeled set of

documents and then deriving queries to retrieve additional documents. Flake et al. (2002) extracted category-specific query modifications from a non-linear SVM classifier. Domain-specific search engines which depend on classification of documents are becoming increasingly popular because they offer increased accuracy and extra features not possible with general, web-wide search engines (McCallum et al., 2000). Unfortunately, they are also difficult and time-consuming to maintain. We propose the use of data mining techniques to greatly automate the creation and maintenance of domain-specific search engines. In our work, on efficient IE, we consider query generation techniques based on a term weighting scheme that is related to some of the techniques in (Salton, 1989), as well as other query generation strategies that exploit data mining results. We describe new research in reinforcement learning, text classification and IE that enables efficient text search and retrieval, and identifies informative text segments. In the traditional sense of QE, our approach is similar to local context analysis (Xu & Croft, 1996). Local context analysis combines global concepts generated with the INQUERY search engine. The concepts are selected from the top-ranked documents based on co-occurrence with query terms. The usability of local context analysis is, however, lessened by the drawbacks that the global concepts generated are not key concepts for the collection. Term co-occurrence

used to rank concepts is problematic as well. Compared to local context analysis, our approach is more sophisticated and collection independent.

2.1.2 Pseudo-relevance Feedback

The quality of a query fed to an IR system has a direct impact on the success of the search outcome. In fact, one of the most important but frustrating tasks in IR is query formulation (e.g., French et al., 1997). Relevance feedback is a popular and widely accepted query reformulation strategy. The main idea consists of selecting important terms, or expressions, attached to the documents that have been identified as relevant by the user, and of enhancing the importance of these terms in a new query formulation. The expected effect is that the new query will be moved towards the relevant documents and away from the non-relevant ones.

As outlined in Section 1, pseudo-relevance feedback methods improve the retrieval performance on average but the results are not as good as relevance feedback. In pseudo-relevance feedback, problems arise when terms or phrases taken from assumed-to-be relevant documents that are actually non-relevant are added to the query causing a drift in the focus of the query. To tackle this issue, Mitra, et al. (1998) incorporated term co-occurrences to estimate word correlation for refining the

set of documents used in query expansion. They made use of only individual terms for query expansion whereas we utilize keyphrases for query expansion. In addition, they vary window size for matching query but in our technique window size is determined by sentence length.

Mihalcea and Moldovan (2000) found that using the selected passages from documents for query expansion is effective in reducing the number of inappropriate feedback terms taken from non-relevant documents. Lam-Adesina and Jones (2001) applied document summarization to query expansion. In their approach, only terms present in the summarized documents are considered for query expansion. Whereas Lam-Adesina and Jones add all terms from the summaries to the query, we use only the top N ranked keyphrases chosen with our selection criteria. Lam-Adesina and Jones adopted a summarization technique based on sentence extracted summaries that are found by scoring the sentences in the documents. The scoring method is simply a sum of the scores gained by the four summarization methods: 1) Luhn's keyword cluster, 2) title terms frequency, 3) location/header, and 4) query-bias methods. Whereas their technique is based on simple mathematical properties of terms, our techniques are information theory-based as well as mathematically solid.

Liu et al. (2004) used noun phrases for query expansion. Specifically, four types of noun phrases were identified: proper names, dictionary phrases, simple phrases, and complex phrases. A document has a phrase if all the content words are in the phrase within the defined window, and these documents that have matched phrases are considered to be relevant. They also apply a similarity measure to select the content words in the phrases to be positively correlated in the collection. By comparison, we utilize keyphrases including verb phrases from the top N-ranked documents retrieved by the original query whereas Liu et al. make use of only noun phrases in queries. In addition, our approach combines phrase co-occurrence with the Information Gain of a keyphrase.

Because we also investigate whether adding concepts from WordNet to keyphrases improves the retrieval performance, we briefly survey some related works to our approach. Liu, et al. (2004) add selected synonyms, hyponyms, and compound words based on their word sense disambiguation technique. Our approach to word sense disambiguation is different in that we disambiguate word sense by similarity criteria between all the non-stopwords from the synonyms and definitions of the hyponym synsets and keyphrases extracted from the retrieved documents.

Voorhees (1998) used WordNet for adding synonyms of query terms whereas we use WordNet to add synonyms and substantial hyponyms of the top N-ranked keyphrases.

2.2 Information Extraction

Many of the earlier systems extracted information on relations by matching the sentences in the text files with hand-tailored patterns in regular expression on some pre-defined set of verbs representing a certain type of reaction (Grishman et al., 2002; Yangarber & Grishman, 1998). Although information can be represented in various forms in natural language text, many patterns of surface expressions need to be prepared for one event. Thus, although pattern-matching based IE can be effective and quick on limited types of events in a limited domain, the workload of preparing patterns for extraction would be too expensive if we expand our attention to texts from other domains or to a wider scope of events.

Most machine learning methods and algorithms that have been developed to automatically generate extraction patterns use special training resources, such as texts annotated with domain-specific tags (e.g., AutoSlog (Riloff, 1996), CRYSTAL (Soderland et al., 1995), RAPIER (Califf 1998), SRV (Freitag, 1998), WHISK (Soderland, 1999) and LIEP (Huffman 1996)). Roth and Yih (2001) have presented a

system Snow-IE, which they use to learn the relational representation desirable for the IE task using propositional learning mechanisms. A key limitation of using machine learning methods to induce information-extraction methods is that the process of labeling a training example is expensive. Another important bottleneck is the availability of high-quality, pre-classified corpora in IE from a text database. Creating a pre-classified corpus entails a high workload for domain experts, and a corpus for a specific domain can usually not be directly transferred to other domains, thus making portability a very challenging issue. AutoSlog is one the earliest attempts to automate the construction of an information extraction system. Although AutoSlog creates a dictionary of extraction patterns using machine learning technique, it is not a fully automated system because a human expert has to intervene in the loop to validate extraction patterns. LIEP can be viewed as another version of AutoSlog with capabilities for learning patterns for multiple slots. Crystal learns more expressive extraction patterns with both semantic and exact work constraints, and was proposed to be combined with a preprocessing engine called WebFoot (Soderland, 1997) for page segmentation based on page layout cues. WHISK focused on learning information extraction patterns for online information that can be either structured, semi-structured, or in free format. SRV applied its ability for learning extraction

patterns written in first-order logic expressions on several attributes and relational structure of the documents, to online web documents. RAPIER learns extraction rules describing constraints on slot fillers and their surrounding context using a specific-to-general search.

Most learning algorithms rely on feature-based representation of objects. That is, an object is transformed into a collection of position-independent features f_1, $f_2, \ldots, f_n$, (f_i can be a n-gram word in the document), thereby producing a N-dimensional vector (also known as bag-of-word representation). The limitation of this representation is that in many cases, data cannot be easily expressed via features. For example, in most NLP problems, feature-based representations produce inherently local representations of objects, because it is computationally infeasible to generate features involving long-range dependencies. Kernel methods (Vapnik,1995; Zelenko et al., 2003) and relational learning are an attractive alternative to feature-based representations. One practical problem in applying kernel methods or relational learning to IE in large text collection is their speed. The two approaches are relatively slow compared to feature classifiers, whose computation complexity may be too high for practical purposes.

Recently, a number of systems use unlabeled examples for training. This direction of research is closest to our work. Specifically, the approach we are following falls into the broad category of bootstrapping techniques. Bootstrapping has been an attractive alternative in automatic text processing. Yarowsky (1995) demonstrates a boostrapping technique for disambiguating senses of ambiguous nouns. Yi and Sundaresan (1999) describe an extension of DIPRE (Brin, 1998) to mine the web for acronyms and their expansion. Blum and Mitchell (1998) present a methodology and theoretical framework for combining unlabeled examples with labeled examples to boost performance of a learning algorithm for classifying web pages. While the underlying principles of using the systems' output to generate the training input for the next iteration is the same for all of these approaches, the tasks are different enough to require specialized methodologies.

HMMs have been drawn much attention from IE community as a competitive extraction algorithm (Freitag, 1998). Instead of using explicit patterns and rules proposed in the systems mentioned above, HMMs rely on statistical token sequence generation model rather than explicit extraction patterns. Since this dissertation adopts HMMs to fill in the IE part of RIKE framework for building a knowledge extraction system, HMMs are described in more detail in the next section.

2.2.1. Hidden Markov Model

The basic theory of HMMs was developed in the late 1960s, but only in the last decade have HMMs been extensively applied in a large number of problems such as speech recognition (Rabiner, 1989), handwritten character recognition (Hu et al., 1996), DNA and protein modeling (Hughey & Krogh, 1996), gesture recognition (Eickeler et al., 1998), and to computer vision problems. Since Leek (1997) used HMMs for finding the binary relationship between gene names and gene locations, several interesting techniques based on HMMs have been proposed (Freitag, 1999; McCallum et al., 2000; Skounakis et al., 2003; Syemore et al., 1999).

HMMs are based on the *Markov Assumption*, proposed by the mathematician, Andrei Markov, in the early decades of the 20th Century. The Markov assumption is that the probability of any event in the sequence depends only on the preceding event, and on none of the earlier ones. Suppose we look at a sequence of random events. If we assure the Markov assumption holds for this sequence, then to predict the next variable in the sequence, we only need to consider the value of the current variable. For instance, if each variable represents your bank account balance at time *N*, we can

probably accept that your balance at time *N+1* does not depend on what it was 100 or even 10 time units ago, rather it will probably only depend on it's present total. Now if we take this variable sequence *X*, as having values that are contained in a finite set *S*, a set of states:

$$\boldsymbol{X = (X_1, X_2, \ldots\ldots, X_T)\ S = \{s_1, s_2, \ldots\ldots, s_N\}} \qquad (2.1)$$

For the sequence *X*, the Markov assumption can be expressed as:

$$\boldsymbol{P\ (\ X_{t+1} = s_k \mid X_1, \ldots . X_t) = P\ (\ X_{t+1} = s_k \mid X_t\ )} \qquad (2.2)$$

That is, the probability of variable X_{t+1} equaling state s_k, given the sequence of variables up to and including X_t, is equal to the probability of this happening given only the value of X_t, the last state in the sequence. The sequence *X* is referred as a *Markov Chain*.

A Markov Chain can be described by a matrix *A*, containing all the state transition probabilities (where position a_{ij} in the matrix would contain the probability of going to state s_j given we were just in state s_i). In addition to this, we need to know what the initial state is, or more precisely what the set of initial state probabilities is (the probability that a given state is the initial one).

Each state can emit a variety of tokens where each token is associated with a probability. That is to say, in a given state s_k, there is a set of emission tokens and their associated emission probabilities. With the emission sequence X, it is not known which state sequence generates the given output, since more than one state can generate the same token. Hence the sequence gives us some probabilistic function of the actual state sequence that generated it. This model is called a Hidden Markov Model (HMM), because the actual state sequence that generated the output is hidden from us (Figure 2).

Formally, a Hidden Markov Model can be represented by the quintuple $(S,K,\prod,A,B)$, where:

S = Set of states in the HMM, $S = \{s_1,s_2,.....,s_N\}$

K = Output Alphabet, $K = \{k_1,k_2,.....,k_M\}$

$\prod$ = Initial state probabilities $\prod = \{\pi_i\}$, $\pi_i = P(s_i \text{ at } t=1), i \varepsilon S$

A = Transition probabilities $A = \{a_{ij}\}$, $= P(s_i \text{ at } t+1 \mid s_j \text{ at } t)$

B = Emission probabilities $B = \{b_{mj}\}$, $= P(k_m \text{ at } t \mid s_j \text{ at } t)$

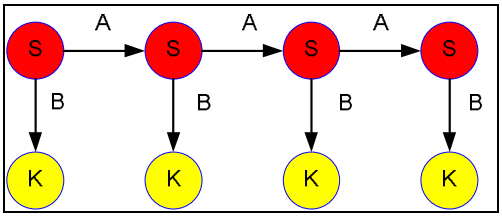

Figure 2: HMM Formalism

HMMs allow us to model a sequence of events generated by an underlying probabilistic model. This is the general approach taken by information extraction, where we can view an arbitrary piece of text as having been produced by some underlying HMM (where most words are produced by some junk-token states, but important information is produced by other states, like name-token states or time-token states).

2.3 Biomedical Literature Mining

The sequencing of the human genome powered by advanced software and hardware opened the door to the new era of large-scale genomics and proteomics. In this new era, large-scale experiments with thousands of genes or proteins can be conducted. However, the gigantic volume of biomedical data challenges us on how to interpret it. To tackle this issue, rigorous studies have recently been carried out to apply IE to biomedical data. Such research efforts began to be called biomedical

literature mining or text mining in bioinformatics (de Bruijn & Martin, 2002; Hirschman et al., 2002; Shatkay & Feldman, 2003).

This section attempts to synthesize the works that have been done in the field. Taxonomy helps us understand the accomplishments and challenges in this emerging field. In this section, we use the following set of criteria to classify the biomedical literature mining related studies:

1) What are the target objects to be extracted?

2) What techniques are used to extract the target objects from the biomedical literature?

3) How are the techniques or systems evaluated?

4) From what data sources are the target objects extracted?

Figure 3 shows a overview of a typical biomedical literature mining system.

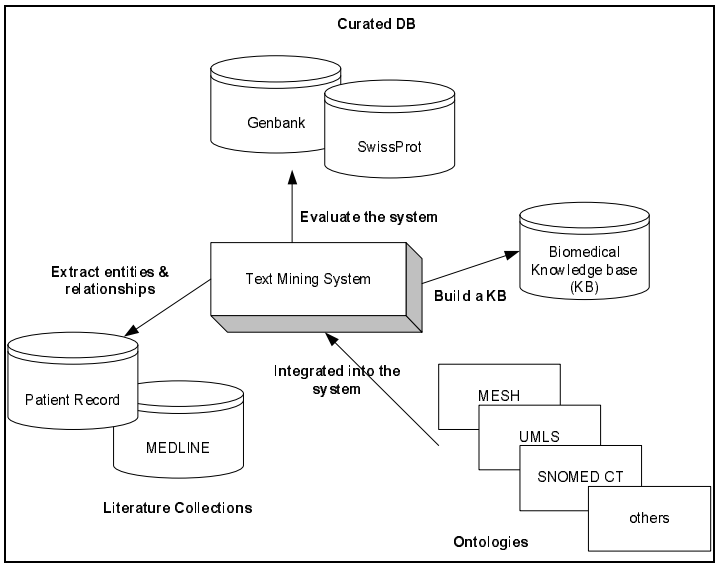

Figure 3: An Overview of a Biomedical Literature Mining System

2.3.1 Target Objects

In terms of what is to be extracted by the systems, most studies can be broken into the following two major areas: 1) named entity extraction, named entities may include *proteins* or *genes* and 2) relation extraction such as relationships between proteins. Most of these studies adopt information extraction techniques, using a curated lexicon or natural language processing for identifying relevant tokens such as words or phrases in text (Shatkay & Feldman, 2003).

Fukuda et al. (1998) extract protein names with hand-crafted rules. Although they reported that experimental results were competitive based on F-value of 0.92, the results were not replicated and their method relied on manually created rules. Proux et

al. (2000) use single word names only with selected test set from 1200 sentences coming from Flybase. Collier et al. (2000) adopt Hidden Markov Models (HMMs) for 10 test classes with small training and test sets. Krauthammer et al. (2000) use BLAST database with letters encoded as 4-tuples of DNA. Demetriou and Gaizuaskas (2002) pipeline the mining processes including hand-crafted components and machine learning components. For the study, they use large lexicon and morphology components. Narayanaswamy et al. (2003) use a Part of Speech (POS) tagger for tagging the parsed MEDLINE abstracts. Although Narayanaswamy and his colleagues (2003) implement an automatic protein name detection system, the number of words used is 302 and thus it is difficult to see the quality of their system in that the size of the test data is too small. Yamamoto et al. (2003) use morphological analysis techniques for preprocessing protein name tagging and apply Support Vector Machine (SVM) for extracting protein names. They found that increasing training data from 390 abstracts to 1600 abstracts improved F-value performance from 70% to 75%. Lee et al. (2003) combined an SVM and dictionary look-up for named entity recognition. Their approach is based on two phases, the first phase is identification of each entity with an SVM classifier and the second phase is post-processing to correct the errors by the SVM with a simple dictionary look-up. Bunescu et al. (2004) studied

protein name identification and protein-protein interaction. 5,206 names extracted from MEDLINE database were used in their experiment. Among several approaches used in their study, the main two ways are one using POS tagging, and the other using the generalized dictionary-based tagging. Their dictionary-based tagging presents higher F-value. Table 1 summarizes the works in the areas of named entity extraction in biomedical literature.

Table 1: A Summary of Works in Biomedical Entity Extraction

Author	***Named Entities***	***Database***	***No of Words***	***Learning Methods***	***F value (%)***
Fukuda et al. (1998)	Protein	MEDLINE	20000	Hand-crafted rules	93
Collier et al. (2000)	Proteins and DNA	MEDLINE	30000	HMM	73
Krauthammer et al. (2000)	Gene and Protein	Review articles	5000	Character sequence mapping	75
Demetriou and Gaizauskas (2002)	Protein, Species, and 10 more	MEDLINE	30,000	PASTA template filing	83
Narayanaswamy (2003)	Protein	MEDLINE	302	Hand –crafted rules and co-occurrence	76
Yamamoto et al. (2003)	Protein	GENIA	1600 abstracts	BaseNP recognition	75
Lee et al. (2003)	Protein DNA RNA	GENIA	10,000	SVM	77
Bunescu (2004)	Protein	MEDLINE	5,206	RAPIER, BWI, TBL, k-NN , SVMs, MaxEnt	56

The second target object type of biomedical literature extraction is relation extraction. Leek (1997) applies HMM techniques to identify gene names and chromosomes through heuristics. Craven and Kumlien (1999) extract location

relations for proteins in Yeast database based on HMM techniques. Blaschke et al. (1999) extract protein-protein interactions based on co-occurrence of the form "… p1…I1… p2" within a sentence, where p1, p2 are proteins and I1 is an interaction term. Protein names and interaction terms (e.g., activate, bind, inhibit) are provided as a "dictionary." Rindflesch (1999) extracts binding relations for Unified Medical Language System (UMLS) from MEDLINE. Proux (2000) extracts an "interact" relation for the gene entity from Flybase database. Pustejovsky (2002) extracts an "inhibit" relation for the gene entity from MEDLINE. Jenssen et al. (2001) extract gene-gene relations based on co-occurrence of the form "… g1…g2…" within a MEDLINE abstracts, where g1 and g2 are gene names. Gene names are provided as a "dictionary", harvested from HUGO, LocusLink, and other sources. Although their study uses 13,712 named human genes and millions of MEDLINE abstracts, no extensive quantitative results are reported and analyzed. Friedman et al. (2001) extract a pathway relation for various biological entities from a variety of articles. In their work, the precision of the experiments is high (from 79-96%). However, the recalls are relatively low (from 21-72%). Bunescu et al. (2004) conducted protein/protein interaction identification with several learning methods such as pattern matching rule induction (RAPIER), boosted wrapper induction (BWI), and

extraction using longest common subsequences (ELCS). ELCS automatically learns rules for extracting protein interactions using a bottom-up approach. They conducted experiments in two ways; one with manually crafted protein names and the other with extracted protein names by their name identification method. In both experiments, Bunescu et al. compared their results with human-written rules and showed that machine learning methods provides higher precisions than human-written rules. Table 2 summarizes work on relation extraction in biomedical literature.

Table 2: A Summary of Relation Extraction for Biomedical Data

Authors	***Relation***	***Entity***	***DB***	***Learning Methods***	***Precision***	***Recall***
Leek (1997)	Location	Gene	OMIM	HMM	80%	36%
Blaschke (1999)	Interact	Protein	MEDLINE	Co-occurrence	n/a	n/a
Craven (1999)	Location	Protein	Yeast	HMM	92%	21%
Rindflesch (1999)	Binding	UMLS	MEDLINE	Co-occurrence	79%	72%
Proux (2000)	Interact	Gene	Flybase	Co-occurrence	81%	44%
Pustejovsky (2001)	Inhibit	Gene	MEDLINE	Co-occurrence	90%	57%
Jenssen (2001)	Location	Gene	MEDLINE	Co-occurrence	n/a	n/a
Friedman (2001)	Pathway	Many	Articles	Co-occurrence & thesauri	96%	63%
Bunescu (2004)	Interact	Protein	MEDLINE	RAPIER, BWI, ELCS	n/a	n/a

2.3.2 Techniques Used

The most commonly used extraction technique is based on co-occurrence. The basic idea is that entities are extracted based on frequency of co-occurrence of biomedical named-entities such as proteins or genes within sentences. This technique was introduced by Blaschke et al. (1999). Their goal was to extract information from scientific text about protein interactions among a pre-determined set of related programs. Since Blaschke et al.'s study, numerous other co-occurrence based systems have been proposed in the literature. All are associated with information extraction of biomedical entities from the unstructured text corpus. The common denominator of the co-occurrence based systems is that they are based on co-occurrences of names or identifiers of entities, typically along with activation/dependency terms. These systems are differentiated one from another by integrating other machine learning techniques such as syntactical analysis or POS tagging (Tanabe et al., 1999; Thomas et al., 2000; Stapley and Benoit, 2000), as well as ontologies and controlled vocabularies (Hahn et al., 2002; Pustejovsky et al., 2002; Yakushiji et al., 2001). Although these techniques are straightforward and easy to develop, from the performance standpoint, recall and precision are much lower than any other machine learning techniques (Ray & Craven, 2001).

In parallel with co-occurrence based systems, the researchers began to investigate other machine learning or NLP techniques. One of the earliest studies was done by Leek (1997) who utilized Hidden Markov Models (HMMs) to extract sentences discussing gene location of chromosomes. HMMs are applied to represent sentence structures for natural language processing, where states of a HMM correspond to candidate POS tags and probabilistic transitions among states represent possible parses of the sentence, according to the matches of the terms occurring in it to the POSs. In the context of biomedical literature mining, HMM is also used to model families of biological sequences as a set of different utterances of the same word generated by a HMM technique (Baldi et al., 1994).

Craven et al. (1999) have proposed a more sophisticated HMM based technique to distinguish fact-bearing sentences from uninteresting sentences. The target biological entities and relations that they intend to extract are protein sub-cellular localizations and gene-disorder associations. With pre-defined lexicon of locations and proteins and several hundreds of training sentences derived from Yeast database, they trained and tested the classifiers over a manually labeled corpus of about 3000 MEDLINE abstracts. Ray and Craven (2001) further extended to use

HMMs to represent the sentence structure, and to identify sentences discussing gene-disease association.

There have been several studies applying natural language tagging and parsing techniques to biomedical literature mining. Rindeflesch et al. (2000) and Friedman et al. (2001) propose methods parsing sentences and using thesauri to extract facts about genes and proteins from biomedical documents. Rindeflesch et al. (2000) explain the effects of drugs on gene-activity within the cell. Friedman et al. (2001) extract interactions among genes and proteins as part of regulatory pathways.

2.3.3 Evaluation

One of the pivotal issues yet to be explored further for biomedical literature mining is how to evaluate the techniques or systems. The focus of the evaluation conducted in the literature is on extraction accuracy. The accuracy measures used in IE are precision and recall ratios. For a set of N items, where N is either terms, sentences, or documents and the system needs to label each of the terms as “positive” or as “negative” according to some criterion – “positive” if a term belongs to a predefined document category or a term class. Recall is defined as the number of relevant documents retrieved divided by the total number of relevant documents in the collection. For example, suppose there are 80 documents relevant to widgets in

the collection. If the system X returns 60 documents and 40 of which are about widgets, then X's recall is 40/80 = 50%. Precision is defined as the number of relevant documents retrieved divided by the total number of documents retrieved. In our example, X's precision is 40/60 = 67%. As discussed earlier, the extraction accuracy is measured by precision and recall ratio. Although these evaluation techniques are straightforward and are well accepted, calculating recall ratios may be criticized when the total number of true "positive" terms is not clearly defined.

Participants in MUC tested the ability of their systems to identify entities in text to resolve co-reference, extract and populate attributes of entities, and perform various other extraction tasks from written text. As identified by Shatkay and Feldman (2003), the important challenge in biomedical literature mining is "the creation of gold-standards and critical evaluation methods for systems developed in this very active field." The framework of evaluating biomedical literature mining systems was recently proposed by Hirschman et al. (2002). According to this framework the following elements are needed for a successful evaluation: 1) challenging problem, 2) task definition, 3) training data, 4) test data, 5) evaluation methodology and implementation, 6) evaluator, 7) participants, and 8) funding. In addition to these elements for evaluation, the existing biomedical literature mining

systems encounter the issues of portability and scalability, and these issues need to be taken as part of evaluation.

2.3.4 Data Sources

In terms of data sources that target biomedical objects are extracted from, most of the biomedical data mining systems focus on mining MEDLINE abstracts of National Library of Medicine. The principal reason for relying on MEDLINE is related to complexity. Abstract occasionally are more easy to mine since many papers contain less precise and less well supported sections in the text that are difficult to distinguish from more informative sections by machines (Andrade & Bork, 2000; Ding et al., 2002). The current version of MEDLINE contains nearly 12 million abstracts stored on approximately 43GB of disk space. A prominent example of methods that target entire papers is still restricted to a small number of journals (Friedman et al., 2000; Krauthammer et al., 2002). The task of unraveling information about function from MEDLINE abstracts can be approached from two different view points. One approach is based on computational techniques for understanding texts written in natural language with lexical, syntactical and semantic analysis (Cowie & Lehnert, 1996). In addition to indexing 'terms' in documents, natural language processing (NLP) methods extract and index higher level semantic structures

composed of terms, and relationships between terms (Baeza-Yates & Ribeiro-Neto, 1999). However, this approach is confronted with the variability, fuzziness and complexity of human language (Andrade & Bork, 2000). The Genies system (Friedman et al., 2000; Krauthammer et al., 2002) for automatically gathering and processing of knowledge about molecular pathways and the Information Finding from Biological Papers (IFBP) transcription factor database (Ohta et al., 1997) are natural language processing based systems.

An alternative approach that may be more relevant in practice is based on the treatment of text with statistical methods. In this approach, the possible relevance of words in a text is deduced from the comparison of frequency of different words in this text with the frequency of same words in reference sets of text. Some of the major methods using the statistical approach are AbXtract (Andrade et al., 1999) and the automatic pathway discovery tool of Ng and Wong (1999). There are advantages to each of these approaches (grammar or pattern matching). Generally, the less syntax is used, the more domain-specific the system is. This allows the construction of a robust system relatively quickly, but many subtleties may be lost in the interpretation of sentences. Recently GENIA corpus has been used for extracting biomedical named

entities (Collier et al., 2000; Yamamoto et al., 2003). The reason for recent surge of using GENIA corpus is because GENIA provides annotated corpus that can be used for all areas of NLP and IE applied to the biomedical domain that employ supervised learning. With the explosion of results in molecular biology there is an increased need for IE to extract knowledge to build databases and to search intelligently for information in online journal collections. For these purposes, GENIA was built with a corpus of annotated abstracts taken from MEDLINE database. GENIA corpus is annotated with a subset of the substances and the biological locations involved in reactions of proteins, based on a data model of the biological domain, in XML format.

CHAPTER 3: APPROACH AND METHOD

3.1 Overview

KE is the process by which relevant data are extracted from plain text files in a structured format such as relations in a database system. This process includes: locating the relevant data from the text databases, extracting the text segments which contain the data, and separating the facts contained in these segments and structuring the data in a database. The architecture of RIKE is shown in Figure 4. RIKE uses a pipelined architecture and extracts with minimal human intervention.

The underlining algorithm is as follows:

1. Starting with a set of user-provided seed instances (the seed instance can be quite small); our system retrieves a sample of documents from the text databases. At the initial stage of the overall document retrieval process, we have no information about the documents that might be useful for the goal of extraction. The only information we require about the target relation is initial input provided by the users, including the specification of the relation attributes to be used for document retrieval. We construct some simple queries by using the attribute values of the initial seed

instances to extract the document samples of a pre-defined size using the search engine.

2. The instance set induces a binary partition (a split) on the documents: those that contain instances or those that do not contain any instance from the relation. The documents are thus labeled automatically as either positive or negative examples, respectively. The positive examples represent the documents that contain at least one instance. The negative examples represent documents that contain no instances.

3. RIKE next applies data mining and IR techniques to derive queries targeted to match—and retrieve— additional documents similar to the positive examples

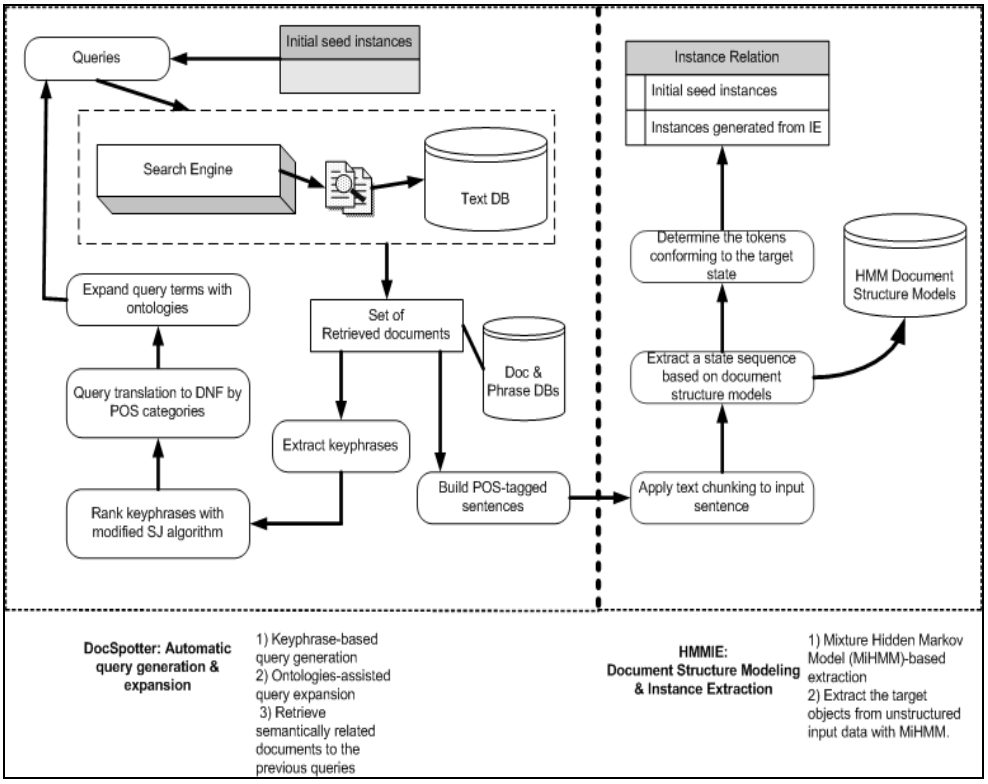

Figure 4: System Architecture of RIKE

4. RIKE then applies a HMM-based state sequence extraction technique over the documents. It models a set of document structures using the training documents. These models are kept in the model base, which will serve as an engine for extracting state sequence from the documents.

5. The system queries the text databases using the automatically learned queries from Step 3 to retrieve a set of promising documents from the databases and then returns to Step 2. The whole procedure repeats until no

new instances can be added into the relation or we reach the pre-set limit of a maximum number of text files to process.

3.2 RIKE System Architecture

In this section, we briefly describe the architecture of RIKE which is based on design pattern and object-oriented design methodologies. We adapt Unified Modeling Language (UML) notations and diagrams to explain our architecture. Throughout the development cycle of RIKE, object-oriented principles are applied. UML diagrams we develop include use case, class, and activity diagram.

3.2.1 Use Case Analysis

The UML modelling technique that provides useful knowledge about the usage of a system is the use case diagram (Fowler & Scott, 1997). Use case diagrams document the functionality and users of the system. Use case diagrams have four major elements: The actors that the system interacts with, the system itself, the use cases, or services, that the system knows how to perform, and the relationships between these elements.

As illustrated in Figure 5, an actor is shown as an agent who interacts with the system agent. This use case diagram shows RIKE consisting largely of three components: 1) extracting keyphrases, 2) expanding queries, and 3) extracting the target relations.

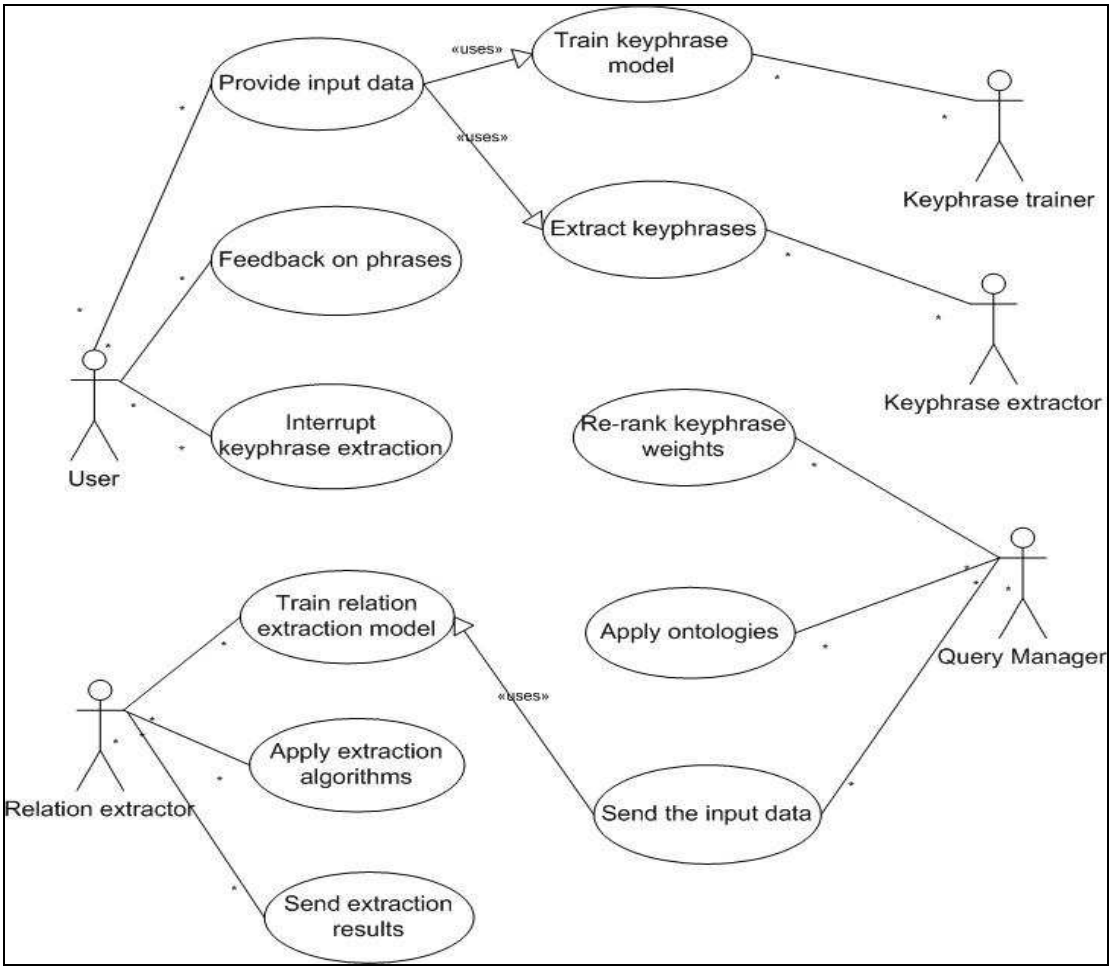

Figure 5: Use Case Diagram for RIKE

3.2.2 Class Diagram

In this section, we present the structure of RIKE at class diagram level. Class diagrams provide a static representation of the structure of a system. Class diagrams appear in various levels of detail depending on the phase of the lifecycle (Fowler, 2003). Figure 6 depicts a high-level conceptual class diagram of RIKE.

The following seven major components are shown in the class diagram: 1) ModelBuilder, 2) DBHandler, 3) POSHandler, 4) ModelManager, 5) KeyphraseHandler, 6) QueryExpander, and 7) RelationExtractor.

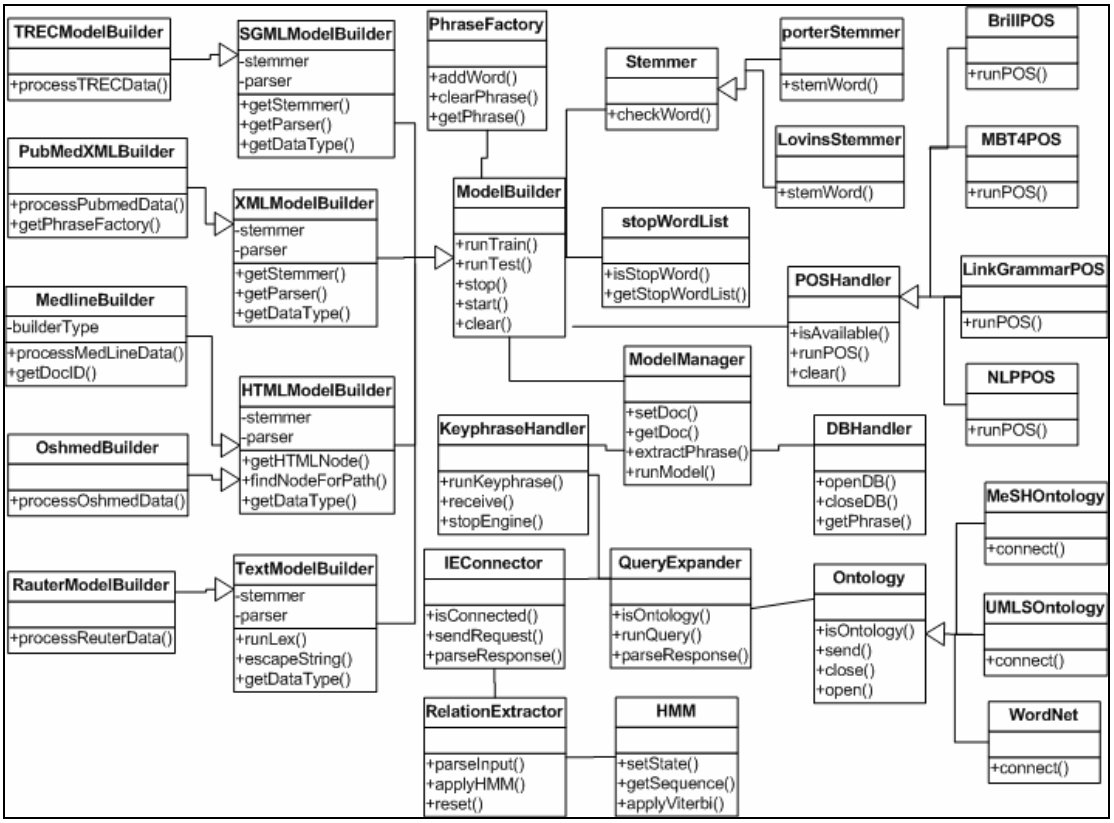

Figure 6: Class Diagram for RIKE

The ModelBuilder component consists of classes processing various input formats such as HTML and XML. The DBHandler component stores statistics on candidate phrases and input documents. The POSHandler component interfaces with the four POS Tagging libraries implemented in RIKE. The ModelManager component applies discretization and WordNet's conversion capability of verb to noun. The KeyphraseHandler component takes care of extracting keyphrases based on the information gain data mining measure. The QueyExpander component is responsible for managing queries. The initial query is expanded by re-ranking the keyphrases based on combination of Information Gain and modified Robertson and Spark Jones ranking algorithm. The RelationExtractor component handles receiving requests from QueryExpander and applying Mixture Hidden Markov models to extract relations.

3.2.3 Activity Diagram

To help understand how the system works, an Activity Diagram is provided (Figure 4). Activity diagrams represent the business and operational workflows of a

system and show the activity and the event that causes the object to be in the particular state (Hofmeister, 1999).

As illustrated in Figure 7, depending on the process mode of the system, RIKE handles test data or training data. Consequently, it either trained model or extracts keyphrases. Which path RIKE takes is determined by the configuration settings in the form of XML.

Figure 7: Activity Diagram for RIKE

3.3 DocSpotter

DocSpotter is a novel querying technique to iteratively retrieve promising documents for information extraction. DocSpotter applies several Data Mining and

Natural Language Processing techniques. First, DocSpotter introduces a keyphrase-based term selection technique combined with data mining and natural processing techniques. Second, it proposes a query weighting algorithm combined with modified Robertson Spark-Jones (RSJ) and Information Gain algorithm. Third, it translates a query to the Disjunctive Normal Form (DNF) by POS term categories. Figure 8 illustrates how the novel query expansion algorithm works in DocSpotter.

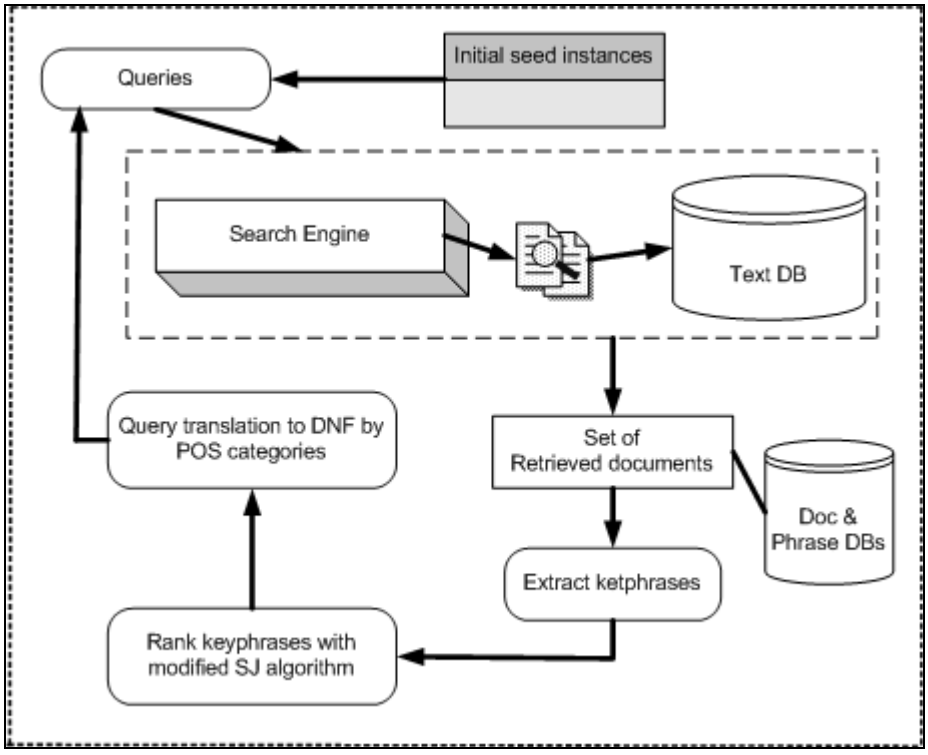

Figure 8: System Architecture of DocSpotter

The outline of the approach described in Figure 8 is as follows:

- Step 1: Starting with a set of user-provided seed instances (the seed instance can be quite small) our system retrieves a sample of documents from the databases. At the initial stage of the overall document retrieval process, we have no information about the documents that might be useful for extraction. The only information we require about the target relation is a set of user-provided seed instances, including the specification of the relation attributes to be used for document retrieval. We use some simple queries. Specifically, we just use the attribute values of the initial seed instances) to extract the document sample of pre-defined size from the search engine.
- Step 2: On the retrieved document set, we parse each document into sentences and apply the keyphrase extraction technique proposed in Song et al. (2003) to extract keyphrases from the input documents.
- Step 3: Applying a hybrid querying expansion algorithm that combines the modified Robertson and Spark-Jones ranking algorithm with Information Gain-based keyphrase ranking to derive queries targeted to match—and—retrieve additional documents similar to the positive examples.

- Step 4: Running the information extraction over the documents, it produces a set of extracted patterns from the documents and these patterns are kept in the pattern base, which will be used to generate more instances from the documents.
- Step 5: Run the new queries from Step 4 to retrieve a set of promising documents form the databases, go to Step 2. The whole procedure repeats until the no new additional documents are retrieved.

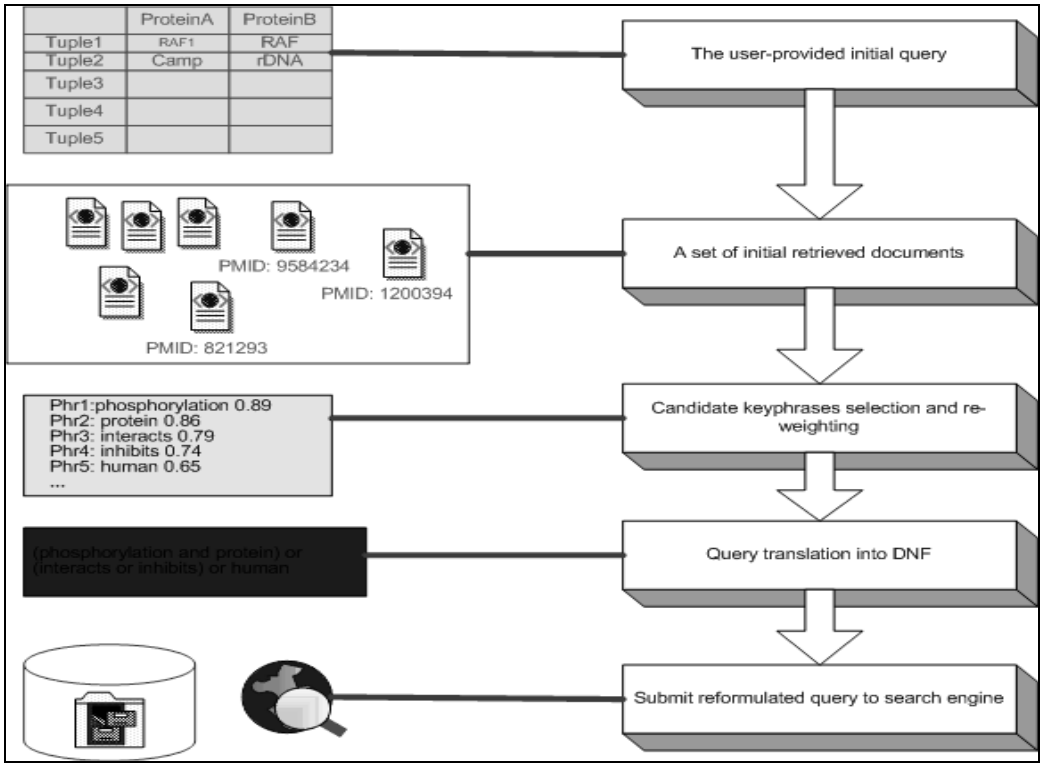

Figure 9: Process Diagram of DocSpotter

Figure 9 shows how DocSpotter works and what output it generates in each step.

In the following section, we discuss how candidate keyphrases are generated, an extraction model is built, and keyphrases for a new document are extracted. As illustrated in Figure 10, DocSpotter comprises the following three stages: 1) building extraction model, 2) extracting keyphrases, and 3) expanding queries. Input of the "building extraction model" stage is training data and input of the "extracting keyphrases" stage is test data or production data. Both training and test data are processed by the three components: 1) Data Cleaning, 2) Data Tokenizing, and 3) Data Discretizing. In Figure 10, the dotted line represents the processing logic for "building the extraction model" whereas the solid line indicates the processing logic for "extracting keyphrases." The detailed descriptions are provided in the following subsections. These two stages are fully automated. Depending on the configuration parameters, DocSpotter processes either "building extraction model" mode or "extracting keyphrases" model. The outcomes of both processes by DocSpotter are stored in XML.

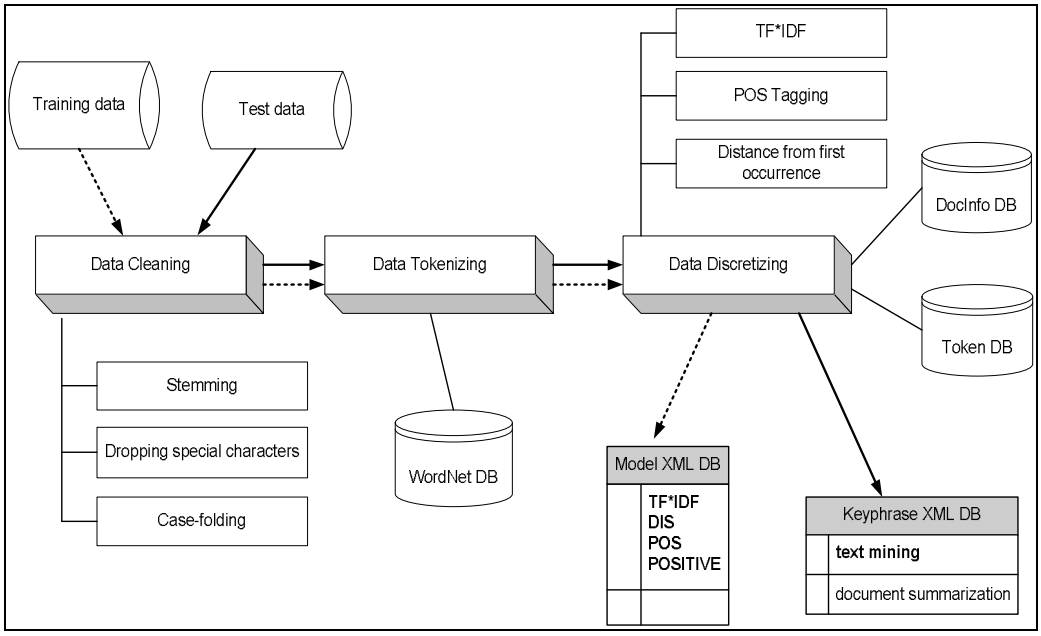

Figure 10: Procedures of Data Processing for Keyphrase Extraction

DocSpotter stores candidate keyphrases and associated information such as frequency and stemmed form into BerkeleyDB, an embedded database management system (Sleepycat, 1990). Using BerkeleyDB enhances performance of the system and provides a persistent data storage. After data tokenizing, extracted candidate phrases are stored into several token-related DBs such as TokenID DB and TokenList DB. And information on document units is stored in the Document DB. This approach is particularly beneficial for the real application system to process a large number of candidate keyphrases. To our knowledge, DocSpotter is the first system that applies a persistent data storage technique to store and retrieve keyphrase

candidates. The major benefit of using a persistent storage such as BerkeleyDB is that it enables the system to gain better performance in a real-world application where the system has to handle gigabyte candidate keyphrases.

3.3.1 Procedure of Selecting Candidate Phrases

In this section, we describe the selection criteria and rules for candidate phrases applied in DocSpotter. Raw text data are filtered to regularize the text and determine initial phrase boundaries. The input stream is split into tokens (sequences of letters, digits, and internal periods). DocSpotter then considers all the subsequent phrases in the document and determine which candidate phrases are suitable. Witten et al. (1999) employed the following rules for selecting candidate phrases:

1.Candidate phrases are limited to a certain maximum length (N consecutive words).

2.Candidate phrases cannot be proper names (i.e. single words that only ever appear with an initial capital)

3.Candidate phrases cannot begin or end with a stop word

DocSpotter adopts these rules for extracting candidate phrases. Our stop word list consists of 650 unimportant words such as articles, prepositions, and conjunctions. All continuous sequences of words in the document are evaluated as candidate phrases with the three rules above. With regard to candidate selection criteria, DocSpotter is different from KEA in that KEA considers the contiguous sequences of words only in each input line, whereas DocSpotter extracts N consecutive words in a single sentence. It is important to tokenize words on a sentence basis because our POS tagging algorithms take a single sentence as input to apply POS tags to each token.

3.3.2 Case-folding and stemming

The final step in determining the candidate phrases is to case fold all words and to stem them. For this study, we chose Porter algorithm (Porter, 1980). Stemming and case-folding permit us to treat different variations on a phrase as the same thing. For example, “data transformation” and “data transform” are semantically very similar, but without stemming they would have to be treated as different phrases.

3.3.3 Procedure of Feature Selection

The following three feature sets were chosen and calculated for each candidate phrase: 1) TF*IDF, 2) Distance from First Occurrence, and 3) POS Tagging.

*TF*IDF*

TF*IDF was first proposed by Salton and Buckley (1988). It is a measure of importance of term in a document or class. As indicated by formula 3.1, TF*IDF is term frequency in a document or class, relative to overall frequency. The TF*IDF feature is a well-known weighting scheme in information retrieval.

$$W_{ij} = tf_{ij} * \log_2 \frac{N}{n} \qquad (3.1)$$

W_{ij} weight of term T_i in document D_j and tf_{ij} is frequency of term T_j in document D_j. N is the number of documents in a collection and n is the number of documents where term T_j occurs at least once.

Distance from First Occurrence

The distance from the first occurrence (FO) feature is calculated as the number of words that precedes the phrase's first appearance, divided by the number of words in the document (Formula 3.2).

$$DFO = \sum w_{i-1} / NP \qquad (3.2)$$

w_{i-1} is the number of phrases preceding the target phrase and NP is the total number of phrases in the document.

POS Tagging

POS tagging plays an important role in NLP applications like speech recognition, natural language parsing, information retrieval and information extraction (Manning and Schütze, 1999). Part-of-speech tagging is the process of assigning a part-of-speech like noun, verb, pronoun, preposition, adverb, adjective or other lexical class marker to each word in a sentence (Charniak, 2000). The input to a tagging algorithm is a string of words of a natural language sentence and a specified tagset (a finite list of Part-of-speech tags). The ideal output would be a single best POS tag for each word.

We combine several POS tagging techniques such as 1) NLParser (Charniak, 2000), 2) Link-Grammar (Lafferty et al., 1992), and 3) PCKimmo (Antworth, 1993) to improve POS tagging accuracy. According to Kwok et al. (2001), the best POS parsers employ statistical techniques, and among those statistical techniques based parsers, NLParser was evaluated as the best one by Kwok et al. (2001). The Link-

Grammar is a syntactic parser, based on link grammar, an original theory of English syntax (Lafferty et al., 1992). Given a sentence, Link-Grammar assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. PCKimmo is a rule based POS tagger but its data-processing capabilities are limited. With some minor modifications to these parsers, we seamlessly integrate these three parsing systems into DocSpotter. This combined approach to POS techniques enables us to assign the best tag to lexical tokens, constituting candidate phrases by utilizing outstanding features of each POS technique.

3.3.4 Discretization

Since the features selected in DocSpotter described above are continuous, we need to convert them into nominal forms to make our machine learning algorithm applicable. Among many discretization algorithms, we chose equal-depth (frequency) partitioning method which allows for good data scaling (Dougherty et al., 1995) eual-depth discretization divides the range into N intervals, each containing approximately the same number of samples. A similar approach to discretization was proposed by KEA (Frank et al., 1999), but it only used two features, TF*IDF and DIS.

During the training process, each feature is discretized. In DocSpotter, the value of each feature, a candidate phrase, is replaced by the range to which the value belongs. Table 3 shows the results of discretization by equal-depth partitioning method which is used for the training and testing data.

Table 3: Discretization Table

	Discretization Range				
Feature	1	2	3	4	5
TF*IDF	< 0.003	>= 0.003 && < 0.015	>= 0.015 && < 0.050	>= 0.050 && < 0.100	>= 0.100
DFO	< 0.150	>= 0.150 && < 0.35	>=0.350 && < 0.500	>= 0.500 && < 0.700	>= 0.700
POS	< 0.001	>= 0.001 && < 0.200	>=0.200 && < 0.700		

In the process of model building, each phrase extracted from each training document is marked as either a keyphrase or non-keyphrase, by comparing with the actual keyphrases for the document provided by the author. DocSpotter builds the model that predicts the class using the values of the three feature sets. The extraction model is stored in a XML file to make it portable. Figure 11 shows the sample model XML built with the training data. In the given sample model below, a XML element, called TUPLE, represents a candidate phrase. Each tuple has three attributes, TF_IDF,

DIS, and POS. The attributes for TF_IDF and DIS have five values each, and the attributes for POS have three values. Each tuple also has a value indicating whether it matches pre-determined keyphrases or not.

```
<MODEL>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_2" POS="pos_2">negative</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_2" POS="pos_2">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_5" DIS="dis_2" POS="pos_2">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_2" POS="pos_1">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_5" DIS="dis_4" POS="pos_1">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_5" DIS="dis_5" POS="pos_1">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_5" DIS="dis_1" POS="pos_1">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_1" POS="pos_1">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_1" POS="pos_1">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_1" POS="pos_2">negative</TUPLE>
  <TUPLE TF_IDF="tf_idf_5" DIS="dis_1" POS="pos_2">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_4" POS="pos_2">negative</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_1" POS="pos_1">negative</TUPLE>
  <TUPLE TF_IDF="tf_idf_5" DIS="dis_2" POS="pos_1">positive</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_2" POS="pos_1">negative</TUPLE>
  <TUPLE TF_IDF="tf_idf_4" DIS="dis_2" POS="pos_1">negative</TUPLE>
</MODEL>
```

Figure 11: Keyphrase Model to Train Data

3.3.5 Query Expansion with Keyphrases

DocSpotter uses a modification of Robertson-Spark Jones QE algorithm (Robertson and Spark-Jones, 1976). DocSpotter generates a list of keyphrases for each document and stores it in a XML file (Figure 12). As illustrated in Table 4, for each document identified by the filename (the value of the ID attribute is filename), a

list of keyphrases is extracted from the document with the weight calculated by Information Gain.

```
<KEY_PHRASE>
- <DOC ID="10004">
    <PHRASE RANK="0.965011">algorithm</PHRASE>
    <PHRASE RANK="0.949753">amount computation</PHRASE>
    <PHRASE RANK="0.954427">checkpoints recovery</PHRASE>
    <PHRASE RANK="1.00497">collection</PHRASE>
    <PHRASE RANK="0.985145">collection algorithm</PHRASE>
    <PHRASE RANK="1.00497">collection paper</PHRASE>
    <PHRASE RANK="0.928852">computation</PHRASE>
    <PHRASE RANK="0.928852">dissemination</PHRASE>
    <PHRASE RANK="0.928852">information</PHRASE>
    <PHRASE RANK="0.949753">information dissemination</PHRASE>
    <PHRASE RANK="0.933631">recovery</PHRASE>
    <PHRASE RANK="0.966488">snapshot</PHRASE>
    <PHRASE RANK="1.00497">snapshot collection</PHRASE>
    <PHRASE RANK="0.985145">snapshot collection algorithm</PHRASE>
    <PHRASE RANK="0.949753">snapshot information</PHRASE>
  </DOC>
</KEY_PHRASE>
```

Figure 12: A List of Keyphrases Extracted by DocSpotter

Keyphrase Ranking

Automatic query expansion requires a term-selection stage. The ranked order of terms is of primary importance in that the terms that are most likely to be useful are close to the top of the list. We re-weight candidate keyphrases with Information Gain.

Specifically, candidate keyphrases are ranked by an Information Grain, GAIN(P), measure of expected reduction in entropy based on the "usefulness" of an attribute A.

This is one of the most popular measures of association used in data mining. For instance, Quinlan (1993) uses Information Gain for ID3 and its successor C4.5 which are widely-used decision tree techniques. ID3 and C4.5 construct simple trees by choosing at each step the splitting feature that "tells us the most" about the training data. Mathematically, Information Gain is defined as:

$$GAIN(P_i) = I(p,n) - E(P_i) \tag{3.3}$$

Where P_i is value of candidate phrase that falls into a discretized range. I(p,n) measures the information required to classify an arbitrary tuple.

$$I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s} \tag{3.4}$$

S contains S_i tuples of class C_i for i=(1,…,m).

E(w) is the entropy of attribute W with values of a_1, a_2,…, a_v where a_1 is the probability of W_1 occurring .

$$E(w) = \sum_{j=1}^{v} \frac{s_{1j} + ... + s_{mj}}{s} I(s_{ij}, ..., s_{mj}) \tag{3.5}$$

Each candidate phrase, extracted from a document, is ranked by the probability calculated with GAIN(P). In our approach, I(p,n) is stated such that class p: candidate phrase = “keyphrase” and class n: candidate phrase = “non-keyphrase.”

For example, suppose that a candidate phrase, “text mining algorithm”, has 0.0134 for TF*IDF feature. According to Table 3, it falls into Range 3 for TF*IDF. Let us also assume that it appears twice as a keyphrase in the range, tf_idf_3. In the given scenario, the number of tuples in tf_idf_3 is 20. The number of keyphrases is 120 out of the total 1000 candidate phrases. For this case, I(120,880) is 0.529361, E(tf_idf3) is 0.046900, and GAIN(Ptf_idf_3) = 0.482462. GAIN(P) for DFO and POS is calculated in the same way. The overall rank of a candidate phrase is determined by the sum of GAIN(P_i). In the above example, the candidate phrase is ranked fifth as a keyphrase for the given test document.

Many query re-ranking algorithms are reported in the literature (Ro, 1998; Sager et al., 1976). These algorithms attempt to quantify the value of candidate query expansion terms. Formulae estimate the term value based on qualitative or quantitative criteria. The qualitative arguments are concerned with the value of the particular term in retrieval. On the other hand, the quantitative argument involves some specific criteria such as a proof of performance. One example of the qualitative-based formula is the relevance weighting theory.

While there are many promising alternatives to this weighting scheme in the IR literature (Xu & Croft, 2000), we chose the Robertson-Sparck Jones algorithm (1976) as our base because it has been demonstrated to perform well, is naturally well suited to our task, and incorporating other term weighting schemes would require changes to our model.

The F4.5 formula was proposed by Robertson and Jones. It has been widely used in IR systems with some modifications (Okapi). Although a few more algorithms were derived from F4.5 formula by Robertson and Jones, in this paper, we modify the original for keyphrases as shown:

$$P(w)^{(1)} = \log \frac{\left(\frac{r+0.5}{R-r+0.5} \right)}{\left(\frac{n-r+0.5}{N-n-R+r+0.5} \right)} \tag{3.6}$$

P(w) is the keyphrase weight, N is the total number of sentences, n is the number of sentences in which that query terms co-occur, R is the total number of relevant sentences, and r is the number of relevant sentences in which the query terms co-occur.

We combine Information Gain with the modified F4.5 formula to incorporate keyphrase properties gained as follows:

$$KP(r) = \sqrt{\frac{GAIN(p) * P(w)}{2}} \qquad (3.7)$$

All candidate keyphrases are re-weighted by KP(r) and the top N ranked keyphrases are added to the query for the next pass. The N number is determined by the size of the retrieved documents.

Query Translation into DNF

A major research issue in IR is easing the user's role of query formulation through automating the process of query formulation. There are two essential problems to address when searching with online systems: 1) Initial query formulation that expresses the user's information need; and 2) query reformulation that constructs a new query from the results of a prior query (Salton et al., 1983). The latter effort implements the notion of relevance feedback in IR systems and is the topic of this section.

An algorithm for automating Boolean query formulation was first proposed in 1970. This method employs a term weighting function first described in Frants and Sapiro (1991) to decide the "importance" of terms which have been identified. The terms were then aggregated into "sub-requests" and combined into a Boolean expression in disjunctive normal form (DNF). Other algorithms that have been proposed to translate a query to DNF are based on classification (French et al., 1997),

decision-trees (Chang et al., 1996), and thesauri (van der Pol, 2003). Hearst (1996) proposed a technique for constructing Boolean constraints, which was revisited by Mitra, et al. (1998).

Our POS category-based translation technique differs from others in that ours is unsupervised and is easily integrated into other domains. In our technique, there are four different phrase categories defined; 1) Ontology phrase category, 2) Non-Ontology noun phrase category, 3) Non-Ontology proper noun phrase category, and 4) Verb phrase category. Phrases that have corresponding entities in ontologies such as WordNet and MESH belong to the ontology phrase category. We include the Verb phrase category as a major category because important verb phrases play a role in improving the retrieval performance (Gauch et al., 1997). Keyphrases within the category are translated into DNF and categories are then translated into Conjunctive Normal Form.

The sample keyphrases for query 350 for TREC-6 are shown in Figure 9. As explained earlier, within the same category the phrases are combined with the OR Boolean operator. Between categories, the terms are combined with the AND Boolean operator. Thus, the query shown in Figure 13 is translated as follows:

((occupational health OR workplace disorders OR physical injury OR computer terminal OR terminals activity) AND (computer screen OR workers computer) AND report).

```
<keyphrases id="350">
        <keyphrase weight="0.39282" category=”2” >
          computer screen </keyphrase>
        <keyphrase weight="0.38114" category=”1” >
          occupational health </keyphrase>
        <keyphrase weight="0.38566" category=”1” >
          workplace disorders </keyphrase>
        <keyphrase weight="0.38432" category=”1” >
          physical injury</keyphrase>
        <keyphrase weight="0.38427" category=”1” >
          computer terminal </keyphrase>
        <keyphrase weight="0.38320" category=”2” >
          workers computer </keyphrase>
        <keyphrase weight="0.38293" category=”4”>
          report </keyphrase>
        <keyphrase weight="0.38174" category=”1” >
          terminals activity </keyphrase>
</keyphrases>
```

Figure 13: Sample Keyphrases Extracted for Query Expansion

3.3.6 Query Expansion with Ontologies

DocSpotter adopts three ontologies for query expansion: WordNet, MeSH, and UMLS. The detailed descriptions of these ontologies are provided in the section of Impact of Ontologies on Retrieval Performance. In this section, instead of re-stating what each ontology does, we focus on what algorithms are used in each ontology for query expansion.

Query Expansion with WordNet

For the top N-ranked keyphrases, our technique can traverse ontologies such as WordNet. If there is a corresponding phrase, the keyphrase is categorized as an Ontology phrase category. With WordNet, we encounter a complication with multiple senses of given a phrase.

To tackle this problem, we introduce a straightforward Word sense disambiguation technique, based on similarities between WordNet phrases and the keyphrases extracted by our technique. In WordNet, a group of synonyms with the same meaning composes a “synset”. The synsets are linked to each other through relationships such as hyponyms, hypernyms, and holonyms. If no synsets are found for the given phrase, we traverse down in the synset list to find the sysnet. For multiple synsets, all the non-stopwords are captured from synonyms and their

descriptions, hyponyms and their descriptions, and other relations for each synset. These terms and phrases are then compared with the keyphrase list by the similarity function Sim(S).

$$Sim(S) = \sum_{i=1}^{M} \max_{j \in \{1,..,n_i\}} w(p_{ij}) \tag{3.8}$$

where $w(p_{ij})$ is the frequency of phrase p_{ij} if it occurs in a synset, S, and is 0 otherwise. The synset with the highest similarity value is chosen and synonyms from the synset are added for query expansion.

Query Expansion with MeSH

We employ two approaches to applying MeSH to query expansion. First, we query MeSH database with the top N selected keyphrases. If the corresponded MeSH terms are found, we add the MeSH terms to expanded queries. The second approach is a more sophisticated than the first one. A majority of studies of applying MeSH for query expansion belong to the second approach (Hersh, 1994; Srinivasan, 1997). Correctly identifying MesH terms when searching MEDLINE can significantly increase the precision of search results. Works attempting to correlate extracted noun phrase entities from queries with only the definitions of MeSH defined in the UMLS Metathesaurus showed mixed results (Aranson, 1997; Hersh, 2000).

However, we saw a significant increase in the precision of search results using relevance feedback techniques that iteratively augment the original query with MeSH for highly ranked documents from previous search results (Srinivasan, 1997).

For each query, we examine the top N set of relevant documents for that query determined by keyphrases and maintain a set of vectors of MeSH terms assigned to those documents. We then calculate the TF*IDF value of MeSH terms in the list of vectors. We apply the same TF*IDF formula used above to calculate TF*IDF value for MeSH terms. We sort the MeSH terms in decreasing order of the TF*IDF values and use top N MeSH terms for query expansion. This approach is employed with our pseudo-relevance feedback technique.

Query Expansion with UMLS

We integrate UMLS to DocSpotter for query expansion. Query expansion assisted with UMLS improves retrieval performance (Aronson et al., 2000; Chu et al., 2005; Hersh et al., 1997). In particular, we utilize UMLS Metathesaurus, one of three knowledge sources provided by NLM, to expand queries in our automatic pseudo-relevance feedback procedure. Specifically, three UMLS Metathesaurus APIs,

GetSemanticType, GetCooccurrences, and FindConcept, are used for expansion. After we retrieve responses, we combine the UMLS terms for question expansion.

3.4 HiMMIE

The goal of HMMIE is to take a set of abstracts in MEDLINE and learn a generative probabilistic model of the underlying state transition structure of the document from a set of tagged training data. This model is not just one HMM however, it is a mixture of HMM's organized in a hierarchical structure to help the system cope with data sparseness. Given a trained probabilistic mixture model of the data, the system should then be able to apply this model to new unseen MEDLINE documents to predict which portions of these documents are likely targets according to the training data template.

The Mixture Hidden Markov Model (MiHMM) is defined as a mixture of HMMs organized in a hierarchical structure to help the IE system cope with data sparseness. MiHMM takes a set of documents with contextual cues such as Part of Speech (POS) tags and learn a generative probabilistic model of the underlying state transition structure of the document from a set of tagged training data. Given a trained probabilistic mixture model of the data, the system is able to apply this model to new

unseen input documents to predict which portions of these documents are likely targets according to the training data template.

MiHMM is different from existing HMMs as follows: (a) it employs probabilistic mixture of HMMs that is hierarchically structured. (b) it incorporates contextual and semantic cues into the learned models to extract knowledge from the unstructured text collections without any document structures.

Thus using MiHMM for IE has the following advantages over other approaches: (a) it overcomes the problem of the traditional HMMs with modeling the rich representation of text where features overlap among state units such as word, line, sentence, and paragraph. By incorporating sentence structures into the learned models, MiHMM provides better extraction accuracy than the tranditional HMMs. (b) it resolves the issues with the tranditional HMMs for IE that operate only on the semi-structured such as HTML documents and other text sources in which language grammar does not play a pivotal role. The solution proposed by MiHMM is to incorporate contextual as well as semantic cues such as Part of Speech (POS) tags into HMMs. In doing so, MiHMM models the unstructured documents such as MEDLINE abstracts and extracts the target entities from the unstructured documents.

3.4.1 Algorithm Description

This section briefly covers the basic algorithms used by HMMIE, a MiHMM-based text extraction system.

The Model Structure of State and Observation Sequences

Figure 14 illustrates the basic format by which a document is modeled with a state and observation sequence. The example shown in Figure 14 uses a web page that contains instructor name, which is the common example used by most of the HMM-based IE systems.

There are two fundamental issues to note in this model:

1. The observation sequence is always observed but the state sequence is only known for the training data. In this data, the start and end states are trivially known, the target state is marked up as described in the next section and is therefore known, the prefix and suffix states can be easily determined by knowledge of the target state and the context width defined in the training template, and all other states are considered to be background states. Knowledge of both the observation and hidden states makes training the model a simple case of maximum likelihood probability estimation.

2. The goal of the template extractor will be to generate the state sequence for an untagged document using the previously trained model. From this sequence it should be fairly obvious that the text to be extracted is the text that is labeled as a target state by the extraction algorithm.

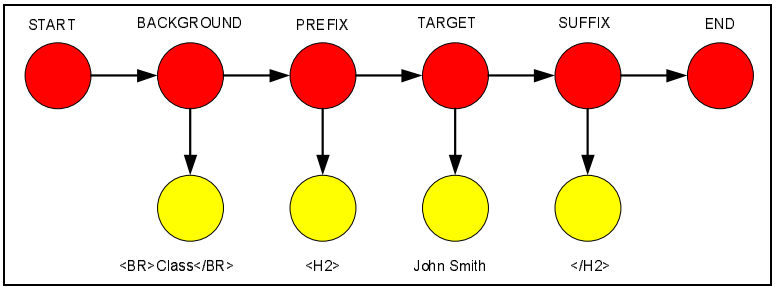

Figure 14: An Example of State and Observation Model Structure

Modeling the Document Structure

In a generic Hidden Markov Model, it is typical to define a number of states and a number of transitions between those states. The more complex the HMM, the better it can represent a document, but also the more data that is needed to dependably train it. Consequently there is a tradeoff between representational efficacy and training efficiency and this tradeoff varies from domain to domain.

Therefore, rather than employing on just one model, it is often easier to use a mixture of models and decide later how much to weight each model. This approach

is quite effective because a document can be modeled at varying degrees of granularity by effectively using a hierarchical model. At the same time it also retains the advantages of each model. That is, if the data is sparse, the simpler model will likely perform better and thus be weighted more during the extraction phase. If there is an abundance of data, the more complex model can be robustly trained and be weighted more heavily during the extraction phase.

For HMMIE, a basic set of three mixture models were used and are shown in Figure 15. In Figure 15, S denotes the start state, E denotes the end state, B denotes the background state, C denotes the context state, T denotes the target state, Pf denotes the prefix state, Sf denotes the suffix state. A similar mixture model was proposed by Freitag and McCallum (2000). Comparing our approach to Freitag and McCallum's, their model utilizes fairly complex prefix and suffix structures. Where our simpler models break down however is where the text to be extracted is heavily context dependent such that a reversal of two context words could reverse the classification of an excerpt of text as a target. The models used here are obviously incapable of modeling these cases since all prefixes and suffixes are treated in an order-independent manner.

Despite the simplicity of these models however, they yield the one main benefit that they can be trained on very sparse data.

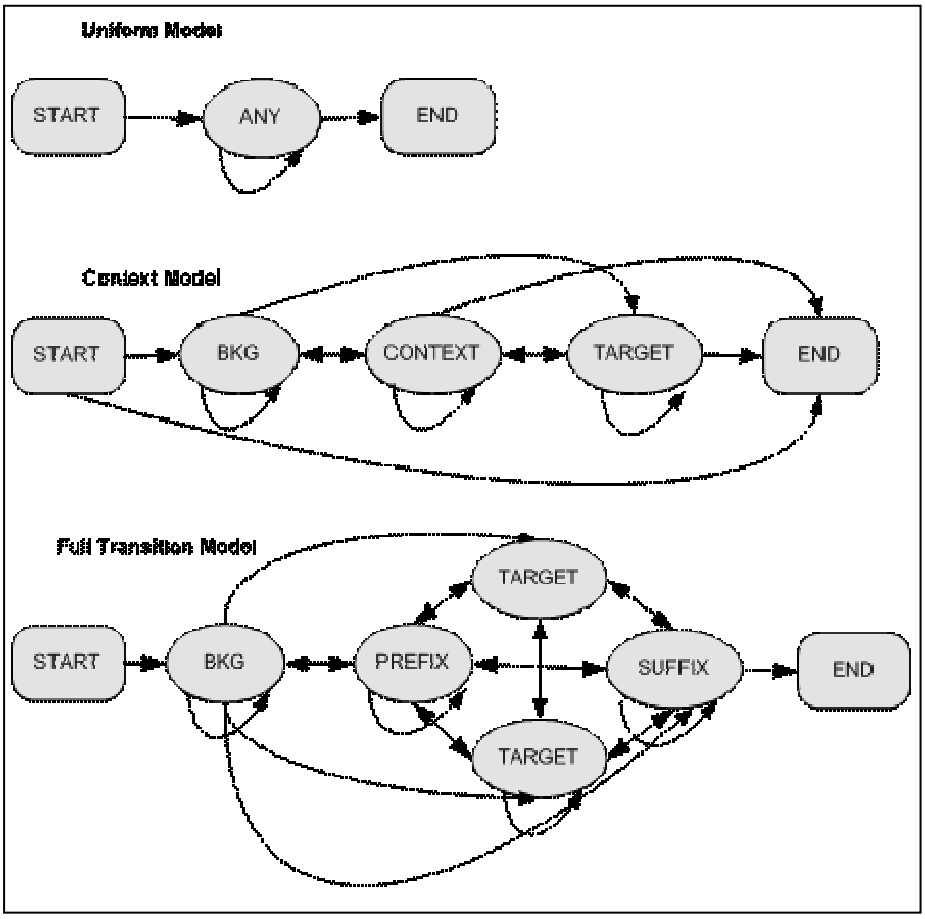

Figure 15: A Mixture Hidden Markov Model

Training the Model

As discussed previously, training the model is a straightforward process of maximum likelihood parameter estimation. From the document training set we can easily obtain the sufficient statistics concerning the frequency that a given state or

observation occurred and the frequency with which a state transition or observation emission was made.

Thus, state transition probability can be trained as follows:

$$P(S_{t+1}|S_t) = \frac{Freq(S_{t+1} \& S_t)}{Freq(S_t)} \qquad (3.9)$$

And likewise, the observation emission probability can be trained as follows:

$$P(O_t|S_t) = \frac{Freq(O_t \& S_t)}{Freq(S_t)} \qquad (3.10)$$

There might be some confusion as to how all three models are trained from the same set of sufficient statistics, but it is straightforward to show that the sufficient statistics for the full transition model can be converted to the sufficient statistics for any of the simpler models. Because the prefix and suffix states are collapsed to the same context state in the context model, one can simply combine these sufficient statistics and estimate the transition probabilities for the context model accordingly.

This suffices for training the individual models but it does not explain how to combine the models. To do this, we use a concept known as “shrinkage” which states quite simply that each model contributes a weighted estimate to the overall

calculation subject to the constraint that the sum of the weights is 1.0. A bit more formally, we can estimate a mixture state transition parameter using the following equation:

$$P(S_{t+1}|S_t)=\lambda_{Uniform}\cdot P_{Uniform}(S_{t+1}|S_t)+\lambda_{Context}\cdot P_{Context}(S_{t+1}|S_t)+\lambda_{Full}\cdot P_{Full}(S_{t+1}|S_t) \quad (3.11)$$

$$\lambda_{Uniform} + \lambda_{Context} + \lambda_{Full} = 1.0 \qquad (3.12)$$

Subject to the constraint that:

One can perform an analogous computation to compute the mixture estimation of the observation emission probabilities. One can empirically learn the λ weights using the EM algorithm. However for purposes of testing and time, these weights are simply specified in the template file which contains the model structure and parameters.

Extracting a State Sequence

Once an HMM has been trained on a set of tagged training data, it can then be applied to data to label the most probable state sequence for a document simply given a set of observations, which is, in fact, the document itself.

The trained mixture appears to be a single HMM for all intents and purposes. At least the purpose of the mixture model class is to make transparent the fact that the estimates for the transition and emission probabilities are actually based on multiple sub-models. Consequently, we can apply the standard HMM algorithm to the mixture model to extract the most probable state sequence given a set of observations. This is known as the Viterbi algorithm (Viterbi, 1967) and is covered in many texts (e.g., Rabiner, 1989).

It suffices to say that the Viterbi algorithm is a dynamic programming algorithm requiring time **O**(TS2) (where T is the number of time steps and S is the number of states) where at each time step it computes the most probable path for each state given that the most probable path for all previous time steps has been computed.

3.4.2 System Description

Figure 16 illustrates the system architecture of HiMMIE. The system consists of two major components: 1) sentence chunking by the SVM component and 2) relation extraction by the MiHMM component.

For sentence chunking by the SVM component, the input data is plain text consisting of titles and abstracts. The input data is seperated into sentences. A set of

regular expression rules is applied to parse sentences. For a parsed sentence, we applied an integrated POS tagging technique proposed by Song et al. (2003) to tag sentences with POS. With SVM-based text chunking technique, these POS tagged sentences are then grouped into chunks of different phrase types such as noun, verb, and preposition chunk.

In relation extraction by MiHMM component, MiHMM is applied to the grouped phrases by the SVM text chunking technique. The target state, which is a target noun group containing proteins, is extracted with hierarchically structured HMMs. Finally protein-protein pairs are extracted from the target states within a sentence by re-applying MiHMM to the target state groups.

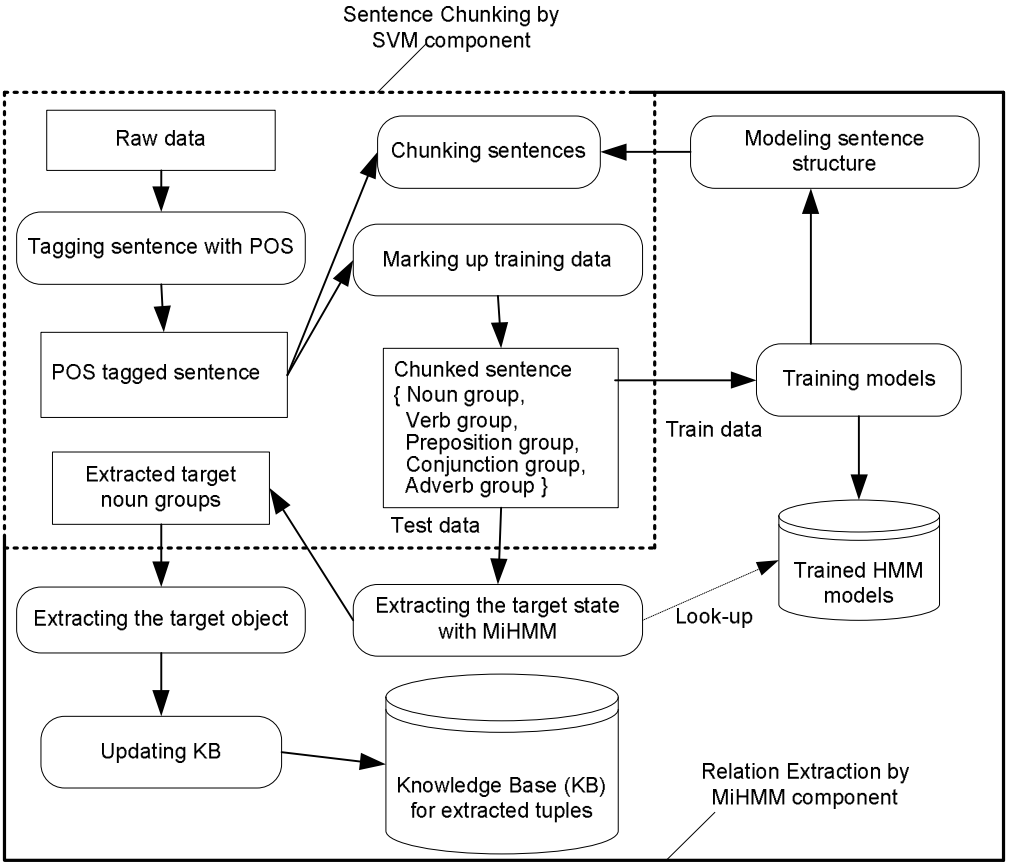

Figure 16: System Architecture of HiMMIE

The result of running HiMMIE is a set of tuples related to protein-protein pair. HiMMIE stores these tuples in the knowledge base and reset the token statistics for the next input data. The detailed description of the components is provided in the sub-sections below.

Figure 17 illustrates the procedure of converting a raw sentence from PubMed to the phrase-based units grouped by the SVM text-chunking technique. The top box shows

a sentence that is part of abstracts retrieved from Pubmed. The middle box illustrates the parsed sentence by POS taggers. The bottom box shows the final conversion made to the POS tagged sentence by the SVM based text chunking technique.

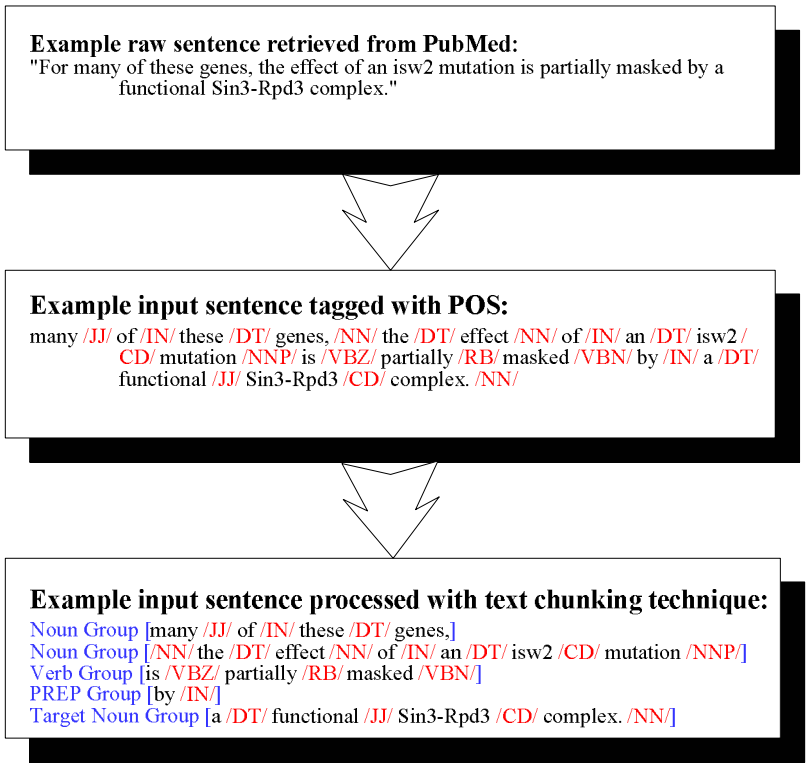

Figure 17: A Procedure of Sentence Parsing

JJ denotes adjective, IN denotes preposition, DT denotes determiner, CD cardinal number, NN denotes singular noun, NNP denotes proper noun, VBZ and VBN denote verb, RB denotes adverb

We use the following tagging types for training HiMMIE.

Table 4: Tagging Types Used in HiMMIE

Non Target Tagging Type	Target Tagging Type
NP_GROUP	NP_CONJ_GROUP_PROTEIN1_PROTEIN2
VP_GROUP	NP_CONJ_GROUP_PROTEIN2_PROTEIN1
PP_GROUP	NP_GROUP_PROTEIN1
CONJ_GROUP	NP_GROUP_PROTEIN2
ART_GROUP	NP_CONJ_GROUP_PROTEIN1
CM_GROUP	NP_CONJ_GROUP_PROTEIN2
ADV_GROUP	NP_GROUP_PROTEIN1_PROTEIN2
ADJ_GROUP	NP_GROUP_PROTEIN2_PROTEIN1
PREP_GROUP	VP_GROUP_PROTEIN1
NP_CONJ_GROUP	VP_GROUP_PROTEIN2
LEX_GROUP	VP_GROUP_PROTEIN1_PROTEIN2
PART_GROUP	ADJ_GROUP_PROTEIN1

3.4.3 Sentence Chunking by SVM component

Text chunking is defined as dividing a text in syntactically correlated parts of words (Kudo & Matsumoto, 2000). Chunking is recognized as series of processes - first identifying proper chunks from a sequence of tokens (such as words), and second classifying these chunks into some grammatical classes. Major advantages of using text chunking over full parsing techniques are that partial parsing such as text chunking is much faster, more robust, yet sufficient for IE.

Support Vector Machine (SVM) based text chunking was reported to produce the highest accuracy in the text chunking task (Kudo & Matsumoto, 2000). The SVMs-based approach such as other inductive-learning approaches takes as input a

set of training examples (given as binary valued feature vectors) and finds a classification function that maps them to a class.

In general, SVM models can be characterized as follows: First, SVMs are known to robustly handle large feature sets and to develop models that maximize their generalizability. This makes them an ideal model for IE. Generalizability in SVMs is based on statistical learning theory and the observation that is useful to misclassify some of the training data so that the margin between other training points is maximized (Cortes & Vapnik, 1995). This is particularly useful for real world data sets that often contain inseparable data points. Although training is generally slow, the resulting model is usually small and runs quickly as only the patterns that help define the function that separates positive from negative examples. In addition, SVMs are binary classifiers and so we need to combine several SVM models to obtain a multiclass classifier.

Due to the nature of the SVM as a binary classifier it is necessary in a multi-class task to consider the strategy for combining several classifiers. In this paper, we use Tiny SVM (Kudo & Matsumoto, 2000) in that Tiny SVM performs well in handling a multi-class task.

3.4.4 Relation Extraction by MiHMM component

Figure 18 is a schematic representation of how our MiHMM works. Our phrase group includes 18 phrase types. Our models are fully connected, which means that the model can emit a segment of any type at any given position within the sentence. The bold boxes in Figure 18 indicate the target noun group that contains either proteins or a protein-protein pair. Each box represents phrase group and circles inside the box show the POS tags assigned to words in order that appears in the sentence.

In a generic Hidden Markov Model, it is typical to define a number of states and a number of transitions between those states. The more complex the HMM, the better it can represent a document, but also the more data that is needed to dependably train the model. Consequently there is a tradeoff between representational efficacy and training efficiency, and this tradeoff varies from domain to domain.

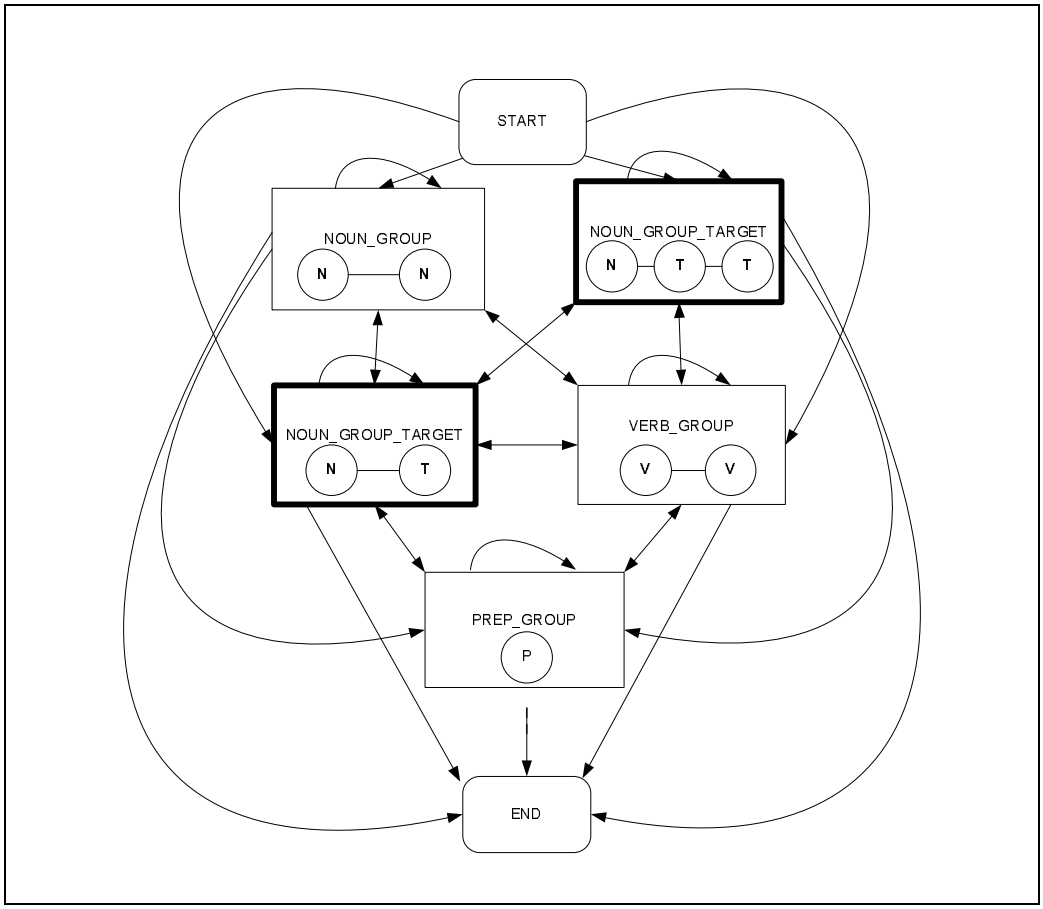

Figure 18: Noun Phrase based Mixture Hidden Markov Models
N denote noun, P denotes preposition, T denotes target, V denote verb.

Therefore, rather than deciding on just one model, it is often easier to use a mixture of models and decide later how much to weight each model. This approach is effective because it allows one to model a document at varying degrees of granularity by using a hierarchical model. At the same time it also retains the advantages of each model. That is, if the data is sparse, the simpler model will likely perform better and thus be weighted more during the extraction phase.

For HMMIE, a basic set of three mixture models was used and are shown in Figure 19. A similar mixture model was proposed by Freitag and McCallum (2000). In comparison to Freitag and McCallum', their model utilizes fairly simple prefix and suffix structures. Despite the simplicity of these models however, the primary benefit of these models that they can be trained on very sparse data.

Training the model is a process of maximum likelihood parameter estimation. From the sentence training set we can easily obtain the information concerning the frequency that a given state or observation occurred and the frequency with which a state transition or observation emission was made.

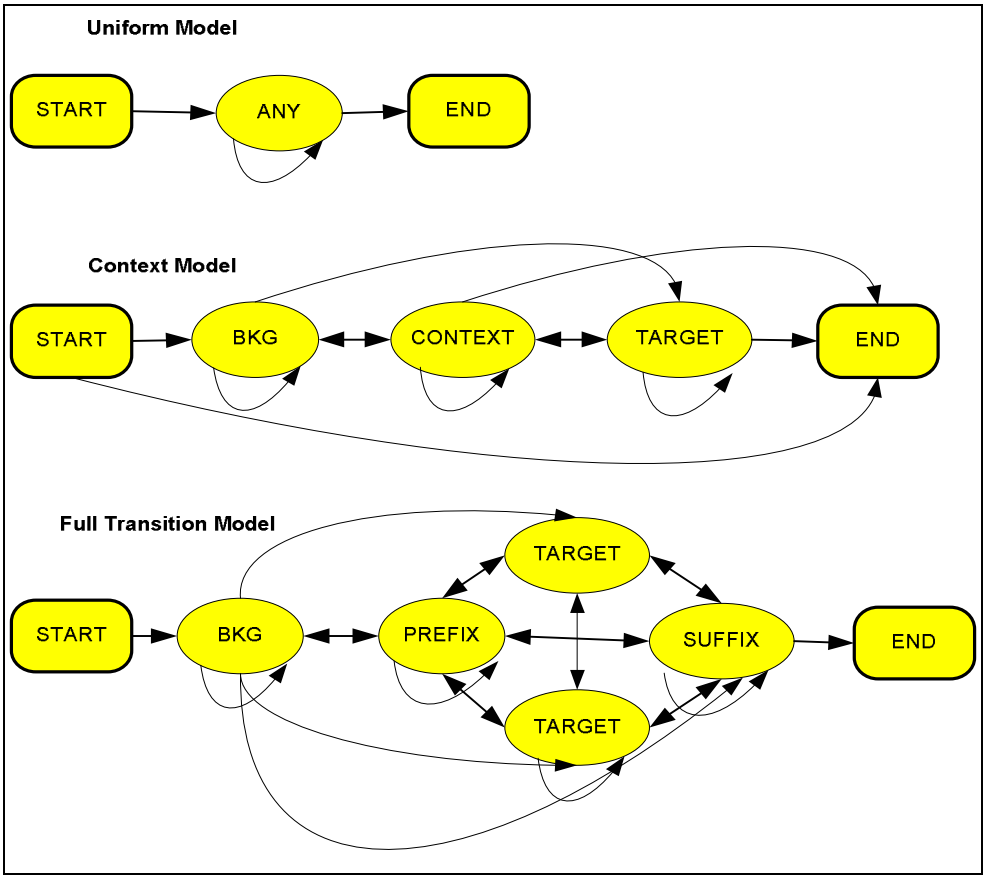

Figure 19: Graphic representation of MiHMM
BKG denotes Background

The parameters of the model are the transition probabilities $P(q \to q')$ that one state follows another and the emission probabilities $P(q \uparrow q')$ that a state emits a particular output symbol. The probability of a string x being emitted by an HMM is computed as a sum over all possible paths by:

$$P(x \mid M) = \sum_{q_1,\ldots q_t \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \to q_k) P(q_k \uparrow x_k) \qquad (3.13)$$

where q_0 and q_{l+1} are restricted to be q_I and q_F respectively, and is an end-of-string token. The forward algorithm can be used to calculate this probability (Rabiner, 1989). The observable output of the system is the sequence of symbols that the states emit, but the underlying start sequence itself is hidden. One common goal of learning problems that use HMMs is to recover the state sequence $V(x \mid M)$ that has the highest probability of having produced an observation sequence:

$$V(x \mid M) \quad \arg\max_{q_1 \ldots q_t \in Q^l} \prod_{k=1}^{l+1} P(q_{k-1} \quad q_k) P(q_k \quad x_k) \qquad (3.14)$$

Fortunately, the Viterbi algorithm (Viterbi, 1967) efficiently recovers this state sequence.

CHAPTER 4: EVALUATION

Our purpose in developing RIKE is to show that integrating keyphrase-based query expansion into IE can be used to develop a scalable knowledge extraction system. RIKE is designed to be particularly suitable for this task. We test RIKE on two different extraction tasks for a biomedical domain. Since RIKE consists of two major components, keyphrase extraction-based query expansion and mixture HMMs for information extraction, the performance evaluation needs to account for effectiveness of each component to the overall evaluation.

The first set of experiments aims at evaluating the performance of DocSpotter. The second set of experiments investigates whether integrating ontologies into query expansion improves the performance of the system in terms of accuracy. The last set of experiments focus on evaluating the performance of HiMMIE.

To this end, we compare the performance of each component of RIKE with the well-receiving existing counterpart algorithms. The details of our experimental evaluation are provided in the sub-sections.

4.1 Evaluation of DocSpotter

In this section, we explain methodologies and strategies used for DocSpotter evaluation. First, we describe the data sources and the search engines integrated to DocSpotter. Second, we outline the tasks used for the experiments. Third, we describe the other query expansion algorithms implemented for the performance comparison.

4.1.1 Data Sources and Search Engines Integrated into DocSpotter

The following two data sources and three different underlying search engines are integrated into DocSpotter: 1) MEDLINE data – PubMed Engine, 2) MEDLINE data – Lemur Engine, and 3) TREC data – Zettair Engine. The purpose of incorporating multiple datasets and search engines are two-fold. The first is to investigate whether DocSpotter generates the consistent results regardless of the data sources and search engines. The second is to demonstrate the system's robustness and ability to integrate various types of data into the system without major architecture changes.

4.1.1.1 MEDLINE Data - PubMed Engine

MEDLINE is the National Library of Medicine (NLM)'s premier bibliographic database covering the fields of medicine, nursing, dentistry, veterinary medicine, health care systems, and the pre-clinical sciences. MEDLINE contains bibliographic citations and author abstracts from more than 4,000 biomedical journals and is the largest English language biomedical bibliographic database with more than 12 million abstracts stored as plain text files.

The sheer size of MEDLINE is challenging for scientists involved in biomedical research. To expedite the progress of functional bioinformatics, several human-curated knowledge bases have been built from MEDLINE. Among the manually created knowledge bases, we select two databases for our experiment, Online Mendelian Inheritance in Man (OMIM) and Database of Interacting Proteins (DIP).

PUBMED Search Engine

PubMed, a web-based search engine, has beens developed by the National Center for Biotechnology Information (NCBI) at the National Library Medicine

(NLM). PubMed provides access to bibliographic information that includes MEDLINE, OLDMEDLINE. It also covers the out-of-scope citations from certain MEDLINE journals, primarily general science and chemistry journals, for which the life sciences articles are indexed for MEDLINE.

In order to retrieve MEDLINE records from PubMed, two XML APIs are implemented in DocSpotter. The first XML API is to retrieve record IDs and the second XML API is to retrieve full bibliographic information (Table 5). The communication between PubMed and DocSpotter takes place via HTTP protocol.

Table 5: Elements of Querying PubMed API

API Element	Description	Accepted Values
BASE URL	Web Service	http://eutils.ncbi.nlm.igv.entrez/eutils/esearch.fcgi?
DB	Database name	PubMed, MeSH, protein
QUERY KEY	The value used for a history search number or previously returned in XML results from PubMed	Numeric value return from PubMed
SEARCH TERM	The query terms used for PubMed search	“Protein A and Protein B and interacts”
DATE RANGES	Limit results bounded by two specific dates	Mindate= Maxdate=
RETRIEVAL MODE	Mode to display results	Retmode=xml

Once the results are received from PubMed upon the above XML request, DocSpotter parses the results in XML form with either the DOM or the SAX parsing technique. An example of the results is shown in the Table 6.

Table 6: Results of Retrieving MEDLINE IDs from PubMed

```
<eSearchResult>
   <Count>8</Count>
   <RetMax>8</RetMax>
   <RetStart>0</RetStart>
   <QueryKey>1</QueryKey>
   <WebEnv>0rReMf8LRvguGgqk0tMlAOG1zAM6rA-bJMZnVxEma1qy-
kd0rMoW</WebEnv>
   <IdList>
      <Id>15121178</Id>
      <Id>11799243</Id>
      <Id>10903573</Id>
      <Id>10894179</Id>
      <Id>10521446</Id>
      <Id>10504198</Id>
      <Id>10051665</Id>
      <Id>7595547</Id>
   </IdList>
   <TranslationSet>
   …
   </eSearchResult>
```

With the record IDs received in the XML, we query PubMed again to get the MEDLINE records including title, author, abstract, etc. The XML API used to get the full bibliographic record is shown in Table 7.

Table 7: Retrieval API for the Results from PubMed

API Element	Description	Accepted Values
BASE URL	Web Service	http://eutils.ncbi.nlm.gov/entrez/eutils/epost.fcgi?
DB	Database name	PubMed, MeSH, protein
RECORD IDENTIFIER	MEDLINE Record IDs	id=198383,238382,29393
RETRIEVE MODE	Mode to display the results	Retmode=xml

Below is an example of results with XML tags to respond to the XML request for full bibliographic record (Table 8). The response returned from PubMed includes 1) PubMed record ID, 2) title, 3) abstract, 4) authors, 5) MeSH subject terms, 6) publication dates, and 7) links to other medical related DBs.

Table 8: Results of Retrieving MEDLINE Full-texts from PubMed


```
<PubmedArticleSet>
<PubmedArticle>
    <MedlineCitation Owner="NLM" Status="MEDLINE">
        <PMID>10023546</PMID>
        <DateCreated>
            <Year>1999</Year>
            <Month>04</Month>
            <Day>21</Day>
        </DateCreated>
        <Article PubModel="Print">
<ArticleTitle>Organelle isolation by magnetic immunoabsorption.</ArticleTitle>
            <Pagination>
                <MedlinePgn>336-43</MedlinePgn>
            </Pagination>
            <Abstract>
                <AbstractText>Currently, most organelle isolation procedures
rely on physical parameters and centrifugation for separation. Here, we report the
rapid and gentle isolation of a variety of organelles by immunolabeling whole cell
lysates with organelle-specific antibodies and streptavidin magnetic particles
followed by separation in a magnetic field. Using magnetic immunoabsorption, we
have been able to specifically label mouse metaphase chromosomes and a variety of
plant organelles, including: amyloplasts, choroplasts and nuclei from whole cell
lysates of various plant tissues. We find that the distinct magnetic properties, surface
characteristics and mean diameter-size ranges of different particle preparations
significantly influence their specific utility for organelle isolations. By using an
internal-field magnetic separation device, we have developed a method for
quantitative recovery of labeled organelles in microarrays and tested a variety of
antibodies to chloroplast outer envelope proteins for their ability to immune-isolate
chloroplasts.</AbstractText>

            </Abstract>
```

Each record is passed to the DocSpotter core component, keyphrase extraction for further analysis.

4.1.1.2 MEDLINE Data – Lemur Engine

In this section, we build a more strictly controlled experimental environment. Since we do not know in advance how many PubMed records containing protein-protein pairs are available in PubMed engine, we developed our own indexes with a large number of PubMed records. The majority of these records do not contain protein-protein pairs, and only some small set of records in indexes contain protein-protein interaction. The current index consists of 264363 PubMed records. Out of these 264363 records, there are 4521 records containing protein-protein interaction pairs.

We decided to include the records containing protein-protein pairs from other sources than human experts-curated databases such as OMID and DIP. The reason for this decision is because we are interested in how well DocSpotter can find the records that have not been covered in these two databases. Out of 4521 records, 4100 records identified by OMID and DIP and the rest records, 421, are from other sources (Basu,

2004).

LEMUR Search Engine

Lemur is developed by collaboration between the Computer Science Department at the University of Massachusetts and the School of Computer Science at Carnegie Mellon University. Lemur is designed to facilitate research in language modeling and information retrieval (OgilVie & Callan, 2001). Lemur supports indexing of large-scale text databases, the construction of simple language models for documents, queries, or sub-collections, and the implementation of retrieval systems based on language models as well as a variety of other retrieval models such as Okapi.

Table 9: Statistics of the MEDLINE Data Indexed by LEMUR

No. of Records Indexed	No. of Terms	No. of Unique Terms	Document Length
264363	61515989	303077	232

Since the default LEMUR code does not include parsing classes for PudMed records, we extended LEMUR to work with PubMed records. After we implemented

the extended parsing component for PubMed data, we indexed it with the inverted index method provided by Lemur. In addition, we built B-tree indexes to store record number and its position to retrieve full bibliographic records from Lemur. Table 10 shows the parameters used for indexing MEDLINE data.

Table 10: Lemur Index Parameters

Indexing Parameter	Values
DOCUMENT FORMAT	Pubmed
DATA FILE	/storage/TREC/doc/pubmed_dir
MANAGER	Pubmed_manager
MANAGER TYPE	Elem
INDEX	Pubmed_index
INDEX TYPE	Inv
STOPWORD	./stop_word_list.txt
STEMMER	Krovetz

Once indexes are built which is done off-line, DocSpotter is ready to interact with Lemur engine. For our experiments, we implemented a persistent daemon based on Lemur retrieval APIs in C++ that DocSpotter can interact with. The daemon runs on a socket program and listens to a port. Although Lemur has recently been

enhanced with a CGI interface to the indexes, at the time we conducted the experiments, the CGI interface was not available.

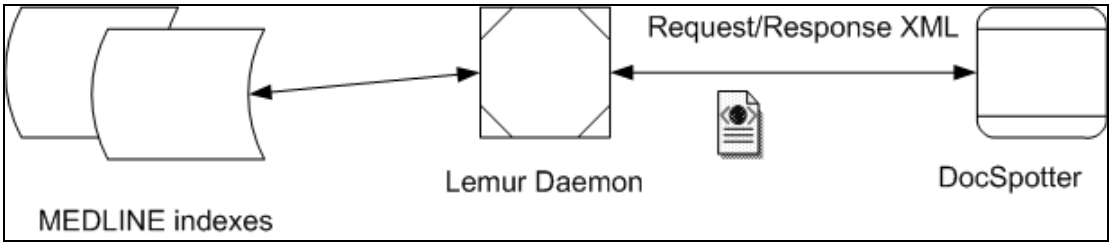

Figure 20: Interacting between Lemur and DocSpotter

Figure 20 shows how DocSpotter interacts with the Lemur daemon to query and retrieve Lemur indexes. The following two figures are examples of the request and response XMLs between DocSpotter and Lemur daemon.

```
<LEMUR_SEARCH>
    <QUERY>((proteins activations kinases) AND (binds mediated))</QUERY>
    <MAX>50</MAX>
    <ID_MAX>1000</ID_MAX>
    <RANK_START>0</RANK_START>
</LEMUR_SEARCH>
```

Figure 21: Request XML to Lemur Daemon

Figure 21 shows an example of a request sent to Lemur for document retrieval. Figure 22 shows an example of retrieved documents received from Lemur.

```
<RESPONSE>
  <RESULTS>
    <RESULT>
      <DOCNO>1285991</DOCNO>
      <TI>Pathogenesis of disseminated intravascular coagulation in sepsis</TI>
       <TEXT>To review new insights in the pathogenetic mechanism …. </TEXT>
      <DOCNO>19394334</DOCNO>
      …
    </RESULT>
  </RESULTS>
</RESPONSE>
```

Figure 22: Response XML from Lemur Daemon

4.1.1.3 TREC Data – Zettair Engine

TREC Data

The Text Retrieval Conference (TREC) is sponsored by both the National Institute of Standards and Technology (NIST) and U.S. Department of Defense.

TREC supports research within the information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies. The NIST TREC Document Databases, hereafter called TREC data, are distributed for the development and testing of IR systems and related natural language processing research. The document collections consist of the full text of various newspaper and newswire articles plus government proceedings. The documents have been used to develop a series of large IR test collections known as the TREC collections. An IR test collection consists of three parts: a set of documents, a set of questions, called topics in TREC that can be answered by some of the documents, and relevance judgments that list the documents relevant to each question.

The keyphrase-based query expansion method is evaluated using the TREC-5, TREC-6, and TREC-7 ad hoc test set. The ad hoc task investigates the performance of systems that search a static document collection using new query statements. The document set consists of approximately 628,531 documents distributed on three CD-ROM disks (TREC disks 2, 4, and 5) taken from the following sources: Federal Register (FR), Financial Times (FT), Foreign Broadcast Information Service (FBIS), LA Times (LAT), Wall Street Journal, AP Newswire, and Information from Computer Select disks.

The format of the documents on the TREC disks is a labeled bracketing expressed in the style of SGML. The different datasets on the disks have identical major structures but have different minor structures. Every document is bracketed by <DOC></DOC> tags and has a unique document identifier, bracketed by <DOCNO></DOCNO> tags.

Tables 11, 12, and 13 show the statistics of records contained in three disks respectively.

Table 11: Statistics of TREC Disk 5

Data Description	Size of Dataset
Foreign broadcast information service	Approx. 130,000 documents Approx. 470 MB
Los Angeles Times (from 1989 to 1990)	Approx. 130,000 documents Approx. 475 MB

Table 12: Statistics of TREC Disk 4

Data Description	Size of Dataset
Congressional Record of the 103rd Congress	Approx. 30,000 documents Approx. 235 MB
Federal Register (1994)	Approx. 55,000 documents Approx. 395 MB
Financial Times (1992-1994)	Approx. 210,000 documents Approx. 565 MB

Table 13: Statistics of TREC Disk 2

Data Description	Size of Dataset
Wall street journal (1986, 1987, 1988, and 1989)	Approx. 100,000 documents Approx. 255 MB
AP Newswire (1989)	Approx. 85,000 documents Approx. 248 MB
Information from Computer Select disks (Ziff-Davis Publishing)	Approx. 75,000 documents Approx. 188 MB
Federal Register (1988)	Approx. 26,000 documents Approx. 211 MB

ZETTAIR

Zettair is the search engine developed by the Search Engine group at RMIT University (Billerbeck & Jobel, 2004). Zettair, formerly called Lucy, was designed to be simple and flexible search engine to index and search HTML and TREC collections. For our experiments, we used Zettair to index and search TREC data. The reason that we chose Zettair is because it is easy to add a query expansion technique such as ours on top of it. In addition, Billerbeck and Jobel (2004) reported that ZETTAIR produced comparable results with Okapi, another state-of-the-art system.

Building indexes for TREC data with Zettair is simple and straightforward. The following parameters were used to build indexes: 1) TREC data file names, 2)

system memory required for indexing, 3) the name of indexes, and 4) the configuration file name containing information about TREC SGML tags.

We decided to use Zettair C API directly in the DocSpotter instead of implementing a persistent daemon on top of the Zettair C API. The rationale is that it requires major changes to the core component of Zettair to implement a persistent daemon. Since our goal is not to make quality re-usable driver classes for search engine but to demonstrate the flexibility of DocSpotter's architecture and also to examine whether there is any impact of different search engine techniques on the DocSpotter performance.

4.1.2 Other Query Expansion Algorithms

In this section, we describe two competitive query expansion algorithms we used to compare with DocSpotter. These are a rule-based query expansion technique, SLIPPER (Agichtein and Gravano, 2003) and a statistical expansion technique, BM25 (Mitra et al., 1998).

4.1.2.1 SLIPPER

We chose SLIPPER to compare the performance of DocSpotter in generating queries. SLIPPER is an efficient rule-learning system, which is based on confidence-ruled boosting, a variant of AdaBoost (Cohen & Singer, 1999). Like RIPPER, the previous version of SLIPPER, SLIPPER supports set-valued features which make it useful for text categorization using a bag of words. In addition, SLIPPER is almost 20 times faster than RIPPER (Cohen & Singer, 1999). One of the main ideas is to maintain a distribution or set of weights over the training set. The weight of this distribution on training example *i* on round *t* is denoted $D_t(i)$. Initially, all weights are set equally, but on each round, the weights of incorrectly classified examples are increased so that the weak learner is forced to focus on the hard examples in the training set. SLIPPER uses boosting to create an ensemble of rules. The weak learner that is boosted finds a single rule, using essentially the same process as used in the inner loops of IREP (Furnkranz & Widmer, 1994) and RIPPER (Cohen, 1995).

SLIIPPER learns concise rules such as ***protein AND interacts" --> Useful***, which shows that if a document contains both term protein and term interacts, it is declared to be useful. These classification rules generated by SLIPPER are then

translated into conjunctive queries in the search engine syntax. For instance, the above rule is translated into a query "protein AND interacts."

4.1.2.2. Okapi BM25

The Okapi BM25 probabilistic model was developed by Robertson (1971) and has been widely adopted in many experimental information retrieval systems. The Okapi BM25 is based on the following simple heuristics:

1) The more occurrences of a query term in a document, the more likely it is that the document is relevant.

2) A long document containing the same number of occurrences of a query term as a short one is less likely to be relevant.

The Okapi BM25 weighting function is a very well known mathematical formulation of these heuristics. The algorithms used in the experiments are denoted as follows:

BM25: The standard Okapi BM25 formula is used as the baseline:

$$BM25 = \sum_{t \in q} \log\left(\frac{N - f_t + 0.5}{f_t + 0.5}\right) \times \frac{(k_i + 1) f_{d,t}}{K + f_{d,t}} \qquad (4.1)$$

where t is a term of query q. f_t is the number of occurrences of a particular term across the document collection that contains N documents and $f_{d,t}$ is the frequency of a

particular term *t* in document *d*. *K* is $k_1((1-b)+b * L_d / AL$ .where k_1 and *b* are parameters set to 1.2 and 0.75, respectively. L_d is the length of a particular document and *AL* is the average document length.

4.1.3 Task Descriptions

In this section, we describe the tasks used to evaluate DocSpotter's performance with three different datasets and search engines introduced in the previous section.

4.1.3.1 Initial Query Sets for MEDLINE

The tasks with MEDLINE data intended to be used for the experiments with MEDLINE - PubMed engine and MEDLINE - Lemur engine. In these tasks, the focus is on retrieving MEDLINE records that contain protein-protein interaction pairs. To explore the flexibility and generality of DocSpotter, we studied query expansion for MEDILINE articles. The task we selected is to retrieve documents containing protein-protein interaction pairs. The total 25 initial queries were provided for the experiments. The initial queries consist of 3 to 5 protein-protein interaction pairs.

Figure 23 shows the initial query used to retrieve the documents from PUBMED. APPENDIX B shows the twenty queries used for the experiments.

```
<init_query>
   <terms protein1=“MAP4" protein2="Mapmodulin"/>
   <terms protein1=“WIP" protein2="NCK"/>
   <terms protein1="GHR" protein2="SHB"/>
   <terms protein1="SHIP" protein2="DOK"/>
   <terms protein1="LNK" protein2="GRB2"/>
   <terms protein1="CRP" protein2="Zyxin"/>
</init_query>
```

Figure 23: Initial Query Used for Protein-protein Interaction Tasks.

The initial queries are sent to either the PubMed or Lemur engine to retrieve the initial set of retrieved documents. DocSpotter is then applied to extract keyphrases and expand queries based on the top N-ranked keyphrases.

4.1.3.2 Initial Query Sets for TREC

Search requests in the form of TREC topics consist of three parts: title, description, and narrative. The title consists of individual words that best describe the information need, the description field is a one-sentence description of the topic area,

while the narrative gives a concise description of what makes a document relevant or not. The different parts of the TREC topic allow investigation of the effect of different query lengths on retrieval performance. For our investigation only the title field of the topics is used because it is most similar to the form of queries entered by typical users.

Table 14: Documents and Queries Used in TREC Ad Hoc Tasks

Task	Documents	Queries
TREC5	TREC disks 2,4	251-300
TREC6	TREC disks 4,5	301-350
TREC7	TREC disks 4,5	351-400

The query sets and document collections used in these tasks are shown in Table 12. The sample query used in the experiments is shown in Figure 24. In TREC terminology, “topic” serves as queries to retrieve documents from the search engines. As shown in Figure 24, it is called “topic” to be distinguished from a query in that “topic” is a natural language statement of information need. The topics are formatted using a very simple SGML-style tagging. Different topic sets have different fields

included in the topic statements.

<top>

<num> Number: 301

<title> Integration Organized Crime

<desc> Description:

Identify organizations that participate in international criminal activity, the activity, and if possible, collaborating organizations and the countries involved.

<narr> Narrative:

A relevant document must as a minimum identify the organization and the type of illegal activity (e.g., Columbian cartel exporting cocaine). Vague references to international drug trade without identification of the organization(s) involved would not be relevant.

</top>

Figure 24: Topic 301 Used for Evaluating TREC Data

4.2 Evaluation of HiMMIE

In this section, we describe the overall methodologies and strategies adopted to evaluate our information extraction component, HiMMIE. The goal of HiMME is to extract as many valid tuples as possible from the text collections and to combine them into one table. Input data to HiMMIE is formatted in DocSpotter. The detailed

descriptions are provided in the section of System Description.

In our experiments, we do not attempt to capture every instance of such tuples. Instead, we exploit the fact that these tuples tend to appear multiple times in the types of collections that we consider. As long as we capture one instance of such a tuple, we consider our system to be successful for that tuple. To evaluate this task, we adapt the recall and precision metrics from IR to quantify how accurate and comprehensive our combined table of tules is. Our metrics for evaluating the performance of an extraction system over a collection of document *D* are based on determining the set of all the tuples that appear in the collection *D*.

OMIM (McKusick, 1998) is a catalog of human genes and genetic disorders. It was developed for the World Wide Web by NCBI, the National Center for Biotechnology Information. The database contains textual information and references. It also contains copious links to MEDLINE and sequence records in the Entrez system, and links to additional related resources at NCBI and elsewhere. With OMIM, our task is to extract gene-disease interaction. OMIM was used by Ray and Craven (2001) to extract gene-disorder interaction. The data set used contains 892 positive instances and 11,478 negative instances. They manually labeled the sentences in the

MEDLINE abstracts gathered through OMIM entries. A sentence which contained words that matched a tuple was taken to be a positive instance. For our experiment, we use the data set compiled by Ray and Craven (2001).

DIP (Xenarios et al., 2001) is such a knowledge base about the biological relationships of protein-protein interactions and is constructed by human experts manually for many man-year efforts. However, it is becoming more and more difficult for curators to keep up with the increasing volume of literature. Thus, automatic methods such as RIKE are very useful to speed up this step of database construction. In our evaluation, the manually curated knowledge base DIP serves as an ideal testbed to check the performance of our RIKE system. In the DIP database, as shown in Table 13, it has the information of protein names, protein-protein interaction pairs and the MEDLINE abstracts from which the protein-protein pairs are manually extracted for a few species (e.g., human being, yeast, fruit fly, house mouse, Helicobacter pylori, Escherichia coli). For example, in DIP, for 7050 fruit fly proteins, there are 21017 protein-protein interactions, which are extracted from 7065 articles out of the 12 million.

Table 15: Protein-protein Interactions from DIP

Organism	**Protein**	**Protein Interactions**	**# of relevant abstracts from MedLine**
Drosophila melanogaster (fruit fly)	7050	21017	7065
Saccharomyces cerevisiae (yeast)	4726	15364	4740
Helicobacter pylori	710	1425	710
Homo sapiens (Human)	753	1128	848
Escherichia coli	421	516	418
Mus musculus (house mouse)	191	279	298

We conduct a series of the experimental evaluation studies. We start with a few protein-protein interaction pairs or gene-disease interaction pairs and then let RIKE take over and automatically construct queries, select the relevant articles from MEDLINE and extract the protein-protein interaction for each species. We repeat the experiments for each species several times with different seed instances and then take the average of the articles numbers.

We use OMIM and DIP as references to compare the number of articles that need to be retrieved and extracted by key-word-based approach and RIKE approach in order to rebuild the protein-protein interactions or gene-disease interactions for each species. We expect the articles selected from RIKE will be significantly fewer

than the key-word-based approach, but larger than the numbers in OMIM or DIP. Each article in OMIM or DIP is manually selected by curators and the number indicated by DIP is the minimum number of articles that need to examine in order to find the protein-protein interaction or gene-disease interactions for the corresponding species.

4.2.1. Other Information Extraction Algorithms

In this section, we identify several key algorithms proposed in IE from our literature review. Among them, we implemented five IE algorithms that were reported to produce high extraction accuracy. We compare HiMMIE with these five IE algorithms.

4.2.1.1 Dictionary-based

We developed a dictionary-based extraction system along the lines proposed by Blaschke et al. (1999). As described in Blaschke et al. (1999), the following six steps were taken to extract protein-protein interactions: 1) the protein names are collected from the Database of Interacting Proteins (DIP) and Protein-Protein

Interaction Database (PPID) databases. The synonyms of the target proteins are manually provided. 2) The 14 verbs, indicating actions related to protein interaction, are used. 3) Abstracts are provided from MEDLINE. 4) The passages containing target proteins and actions are identified. 5) The original text is parsed into fragments preceding grammatical separators. 6) The final step is to build protein-protein pairs.

4.2.1.2 RAPIER

To evaluate the performance of HiMMIE, we compare it with RAPIER. RAPIER (Califf & Mooney, 1999) is a well-known IE system that was developed with a bottom-up inductive learning technique for learning information extraction rules. In order to use the slot-filling IE systems like RAPIER for extracting relations, we adapt the Role-filler approach proposed by Bunescu et al. (2004).

The Role-filler approach allows for extracting the two related entities into different role-specific slots. For protein interactions, Bunescu et al. (2004) name the roles “interactor” and “interactee”. As indicated by the role names, protein-protein interactions are defined with the assumption that the proteins appear in the same sentence. Bunescu et al. (2004) extracted the related pairs using the following criteria:

1) the interactors and interactees appear in the same sentence, 2) each interactor is associated with the next occurring interactee in the segment, and 3) If the number of the interactors and the interactees is unequal, use the last interactor (interactee) for building the remaining pairs.

4.2.1.3 Single POS HMM

In order to verify that our MiHMM models are superior to simple HMMs, we develop a simple HMM, based on single terms and a single model that incorporate less grammatical information. We implemented single-level HMMs whose states emit words, but are typed with part-of-speech (POS) tags so that a give state can emit words with only a single POS. The Viterbi algorithm extracts information from documents modeled by an HMM. With the fixed structure, the objective of learning is to give high probabilities to training documents. The result of learning is estimated probabilities for vocabularies and transitions.

4.2.1.4 Support Vector Machine (SVM)

SVM has achieved state of the art performance in several classification tasks. SVM is a machine learning technique based on the statistical learning theory, which implements the structural risk minimization inductive principle with the purpose of obtaining a good generalization from limited size data sets. In the supervised machine learning approach which we adopt here, we aim to estimate a classification function *f*,

$$f : x^N \rightarrow \{\pm 1\} \qquad (4.2)$$

so that error on unseen examples is minimized, using training examples that are *N* dimensional vectors x_i with class labels y_i.

From the training set $S < (x_i, y_i),...,(x_r, y_r) >$, we learn a classifier $c : x \rightarrow y$ with $y = sign(wx + b)$, using the SVMlight () with linear kernel and default parameter settings to find parameters *w* and *b*. The classification function returns either +1 if the test data is a member of the class, or -1 if it is not. SVMs use linear models to discriminate between tow classes. This raises the question of how can they be used to capture non-linear classification functions? The answer to this is by the use of a non-linear mapping function called a kernel,

$$\Phi : \chi^N \rightarrow \Gamma \qquad (4.3)$$

which maps the input space χ^N into a feature space Γ. The resulting sequence classifier is applied to the incoming testing data.

4.2.1.5 Maximum Entropy

Maximum Entropy (Berger et al., 1996) is widely used method for inducing probabilistic classifiers. The classification problem is viewed in terms of a random access that produces an output value *y* from a finite set *Y*, based on a contextual information *x*, a member of a finite set *X*. The idea of maximum entropy modeling is to choose the probability distribution *p* that has highest entropy out of those distributions that satisfy a certain set of constraints. The constraints restrict the model to behave in accordance with a set of statistics collected from the training data. The statistics are expressed as the expected values of appropriate functions defined on the contexts *h* and tags *t*. In a tagging scenario, this means associating a tag *y* with each token, whereas the context *x* can be derived from the text centered at the current token position. In maximum entropy modeling we are looking for a probability distribution $p(y \mid x)$ expressed in terms of a set of user specified features $f_i(x, y) \in F$:

$$p(x \mid y) = \frac{1}{Z(x)} \exp(\sum_i \lambda_i f_i(x, y)) \qquad (4.4)$$

where $Z(x) = \sum_y \exp(\lambda_i f_i(x, y))$ is a normalizing constant. Each feature f_i is a binary function based on the current context *x* and its proposed classification *y*. In the case of maximum entropy tagging, called MaxEnt, we adopted the same tagging types used for HiMMIE. See Table # for the complete tagging types. We implemented our MaxEnt algorithm based on the Stanford Tagger (Toutanova et al., 2003). The Stanford Tagger improves maximum entropy-based tagging algorithm proposed by Ratnaparkhi (1996) which is based on learning a log-linear conditional probability model from tagged text. The improvement was made to the parameter estimation procedure for iterative scaling.

4.3 Evaluation of the Impact of Ontologies on the Retrieval Performance

One of the hypotheses explored in this dissertation is whether ontologies improve the retrieval performance of the system. In order to investigate the impact of ontologies on the system performance, we chose the following three ontologies: WordNet, MeSH, and UMLS. These ontologies are integrated into DocSpotter. Ontologies or lexical reference systems are successfully used for query expansion in limited domains (Hersh et al., 1994). For instance, clinical queries to PubMed can be automatically expanded to yield either specific or general results. In particular, in the

biomedical domain, query expansion using MeSH or UMLS looks more promising due to the considerable effort being devoted to creating useful cross-linkages across data sources (Hersh et al., 2003).

4.3.1 WordNet

WordNet is an online lexical reference system in which English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing one underlying lexical concept. Different relations link the synonym sets. WordNet was developed in the Cognitive Science Laboratory at Princeton University.

Published results on sense based query expansions are not very recent (Sanderson, 1994; Voorhees, 1993). More recent work (Gonzalo et al., 1998) analyzes the effect of expanding a query with WordNet synsets, in a “canned” experiment where all words are manually disambiguated. Gonzalo et al. show that a substantial increase in performance is obtained with less than 10% errors in the word send disambiguation task. Word sense disambiguation can be one of the hardest problems in AI.

For our experiments, we use WordNet 2.0. The detailed descriptions of how

WordNet is applied to query expansion in DocSpotter are provided in the section of Query Expansion with WordNet.

4.3.2 MeSH

MeSH, Medical Subject Heading, is developed and maintained by the National Library of Medicine. It consists of sets of terms naming descriptors in a hierarchical structure that permits searching at various levels of specificity. Applying MeSH to query expansion, Srinivasan (1996a, 1996b) reports a number of experiments which illustrate her overall approach. These experiments are based on a test collection produced by Hersh et al. (1994). Applying MeSH to query expansion, Srinivasan (1996a, 1996b) reports on a number of experiments which illustrate her overall approach. These experiments are based on a test collection produced by Hersh et al. (1994) which comprises 75 queries and 2,344 MEDLINE citations. Each citation includes a title, an abstract, and MeSH indexing terms assigned by human. The retrieval system used is SMART (Salton, 1971). All results are reported in terms of 11-point average precision.

Srinivasan employs retrieval feedback based on relevance feedback.

Srinivasan reports significant gains in average precision using her methodology. Average precision for retrieval of relevant documents in the baseline experiment, where both documents and queries consist of text-only, is 51,7%. Adding the MeSH terms to the documents but leaving the queries unexpanded increases average precision to 55.6%, which is a 7.5% increase over the baseline. Using expanded queries on MeSH-indexed text further increases average precision to 60.2%. This represents a 16.4% increase over the baseline average precision and an 8.3% increase experiment based on indexed text, but with unexpanded queries. These experiments demonstrate that both human indexing as represented by the MeSH terms in the documents and retrieval feedback contribute to the overall improvements in average precision of 16.4% over the baseline figure.

The detailed descriptions of how MeSH is applied to query expansion in DocSpotter are provided in the section of Query Expansion with MeSH.

4.3.3 The Unified Medical Language System (UMLS)

UMLS has been developed by the National Library of Medicine to facilitate the development of intelligent systems to understand the meaning of the language of

biomedicine and health (UMLS, 2005). In UMLS, there are three knowledge sources: Metathesaurus, Semantic Network, and SPECIALIST lexicon. The Metathesaurus is a large, multi-purpose, and multi-lingual vocabulary database that contains information about biomedical and health related concepts, their various names, and the relationships among them. The Metathesaurus is organized by concept or meaning. It links alternative names and views of the same concept together and identifies useful relationships between different concepts.

The Semantic Network provides a consistent categorization of all concepts represented in the UMLS Metathesaurus and provides a set of useful relationships between these concepts. All information of specific concepts is found in the Metathesaurus.

The SPECIALIST lexicon provides the lexical information needed for the SEPCIALIST Natural Language Processing (NLP) system. The lexicon entry for each word or term records individual syntactic, morphological, and orthographic information.

Major researches conducted on query expansion with UMLS have used the Metathesaurus (Aronson et al., 1997; Hersh et al., 1994; Ruiz & Shrinivasan, 1998).

Although the major goal of the UMLS Metathesaurus is to provide linkages among different vocabularies, a secondary goal is to expand terminology for applications such as IR systems where searchers may use different terms than indexers or authors of the documents they desire to retrieve (Lindberg et al., 1993). The detailed descriptions of how UMLS is applied to query expansion in DocSpotter are provided in the section of Query Expansion with UMLS.

4.4 Experimental Methodology

To evaluate RIKE as effectively and accurately as possible, we decide to evaluate RIKE separately by component, DocSpotter and HiMMIE. DocSpotter is evaluated in IR standpoint and HiMMIE is evaluated in IE perspective. The following sections describe the methodologies employed to evaluate these two systems.

4.4.1 Methodology for DocSpotter

The performance of DocSpotter is evaluated in three different cases: MEDLINE –PubMed, MEDLINE – Lemur, and TREC – Zettair. In IR, experimental methodology has been the prominent issue since 1960s. TREC was established in 1992 to evaluate large-scale IR which retrieves documents from a gigabyte collection.

It is known to be a most well-received IR evaluation setting.

Since an IR system, particularly in the context of web-based system, is complex and large scale, it requires evaluating several aspects of the system: assistance in formulating queries, speed of retrieval, resources required, presentation of documents, and ability to find relevant documents (Allen, 2001). In a broad sense, these aspects could be classified into two: usability or retrieval effectiveness. Although all of these aspects need to be taken into account, the most common evaluation for IR system focuses on retrieval effectiveness. The retrieval effectiveness is generally measured by recall and precision.

Precision is the proportion of a retrieved set that is relevant. Precision is the proportion of all relevant documents in the collection included in the retrieved set. For ranked retrieval, we compute value at fixed recall points (e.g., precision at 20% recall) and compute value at fixed rank cutoffs (e.g., precision at rank 20).

Since these two measures have an inverse relationship and often we need one unified measure for evaluation, there are several proposals on the single-valued measures (Baeza-Yates & Ribeiro-Neto, 1999). One of the earliest proposals on most single-valued measures is E (Rijsbergen, 1979). The E measure is based on combining

precision and recall into one.

$$E = 1 - \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} \tag{4.5}$$

It is used to emphasize precision, essentially a weighted average of precision. Large α increases importance of precision. This formula can transform by $\alpha = 1/(\beta^2 + 1), \beta = P/R$:

$$E = 1 - \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{4.6}$$

When $\beta = 1$, precision and recall are given equal importance.

There is another well-established single-valued measure called F measure. The F measure is calculated by substituting E measure from 1. Among various parameter combinations used in F measure, F1 measure is popular: F with $\beta = 1$. F measure is particularly popular with classification researchers. We use the F measure for evaluating HiMMIE.

Table 16: Grid for Precision and Recall

	Relevant	Non-relevant	Total
Retrieved	A	B	A+B
Not retrieved	C	D	C+D
Total	A+C	B+D	A+B+C+D

Recall is proportion of retrieved documents amongst the relevant documents ($\frac{A}{A+C}$). Precision is proportion of relevant documents amongst the retrieved documents ($\frac{A}{A+B}$).

4.4.2 Methodology for HiMMIE

In IE, the evaluation of system performance is done with an answer key that contains the annotations and their attributes (also called slots) the system should find from the input. Precision (P) and recall (R) have been used regularly to measure the performance of IE as well as IR. Recall denotes the ratio of the number of slots the system found correctly to the number of slots in the answer key, and precision is the ratio of the number of correctly filled slots to the total number of slots the system filled (Equations 4.7 and 4.8). Precision deals with substitution and insertion errors while recall deals with substitution and deletion errors.

$$R = \frac{C}{K} \tag{4.7}$$

where C is the number of correct slots.

$$P = \frac{CF}{S} \quad (4.8)$$

where CF is the number of correctly filled slots. S is the number of slots the system filled.

If both precision and recall are 1.0, then the results are completely correct. Lower precision indicates that the system is extracting some incorrect entities. Lower recall indicates that the system is failing to find enough of correct entities. In addition to precision and recall, we report another measure for the experiments, F-measure, which was introduced by the MUC conferences (DARPA, 1992) to the IE community. The F-measure was originally proposed by van Rijsbergen (1979) in the context of IR. F-measure combines precision and recall in order to provide a single number measurement for information extraction systems (4.9). As illustrated in Figure 25, there is a tradeoff between recall and precision. It is impossible to produce 100% recall and 100% precision at the same time. Since F-measure provides a useful tool for examining the relative performance of systems when one has better precision and the other better recall, we report that number where it is useful.

$$F_b = \frac{(b^2 + 1)PR}{b^2 P + R} \quad (4.9)$$

when P is precision, R is recall, *b=0* means *F*=precision, b=∞ means F=recall.

$b = 1$ means recall and precision are equally weighted, $b = 0.5$ means recall is half as important as precision. $b = 2.0$ means recall is twice as important as precision. because $0 \le P, R \le 1$, a larger value in the denominator means a smaller value overall.

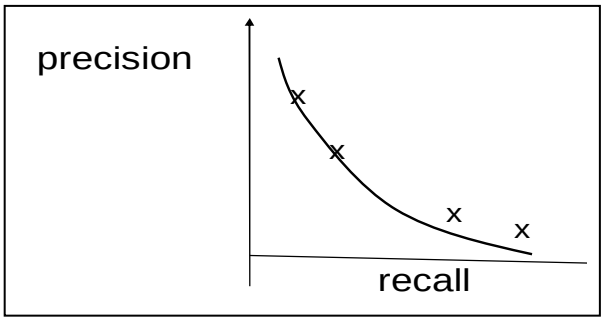

Figure 25: Graphical Representation of Recall and Precision Relationship

4.5 Evaluation Strategy

Our major goal of conducting a series of experiments is to scrutinize three hypotheses discussed in this dissertation. In doing so, we investigate how RIKE performs in retrieving promising documents that satisfy the users' needs and extracting the desired entities or relations in the retrieved documents.

The major goal of RIKE is to extract as many valid tuples as possible from the text collection and to combine them into one table. RIKE does not attempt to capture every instance of such tuples in the text databases. Instead, RIKE exploits

the fact that all these instances will tend to appear multiple times in the types of collections. As long as we capture one instance of such a tuple, we consider RIKE to be successful for that instance. This is different from the goal of traditional IE. Traditional IE systems aim at extracting all the relevant information from each document as completely as possible, while our system extracts instances from all the documents in the collection and combines them into one table.

To evaluate RIKE, two components of RIKE, DocSpotter and HMMIE are compared with several well-known algorithms in IR and IE. First, we compare Keyphrase based query expansion employed in DocSpotter with a highly efficient rule-based text document classifier SLIPPER (Cohen, 1995) and a statistical query expansion algorithm, BM25. Next, HMMIE is compared with 1) Rapier, a relational learning IE system, 2) Single POS HMM, a traditional Hidden Markov, Model, 3) MaxEnt, Maximum Entropy-based extraction, and 4) SVM, Support Vector Machine-based extraction. With these experimental settings, we can verify what combination of techniques produce the best experimental results.

CHAPTER 5: EXPERIMENTAL RESULTS AND ANALYSES

In this chapter, we briefly describe the experimental settings designed to evaluate DocSpotter and HiMMIE and report the experimental results. The detailed descriptions are provided in the chapter 4. The experiments are carried out to investigate three hypotheses studied in this dissertation: 1) evaluating query expansion algorithms, 2) evaluating the impact of ontologies on retrieval accuracy, and 3) evaluating information extraction algorithms.

5.1 Experimental Results with DocSpotter

As stated in the previous chapter, the keyphrase-based query expansion algorithm was evaluated with three different data and search engines. The subsections below report the experimental results with these combinations. We used 25 initial queries for both MEDLINE – PubMed and MEDLINE – Lemur[1]. We used 200 topics (queries) for TREC – Zettair. In order to examine whether query drift exists, we ran ten iterations in each query expansion experiment. Two measures, average precision and precision at top 20 documents, were utilized for performance evaluation of our

[1] See the detailed description for 25 queries used for the experiments in APPENDIX B.

query expansion algorithm (p@20). The four query expansion algorithms below are used for the experiments:

BM25: Okapi BM25 algorithm.

SLP: SLIPPER**,** a Rule-based AdaBoost algorithm

KP: Apply the Keyphrase-based query expansion algorithm.

KP+C: In addition to the KP formula, this algorithm employs Boolean constraints by POS type of keyphrases.

5.1.1 Evaluation of Query Expansion with TREC – Zettair

In this section, we report experimental results with query expansion algorithms on TREC data and Zettiar. As described in the Chapter 4, we use TREC disk 2, 4, and 5 for our experiments with TREC and Zettair.

Table 17: Results for TREC 5 with Four Query Expansion Algorithms

Algorithm	**TREC 5**	
	Avg. P	P@20
BM25	0.1623	0.3252
SLP	0.1278	0.2531
KP	0.1938	0.3368
KP+C	0.1985	0.3398

Table 15 shows the overall performance of the four algorithms executing the query set 251-300 on TREC 5 data. The results indicate the improvements in average precision as well as P@20 of each algorithm compared to its preceding algorithm. Among the algorithms, KP+C in P@20 shows the best improvement among the algorithms.

Figure 26 indicates our keyphrase-based query expansion integrated with POS categorization (KP+C) outperforms over ten query iterations in average precision. The lowest average precision for KP+C is 0.2114 and the highest average precision is 0.2593. The next best algorithm is the baseline keyphrase-based query expansion (KP) which the range of average precision is between 0.1961 and 0.2492. It is followed by Okapi BM25 algorithm that the average precision ranges from 0.1797 to 0.2322. The lowest average precision is produced by a rule-based query expansion (SLP) that the range is 0.1268 and 0.1433.

To confirm the differences among the conditions, we conducted an ANOVA for the P@20 TREC5 results. This showed an overall effect of condition $F(3,196)=17.64$, $p<0.01$. We also conducted individual *t*-tests essentially as specific comparisons. Our prediction that KP would be better than BM25 was confirmed

$t(49)=-7.37$, $p<0.01$ (one-tailed). Similarly, our prediction that KP+C would be better than KP was confirmed $t(49)=-4.72$, $p<0.01$ (one-tailed).

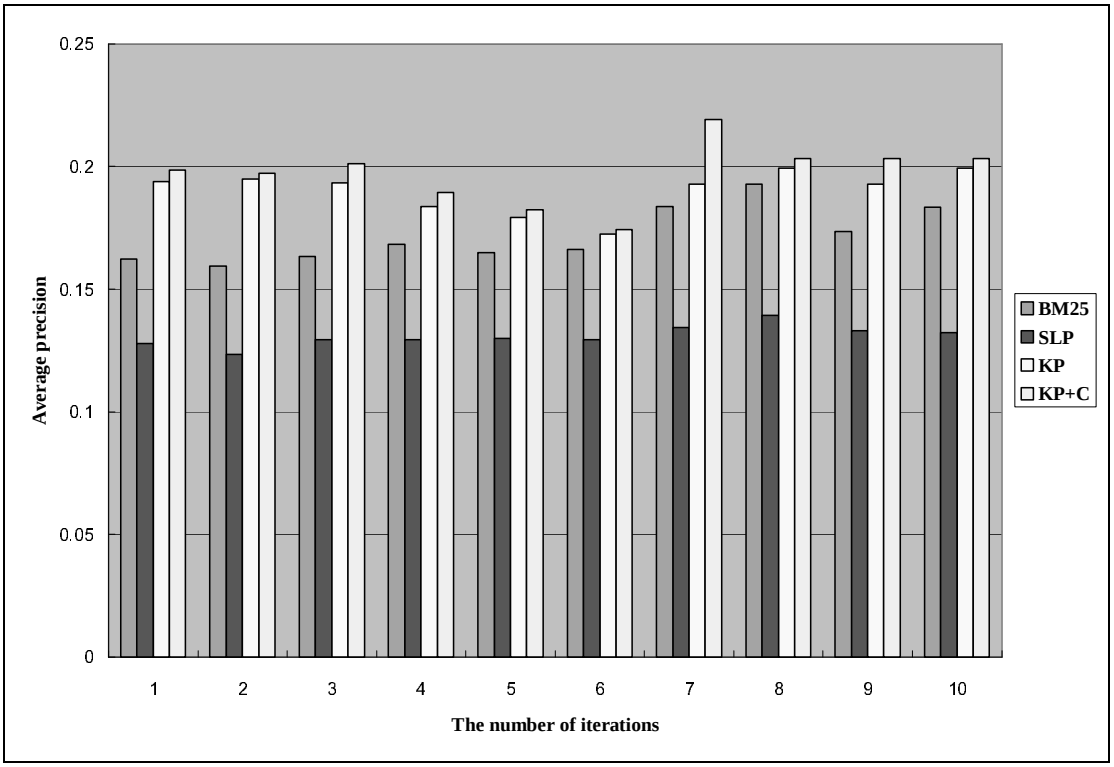

Figure 26: TREC 5 - Zettair with Four Query Expansion Algorithms

We also conducted the same experiments with TREC 6 and TREC 7. As with TREC 5, we used average precision and precision at top 20 (p@20) for performance measures for the query expansion algorithms.

Table 18: Results for TREC 6 with Four Query Expansion Algorithms

Algorithm	**TREC 6**	
	Avg. P	P@20
BM25	0.1797	0.3160
SLP	0.1268	0.2433
KP	0.2098	0.3390
KP+C	0.2114	0.3424

Tables 15 and 16 show similar results to those obtained for TREC 5. Two new algorithms (KP and KP+C) proposed in this dissertation improve the retrieval performance on TREC 6 and 7. As with TREC 5, the KP+C algorithm outperforms BM25, SLP, and KP in average precision and in P@20.

Table 19: Results for TREC 7 with Four Query Expansion Algorithms

Algorithm	**TREC 7**	
	Avg. P	P@20
BM25	0.2229	0.3837
SLP	0.1533	0.2343
KP	0.2343	0.3878
KP+C	0.2458	0.4024

The experimental results over ten query iterations with TREC 6 are shown in Figure 22. The experiment results with TREC 7 are shown in Figure 23. As shown in Figure 22, the best performance was observed with the keyphrase-based query expansion with POS categorization (KP+C) in average precision. The average precision range was in between 0.2343 and 0.2510. As with TREC 4, the worst performance was produced by a rule-based query expansion (SLP). The lowest average precision for SLP was 0.1234 and the highest average precision for SLP was 0.1392. The similar pattern of these experimental results is found with TREC 7. The best performance was produced by KP+C (highest 0.1742 and lowest 0.1502) and the worst performance was from SLP (highest 0.1023 and lowest 0.1093).

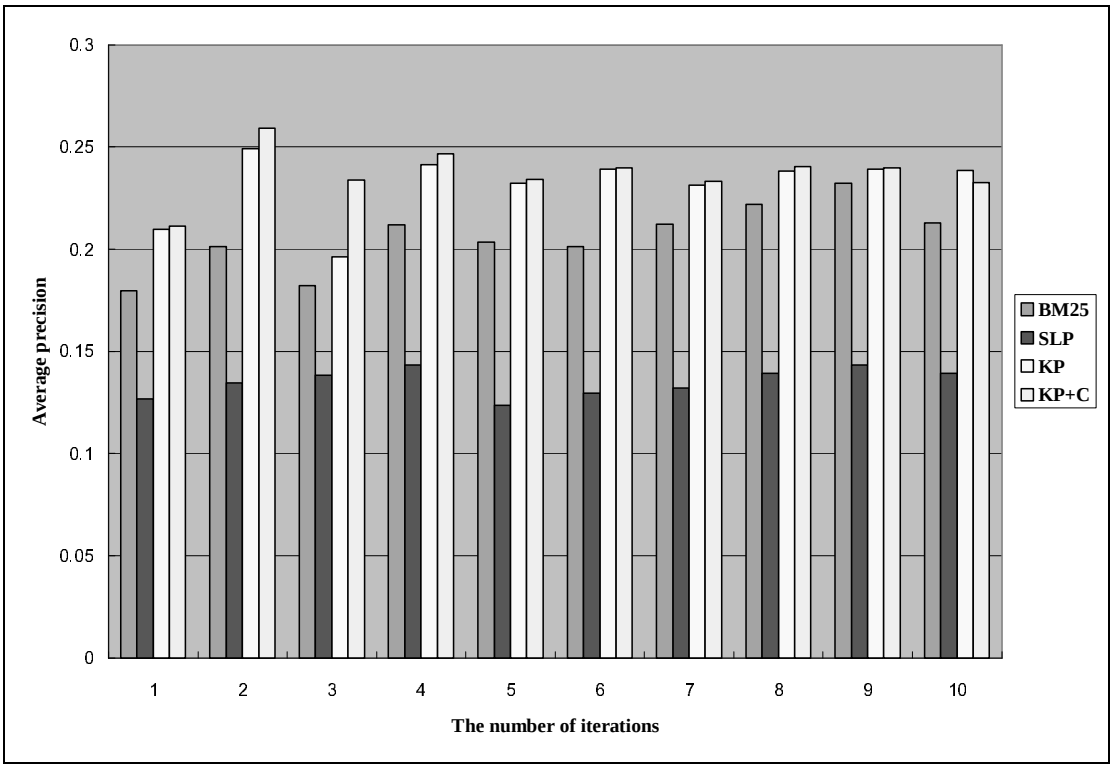

Figure 27: TREC 6 - Zettair with Four Query Expansion Algorithms

As shown in Figures 27 and 28, our two query expansion algorithms based on keyphrase extraction are superior to Okapi BM25 and SLIPPER in all cases. The interesting observation from these experiments is that SLIPPER is not a competitive query expansion algorithm contrasted to the findings reported by Agichtein and Gravano (2003). Agichtein and Gravano used RIPPER, the ancestor of SLIPPER, as one of their three query expansion algorithms integrated into SNOWBALL. In their experiments, SNOWBALL is demonstrated to be an excellent retrieval mechanism. Their results differ from ours in terms of usage of RIPPER or SPLIPPER for query expansion. The poor performance of SLIPPER could be attributed to the fact that

SLIPPER is not suitable for high precision-oriented retrieval tasks like ours.

Moreover, RIPPER algorithm could be inferior to SLIPPER.

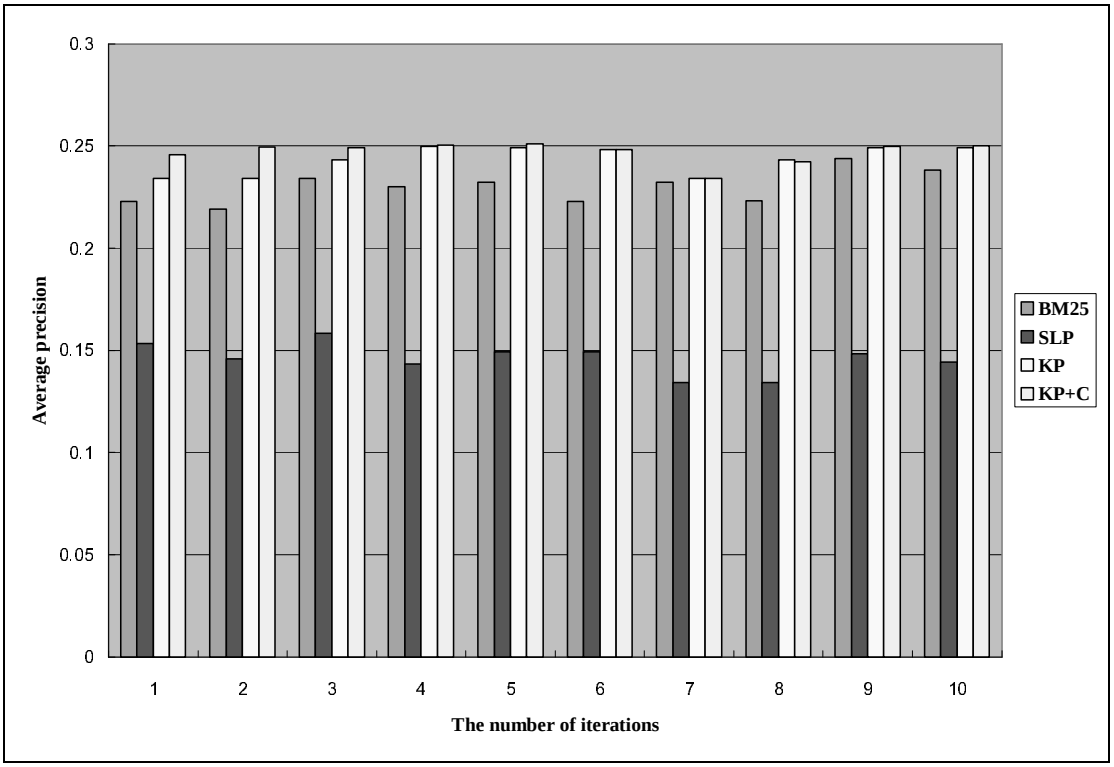

Figure 28: TREC 7 - Zettair with Four Query Expansion Algorithms

Our keyphrase-based technique combined with the POS phrase category produces the highest average precision. One of the best results on TREC 5 is 19.44 and 32.40 in average precision and P@20 respectively. On TREC 6, their best results are 20.34 and 33.50 in average precision and P@20. The algorithm KP+C produces 21% and 48% better than these results on TREC 5 in average precision and P@20.

On TREC 6, it is 39% and 22% which are better than the results reported in Mitra et al. (1998).

5.1.2 Evaluation of Query Expansion with MEDLINE - PubMed

The experimental results for MEDLINE are shown in Table 18. Our keyphrase-based technique combined with the POS phrase category produces the highest average precision. Our two algorithms (KP and KP+C) improve the retrieval performance on the tasks of retrieval documents containing protein-protein interaction pairs. As with our TREC results, the KP+C algorithm gives the best average precision. The worst performance was produced by a rule-based algorithm (SLP) both in average precision and precision at top 20.

Table 20: Results for MEDLINE with Four Query Expansion Algorithms

Algorithm	**MEDLINE**	
	Avg. P	P@20
BM25	0.1282	0.2727
SLP	0.1051	0.2366
KP	0.1324	0.2844
KP+C	0.1522	0.2996

Compared to the experimental results with TREC-Zettair, the overall performance of the query expansion algorithms is poor in terms of average precision and precision at top 20. There might be two possible reasons that cause the overall poor performance. First, PubMed is based on exact match retrieval model, and it makes the keyphrase-based query expansion less effective. Second, the size of the database is too big. The number of records in MEDLINE is over 12 millions.

Figure 29 indicates that both keyphrase-based query expansion with MeSH ontology (KP+M) and keyphrase-based query expansion with UMLS (KP+U) made significant performance improvement.

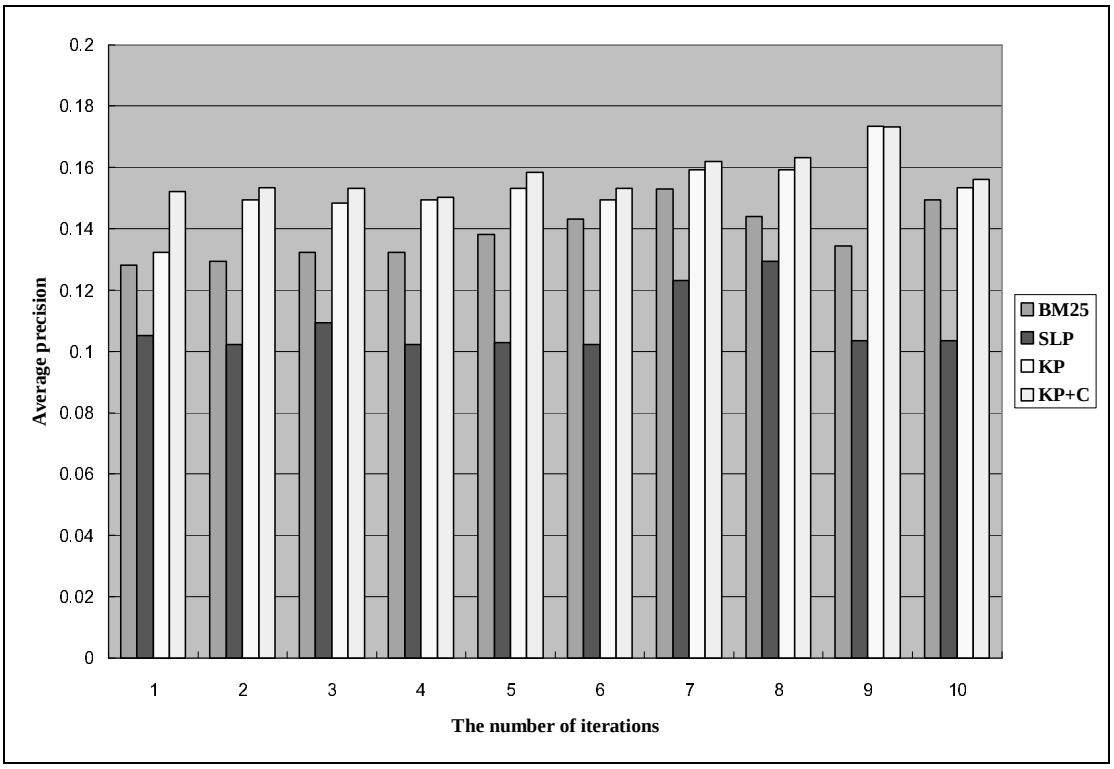

Figure 29: MEDLINE - PubMed with Four Query Expansion Algorithms

We also explored the effect of a sequence of query expansion iterations. Table 18 shows the results for four query expansion iterations. The second column is the number of retrieved documents from MEDLINE for each iteration. The third column shows the number of retrieved documents containing protein-protein pairs. The fourth column is the F-Measure (Van Rijsbergen, 1979). In the F-Measure, we use $b=2$ because recall is more important than precision in the tasks of retrieving the documents containing protein-protein interaction pairs. Our results show that the F-Measure generally increases as the number of iterations increases.

Table 21: Query Expansion Iterations for MEDLINE

Iteration	No of retrieved documents	No of documents containing protein-protein pairs	F-Measure (%)
1	30	18	47.76
2	609	289	51.65
3	832	352	51.27
4	1549	578	53.69
5	1312	545	53.21

5.1.3 Evaluation of Query Expansion with MEDLINE - Lemur

In this section, we report experimental results with query expansion algorithms on MEDLINE – Lemur. As described in the chapter 4, approximately 0.26 million MEDLINE records were indexed and searched with Lemur. As indicated in Table 19, the best performance was produced by the keyphrase-based query expansion algorithm with the POS phrase category (KP+C) both in average precision and P@20. The baseline query expansion algorithm was the next highest followed by BM25.

Table 22: MEDLINE-Lemur with Four Query Expansion Algorithms

Algorithm	**MEDLINE (0.26 million)**	
	Avg. P	P@20
BM25	0.2433	0.3798
SLP	0.1975	0.3241
KP	0.2645	0.3912
KP+C	0.2692	0.3933

Overall, these results with MEDLINE-Lemur are not different from the previous results. One interesting observation is that MEDLINE-Lemur produces the higher scores of average precision and P@20 than other dataset and search engine combinations. The reasons for these higher scores are because 1) the data collection drawn from MEDLINE is homogeneous in terms of the subject matters of the collection and 2) Lemur search engine is the better search engine to work with than PubMed and Zettair.

5.2 Experimental Results with the Impact of Ontologies on Retrieval

In this section, we report the experimental results about whether ontologies influence the retrieval performance. The following three ontologies are integrated into our query expansion system, DocSpotter: WordNet, MeSH, and UMLS. The results

of query expansion with these ontologies are compared with the ones with query expansion without ontologies.

KP: A keyphrase-based query expansion algorithm

KP+W: A keyphease-based query expansion algorithm integrated with WordNet

KP+M: A keyphrase-based query expansion algorithm integrated with MeSH ontology

KP+U: A keyphrase-based query expansion algorithm integrated with UMLS

The experiments are conducted on three test cases: MEDLINE-PubMed, MEDLINE-Lemur, and TREC-Zettair.

Table 23: Results for MEDLINE-Lemur with Ontolgies Impacts

Algorithm	**MEDLINE (0.26 million) – Lemur**	
	Avg. P	P@20
KP	0.2645	0.3912
KP+W	0.2615	0.3878
KP+M	0.2761	0.3967
KP+U	0.2773	0.4034

Table 20 shows the results of the ontologies impact on retrieval performance with MEDLINE-Lemur. The keyphrase-based technique integrated with UMLS

produces the highest average precision and P@20. The keyphrase-based technique with MeSH produces the next highest average and P@20 scores. The keyphrase-based technique with WordNet produces the lowest average and P@20 scores. We also conducted a set of experiments to examine whether ontologies have the impact on retrieval performance over numerations. Figure 30 shows the results of the experiments. The results lead us to identify the advantages of using ontologies for query expansion. Particularly, medical ontologies such as MeSH and UMLS make substantial improvement of the retrieval performance. In contrast, a general ontology such as WordNet fails to show that it improves the retrieval performance in the bio--medical domain.

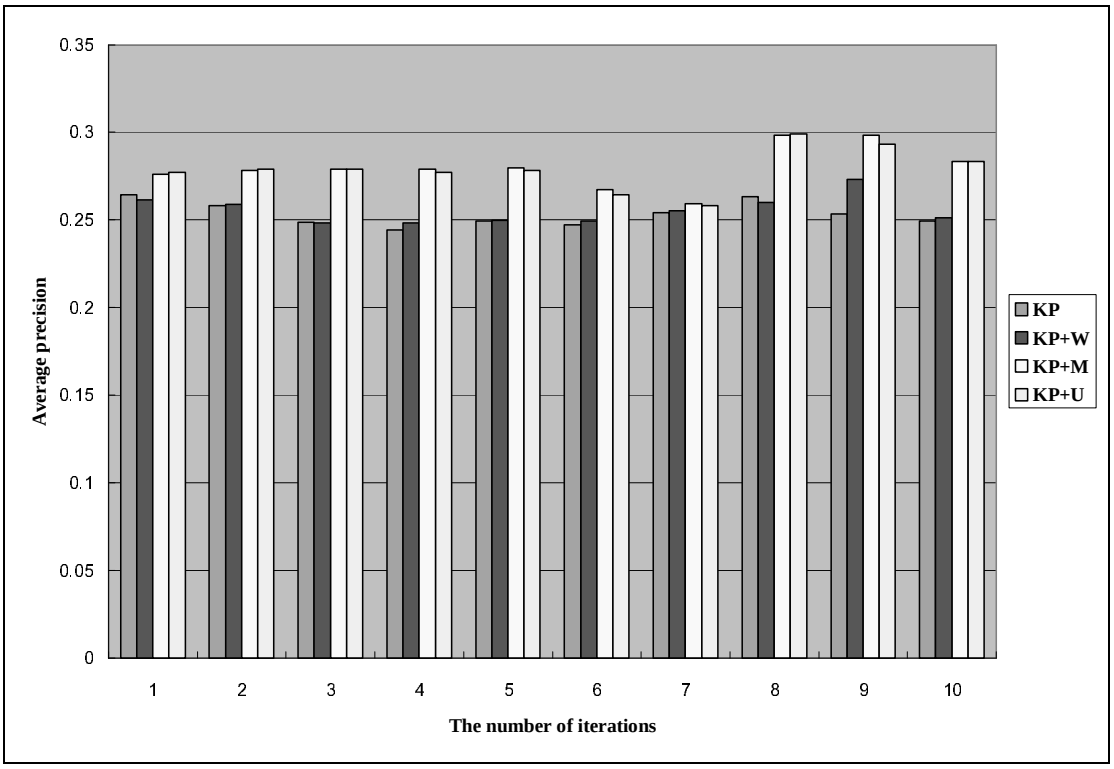

Figure 30: Ontologies' Impact on Retrieval Performance for MEDLINE-Lemur

We conducted the same set of experiments with MEDLINE – PubMed. One of the interesting findings is that the overall performance of query expansion with ontologies is inferior to the experiments with the other dataset and search engines.

Table 24: Results for MEDLINE-PubMed with Ontolgies Impacts

Algorithm	**MEDLINE – PubMed**	
	Avg. P	P@20
KP	0.1324	0.2844
KP+W	0.1325	0.2856
KP+M	0.1487	0.2911
KP+U	0.1483	0.2913

Table 21 shows the results of the ontologies' impact on retrieval performance with MEDLINE-PubMed. The keyphrase-based technique integrated with UMLS (KP+U) produces the highest average precision. The keyphrase-based technique with MeSH (KP+M) produces the highest P@20. The keyphrase-based technique with WordNet produces the next highest average and P@20 scores. The keyphrase-based technique without ontologies produces the lowest average precision and P@20. Figure 31 indicates how ontologies influence the retrieval performance over ten iterations. Compared to the results with MEDLINE-Lemur, the impact of MeSH and UMLS on the retrieval performance is amplified. In addition, the overall performance of query expansion with ontologies is worse than the one with other datasets and search engines. This phenomenon is consistently observed in other experimental cases.

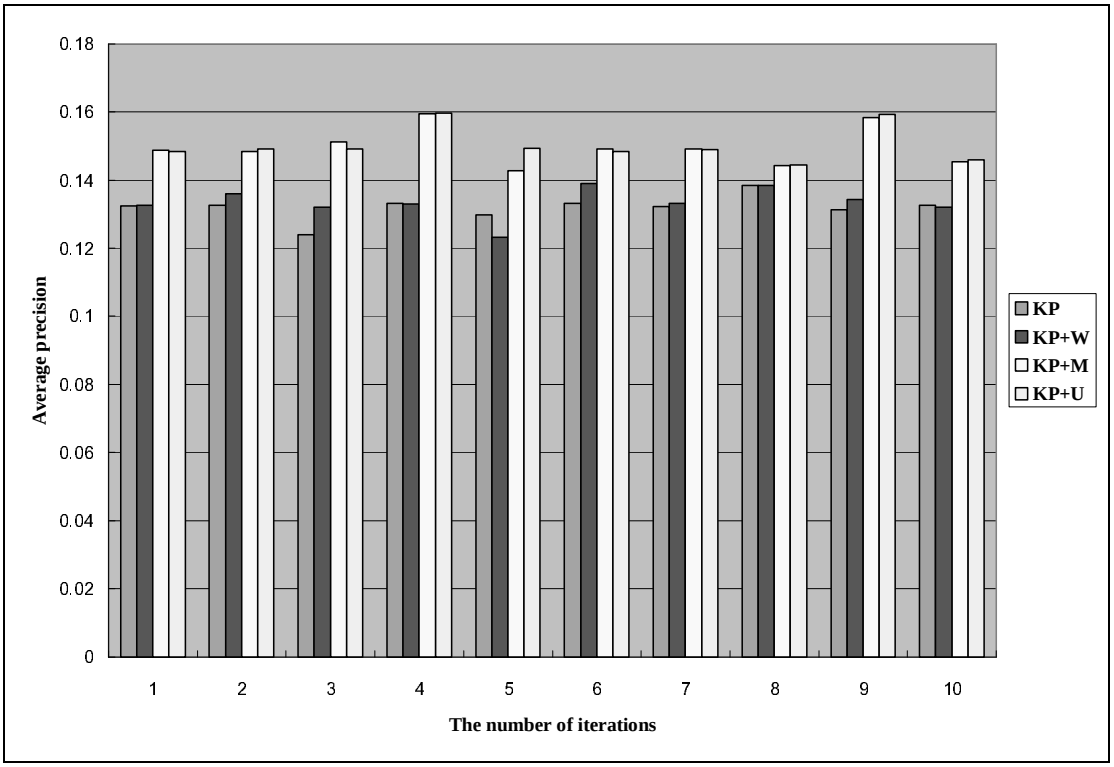

Figure 31: Ontologies' Impact on Retrieval Performance for MEDLINE-PubMed

In the experiments with TREC data – Zettair, we excluded testing the impact of UMLS and MeSH on the retrieval performance in that TREC data deals with the general topics in newswires and government documents.

Table 25: Results for TREC 5-Zettair with Ontolgies Impacts

Algorithm	**TREC5 – Zettair**	
	Avg. P	P@20
KP	0.1938	0.3368
KP+W	0.2012	0.3371

Table 22 shows the results of the ontologies impact on retrieval performance with TREC 5 - Zettair. The keyphrase-based technique integrated with UMLS (KP+U) produces the highest average precision and P@20. The keyphrase-based technique with MeSH (KP+M) produces the next highest average precision and P@20. The keyphrase-based technique with WordNet produces the next highest average and P@20 scores. The keyphrase-based technique without ontologies produces the lowest average precision and P@20.

Table 26: Results for TREC 6-Zettair with Ontologies Impacts

Algorithm	**TREC 6 – Zettair**	
	Avg. P	P@20
KP	0.2098	0.3390
KP+W	0.2114	0.3424

Table 23 shows the overall results of the ontologies impact on retrieval performance with TREC 6 - Zettair. The keyphrase-based technique integrated with UMLS (KP+U) produces the highest average precision and P@20. The keyphrase-based technique with MeSH (KP+M) produces the next highest average precision and P@20. The keyphrase-based technique with WordNet produces the next highest average and P@20 scores. The keyphrase-based technique without ontologies produces the lowest average precision and P@20.

Table 27: Results for TREC 7-Zettair with Ontologies Impacts

Algorithm	**TREC 7 – Zettair**	
	Avg. P	P@20
KP	0.2343	0.3878
KP+W	0.2319	0.3985

Table 24 shows the overall results of the ontologies impact on retrieval performance with TREC 7 - Zettair. The keyphrase-based technique integrated with UMLS (KP+U) produces the highest average precision and P@20. The keyphrase-based technique with MeSH (KP+M) produces the next highest average precision and P@20. The keyphrase-based technique without ontologies produces the next highest average and P@20 scores. The keyphrase-based technique with WordNet produces the lowest average precision and P@20.

In summary, integrating ontologies into the keyphrase-based query expansion improves the retrieval performance. In the cases of MEDLINE-PubMed and MEDLINE-Lemur, UMLS produced the best performance among ontologies. MeSH also shows the promising results for retrieval enhancement.

5.3 Experimental Results with HiMMIE

Sample Output

The sample outcome of running HiMMIE is illustrated in Table 25. Doc ID indicates the PubMed record ID that contains the abstract that the target protein pairs are extracted from.

Table 28: Sample Results of Running HiMMIE

Doc ID: PUBMED8681382 Sentence ID: 1 Target Protein 1: yta10p Target Protein 2: yta12p
Doc ID: PUBMED8182122 Sentence ID: 5 Target Protein 1: spo7p Target Protein 2: nem1p

Sentence ID is the ID assigned to the sentence by HiMMIE in the PubMed record. Target protein 1 and target protein 2 indicates the protein pairs that HiMMIE extracted for the task of protein-protein interaction.

We conducted experiments to evaluate the performance of HiMMIE on the task of protein-protein interaction extraction. In experiments the machine learning systems were trained using the abstracts with proteins and their interactions, processed by the text chunking technique. With these set of data, the IE systems extract interactions among these proteins. This gives us a measure of how the protein interaction extraction systems alone perform.

Performance is evaluated using ten-fold cross validation and measuring recall and precision. As the task of interest is only to extract interacting protein-pairs, in our evaluation we do not consider matching exact position and every occurrence of interacting protein-pairs within the abstract.

To evaluate our IE systems, we construct a precision-recall graph. Recall denotes the ratio of the number of slots the system found correctly to the number of slots in the answer key, and precision is the ratio of the number of correctly filled slots to the total number of slots the system filled

Table 29: Comparison of Extraction System Performance in First Round

Extraction System	*Precision*	*Recall*	*F-Measure*
RAPIER	60.17%	34.12%	44.13%
SVM	68.98%	48.23%	51.44%
MaxEnt	69.32%	49.03%	53.04%
Single POS HMM	67.40%	47.23%	50.58%
HiMMIE	70.23%	51.21%	58.33%

Our experiments show that RAPIER produces relatively high precision but low recall. The similar results are observed in the Single POS HMM method which gives also high precision but low recall. MaxEnt produces the second best results, although recall is relatively lower than precision. SVM produces the better results than RAPIER or Single POS HMM but the worse results than MaxEnt and HiMMIE. Among these five systems, HiMMIE outperforms RAPIER, single POS HMM, SVM, and MaxEnt in terms of precision, recall, and F-measure. As shown in Table 2, F-Measure of HiMMIE is 58.33% whereas RAPER is 44.13%, SVM is 51.44%, and single POS HMM is 50.58% and MaxEnt is 53.04%.

Figure 32 shows the precision-recall graphs of HiMMIE, RAPIER, SVM, MaxEnt, and single POS HMM-based extraction for the protein-protein interaction

data set. The curve for HiMMIE is superior to the curves for RAPIER, SVM, MaxEnt, and single POS HMM.

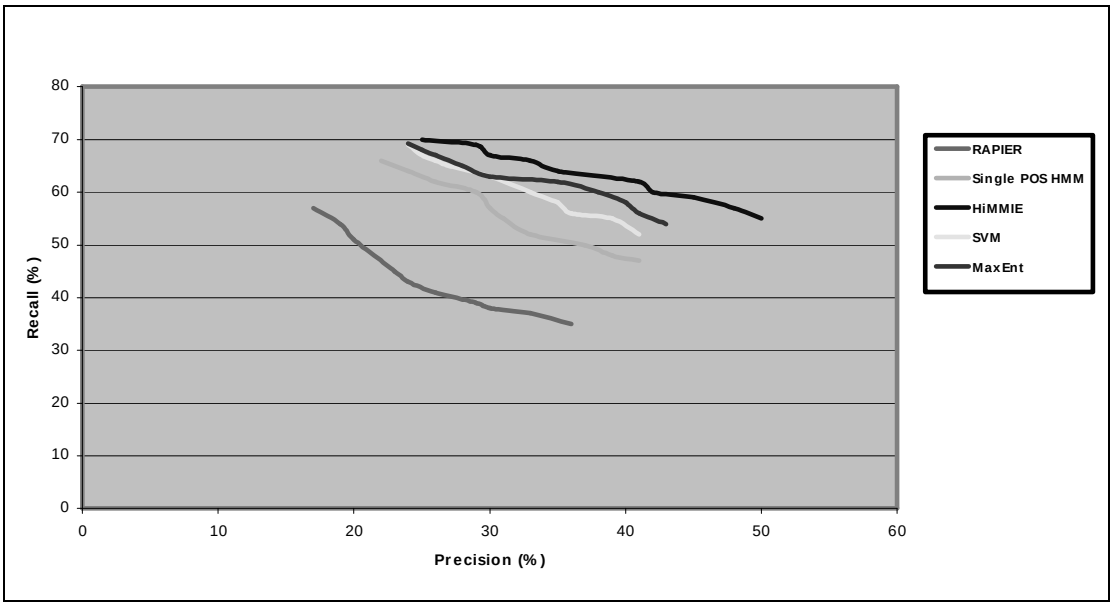

Figure 32: Extracting Protein-protein Interaction in First Round

We conducted another set of tests to investigate whether the results observed above are reproduced. To this end, we used input data that is obtained from DocSpotter as discussed. Since iterative query expansion is able to retrieve multiple set of documents, we use a set of documents retrieved in each round.

Table 30: Comparison of Extraction System Performance in Second Round

Extraction System	*Precision*	*Recall*	*F-Measure*
RAPIER	57.12%	37.53%	44.87%
SVM	70.32%	42.37%	52.51%
MaxEnt	70.89%	43.22%	53.27%
Single POS HMM	66.58%	44.01%	52.80%
HiMMIE	71.43%	51.91%	59.36%

Table 26 shows the experimental results with a new set of incoming data from DocSpotter. The apparent pattern of the results is resemble with the one reported in the previous run (Table 25). As indicated in Table 26, HiMMIE produces the best performance, precision 71.43%, recall 51.91%, and F-measure, 59.36%. The next highest score is produced by MaxEnt. RAPIER produces the lowest precision, recall, and F-Measure. Figure 33 shows the precision-recall graphs of HiMMIE, RAPIER, SVM, MaxEnt, and single POS HMM-based extraction for the protein-protein interaction data set. The curve for HiMMIE is superior to the curves for RAPIER, SVM, MaxEnt, and single POS HMM.

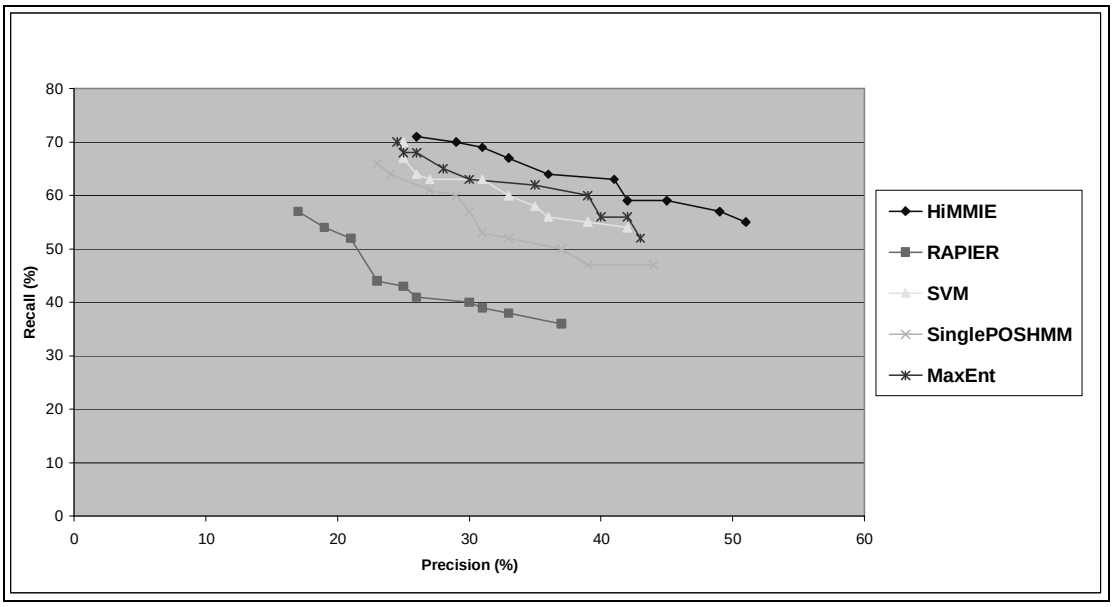

Figure 33: Extracting Protein-protein Interaction in Second Round

We repeated the same experimental tests over the 10 different datasets that are sent from DocSpotter. Figure 34 shows the results of the five extraction methods, HiMMIE, Single POS HMM, RAPIER, SVM, and MaxEnt in F-Measure. HiMMIE outperforms the other four algorithms. HiMMIE produces between 56.32% and 61.12% in F-Measure. Single POS HMM produces between 42.84% and 53.32% in F-Measure. RAPIER produces between 42.23% and 45.95% in F-Measure. SVM produces between 50.20% and 57.43% in F-Measure. MaxEnt produces between 51.23% and 58.23% in F-Measure.

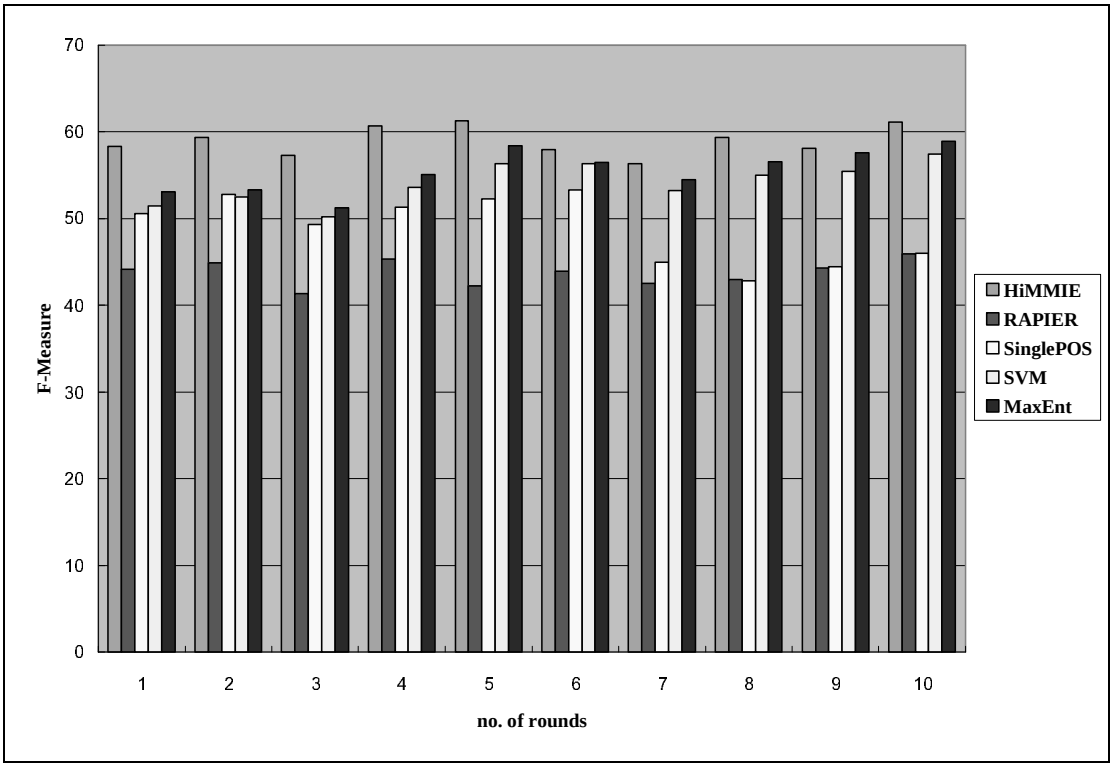

Figure 34: Overall Extraction Performance of the Five Algorithms

CHAPTER 6: CONCLUSION AND FUTURE STUDIES

Knowledge extraction is an emerging research area where various research fields need to be incorporated. This multidisciplinary nature of the field has led this dissertation to address and investigate the research problems in the context of assisting the users to find the desired information in an intelligent manner. In this dissertation, we proposed a hybrid knowledge extraction algorithm drawn from several research fields such as pseudo-relevance feedback, data mining, natural language processing, and information extraction. Specifically, we developed a robust novel extraction algorithm that consists of 1) a keyphrase-based query expansion to spot the promising documents and 2) a Mixture Hidden Markov model-based information extraction. We also conducted a series of experiments to validate three research hypotheses formed in this dissertation.

The major contributions of this dissertation are four-fold:

1. Automatic query generation for effective retrieval from text databases. This thesis introduced novel automatic query-based techniques to retrieve the articles/documents that are promising for the extraction of relations from text. The

proposed technique is based on keyphrases and POS-tagged categorization. The keyphrases are extracted from the retrieved documents and weighted with an algorithm based on information gain and co-occurrence of phrases. It automatically discovered the characteristics of documents that are useful for extraction of a target relation and generates queries in each iteration to select potentially useful articles from the text databases.

2. A statistical generative model, Mixture Hidden Markov Model (MiHMM) for automatic relation extraction. This thesis proposed MiHMM, a mixture of Hidden Markov Models (HMMs), organized in a hierarchical structure to help the IE system cope with data sparseness. MiHMM takes a set of sentences with contextual cues that were identified by a Support Vector Machine-based text chunking technique. MiHMM then learns a generative probabilistic model of the underlying state transition structure of the sentence from a set of tagged training data. Given a trained probabilistic mixture model of the data, the system then applies this model to new unseen input documents to predict which portions of these documents are likely targets according to the training data template. MiHMM is different from existing HMM-based approaches as follows: (a) It employs probabilistic mixture of HMMs that is hierarchically structured. (b) It incorporates contextual and semantic cues into

the learned models to extract knowledge from the unstructured text collections without any document structures. (c) It adopts a SVM text chunking technique to partition sentences into grammatically related groups. Thus using MiHMM for extracting biomedical entities has the following advantages over other approaches: (a) it overcomes the problem of the single POS HMMs with modeling the rich representation of text where features overlap among state units such as word, line, sentence, and paragraph. By incorporating sentence structures into the learned models, MiHMM provides better extraction accuracy than the single POS HMMs. (b) it resolves the issues with the single POS HMMs for IE that operate only on the semi-structured such as HTML documents and other text sources in which language grammar does not play a pivotal role.

3. Ontologies-assisted query expansion coupled with Keyphrases. In this disseration, we incorporated three ontologies such as WordNet, MESH, and UMLS into our query expansion technique. WordNet ontology was used to capture the noun form of the important verb terms such as “retrieve” or “interact” (Song et al., 2004). In our experiments, WordNet fails to make substantial enhancement of the retrieval performance. However, the experimental results showed that MeSH and UMLS made significant improvement of the retrieval performance. MeSH and UMLS ontologies

were used to expand queries by providing synonymously associated phrases.

4. Minimum human intervention required for knowledge extraction. From a system architecture perspective, RIKE is designed and developed with design pattern and object-oriented design methodologies. The architectural soundness of RIKE supported by object-oriented paradigm makes it possible to implement minimum human-intervention operation of the system. The whole procedure proposed in this dissertation was unsupervised with no human intervention except for a few seed instances provided by users in the very beginning. It makes it easy to port to a new domain without any changes to the extraction system. Also, it introduced a strategy for evaluating the quality of the patterns and the instances that are generated in the each iteration of the extraction process. Only those instances and patterns that are regarded as being "sufficiently reliable" were kept for the following iteration of the system. These new strategies for generating and filtering of patterns and instances improved the quality of the extracted instances and patterns significantly. In addition, RIKE is designed to plug into different search and extraction engines with minimum changes. RIKE is also ease to incorporate with various types of data formats such HTML, SGML, or XML.

6.1 Summary

We have proposed a novel knowledge extraction system, RIKE, which consists of DocSpotter, a keyphrase-based query expansion algorithm, and HiMMIE, a Mixture Hidden Markov model-based information extraction algorithm. We demonstrated that our query expansion algorithm based on keyphrases and the POS phrase category is effective and accurate in terms of average precision and precision at top 20. Encouraged by the previous studies on pseudo-relevance feedback, we applied keyphrase extraction techniques to query expansion. Along with keyphrase-based expansion, we employed a new word sense disambiguation technique in using ontology (e.g., WordNet) to add terms to the query. We also employed a POS phrase category-based Boolean constraint technique.

We also demonstrated that our techniques yielded significant improvements over the well-established BM25 algorithm and SLIPPER for the three TREC collections, TREC 5, 6, and 7, as well as for MEDLINE data on various search tasks including the protein-protein interaction tasks. Among four algorithms implemented, BM25, SLP, KP, and KP+C, the experimental results indicate that the KP+C algorithm proved to be the best.

We validated that ontologies-assisted query expansion improved the retrieval performance in the biomedical domain. However, we did not observe the same level of improvement with WordNet although WordNet-assisted query expansion outperformed BM25 and SLIPPER. To improve the peformance of WordNet-based query expansion, applying ontologies to original query or both original query and keyphrases as carried out in (Liu et al., 2004) can be considered.

We also proved that Mixture Hidden Markov model-based extraction is a high quality extraction algorithm, particularly useful for biomedical entity relations. We compared HiMMIE with four well-known IE techniques: 1) RAPIER, a rule-based machine learning system, 2) SVM, Support Vector Machine-based extraction algorithm, 3) MaxEnt, Maximum Entropy-based extraction algorithm, and 4) single POS HMM. Our experiments showed that HiMMIE outperforms these IE techniques in extracting protein-protein interactions in terms of F-measure.

In this dissertation, we addressed three challenging research problems, proposed a novel knowledge extraction system, and validated the research hypotheses. It is our sincere hope that this dissertation contributes to trailblazing a new research area such as knowledge extraction.

6.2 Recommendations for Further Research

The results of this dissertation stimulate further research in several directions. First, given the results that keyphrase-based query expansion is proven to be effective, it is worthwhile to investigate how effective it is to apply the keyphrase-based technique to other research problems such as text summarization and categorization. Text summarization is the process of identifying salient concepts in text narrative, conceptualizing the relationships that exist among them and generating concise representations of the input text that preserve the gist of its content (Radev et al., 2004). Keyphrase-based text summarization would be an interesting approach to summarization in that summarizing the collections with top N ranked keyphrases generates semantically cohesive passages. In addition, a keyphrases-base approach could be applied to automatic class modeling. For example, keyphrases can be extracted from text descriptions such as functional requirements and class model descriptions. With extracted keyphrases, we can identify a set of core classes and its relationship with other classes.

Second, it is interesting to investigate how HiMMIE performs when it is applied to other types of relation extractions such as subcellular-localization relation extraction. In addition, applying HiMMIE to other domains such as Web data

extraction would be a challenging but interesting research project. In addition to relation extraction, HiMMIE can be applied to entity extraction such as extracting CEO names from newswires.

Third, it is worthwhile to examine how to improve word sense disambiguation. In this dissertation, we did not observe the satisfactory results with our technique of word sense disambiguation that is keyphrases-driven. We need to explore more ways of fine-tuning word sense disambiguation technique such as the techniques proposed by Lie et al. (2004) to improve the retrieval accuracy with WordNet further. In the Hu et al.'s study (2004), we applied a medical ontology, UMLS, to entity tagging and gained promising results. As a follow-up study, we are investigating whether an ensemble approach, combining several ontologies such as WordNet, MeSH, and UMLS into one, to entity tagging would improve the performance.

We are also interested in whether keyphrases help the users understand the content of a collection and provide sensible entry points into it. In addition, we will investigate whether and how keyphrases can be used in information retrieval systems as descriptions of the documents returned by a query, the basis for search indexes, a

way of browsing a collection, and a document clustering technique.

In the context of visualization, it is worthwhile to examine whether keyphrases-based visualization would be usable for the users to navigate the document space. Instead of building a document term matrix from the entire text collections, using top N ranked keyphrases from the documents would generate a more semantically meaningful map than using the entire terms sets from the collection. It also dramatically improves speed of visualizing the text collection.

LIST OF REFERENCES

Agichtein, E. and Gravano, L. (2003). Querying Text Databases for Efficient Information Extraction, in *Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE)*, 113-124.

Agichtein, E. and Gravano, L. (2000). Snowball: Extracting Relations from Large Plain-Text Collections, in *Proceedings of the 5th ACM International Conference on Digital Libraries (DL)*, 85-94.

Alani, H., Kim, S., Millard, D.E., Weal, M.J., Hall, W., Lewis, P.H., and Shadbolt, N.R. (2003). Automatic Ontology-based Knowledge Extraction from Web Documents, *IEEE Intelligent Systems*, 18 (1): 14-21.

Allen, J. (1995). *Natural Language Understanding*. Addison-Wesley Pub Co, New York.

Andrade, M.A. and Valencia, A. (1998). Automatic Extraction of Keywords from Scientific Text: Application to the Knowledge Domain of Protein Families, *Bioinformatics*. 14 (7): 600-607.

Andrade, M. A. and Bork, P. (2000). Automated Extraction of Information in Molecular Biology. *FEBS Letters*, 476:12-7.

Andrade, M. A. and Valencia, A. (1997). Automatic Annotation for Biological Sequences by Extraction of Keywords from MEDLINE Abstracts. Development of a prototype system. In *Fifth International Conference on Intelligent Systems for Molecular Biology* (Gaasterland, T., Karp, P., Karplus, K., Ouzounis, C., Sander, C. et al., eds.), pp. 25-32, AAAI Press, Halkidiki, Greece.

Andrade, M., Blaschke, C. and Valencia, A. (1999). AbXtract: Automatic Abstract eXtraction of Keywords Associated to Protein Function. *Bioinformatics*. 14(7):600-7.

Antworth, Evan L. (1993). Glossing Text with the PC-KIMMO Morphological Parser. *Computers and the Humanities*. pp. 475-484.

Apte, C. and Damerau, F. (1994). Automated Learning of Decision Rules for Text Categorization, *ACM Transactions on Information Systems*, 12(3): 233-251.

Aronson, A.R. and Rindflesh, T.C. (1997). Query Expansion Using the UMLS Metathesaurus. *American Medical Informatics Association Fall Symposium*, 485-489.

Azari D, Horvitz E, Dumais S, and Brill E. (2004) Actions, Answers, and Uncertainty: a Decision-making Perspective on Web-based Question Answering, *Information Processing & Management*, 40 (5): 849-868.

Bachmann, L.M., Cora, R., Estermann, P., and Riet, G. (2002). Identifying Diagnostic studies in MEDLINE: Reducing the Number Needed to Read, *Journal of the American Medical Informatics Association*, 9, 653-658.

Bader, G., Donaldson, I., Wolting, C., Francis Ouellette, B., Pawson T., and Hogue, C. (2001). BIND—The Biomolecular Interaction Network Database, *Nucleic Acids Research*, 29(1), 242-245.

Baeza-Yates, R. and Ribeiro-Neto,B. (1999). *Modern Information Retrieval*.

Baldi, P., Chauvin, Y., Hunkapiller, T., and McClure, M.A. (1994). Hidden Markov Models of Biological Primary Sequence Information. *Proceedings of the National Academy of Sciences of the United States of America*, 91(3), 1059-1063.

Bennett, N.A., He, Q., Powell, K., and Schatz, B.R. (1999). Extracting Noun Phrases for All of MEDLINE. *Journal of the American Medical Informatics Association*, 671-675 Suppl. S.

Berger, A.L, Della Pietra, S.A, and Della Pietra, V.J. (1996) A Maximum Entropy Approach to Natural Language Processing, *Computational Linguistics*, 22(1): 39-71.

Billerbeck, B and Zobel, J. (2004) Questioning Query Expansion: an Examination of Behavior and Parameters, in: *Proceedings of Fifteenth Australasian Database Conference*, 2004; 69-76.

Blum, A. and Mitchell T. (1998). Combining Labeled and Unlabeled Data with Co-training. In *Proceedings of the 1998 Conference on Computational Learning Theory*, 92 – 100.

Blaschke, C., Hirschman, L. &Valencia, A. (2002). Information Extraction in Molecular Biology. *Brief Bio inform*, 3,154-165.

Blaschke, C., Andrade, M.A., Ouzounis, C., and Valencia, A. (1999). Automatic Extraction of Biological Information from Scientific Text: Protein-Protein Interactions, *In Proceedings of the First International Conference on Intelligent Systems for Molecular Biology,* 60-67.

Blaschke, C. and Valencia, A. (2001). Can Bibliographic Pointers for Known Biological Data Be Found Automatically? Protein interactions as a case study, *Comparative and Functional Genomics*. 2 (4): 196-206 AUG.

Blaschke, C. (2001). Applications of Information Extraction Techniques to Molecular Biology. U. Autonoma Madrid, *PHD thesis*, Department of Computer Science.

Blaschke, C. & Valencia, A. (2002). *The Frame-based Module of the Suiseki Information Extraction System.* IEEE Intelligent Systems, *17, 14-20.*

Brin, S (1998). Extracting Patterns and Relations from the World-Wide Web. In *Proceedings of the 1998 International Workshop on the Web and Databases (WebDB'98)*, March.

Bunescu, R, Ge, R, Kate, R.J, Marcotte, E.M, Mooney, R.J, Ramani, A.K, and Wong, Y.W. (2004) Comparative Experiments on Learning Information Extractors for Proteins and Their Interactions. To appear in *Journal Artificial Intelligence in Medicine (Special Issue on Summarization and Information Extraction from Medical Documents).*

Califf, M.E., (1998). Relational Learning Techniques for Natural Language Information Extraction, Ph.D. Dissertation. *The Rept. AI 98-276, Artificial Intelligence Lab, The University of Texas at Austin*

Califf, M.E., Mooney R. (2003). Bottom-Up Relational Learning of Pattern Matching Rules For Information Extraction, *Journal of Machine Learning Research* 2(2003) 177-210

Callan, J. and Connell,M. (2001). Query-Based Sampling of Text Databases, *ACM Transactions on Information Systems,* 19(2): 97-130, 2001.

Cardie, C. (1997). Empirical Methods in Information Extraction, *AI Magazine*, 18(4), 65-80.

Chang, K.C, Garcia-Molina, H., and Paepcke, A. (1996). Boolean Query Mapping Across Heterogeneous Information Sources, *IEEE Transactions on Knowledge and Data Engineering*, 8(4): 515-521.

Chen, H., Shankaranarayanan, G., She, L., and Iyer, A. (1998). A Machine Learning Approach to Inductive Query by Examples: An Experiment Using Relevance Feedback, ID3, Genetic Algorithms, and Simulated Annealing. *Journal of American Society for Information Science*, 49(8), 693-705.

Chu, W.W, Liu, Z., Mao, W., Zou, Q. (2005) A Knowledge-based Approach for Retrieving Scenario-specific Medical Text Documents, *Control Engineering Practice (CEP), Special Section on Biomedical Control*.

Cristianini, N. and Shawe-Taylor, J. (2001). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge University Press. 2001;

Ciravegna, F. (2001). Adaptive Information Extraction from Text by Rule Induction and Generalization, in *Proceedings of the 17th International Joint Conference on AI*, 1251–1256.

Cohen, W. and Singer, Y. (1996). Learning to Query the Web. In *Proceedings of the AAAI Workshop on Internet-Based Information System*,

Collier, N., Nobata,C., and Tsujii, J. (2000) Extracting the Names of Genes and Gene Products with a Hidden Markov Model. *Proceedings of the 18th International Conference on Computational Linguistics (COLING2000)*, 201-207.

Cortes, C. and Vapnik V. (1995). Support-vector Networks. *Machine Learning*, 20(3): 273-297.

Cowie, J., Wakao, R., Guthrie, L., Jin W., Pustejovsky, J. and Waterman S (1992). The Diderot Information Extraction System. In *Proceedings of the 4th Message Understanding Conference*

Cowie, J. & Lehnert, W. (1996). Information Extraction. *Communications of ACM*, 39, 80-91.

Craven, M. and Kumlien, J. (1999). Constructing Biological Knowledge Bases by Extracting Information from Text Sources. *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pp. 77-86.

Croft, W.B., Lavrenko, V., and Cronen-Townsend, S. (2001). Relevance Feedback and Personalization: A Language Model Perspective, *Proceedings of the Joint DELOS-NSF Workshop on Personalization and Recommender Systems in Digital Libraries*, 49-54.

Crouch, C.J. (1988). A Cluster-based Approach to Thesaurus Construction, in *Proceedings of 11th International Conference on Research and Development in Information Retrieval*, 309-320.

Day, D., Aberdeen, J., Hirshman, L, Kozierok, R., Robinso, P., and Vilain, M. (1997). Mixed-Initiative Development Of Language Processing Systems. In *Proceedings of the fifth ACL Conference on Applied Natural Language Processing*, April 1997

De Bruijn, B. and Martin, J. (2002). Getting to the (C)ore of Knowledge: Mining Biomedical Literature. *International Journal of Medical Informatics*, (67): 7-18.

Demetriou, G. and Gaizauskas, R. (2002). Utilizing Text Mining Results: The Pasta Web System. *Proceedings of the Workshop on Natural Language Processing in the Biomedical Domain*, 77-84.

Ding, J., Berleant, D., Nettleton,D. and Wurtele, E. (2002). Mining MEDLINE: Abstracts, Sentences, or Phrases? *Pacific Symposium on Biocomputing*, 326-337.

Dougherty, J., Kohavi, R. and Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. In: *Proceeding of ICML-95, 12th International Conference on Machine Learning*, Lake Tahoe, US, pp.194--202.

Dumais, S. and Chen, H. (2000). Hierarchical Classification of Web Content. *Proceedings of SIGIR 2000*. Athens, Greece. July 24-28, 256-263.

Efthimiadis, E.N. (1996). Query Expansion, in *Annual Review of Information Systems and Technology*, 31, 121-187.

Eickeler, S., Kosmala, A., Rigoll, G. (1998). Hidden Markov Model based Online Gesture Recognition. *Proc. Int. Conf. on Pattern Recognition (ICPR)*, 1755–1757.

Fisher, D., Soderland S., McCarthy J., Feng F. and Lehner W (1995). Description of the Umass Systems As Used For MUC-6. In *Proceedings of the 6th Message Understanding Conference*, 1995

Flake,G., Glovers, E.J., Lawrence, S., and Giles, C.L (2002). Extracting Query Medications From Nonlinear Svms, in *Proceedings of the 11th World Wide Web Conference* (WWW-2002), 2002

Feldman, R., Aumann, Y., Fresko, M., Liphstat, O., Rosenfeld, B., and Schler, Y. (1999). Text Mining via Information Extraction. *Lecture Notes in Artificial Intelligence: Principles of Data Mining and Knowledge Discovery*, 704: 165-173.

Folwer, M and Scott, K. (1997). *UML Distilled: Applying the Standard Object Modeling Language*, Addison-Wesley Longman Ltd.

Frank E., Paynter G.W., Witten I.H., Gutwin C. and Nevill-Manning C.G. (1999). Domain-specific Keyphrase Extraction, In: *Proc. Sixteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers, San Francisco, CA, pp. 668-673.

Frants, V.I., and Shapiro, J. (1991). Algorithm for Automatic Construction of Query Formulations in Boolean Form, *JASIS*, 42(1): 16-26.

Freitag, D. (1998). Toward General-Purpose Learning for Information Extraction. In *Proceedings of the 36th Annual Meetings of the Association for Computational Linguistics*

Freitag, D. and McCallum, A. (2000). Information Extraction with HMM structures learned by stochastic optimization. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, July 30 - August 3 2000. Austin, Texas.

Freitag, D. (2000). Machine Learning for Information Extraction in Informal Domains, *Machine Learning*, 39, 169-202.

French, J.C., Powell, A.L., Gey, F. and Perelman, N. (2001). Exploiting a Controlled Vocabulary to Improve Collection Selection and Retrieval Effectiveness. *10th International Conference on Information and Knowledge Management*.

Friedman, C., Kra, P., Yu, H.,Krauthammer, M. & Rzhetsky, A. (2001). GENIES: a Natural-language Processing System for the Extraction of Molecular Pathways from Journal Articles. *Bioinformatics*, 17 Suppl 1, S74-82.

French, J.C., Brown , D.E., and Kim, N.H. (1997). A Classification Approach to Boolean Query Reformulation, *Journal of the American Society for Information Science*, 48(8): 694-706.

Fukuda, K., Tamura, A., Tsunoda, T., and Takagi, T. (1998). Toward Information Extraction: Identifying Protein Names from Biological Papers. *Pacific Symposium Biocomputing*. pp. 707-18.

Furnkranz, J. and Widmer, G. (1994). Incremental Reduced Error Pruning. In: Machine Learning: Proceedings of the Eleventh Annual Conference. New Brunswick, New Jersey: Morgan Kaufmann.

Gauch, S., Wang, J., and Rachakonda, S.M. (1997). A Corpus Analysis Approach for Automatic Query Expansion and its Expansion to Multiple Databases, *ACM Transaction on Information Systems*.

Gordon, M. (1988). Probabilistic and Genetic Algorithms for Document Retrieval, *Communications of the ACM*, Vol. 31 (10).

Gravano, L., Iperiotis P., and Sahami, M (2002). Qprober: A System for Automatic Classification O Hidden-Web Databases. *ACM Transaction on Information Systems (TOIS)*. vol. 21, no. 1

Grishman, R., Huttunen, and Yangarber, R. (2002). Real-Time Event Extraction For Infectious Disease Outbreaks, in *Proceedings of Human language Technology Conference* (HLT), 2002

Hahn, U., Romacker, M., Schulz, S. (2002). Creating Knowledge Repositories from Biomedical Reports: The MEDSYNDIKATE Text Mining System. *Pacific Symposium on Biocomputing*, pp. 338-349.

Harabagiu S., Moldovan, D, Pagca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V. and Morarescu, P (2000). Falcon: Boosting Knowledge for Answer Engines, *In the Proceedings from the Ninth TREC Conference (TREC-9), 2000*.

Harris, Z., (1985). Distributional Structure, In Katz, J.J., (ed). *The Philosophy of Linguistics*, New York: Oxford University Press, pp 26-47

Hatzivassiloglou, V., Duboue, P. A .& Rzhetsky, A. (2001). Disambiguating Proteins, Genes, and RNA in Text: a Machine Learning Approach. *Bioinformatics*, 17, S97-S106.

Hearst, M.A. (1996). Improving Full-Text Precision on Short Queries Using Simple Constraints, In: *Proceedings of the Symposium on Document Analysis and Information Retrieval*.

Hersh, W., Price, S. and Donohoe, L. (2000). Assessing Thesaurus-Based Query Expansion Using the UMLS Metathesaurus, in *Proceedings of AMIA Symposium*, 344-348.

Hersh, W.R., Bhupatiraju, R.T., and Price, S. (2003) Phrases, Boosting, and Query Expansion Using External Knowledge Resources for Genomic Information Retrieval. *TREC 2003*, 503-509.

Hersh WR, Campbell EM, Malveau SE, Assessing the Feasibility of Large-scale Natural Language Processing in a Corpus of Ordinary Medical Records: a Lexical Analysis, *Proceedings of the 1997 AMIA Annual Symposium*, 1997, 580-584.

Hersh, W. R., Evans, D. A., Monarch, I. A. & Gorman, P. N. (1992). Indexing Effectiveness of Linguistic and Non-Linguistic Approaches to Automatic Indexing. Elsevier Science Publishers, Amsterdam.

Hersh, W.R., Hickam, D.D., Haynes, R.B., and McKibbon, K.A. (1994). A Performance and Failure Analysis of SAPHIRE with a MEDLINE Test Collection. *Journal of the American Medical Informatics Association*, 1(1): 51-60.

Hirschman, L., Park, J.C., Tsujii, J., Wong, L., and Wu, C.H. (2002). Accomplishments and Challenges in Literature Data Mining for Biology. *Bioinformatics*, 18 (12): 1553-1561.

Hirschman, L. (1998). The Evolution of Evaluation: Lessons from the Message Understanding Conferences, *Computer Speech and Language*,12, 281-305.

Hu, J., Brown, M.K., Turin, W. (1996). HMM based On-line Handwriting Recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(10): 1039–1045.

Hu, X., Yoo, I (2004). Scalable Learning Method To Extract Biological Information from Huge Online Biomedical Literature, Chapter 23 in *Computational Web Intelligence: Intelligent Technology for Web Applications*, World Scientific Publisher, in print, 2004

Hu, X., Yoo, I., Song, I.Y., Song, M, Han, J., and Lechner, M. (2004). Extracting and Mining Protein-protein interaction Network from Biomedical Literature, *2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*.

Hu X., Lin T.Y. Song IY, Lin. X, Yoo, I, Song. M (2004). An Ontology-based Scalable and Portable Information Extraction System to Extract Biological Knowledge from Huge Collection of Biomedical Web Documents, In: *Proceedings of the 2004 IEEE/ACM Web Intelligence Conference*, 77-83.

Hu, X.(2004). Scalable Learning Method to Extract Biological Relationships from Medline Abstracts, *IASTED International Conference on Biomedical Engineering*, Feb 16-18, 2004, Innsbruck, Austria

Hu, X. and Cercone, N. (2001). Discovering Maximal Generalized Decision Rules through Horizontal and Vertical Data Reduction**,** *Computational Intelligence: An International Journal*, 17(4), 685-702, 2001

Hu, X. and Cercone, N. (1999). Data Mining via Discretization, Generalization and Rough Set Feature Selection, Knowledge and Information System: An International Journal, 1(1), 33-60, 1999

Huffman, S., (1996). Learning Information Extraction Patterns from Examples, In Wermter, S; Riloff ,E; and Scheler, G., eds., *Connectionist, Statistical, and Symbolic Approaches to learning for Natural Language Processing*. Springer-Verlag, Berlin, 246-260

Hughey, R. and Krogh, A. (1996). Hidden Markov Model for Sequence Analysis: Extension and Analysis of the Basic Method. *Computer Applications in the Biosciences*, 12:95–107.

Humphreys, K., Demetriou, G., Gaizauskas, R. (2000). Two Applications of Information Extraction to Biological Science Journal Articles: Enzyme Interactions and Protein Structures. *Pacific Symposium on Biocomputing*, pp. 505-16.

Iliopoulos, I., Enright, A. J.and Ouzounis, C. A. (2001). Textquest: Document Clustering of MEDLINE Abstracts for Concept Discovery in Molecular Biology. *Pac Symp Biocomput*,384-95.

Jenssen, T.K., Laegreid, A., Komorowski, J., and Hovig, E. (2001). A Literature Network of Human Genes for High-throughput Analysis of Gene Expression. *Nature Genetics*, May; 28(1):21-8.

Joachims, T. (2000). *Learning to Classify Text using Support Vector Machines*, Kluwer, 2002.

Krauthammer, M., Kra, P., Iossifov,I., and Gomez, S. M., Hripcsak, G. et al. (2002). Of Truth and Pathways: Chasing Bits of Information through Myriads of Articles. *Bioinformatics*, 18 Suppl 1, S249-S257.

Kudo, T. and Matsumoto, Y. (2000). Use of Support Vector Learning for Chunk Identification. In Proceedings of CoNLL- 2000 and LLL-2000, Saarbruncken, Germany, 142-144.

Lafferty J., Sleator D., and Temperley D. (1992). Grammatical Trigrams: A Probabilistic Model of Link Grammar. In: *Proceedings of the AAAI Conference on Probabilistic Approaches to Natural Language*, October.

Lam-Adesina A.M., and Jones, G.J.F. (2001). Applying Summarization Techniques for Term Selection in Relevance Feedback, *Proceedings of the 24th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*: 1-9.

Lee, K., Hwang, Y and Rim, H. (2003). Two-Phase Biomedical NE Recognition based on SVMs. *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, 33-40.

Leek, T. R. (1997). Information Extraction Using Hidden Markov Models. *MSc Thesis,* Department of Computer Science, University of California, San Diego.

Lewis, D.D. (1995). Active by accident: Relevance feedback in information retrieval, *AAAI Fall Symposium on Active Learning*.

Lewis, D.D. and Ringuette, M. (1994). A Classification of Two Learning Algorithms for Text Categorization, *Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR94)*, 81-93.

Lindberg, D.A.B., Humphreys, B.L., and McCray, A.T. (1993). The Unified Medical Language System Project, Methods of Information in Medicine, 32: 281-291.

Liu, B., Hsu, W., and Ma, Y. (1998). Integrating Classification and Association Rule Mining. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98, Plenary Presentation), New York, USA.

Liu, S., Liu, F., Yu, C., and Meng, W. (2004). An Effective Approach to Document Retrieval via Utilizing WordNet and Recognizing Phrases, *Proceedings of the 27th annual international Conference on Research and development in Information Retrieval*: 266-272.

Manning C. Manning and Schütze H., (1999). *Foundations of Statistical Natural Language Processing*, MIT Press. Cambridge, MA: May.

McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the Construction of Internet Portals with Machine Learning, in *Information Retrieval*, 127-163.

McCallum, A., Nigam, K., Rennie, A., and Seymore, K. (1999). A Machine Learning Approach to Building Domain-specific Search Engine, in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, 686.

McCallum. A., and Nigam, K. (1998). A Comparision of Event Models for Naïve Bayes Text Classification, *AAAI-98 Workshop on Learning for Text Categorization*.

Mendonca, EA; Cimino, JJ. (2000). Automated Knowledge Extraction from MEDLINE Citations. *Journal of the American Medical Informatics Association*, 575-579.

Miller, G (1990). Wordnet: An On-line Lexical Database, *International Journal of Lexicography* 3(4).

Mihalcea, R., and Moldovan, D. (2000). Semantic Indexing Using WordNet Senses. ACL *Workshop on IR & NLP*.

Mitra, C.U., Singhal, A., and Buckely, C. (1998). Improving Automatic Query Expansion, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*: 206-214.

MUC-4 Proceedings. (1992). Proceedings of the Fourth Message Understanding Conference (MUC-4). San Mateo, CA: Morgan Kaufmann.

MUC-7 Proceedings. (1998). Proceedings of the Seventh Message Understanding Conference (MUC-7). San Mateo, CA: Morgan Kaufmann

Narayanaswamy, M., Ravikumar, K.E. and Vijay-Shanker, K. (2003). A Biological Named Entity Recognizer. *Pacific Symposium on Biocomputing*, 427-438.

National Library of Medicine (2003). *The MEDLINE database*, http://www.mcbi.nlm.nih.gov/PubMed/.

Ng, S. K. and Wong, M. (1999). Toward Routine Automatic Pathway Discovery from On-line Scientific Text Abstracts. *Genome Informatics Series: Workshop on Genome Informatics*, 10: 104-112.

Ohta, Y., Yamamoto, Y., Okazaki,T., Uchiyama, I. and Takagi, T. (1997). Automatic Construction of Knowledge base from Biological Papers. *Proceedings of International Conference on Intelligent System for Molecular Biology*, 5:218-25.

Ogilvie, Paul and Jamie Callan (2001) Experiments Using the Lemur Toolkit. In *the Proceedings of the Tenth Text Retrieval Conference, TREC 2001*. NIST Special Publication 500-250, pp. 103-108.

Okapi. http://www.soi.city.ac.uk/~andym/OKAPI-PACK/

Park, J.C., Kim, H.S., and Kim, J.J. (2001). Bidirectional Incremental Parsing for Automatic Pathway Identification with Combinatory Categorical Grammar, *Pacific Symposium on Biocomputing*, pp.396-407.

Pizzato, L.A.S. and de Lima, V.L.S. (2003). Evaluation of a Thesaurus-based Query Expansion Technique, *Proceedings of Lecture Notes in Artificial Intelligence*, 251-258.

Porter, M.F. (1980). An Algorithm for Suffix Stripping, *Program*, 14(3), pp. 130-137.

Proux, D., Rechenmann, F., and Julliard, L. (2000). A Pragmatic Information Extraction Strategy for Gathering Data on Genetic Interactions. *Proceedings of International Conference on Intelligent System for Molecular Biology*, 8:279-85.

Qiu, Y., and Frei, H. (1993). Concept-based Query Expansion, In: *Proceedings of the 16th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*: 160-169.

Quinlan, J. R. (1993). *Programs for Machine Learning*, San Mateo, CA: Morgan Kaufmann Publishers.

Pustejovsky, J., Castano, J., Zhang, J., Kotecki, M., Cochran, B. (2002). Robust Relational Parsing over Biomedical Literature: Extracting Inhibit Relations. *Pacific Symposium on Biocomputing*, pp. 362-73.

Rabiner, L.R. (1989). A tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of IEEE*, 77(2): 257–286.

Radev, D.R., Jing, H., Stys, M. and Tam, D. (2004). Centroid-based Summarization of Multiple Documents. *Information Processing and Management*, 40:919-938.

Ratnaparkhi, A. (1996). A Maximum Entropy Model for Part-of-Speech Tagging. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 133-142.

Ray, S. & Craven, M. (2001). Representing Sentence Structure in Hidden Markov Models for Information Extraction. *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, WA. Morgan Kaufmann.

Riloff, E. and Jones, R. (1999) .Learning Dictionaries for Information Extraction by Multi-level Bootstrapping, in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99),* 474-479.

Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged text *Prod. of the 13th National Conference on AI (AAAI-96),* pp1044-1049

Rindflesch, T.C., Tanabe, L., Weinstein, J.N., Hunter, L. (2000). EDGAR: Extraction of Drugs, Genes and Relations from the Biomedical Literature. *Pacific Symposium on Biocomputing*, pp. 517-28.

Ro, J.S. (1988). Evaluation of the Applicability of Ranking Algorithms, Pt. I and Pt. II. *Journal of the American Society for Information Science*, 39, 73-78: 147-160.

Robertson, S.E. and Sparck, J.K. (1976) Relevance Weighting of Search Terms, *Journal of the American Society for Information Science*, 27, 129-146.

Robertson, S. (1990). On Term Selection for Query Expansion. In *Journal of Documentation*, Vol 46, 1990

Rocchio, J. (1971). Relevance Feedback in Information Retrieval. In *SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971

Roth, D, Yih, W (2001). Relational Learning via Propositional Algorithm: An Information Extraction Case Study, *IJCAI'2001*

Ruiz, M.E. and Srinivasan, P. (1998). Crosslingual Information Retrieval with the UMLS: An Analysis of Errors. In*: Proceedings of the 61st Annual Meeting of the American Society for Information Science,* Pittsburgh, PA. 153-165.

Sager, W.K.H. and Lockemann, P.C. (1976). Classification of Ranking Algorithms, *International Forum for Information and Documentation*, 1, 12-25.

Salton, G. (1989). Automatic Text Processing*: The Transformation, Analysis and Retrieval of Information by Computer*, Addison-Wesley, 1989.

Salton G. and Buckley, C. (1988). Term-weighting Approaches in Automatic Text Retrieval, *Information Processing and Management*, pp.513-523.

Schapire, R., Singer, Y. and Singhal, A (1998). Boosting and Rocchio Applied to Text Filtering, *Proceedings of ACM SIGIR* 98.

Shatkay H. and Feldman R. (2003). Mining the Biomedical Literature in the Genomic Era: An Overview. *Journal of Computational Biology*, 10 (6): 821-855.

Shatkay, H., Edwards, S. and Boguski, M. (2002). Information Retrieval Meets Gene Analysis. *IEEE Intelligent Systems*, 17(2): 45-53.

Shatkay, H., Edwards, S., Wilbur,W. J. & Boguski, M. (2000). Genes, Themes and Microarrays: Using Information Retrieval for Large-scale Gene Analysis. *Proc Int Conf Intelligent System Molecular Biology*, 8,317-28.

Skounakis, M. and Craven, M. (2003). Evidence Combination in Biomedical Natural-language Processing, in *Proceedings of the 3rd Workshop on Data Mining in Bioinformatics,* 25-32.

Skounakis, M., Craven, M. and Ray, S. (2003). Hierarchical Hidden Markov Models for Information Extraction. *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico. Morgan Kaufmann.

Smalheiser, N.R. and Swanson, D.R. (1998). Using ARROWSMITH: a Computer-assisted Approach to Formulating and Assessing Scientific Hypotheses, *Computer Methods and Programs in Biomedicine*. 57 (3): 149-153 NOV.

Soderland, S., Fisher, D; Aseltine, J.; and Lehnert, W., (1995). CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1314-1319.

Soderland, S. (1999) .Learning Information Extraction Rules for Semi-structured and Free Text, *Machine learning*, 34, 233-272.

Song, M., Song, I.-Y., and Hu, X. (2004). Designing and Developing an Automatic Interactive Keyphrase Extraction System with UML, accepted to be presented at *ASIST2004 Annual Meeting*, Providence, RI, 367-372.

Song, M., Song, I-Y, and Hu, X.(2003). KPSpotter: A Flexible Information Gain-based Keyphrase Extraction System, in *Proceedings of the ACM CIKM 5th Workshop on Web Information and Data Management (WIDM 03),* New Orleans, LA, Nov. 3-8, 2003, pp50-53

Song, M., Song, I-Y, and Chen, P (2004). Design of a Cross Search Engine Using UML and Design Pattern, *Information Systems Frontiers*, 6(1): 77-90.

Sparck-Jones, K (1972). Statistical Interpretation of Term Specificity and Its Application In Retrieval. *Journal of Documentation*. 28. 1. pp 11 - 20. 1972

Spink, A. (1994). Term Relevance Feedback and Query Expansion; Relation to Design, in *Proceedings of 17th International Conference on Research and Development in Information Retrieval (SIGIR94),* 81-90.

Srinivasn, P. (1996). Optimal Document Indexing Vocabulary for MEDLINE. *Information Processing and Management*, 32(5): 503-514.

Srinicasan, P. (1996). Query Expansion and MEDLINE. *Information Processing and Management*, 32(4): 431-443.

Stapley, B. J. and Benoit, G. (2000). Biobibliometrics: Information Retrieval and Visualization from Co-occurrences of Gene Names in MEDLINE Abstracts. *Pacific Symposium on Biocomputing*, 529-40.

Stephens, M., Palakal, M., ukhopadhyay, S., Raje, R. & Mostafa, J. (2001). Detecting Gene Relations from MEDLINE Abstracts. *Pacific Symposium Biocomputing*, 483-95.

Stevens, R., Goble, C. A. &Bechhofer, S. (2000). Ontology-based Knowledge Representation for Bioinformatics. *Brief Bioinformatics*, 1, 398-414.

Sutton, R. (1996). Reinforcement Learning and Information Access, *AAAI Spring Symposium on Machine Learning and Information Access*.

Tai, X.Y., Ren, F., and Kita, K. (2002). Information Retrieval Model based on Vector Space Method by Supervised Learning, *Information Processing & Management*, 38 (6): 749-764 NOV.

Tanabe, L., Scherf, U., Smith, L.H., Lee, J.K., Hunter, L., and Weinstein, J.N. (1999). MedMiner: an Internet Text-Mining Tool for Biomedical Information, with Application to Gene Expression Profiling. *Biotechniques*, 27(6), 1210-4, 1216-7.

Thomas, J., Milward, D., Ouzounis,C., Pulman, S. and Carroll, M. (2000). Automatic Extraction of Protein Interactions from Scientific Abstracts. *Pacific Symposium on Biocomputing*, 541-52.

Thompson, C.A., Califf, M.E. and Mooney, R.J. (1999). Active Learning for Natural Language Parsing and Information Extraction, in *Proceedings of 16th International Machine Learning Conference*, 406-414.

Toutanova, K., Klein, D., Manning, C.D., and Singer, Y. (2003) Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network. In: *Proceedings of HLT-NAACL*, 252-259.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworth 1979

Van Der Pol, R. (2003). Dipe-D: a Tool For Knowledge-based Query Formulation, *Information Retrieval*, 6, 21-47.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag, NY, USA, 1995.

Voorhees, E.M. (1998). Using WordNet for Text Retrieval. In *WordNet, an Electronic Lexical Database*, C. Fellbaum (ed.), MIT Press, Cambridge MA, 285-303

Weigend, A.S., Weiner, E. D., and Peterson, J. O. (1999). Exploiting Hierarchy on Text Categorization, *Information Retrieval*, 193-216.

Weizenbaum, J. (1966). ELIZA – A Computer Program for the Study of Natural Language Communications between Men and Machine, *Communications of the Association for Computing Machinery*, 9, pp. 36–45.

Witten I.H., Paynter G.W., Frank E., Gutwin C. and Nevill-Manning C.G. (1999). KEA: Practical Automatic Keyphrase Extraction. In: *Proc. DL '99*, pp. 254-256.

WordNet 2.0 http://wordnet.princeton.edu/doc.shtml.

Xenarios, I and Eisenberg, D. (2001). Protein Interaction Databases, *Current Opinion in Biotechnology*,12(4), 334-339.

Xu, J. and Croft, W.B. (1996). Query Expansion Using Local and Global Document Analysis. In *Proceedings of the 19th annual international Conference on Research and Development in Information Retrieval*, 4-11.

Yang, J. and Korfhage, R.R (1993). Query Optimization in Information Retrieval Using Genetic Algorithms, *Proceedings of the 5th International Conference on Genetic Algorithms*, Urbana, IL, 603-611.

Yang, Y. (1996). An Evaluation of Statistical Approaches to MEDLINE Indexing. *Proc AMIA Annual Fall Symposium*,358-362.

Yakushiji, A., Tateisi, Y., Miyao,Y. and Tsujii, J. (2001). Event Extraction from Biomedical Papers using a Full Parser. *Pacific Symposium on Biocomputing*, 408-19.

Yamamoto, K., Kudo, T., Konagaya, A. and Matsumoto, Y. (2003). Protein Name Tagging for Biomedical Annotation in Text. *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, 65-72.

Yangarber, R., and Grishman, R. (1998). NYU: Description of the Proteus/PET System as used for MUC-7. In *Proceedings of the seventh Message Understanding Conference (MUC-7).* Morgan Kaufman, 1998

Yangarber, R., Grishman, R., Tapanainen, P., and Huttunen, S. (2000). Unsupervised Discovery of Scenario-Level Patterns for Information Extraction, in *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP-NAACL 2000)*, 282-289.

Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pp 189-196, 1995

Yeh, A.S., Hirschman, L., and Morgan, A.A. (2003). Evaluation of Text Data Mining for Database Curation: Lessons Learned from the KDD Challenge CUP, *Bioinformatics*, 19, 1331-1339.

Yi, J. and Neel, S. (1999). Mining the Web For Acronyms Using The Duality Of Patterns And Relations. In *Proceedings of the 1999 Workshop on Web Information and Data Management*, 1999

Zelenko, D., Aone, C., and Richardella, A. (2003). Kernel Methods for Relation Extraction, *Journal of Machine Learning Research*, 3: 1083-1106.

APPENDIX A: CONFIGURATION INFORMATION OF RIKE

```
<kps_config>
  <name>kpspotter</name>
  <field>ok</field>
  <limit>500</limit>
  <builder>medline</builder>
  <stemmer>porter</stemmer>
  <parser mode="SAX">trec</parser>
  <synonym>WordNet</synonym>
  <base_db_dir>./DB</base_db_dir>
  <base_dir>/home/msong/kmroot/src/project/thesis</base_dir>
  <keyphrase_dir>/home/msong/kmroot/src/project/thesis/phrase_
    dir</keyphrase_dir>
  <query_file
    dir="/home/msong/kmroot/src/project/thesis/query/init"
    init_search="true" online="true" ontology="true" target="trec"
    ontology_method="umls"
    ripper="true">query_set24.xml</query_file>
  <extract_engine>qe_test</extract_engine>
  <query_cut_line>10</query_cut_line>
  <pubmed_dir deleteDB="true" runQE="true" iterative_search="false"
    dir="/home/msong/kmroot/src/project/thesis/stored_file/init">
    /home/msong/kmroot/src/project/thesis/stored_file/init/trec_i
    nit_set.xml</pubmed_dir>
  <category_balance balanced="true" precision_oriented="true" />
  <model
    name="model.xml">/home/msong/kmroot/src/project/thesis</
    model>
  <ripper init_search="false"
    file="ITER_3.xml">/home/msong/kmroot/src/project/thesis/rip
    per</ripper>
  <xrpc port="10101" server="129.25.64.17">/esti/xrpc</xrpc>
```

- <builders>
<name train="**/home/msong/kmroot/src/project/data/tmp**"
test="**/home/msong/kmroot/src/project/data/tmp_test**"
model="**/home/msong/kmroot/src/project/key_phrase_extractio**
n/models/medline_model.xml">**medline**</name>
<name train="**./ieee_training**" test="**./ieee_testing**">**ieee**</name>
<name train="**./ohsumed_database/test_data**"
test="**./ohsumed_database/train_data**">**oshumed**</name>
<name train="**/home/msong/kea/KEA-2.0/CSTR_abstracts_train**"
test="**/home/msong/kea/KEA-**
2.0/CSTR_abstracts_test">**reuters**</name>
<name train="" test="">**http_html**</name>
<name train="" test="">**pubmed_xml**</name>
<name train="" test="">**trec**</name>
</builders>
- <extraction_element>
- <pubmed_xml>
<element name="**root**">**PubmedArticleSet**</element>
<element name="**doc_unit**">**PubmedArticle**</element>
<element
name="**title**">**MedlineCitation/Article/ArticleTitle**</element>
<element
name="**abstract**">**MedlineCitation/Article/Abstract/AbstractText**<
/element>
</pubmed_xml>
- <trec_xml>
<element name="**root**">**RESPONSE**</element>
<element name="**doc_unit**">**DOC**</element>
<element name="**title**">**TTL**</element>
<element name="**abstract**">**TEXT**</element>
<element name="**context**" />
<element name="**doc_no**">**DOCNO**</element>
</trec_xml>
</extraction_element>
- <collection>
- <type name="**pubmed**">

```
<part
  name="url">http://www.ncbi.nlm.nih.gov:80/entrez/eutils/esear
  ch.fcgi?db=PubMed&retmode=xml&usehistory=y</part>
<part name="retmax">500</part>
<part name="retstart">retstart=</part>
<part name="term">term=zanzibar</part>
<part name="start">0</part>
  </type>
- <type name="zettair">
<part name="index">/storage/TREC/doc/disk_pubmed</part>
<part name="max">30</part>
<part name="start">0</part>
<part name="retstart">500</part>
  </type>
- <type name="lemur">
<part name="host">129.25.64.170</part>
<part name="port">12345</part>
<part
  name="index">/storage/TREC/doc/lemur_index/trec_index.ifp</
  part>
<part name="max">30</part>
<part name="start">0</part>
<part name="retstart">1000</part>
  </type>
  </collection>
- <postype>
<type category="1" name="noun" pos="N" />
<type category="1" name="noun" pos="N N" />
<type category="1" name="noun" pos="N N N" />
<type category="1" name="noun" pos="NN" />
<type category="1" name="noun" pos="NN NN" />
<type category="1" name="noun" pos="NN NN NN" />
<type category="1" name="noun" pos="NN NN NNS" />
<type category="1" name="noun" pos="NN NNS NNS" />
<type category="1" name="noun" pos="NN NNS NN" />
<type category="1" name="noun" pos="NNS NNS NN" />
```

<type category="**1**" name="**noun**" pos="**NNS**" />
<type category="**2**" name="**noun**" pos="**NNS NN**" />
<type category="**2**" name="**noun**" pos="**NNS NNS**" />
<type category="**2**" name="**noun**" pos="**NNS NNS NNS**" />
<type category="**2**" name="**noun**" pos="**NNP NNS**" />
<type category="**2**" name="**noun**" pos="**NNS NNP**" />
<type category="**2**" name="**noun**" pos="**NN NNP**" />
<type category="**2**" name="**noun**" pos="**NNP**" />
<type category="**2**" name="**noun**" pos="**NNP NN**" />
<type category="**2**" name="**noun**" pos="**NNP NNP**" />
<type category="**2**" name="**noun**" pos="**NNP NN NNP**" />
<type category="**2**" name="**noun**" pos="**NNP NNP NNP**" />
<type category="**2**" name="**noun**" pos="**NN NNP NNS**" />
<type category="**2**" name="**noun**" pos="**NNPS**" />
<type category="**2**" name="**noun**" pos="**NNPS NNPS**" />
<type category="**2**" name="**noun**" pos="**NNPS NNPS NNPS**" />
<type category="**1**" name="**mix_noun**" pos="**A N**" />
<type category="**1**" name="**mix_noun**" pos="**A N N**" />
<type category="**1**" name="**mix_noun**" pos="**A A N**" />
<type category="**1**" name="**mix_noun**" pos="**A NN**" />
<type category="**1**" name="**mix_noun**" pos="**A NN NN**" />
<type category="**1**" name="**mix_noun**" pos="**A A NN**" />
<type category="**1**" name="**mix_noun**" pos="**A NN**" />
<type category="**1**" name="**mix_noun**" pos="**A NN NN**" />
<type category="**1**" name="**mix_noun**" pos="**A A NN**" />
<type category="**1**" name="**mix_noun**" pos="**AJ N**" />
<type category="**1**" name="**mix_noun**" pos="**AJ NJ N**" />
<type category="**1**" name="**mix_noun**" pos="**AJ AJ N**" />
<type category="**1**" name="**mix_noun**" pos="**AJ NN**" />
<type category="**1**" name="**mix_noun**" pos="**AJ NN NN**" />
<type category="**1**" name="**mix_noun**" pos="**AJ AJ NN**" />
<type category="**1**" name="**mix_noun**" pos="**JJ NN**" />
<type category="**1**" name="**mix_noun**" pos="**JJ NN NN**" />
<type category="**1**" name="**mix_noun**" pos="**JJ JJ NN**" />
<type category="**2**" name="**mix_noun**" pos="**AJ NNS**" />
<type category="**2**" name="**mix_noun**" pos="**AJ NNS NNS**" />

```
<type category="2" name="mix_noun" pos="AJ AJ NNS" />
<type category="2" name="mix_noun" pos="AJ NNP" />
<type category="2" name="mix_noun" pos="AJ NNP NNP" />
<type category="2" name="mix_noun" pos="AJ AJ NNP" />
<type category="2" name="mix_noun" pos="JJ NNS" />
<type category="2" name="mix_noun" pos="JJ NNS NNS" />
<type category="2" name="mix_noun" pos="JJ JJ NNS" />
<type category="2" name="mix_noun" pos="JJ NNP" />
<type category="2" name="mix_noun" pos="JJ NNP NNP" />
<type category="2" name="mix_noun" pos="JJ JJ NNP" />
<type category="3" spec_name="verb" pos="VB" />
<type category="3" spec_name="verb" pos="VN" />
<type category="3" spec_name="verb" pos="VBD" />
<type category="3" spec_name="verb" pos="VBG" />
<type category="3" spec_name="verb" pos="VBN" />
<type category="3" spec_name="verb" pos="VBZ" />
<type category="3" spec_name="verb" pos="VBP" />
  </postype>
<stopword stop_path="/home/msong/kmroot/src/project/thesis"
  stop_name="stopword_list.txt" />
- <mbt4>
<setting>/home/msong/kmroot/src/thirdparty/Mbt4/medline.data.
  settings</setting>
<info name="FilterTreshold" value="1" />
<info name="dumpflag" value="true" />
<info name="EosMark" value="." />
<info name="K_option_name" value="test" />
<info name="knownoutfileflag" value="true" />
<info name="KnownTreeName" value="tree" />
<info name="knowntreeflag" value="true" />
<info name="l_option_name" value="lex" />
<info name="lexflag" value="true" />
<info name="L_option_name" value="tt" />
<info name="klistflag" value="true" />
<info name="TopNumber" value="10" />
<info name="Npax" value="2" />
```

```
<info name="TimblOptStr" value="opt" />
<info name="KtmplStr" value="ko" />
<info name="UtmplStr" value="uo" />
<info name="r_option_name" value="r_name" />
<info name="SettingsFileName" value="set" />
<info name="TestFileName"
  value="/home/msong/kmroot/src/thirdparty/Mbt4/medline.tok"
  />
<info name="UnknownTreeName" value="u_t" />
<info name="U_option_name" value="u_op" />
<info name="unknownoutfileflag" value="true" />
<info name="KeepIntermediateFiles" value="true" />
  </mbt4>
- <linkgrammar>
<data_path>/home/msong/kmroot/src/thirdparty/link-
  grammar/link-4.1/data/</data_path>
<dict_name>4.0.dict</dict_name>
<pp_name>4.0.knowledge</pp_name>
<cons_name>4.0.constituent-knowledge</cons_name>
<affix_name>4.0.affix</affix_name>
  </linkgrammar>
- <pckimmo>
<rule>/home/msong/kmroot/src/thirdparty/pc-parse-
  20030321/pckimmo/test/eng/english.rul</rule>
<lexicon>/home/msong/kmroot/src/thirdparty/pc-parse-
  20030321/pckimmo/test/eng/english.lex</lexicon>
<grammar>/home/msong/kmroot/src/thirdparty/pc-parse-
  20030321/pckimmo/test/eng/english.grm</grammar>
<conf_file>/home/msong/kmroot/src/thirdparty/pc-parse-
  20030321/ktagger/test/eng/engsfm.ctl</conf_file>
- <fields>
<field path_name="WORD" starttag="\\w" endtag="\n" />
<field path_name="LEX" starttag="\\lx" endtag="\n" />
<field path_name="head pos" starttag="\\pos" endtag="\n" />
<field path_name="root" starttag="\\lemma" endtag="\n" />
<field path_name="root_pos" starttag="\\lempos" endtag="\n" />
```

```
    </fields>
    </pckimmo>
- <nlparser>
  <data_dir>/home/msong/kmroot/src/thirdparty/max_ent_ins/DAT
    A/</data_dir>
  <d>0</d>
  <l>75</l>
  <T>200</T>
    </nlparser>
- <brill>
  <LEXICON>/home/msong/kmroot/src/thirdparty/ePost/post/Bin_
    and_Data/LEXICON</LEXICON>
  <BIGRAMS>/home/msong/kmroot/src/thirdparty/ePost/post/Bin_
    and_Data/BIGRAMS</BIGRAMS>
  <LRULEFILE>/home/msong/kmroot/src/thirdparty/ePost/post/Bin
    _and_Data/LEXICALRULEFILE</LRULEFILE>
  <CRULEFILE>/home/msong/kmroot/src/thirdparty/ePost/post/Bin
    _and_Data/CONTEXTUALRULEFILE</CRULEFILE>
    </brill>
  <log_level>1</log_level>
  <log_file>_log.out</log_file>
  </kps_config>
```

APPENDIX B: 25 QUERIES FOR MEDLINE SEARCH

```
<query>

      <terms term1="MAP4" term2="Mapmodulin" />

      <terms term1="WIP" term2="NCK" />

      <terms term1="GHR" term2="SHB" />

      <terms term1="SHIP" term2="DOK" />

      <terms term1="LNK" term2="GRB2" />

      <terms term1="CRP" term2="Zyxin" />

</query>
```

Query 1

```
<query>
      <terms term1="DLG4" term2="CNK2" />
      <terms term1="RhoGAP" term2="Siah" />
      <terms term1="Syntrophin" term2="nNOS" />
      <terms term1="MDM2" term2="E2F-1" />
      <terms term1="H-Ras" term2="SOS-1" />
</query>
```

Query 2

```
<query>
      <terms term1="PKA-R2b" term2="PRKA9" />
      <terms term1="PI3-K" term2="PTP1D" />
      <terms term1="C8a" term2="C8b" />
```

```
    <terms term1="WASF1" term2="ARPC1B" />
    <terms term1="ApoER2" term2="DAB1" />
</query>
```

Query 3

```
<query>
    <terms term1="GRB2" term2="WASL" />
    <terms term1="ETA" term2="GNA11" />
    <terms term1="MTAP2" term2="Src" />
    <terms term1="PKA-R2a" term2="Ezrin" />
    <terms term1="cdc42" term2="MLK-3" />
    <terms term1="BNIP2" term2="Bcl-2" />
    <terms term1="LNK" term2="p56lck" />
    <terms term1="CLTC" term2="SYNJ1" />
</query>
```

Query 4

```
<query>
    <terms term1="Dvl" term2="CK2" />
    <terms term1="Hrs-2" term2="EPS15" />
</query>
```

Query 5

```
<query>
    <terms term1="CCC-R4" term2="MDC" />
    <terms term1="Galectin-1" term2="PTPRC" />
    <terms term1="CD90" term2="Galectin-1" />
    <terms term1="Endophilin-3" term2="Huntingtin" />
</query>
```

Query 6

```
<query>
    <terms term1="SHIP" term2="DOK" />
    <terms term1="RIL" term2="ZRP-1" />
```

```
    <terms term1="Mint-2" term2="LRP" />
    <terms term1="LRP" term2="Synectin" />
    <terms term1="Jip-1" term2="ApoER2" />
    <terms term1="Erk2" term2="PDE4D3" />
    <terms term1="Amphoterin" term2="RAGE" />
    <terms term1="LATS-1" term2="Zyxin" />
    <terms term1="KCIP-1" term2="MAPT" />
</query>
```

Query 7

```
<query>
    <terms term1="TNFR1" term2="STAT-1" />
    <terms term1="STAT-1" term2="TRADD" />
    <terms term1="PCNA" term2="POLD3" />
    <terms term1="Rsk-2" term2="PDK-1" />
    <terms term1="GRK-2" term2="PRKA12" />
</query>
```

Query 8

```
<query>
    <terms term1="CLTC" term2="PRKA12" />
    <terms term1="GL" term2="PYGL" />
    <terms term1="CyclinA1" term2="cdc25C" />
    <terms term1="cdc42" term2="Frabin" />
</query>
```

Query 9

```
<query>
    <terms term1="ARHGAP5" term2="cdc42" />
</query>
```

Query 10

```
<query>
    <terms term1="IKKAP" term2="RIP2" />
    <terms term1="PKA-R1a-a" term2="Myosin" />
```

```
    <terms term1="SOS-1" term2="SHC" />
    <terms term1="Integrin-b3" term2="CD47" />
</query>
```

Query 11

```
<query>
    <terms term1="Crk" term2="SYN1" />
    <terms term1="SYN1" term2="Phox47" />
    <terms term1="MDM2" term2="MTBP" />
    <terms term1="Dvl" term2="Frizzled-R" />
    <terms term1="TRADD" term2="TRAF2" />
    <terms term1="Glc2b" term2="Glc2a" />
    <terms term1="PLD-1" term2="Pea-15" />
</query>
```

Query 12

```
<query>
    <terms term1="STX" term2="SYT1" />
    <terms term1="RAN" term2="NTF-2" />
    <terms term1="PP2B" term2="Cain" />
    <terms term1="KCIP-1" term2="B-Raf" />
</query>
```

Query 13

```
<query>
    <terms term1="ERBB4" term2="BTC" />
    <terms term1="PLCg-1" term2="SOS-1" />
    <terms term1="AP180" term2="AP2B1" />
    <terms term1="VAMP" term2="CALM" />
    <terms term1="Lyn" term2="EVL" />
    <terms term1="DSG" term2="Desmoplakin" />
    <terms term1="GluR2" term2="GluR2" />
    <terms term1="HDAC" term2="KCIP-1" />
```

```
</query>
```

Query 14

```
<query>
    <terms term1="c-Cbl" term2="HGF-SF" />
    <terms term1="MUPP1" term2="NG2" />
    <terms term1="CDC2" term2="GADD45A" />
    <terms term1="EBP50" term2="PLCb" />
</query>
```

Query 15

```
<query>
    <terms term1="Neuropilin1" term2="L1CAM" />
    <terms term1="PKA-R2a" term2="PRKA6" />
    <terms term1="RAF1" term2="KCIP1" />
    <terms term1="RALT" term2="ErbB-2" />
    <terms term1="SOS-1" term2="Abi" />
    <terms term1="HOMER1" term2="mGluR5" />
</query>
```

Query 16

```
<query>
    <terms term1="SDC2" term2="Synbindin" />
    <terms term1="PCNA" term2="GADD45G" />
    <terms term1="CIP1" term2="GADD45G" />
    <terms term1="JAK-2" term2="STAT-1" />
    <terms term1="Src" term2="Connexin-43" />
</query>
```

Query 17

```
<query>
    <terms term1="C8g" term2="C8a" />
    <terms term1="PLD-1" term2="PKCalpha" />
</query>
```

Query 18

```
<query>
	<terms term1="ACT" term2="Presenilin" />
	<terms term1="GRK-2" term2="CALM" />
	<terms term1="GABA" term2="AP2A1" />
</query>
```

Query 19

```
<query>
	<terms term1="ARF1" term2="PSCD1" />
	<terms term1="MAT-1" term2="MCM7" />
	<terms term1="p190RhoGEF" term2="Tubulin" />
	<terms term1="DyneinLC" term2="NRF-1" />
</query>
```

Query 20

```
<query>
	<terms term1="SOCS-3" term2="IGF-1-R" />
	<terms term1="RagA" term2="RagC" />
	<terms term1="PPP1CA" term2="I-4" />
	<terms term1="c-KIT" term2="MITF" />
	<terms term1="c-Abl" term2="Crk" />
	<terms term1="Tiam1" term2="CD44" />
</query>
```

Query 21

```
<query>
	<terms term1="IQGAP-1" term2="MLC1" />
	<terms term1="p114-RhoGEF" term2="Rho" />
	<terms term1="FHIT" term2="Ubc9" />
	<terms term1="Capon" term2="Dexras" />
	<terms term1="Axin" term2="GSK3B" />
	<terms term1="JNKK1" term2="b-arrestin" />
	<terms term1="PPP1CA" term2="Cofilin" />
	<terms term1="cdc42" term2="Pak" />
```

```
    <terms term1="AP1B1" term2="KIF13A" />
</query>
```

Query 22

```
<query>
    <terms term1="PKA-R1a-a" term2="PRKACA" />
    <terms term1="HP1A" term2="Ku-70" />
    <terms term1="FATZ" term2="PP2B" />
</query>
```

Query 23

```
<query>
    <terms term1="JAM" term2="CASK" />
    <terms term1="Phox47" term2="PKCbeta" />
    <terms term1="PICK1" term2="mGluR7" />
    <terms term1="cdc42" term2="WAS" />
    <terms term1="DLG1" term2="Sema4C" />
    <terms term1="G-proteins" term2="N-cadherin" />
    <terms term1="DLG1" term2="Stargazin" />
    <terms term1="DLG2" term2="Stargazin" />
    <terms term1="Kid" term2="Siah" />
    <terms term1="Trio" term2="Filamin" />
</query>
```

Query 24

```
<query>
    <terms term1="BS69" term2="DyneinLC" />
</query>
```

Query 25

VITA

RESEARCH INTERESTS

Text Mining, Information Retrieval, Information Visualization, Knowledge Management, Data Mining, and Digital Libraries

EDUCATION

Fall 2000 – Present	PhD Student, School of Information Science & Technology, Drexel University, Philadelphia, PA (expected to graduate in Spring 2005)
Fall 1993 – Spring 1995	M.A., School of Library and Information Science, Indiana University, Bloomington, Indiana
Fall 1988 – Spring 1992	B.A., School of Library and Information Science, Yonsei University, Seoul, Korea

HONORS AND AWARDS

2004 Dean's Award of Research Day, Drexel University
2003 Dean's Award of Research Day, Drexel University
2001 Current Kudos, Institute for Scientific Information
1999 Current Kudos, Institute for Scientific Information
1998 Clayton A. Shepherd Scholarship, Indiana University
1998 Summer H. W. Wilson Fellowship, Indiana University
1996 Fall-1997 Spring Fellowships of SLIS, Indiana University

TEACHING EXPERIENCE

Summer 2005	Instructor of Object-oriented Programming for Information Systems, Drexel University
Summer 2002	Co-instructor of Workshop on Web Services: Concepts and Techniques, Drexel University
Spring 1998	Lab instructor of Computer-based Information Tools (L401), Indiana University
Fall 1997	Co-instructor of User Needs: Theories and Practices (L503), Indiana University