Ph.D. Proposal, Department of the Computer Sciences,
University of Texas at Austin, February 2001

Text Mining with Information Extraction: Mining Prediction Rules from Unstructured Text

Un Yong Nahm
Department of Computer Sciences
University of Texas
2.124 Taylor Hall
Austin, TX 78712
pebronia@cs.utexas.edu

Supervising Professor: Dr. Raymond J. Mooney

January 25, 2001

Abstract

Text mining is a relatively new research area at the intersection of data mining, natural-language processing, machine learning, and information retrieval. The goal of text mining is to discover knowledge in unstructured text. The related task of *Information Extraction* (IE) concerns locating specific items of data in natural-language documents, thereby transforming unstructured text into a structured database. Although handmade IE systems have existed for a while, automatic construction of information extraction systems using machine learning is more recent. This proposal presents a new framework for text mining, called DiscoTEX (Discovery from Text EXtraction), which uses a learned information extraction system to transform text into more structured data which is then mined for interesting relationships.

DiscoTEX combines IE and standard data mining methods to perform text mining as well as improve the performance of the underlying IE system. It discovers prediction rules from natural-language corpora, and these rules are used to predict additional information to extract from future documents, thereby improving the recall of IE. The initial version of DiscoTex integrates an IE module acquired by the Rapier learning system, and a standard rule induction module such as C4.5rules or Ripper. Encouraging initial results are presented on applying these techniques to a corpus of computer job announcements posted on an Internet newsgroup. However, this approach has problems when the same extracted entity or feature is represented by similar but not identical strings in different documents. Consequently, we are also developing an alternate rule induction system for DiscoTex called, TextRISE, that allows for partial matching of string-valued features. We also present initial results applying the TextRISE rule learner to corpora of book descriptions and patent documents retrieved from the World Wide Web (WWW). Future research will involve thorough testing on several domains, further development of this approach, and extensions of the proposed framework (currently limited to prediction rule discovery) to additional text mining tasks.

1 Introduction

The problem of *text mining*, i.e. discovering useful knowledge from unstructured text, is attracting increasing attention (Hearst, 1999; Feldman, 1999; Mladenić, 2000). This proposal suggests a new framework for text mining based on the integration of *Information Extraction* (IE) and traditional Knowledge Discovery from Databases (KDD), a.k.a. *data mining*. Information Extraction is a form of shallow text understanding that locates specific pieces of data from a corpora of natural-language texts. KDD considers the application of statistical and machine-learning methods to discover novel relationships in large relational databases. However, there has been little if any research exploring the interaction between these two important techniques to perform text mining tasks.

Traditional data mining assumes that the information to be "mined" is already in the form of a relational database. Unfortunately, for many applications, electronic information is only available in the form of unstructured natural-language documents rather than structured databases. Information Extraction, a task that has been a focus since the start of the Message Understanding Conferences (MUCs) (DARPA, 1998), addresses the problem of transforming a corpus of textual documents into a more structured database. This suggests an obvious role that IE can play in text mining when combined with standard KDD methods, as illustrated in Figure 1. In this IE-based text mining framework, called DiscoTEX (Discovery from Text EXtraction), the IE module captures specific pieces of data in raw text, and the resulting database is provided to the KDD module for further mining of knowledge.

Although constructing an IE system is a difficult task, there has been significant recent progress in using machine learning methods to help automate the construction of IE systems (Cardie, 1997; Califf, 1999). By manually annotating a small number of documents with the information to be extracted, a fairly accurate IE system can be induced from this labeled corpus and then applied to a large body of raw text to construct a large database for mining. In this way, a small amount of labeled training data for an IE learning system can be automatically transformed into a large database of structured information ready to be mined with traditional KDD methods. For example, the IE learning system Rapier (Califf & Mooney, 1999) has been used to induce an IE system that transforms newsgroup job postings into a relational database. By applying standard rule induction methods to a database of 5,000 jobs automatically extracted from the newsgroup `austin.jobs`, we have discovered interesting relationships such as "If a computer-related job requires knowledge of `Java` and `graphics` then it also requires knowledge of `PhotoShop`." or "Computer jobs related to `e-commerce` and `database` usually require knowledge of `SQL Server` and `Oracle`."

However, the accuracy of current IE systems, whether built manually or induced from labeled data, is limited despite the recent advancement of information extraction. Therefore, an automatically extracted database will inevitably contain significant numbers of errors. An important question is whether the knowledge discovered from this "noisy" database is significantly less reliable than knowledge discovered from a cleaner traditional database. In this proposal we present experiments showing that knowledge discovered from an automatically extracted database is close in accuracy to that discovered from a manually constructed database, demonstrating that

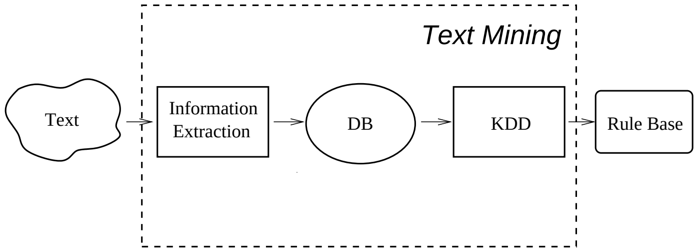


Figure 1: Overview of IE-based text mining framework

combining IE and KDD is a viable approach to text mining.

A less obvious interaction is the benefit that KDD can in turn provide to IE. This proposal also explores the mutual benefit that the integration of IE and KDD can provide, by showing that rules mined by KDD can be used to improve IE performance. The predictive relationships between different slot fillers discovered by KDD can provide additional clues about what information should be extracted from a document. For example, if `Java` ∈ `programming-languages` and the `graphics` ∈ `areas` have been extracted from a job posting, then an IE system might also consider extracting `PhotoShop` ∈ `applications` as an additional slot filler. Since typically the *recall* (percentage of correct slot fillers extracted) of an IE system is significantly lower than its *precision* (percentage of extracted slot fillers which are correct) (DARPA, 1993, 1995, 1998), such predictive relationships might be productively used to improve recall by suggesting additional information to extract. In this proposal, we report experiments in the computer-related job-posting domain demonstrating that predictive rules acquired by applying KDD to an extracted database can be used to improve the recall of information extraction.

One of the assumptions behind the text-mining system described so far is that standard data mining methodologies are suitable for the rule induction component of our "IE + KDD = Text Mining" framework. However, text strings in traditional databases often contain typos, misspellings, and non-standardized variations. This is another important aspect that traditional data mining techniques have not been adequately addressed. The heterogeneity of textual databases causes a problem when we apply existing data mining techniques to text: the same or similar objects are often referred to using different (but similar) textual strings. This issue becomes clear when we consider the World Wide Web (WWW), a vast and dynamic warehouse of text documents, as a potential target for text mining. Since the Web has no centralized moderator, it is highly heterogeneous, making it difficult to apply strict operators on text extracted from web. In order to cope with this problem, we propose a method, TEXTRISE for learning soft-matching rules from text using a modification of the RISE algorithm (Domingos, 1996), a hybrid of rule-based and instance-based (nearest-neighbor) learning methods. Such a hybrid is good match for text mining since rule induction provides simple, interpretable rules, while nearest-neighbor provides soft matching based on a specified similarity metric.

In this proposal, we present preliminary experimental results with an initial version of DISCOTEX and TEXTRISE on a corpus of Internet documents such as computer-related job postings on the newsgroup, book descriptions collected from an online bookstore, and patent documents available on the Web. We believe encouraging results from the experiments indicate that the usefulness of information extraction in text mining. We plan to do full evaluations with our system, to enhance it several directions, and to expand it for other multiple tasks of text mining.

The remainder of this proposal is organized as follows. Section 2 presents background material on text data mining and information extraction. Section 3 describes an implementation of the DiscoTEX framework, that simply combines IE and traditional KDD technologies to discover prediction rules from text. We will also show that DiscoTEX improves performance of IE by exploiting prediction rules. Experimental results obtained on a corpus of job postings from the newsgroup `austin.jobs` are presented and discussed. Section 4 presents and analyzes preliminary results obtained with TextRISE, another implementation of the proposed text mining framework with partial matching rules incorporated in the prediction rule-learning algorithm. Experiments with TextRISE are performed on a corpus of book descriptions from `Amazon.com` and publicly available patent documents. In Section 5, we outline our future research plan, including extensive testing in multiple domains, several enhancements of the current system, and plans for applying the proposed framework to other text mining tasks. Section 6 discusses related work and the final section summarizes and presents our conclusions.

2 Background

This section provides background information on our research by reviewing the goal of text mining and various techniques used for building text-mining systems. The task of Information Extraction (IE) is also discussed in Section 2.2 along with a brief overview of state-of-the-art IE systems.

2.1 Text Mining

2.1.1 What is Text Mining?

Text mining is defined as "the process of finding useful or interesting patterns, models, directions, trends, or rules from unstructured text". Text mining has been viewed as a natural extension of data mining (Hearst, 1999), or sometimes considered as a task of applying the same data mining techniques to the domain of textual information (Dörre, Gerstl, & Seiffert, 1999). This reflects the fact that the advent of text mining relies on the burgeoning field of data mining in a great degree. Traditionally, texts have been mainly analyzed by natural language processing techniques or Information Retrieval (IR) methods. The popularity of the Web and the huge amount of text documents available in electronic media also boosted "the search for the hidden knowledge in collections of text documents".

One of the important goals of text mining is to extract patterns that can be incorporated in other intelligent applications. Such applications include categorization, routing, filtering, segmentation, retrieval, ranking, summarization, clustering, organization and navigation tools for text documents. Many of these tasks often overlap each other. Text categorization (Yang, 1999), or text classification, is the most extensively explored field for applying text-mining techniques because many of the other applications can be cast into the task of text categorization. Text categorization systems conduct a supervised learning task of assigning predefined categories to new text documents based on a classification function learned from a set of labeled documents.

Besides these traditional applications, discovering knowledge from unstructured text is an exiting new area for text mining. For example, Knowledge Discovery in Textual databases (KDT) (Feldman & Dagan, 1995) discovers interesting patterns from text, by establishing a hierarchy of meaningful concepts and looking for mutual connections in between the concept nodes. KDT has evolved into the FACT system (Feldman & Hirsh, 1996) with an aid of a well-known data mining technique, *association rule mining*, and DOCUMENT EXPLORER (Feldman, Fresko, Hirsh, Aumann, Liphstat, Schler, & Rajman, 1998) accompanied with an interactive exploration tool. These approaches are applied to Reuters news articles to find out interesting relationships between concept items, e.g. natural resources of Latin American countries.

Web mining is a specialized extension for text mining to the semi-structured texts available on the Internet (Etzioni, 1996). Finding relevant information, e.g. web search engines, has been the main focus of web mining for years, but recent research topics on web mining are diversified to using the Web as a knowledge base in computer-understandable format for further tasks (Cohen, 1999; Craven, DiPasquo, Freitag, McCallum, Mitchell, Nigam, & Slattery, 2000), personalization of the Web based on interests of individual users (Pazzani, Muramatsu, & Billsus, 1996; Joachims, Freitag, & Mitchell, 1997), locating specific information from hypertext documents (Brin, 1998; Soderland, 1999) or summarization/compression of web documents (Berger & Mittal, 2000). "Web mining" is generally categorized to "web content mining", "web structure mining", and "web usage mining" (Kosala & Blockeel, 2000), but "web mining" in this proposal is used as a narrow sense of "web content mining" because web documents can be viewed simply as a special kind of text documents.

2.1.2 Techniques for Text Mining

Several techniques including conceptual structure, association rule mining, episode rule mining, decision trees, and rule induction methods have been proposed for various text mining tasks.

- **Conceptual Structure**

  KDT brings up an issue of building a hierarchical structure of concepts so that data mining techniques can be applied indirectly to extracted items from unstructured text (Feldman & Dagan, 1995). The concept hierarchy in KDT is a directed acyclic graph of concepts that are interesting to the user. In this hierarchy, nodes for more general concepts have specific ones as their children. For instance, the relationship between `programming-languages` and `Java` is represented as a parent-child nodes to show that the concept for `programming-languages` is more general than that of `Java`. An example tree for computer-science skills for job postings domain is shown in Figure 2. The concept hierarchy is used to label documents, finding the relevant concept for that document and implicitly tagging all of the ancestors of that concept. KDT then statistically analyzes the content of the set of concepts generated for each text document to identify interesting distributions, e.g. highly skewed or highly deviated.

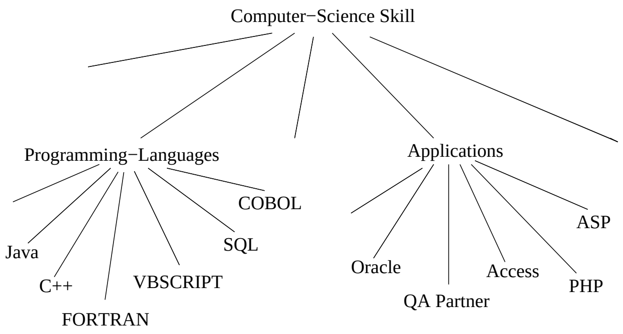


Figure 2: A subtree of the concept hierarchy for computer-science skills

- **Association Rule Mining**

FACT (Feldman & Hirsh, 1996) and DOCUMENT EXPLORER (Feldman et al., 1998) combine the conceptual approach with a standard data mining technique, association rule mining (Agrawal & Srikant, 1994). Particularly, DOCUMENT EXPLORER incorporates an association rule mining module to other text-mining subsystems such as information retrieval and information filtering to build a knowledge management system.

An *association rule* was invented to find interesting correlations between *items* from large databases. For example, the association between `Java` and `C++` is revealed when it is known to us that "In 65% of the cases when required skill of a job announcement includes `Java`, it also includes `C++`." This association is represented as

$$Java \;\Rightarrow\; C{+}{+}\ [65\%]$$

where 65% means the *confidence* of this association rule. However, a rule like

$$COBOL \;\Rightarrow\; FORTRAN\ [80\%]$$

may not be worth consideration in case the number of job announcements requiring `COBOL` and `FORTRAN` together is very small. Therefore an association rule often specifies the *support* of the rule, defined as the number of times items occur together, as follows.

$$\begin{array}{lll} Java & \Rightarrow C{+}{+} & [65\%, 81] \\ COBOL & \Rightarrow FORTRAN & [80\%, 4] \end{array}$$

Even if the confidence of the second rule is very high (80%), the low support of this rule compared to that of the first one makes it uninteresting since it is known that `COBOL` and `FORTRAN` rarely occur together (only 4 times in this case). Usually users are asked to specify a minimum support for an association rule mining algorithm to collect meaningful rules only.

Association can be between more than two items. From a corpus of Reuters news articles, FACT (Feldman & Hirsh, 1996) discovered association rules such as "Saudi Arabia, Iraq, **and** Kuwait $\Rightarrow$ Iran $[100\%, 5]$", "Qatar **and** Iraq $\Rightarrow$ Saudi Arabia $[88\%, 7]$", "Iraq $\Rightarrow$ Iran $[76\%, 63]$", and "Kuwait **and** Bahrain $\Rightarrow$ Saudi Arabia $[75\%, 6]$".

- **Episode Rule Mining**

  Ahonen, Heinonen, Klemettinen, and Verkamo (1998) and Ahonen-Myka, Heinonen, Klemettinen, and Verkamo (1999) applied existing data mining techniques to a text corpora for discovery of *episode rules*. *Episode* is a data mining terminology for temporal values assigned to a sequence of data. Ahonen et al. (1998) showed that the general data mining technique, *episode rule mining*, is applicable to text analysis task of descriptive phrase extraction. Ahonen-Myka et al. (1999) used similar techniques to find co-occurring multi-term phrases in a collection of documents.

  An episode rule discovered from a collection of Finnish legal documents is shown below.

$$chemicals,\ processing\ \Rightarrow\ storage\ [2-3]\ (69\%, 18, 19)$$

  This episode rule tells that in 69% of the cases, where the two words "chemicals" and "processing" occurred within 2 consequent words, also the word "storage" occurred within 3 words. In other words, the size of the window in which "chemicals" and "processing" occur together is 3. The support of the entire episode rule and the support for the left-hand side of the rule are also reported. In this example, "chemicals" and "processing" occur together in 19 times and "storage" followed those two words in 18 times. Episode rule mining is used for language analysis because it preserves the sequential structure of terms in a text document.

- **Decision Trees and Rule Learners**

  Decision tree methods such as C5.0 (an extension of C4.5 (Quinlan, 1993)) and rule learners such as FOIL (Quinlan, 1990) were applied to text mining tasks (Ghani, Jones, Mladenić, Nigam, & Slattery, 2000). Using C5.0, Ghani et al. (2000) discovered interesting patterns, e.g. "Aerospace/defense companies are located in Florida", from the `Hoovers.com` online resource about companies. A rule learner, FOIL was used to learn function-free Horn clauses. For example,

$$arthur\_andersen(X)\ \text{:-}\ headquarter\_city(X, madrid),\ profit(X, ?, ?)$$

  was learned for classifying `auditors` of each company. In English, this rule states that "Companies that have headquarters in Madrid and listed historical finanical information about their profits use Arthur Andersen as their `auditor`".

In addition to the above techniques, natural language processing and symbolic learning algorithms have been used for Information Extraction task as will be shown in Section 2.2. Also, Artificial Neural Networks (e.g. WEBSOM (Honkela, Kaski, Lagus, & Kohonen, 1997)) were found to be useful for text clustering (Merkl, 1998). For tasks such as document matching, ranking, and clustering, Information Retrieval (IR) techniques have been widely used (Baeza-Yates & Ribeiro-Neto, 1999) because a form of soft matching that utilizes word-frequency information typically gives superior results for most text processing problems (Salton, 1989; Cohen, 1998; Yang, 1999).

2.2 Information Extraction

The proposed DiscoTEX framework considers information extraction (IE) as a key component for text mining. In this subsection, we will introduce the information extraction task and list some of recently developed IE systems.

The goal of an information extraction system is to find specific data in natural-language text. DARPA's Message Understanding Conferences (MUC) has concentrated on IE by rigorously evaluating the performance of participating IE systems based on blind test sets of text documents (DARPA, 1993, 1995, 1998). The various MUC's have focused on tasks such as Latin American terrorism, aircraft crashes, joint ventures, micro-electronics, and company management changes. The data to be extracted is typically given by a template which specifies a list of slots to be filled with substrings taken from the document.

Figures 3 and 4 show paired documents and templates from information extraction tasks in two different domains. These templates include only slots that are filled by strings taken directly from the document. Several slots may have multiple fillers for job postings domain as in Figure 3. For example, slots such as `programming-languages`, `platforms`, `applications`, and `areas` usually have more than one filler, while slots related to the job's `title` or location (`city` and `state`) have only one filler for each.

IE is useful for a variety of applications, particularly given the recent proliferation of Internet and web documents. Recent applications for IE include medical patient records (Soderland, Fisher, Aseltine, & Lehnert, 1995), weather reports (Soderland, 1997), seminar announcements (Freitag, 1997), course homepages (Freitag, 1998a), apartment rental ads (Soderland, 1999), and job announcements (Califf & Mooney, 1999). In particular, Califf (1998) suggested extracting information from text documents in order to create easily searchable databases from the information, thus making the online text more easily accessible. For instance, information extracted from job postings in USENET newsgroup `misc.jobs.offered` can be used to build a searchable database of jobs. DiscoTEX is concerned with this aspect of IE, transforming unstructured texts to structured databases.

Information extraction systems are generally very domain-specific, and have complex structures with several modules. They usually include syntactic parsers, specialized lexicons, and discourse processing modules to handle issues such as coreference. Most information extraction systems have been built entirely by hand until recently. In order to reduce human efforts to build an information extraction systems, automatic construction of complex IE systems began to be considered lately by many researchers. Recent attempts to automate the otherwise time-consuming construction of IE systems include AutoSlog (Riloff, 1993) (and AutoSlog-TS (Riloff, 1996b)), Liep (Huffman, 1996), Crystal (Soderland et al., 1995), WHISK (Soderland,

Document

```
Leading Internet Provider using cutting edge Web technology in Austin
is accepting applications for a Senior Software Developer. The candidate
must have 5 years of software development, which includes coding in
C/C++ and experience with databases (Oracle, Sybase, Informix, etc.).
A BS degree or higher in Computer Science or related field are required.
PERL and JAVASCRIPT programming experience will be a definite plus!
This position will require developing applications under Windows95/98
and NT, meeting with customers to define requirements, and the design,
development and implementation of e-commerce, internet/intranet appli-
cations with emphasis on back-end Web site development using C++, Java
and RDBMS.  Salary: $70-85K plus outstanding benefits(Medical, Dental,
Vision, Stock Options); Location: Austin(South), TX; Type of Position:
Full Time
```

Filled Template

```
title: Senior Software Developer
company:
salary: $70-85K
city: Austin
state: TX
language: Perl, C, Javascript, Java, C++
platform: NT, Windows
application: Oracle, Informix, Sybase
area: RDBMS, Internet, Intranet, E-commerce
required years of experience: 5
desired years of experience:
required degree: BS
desired years of degree:
```

Figure 3: Sample message and filled template for job postings

1999), SRV (Freitag, 1998a), SoftMealy (Hsu, 1999), and Rapier (Califf & Mooney, 1999).

AutoSlog is one of the earliest attempts to automate the construction of an information extraction system. Although AutoSlog creates a dictionary of extraction patterns using machine learning techniques, it is not a fully automated system because 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 word 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 also 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

Document

```
Event: Data Mining Colloquium Series
Speaker: Professor Ray Mooney
         Department of Computer Sciences
         University of Texas at Austin
Date: October 27, 2000
Time: 11 AM
Place: ACES Auditorium (2.302)
Coffee: 10:30 AM (same room)
Abstract: Professor Ray Mooney, University of Texas at Austin, will
present a seminar entitled "Text Mining with Information Extraction,"
on Friday, October 27th from 11-12 in ACES 2.302.
```

Filled Template

```
speaker:    Professor Ray Mooney
location:   ACES Auditorium (2.302)
start-time: 11AM or 11
end-time:   12
```

Figure 4: Sample message and filled template for seminar announcements

describing constraints on slot fillers and their surrounding context using a specific-to-general search. Since we adopted Rapier to fill in the IE part of DiscoTEX framework for building a text-mining system, the rule representation scheme and the learning algorithm of Rapier are described in more details in Section 3.1.

Besides these symbolic rule-learning approaches, finite-state automata (FSA) are used to describe patterns for information extraction in SoftMealy (Hsu, 1999), while a hidden Markov model (HMM), a probabilistic generalization of non-deterministic FSA, is taken by Freitag and McCallum (1999). A Naive-Bayesian approach usually used for document classification is also tested by Freitag (1998b). Recent prolification of research on information extraction implies the possibility of using a successfully-built IE component for a larger text-mining system.

3 DiscoTEX: Integrating Data Mining and Information Extraction

In this section, we discuss the details of the proposed text mining framework. We consider the task of constructing a database with Rapier from job announcement postings to the USENET newsgroup in order to show the usefulness of DiscoTEX. An implementation of DiscoTEX, based on Rapier and standard rule-learning methods is presented and evaluated.

	Pre-filler Pattern	**Filler Pattern**	**Post-filler Pattern**
Rule 1)	**word**: [senior, junior]	**list**: max length: 2 **syntactic**:[normal-noun, plural-noun]	[*end-of-sentence*]
Rule 2)	[*empty*]	**word**: [programmer, analyst]	**word**: [needed, for, with]

Figure 5: Sample extraction rules of RAPIER for `title` of jobs

3.1 Information Extraction with RAPIER

RAPIER (Robust Automated Production of Information Extraction Rules) is a relational rule learner for acquiring information extraction rules from a corpus of labeled training examples. Inspired by several inductive logic programming systems, RAPIER learns information extraction rules in a bottom-up fashion from a corpus of labeled training examples. Patterns are learned by RAPIER using a specific-to-general search. Constraints on patterns for slot fillers and their context can specify the specific words, part-of-speech (POS), or semantic classes of tokens. The hypernym links in WordNet (Fellbaum, 1998) provide semantic class information, and documents are annotated with part-of-speech information using the tagger of Brill (1994). In this proposal, we use the simpler version of RAPIER that employs only word and POS constraints since WordNet classes provide no additional advantage in this domain (Califf & Mooney, 1999).

RAPIER's rule representation uses Eliza-like patterns (Weizenbaum, 1966) that make use of limited syntactic and semantic information. The extraction rules are indexed by template name and slot name and consist of three parts: 1) a pre-filler pattern that matches text immediately preceding the filler, 2) a filler pattern that must match the actual slot filler, and 3) a post-filler pattern that must match the text immediately following the filler. Each pattern is a sequence (possibly of length zero in the case of pre- and post-filler patterns) of pattern items or pattern lists. A pattern item matches exactly one word or symbol from the document that meets the item's constraints. A pattern list specifies a maximum length N and matches 0 to N words or symbols from the document that each must match the list's constraints. Figure 5 presents a typical example of extraction patterns learned by RAPIER.

To learn extraction patterns from a set of labeled examples, RAPIER first creates most-specific patterns for each slot in each example specifying the complete word and tag information for the filler and its full context, the context being the entire document. New rules are are created by generalizing pairs of existing rules using a beam search. When the best rule does not produce incorrect extractions, RAPIER adds it to the rule base and removes existing rules that it subsumes. Rules are ordered by an information-theoretic heuristic (Quinlan, 1990) weighted by the rule size.

By training on a corpus of documents annotated with their filled templates, RAPIER acquires a knowledge base of extraction rules that can be tested on novel documents. Califf (1998) and Califf and Mooney (1999) provide more information and results demonstrating that RAPIER performs well on realistic applications such as USENET job postings and seminar announcements.

AIX $\in$ *platform* **and** DB2 $\in$ *application*	$\rightarrow$ Lotus Notes $\in$ *application*
Visual Basic $\in$ *language* **and** OLE $\in$ *area*	$\rightarrow$ UNIX $\notin$ *platform*
Oracle $\in$ *application* **and** QA Partner $\in$ *application*	$\rightarrow$ SQL $\in$ *language*
3D $\in$ *area* **and** Games $\in$ *area* **and** E-Commerce $\notin$ *area*	$\rightarrow$ SQL $\notin$ *language*
Java $\in$ *language* **and** ActiveX $\in$ *area* **and** Graphics $\in$ *area*	$\rightarrow$ Web $\in$ *area*
C++ $\in$ *language* **and** C $\in$ *language* **and** CORBA $\in$ *application*	$\rightarrow$ Windows $\in$ *platform*
UNIX $\notin$ *platform* **and** Windows $\notin$ *platform* **and** Games $\in$ *area*	$\rightarrow$ 3 D $\in$ *area*
HTML $\in$ *language* **and** NT $\in$ *platform* **and** ASP $\in$ *application*	$\rightarrow$ Database $\in$ *area*

Figure 6: Sample rules of DiscoTEX for computer-science job postings

3.2 Rule Induction

Interesting rules can be discovered from a database created by an IE system. For instance, from a collection of the job announcment postings on a USENET newsgroup, we discovered a rule, "If a computer-science job requires AIX for `platform` and DB2 for `applications`, it also requires IMS for `applications`." This reveals a previously unknown fact to someone who does not know all of AIX, DB2, and IMS are products of IBM and both DB2 and IMS are database-related software. In this subsection, we will detail our work to induce prediction rules from the IE templates filled by Rapier.

After constructing an IE system that extracts the desired set of slots for a given application, a database can be constructed from a corpus of texts by applying the IE extraction patterns to each document to create a collection of structured records. Standard KDD techniques can then be applied to the resulting database to discover interesting relationships. Specifically, we induce rules for predicting each piece of information in each database field given all other information in a record. Standard methods for learning classification rules can be employed for this task.

In order to discover prediction rules, we treat each slot-value pair in the extracted database as a distinct binary feature, such as `graphics` $\in$ `area`, and learn rules for predicting each feature from all other features. Similar slot fillers are first collapsed into a pre-determined standard term. For example, "Windows 95" is a popular filler for the `platform` slot, but it often appears as "Win 95", "Win95", "MS Win 95", and so on, and "DBA" in the `title` slot is an abbreviation for "DataBase Administrator". These terms are collapsed to unique slot values before prediction rules are mined from the data. A small domain-dependent synonym dictionary is used to identify such similar terms. Trivial cases such as "Databases" $\rightarrow$ "Database" and "Client/Server" $\rightarrow$ "Client-Server" are handled by manually contrived synonym-checking rules.

We have applied Ripper (Cohen, 1995) to induce rules from the resulting binary data. Ripper is a learning method that forms simple rules in a fairly effective manner. C4.5rules (Quinlan, 1993) was also applied to learn prediction rules. Ripper runs significantly faster than C4.5rules since it has an ability to handle *set-valued features* (Cohen, 1996b) to avoid the step of explicitly translating slot fillers into a large number of binary features.

Discovered knowledge describing the relationships between slot values is written in a form of production rules. If there is a tendency for `Web` to appear in the `area` slot when `ShockWave` appears in the `applications` slot, this is represented by the production rule, "`ShockWave` $\in$ `application`

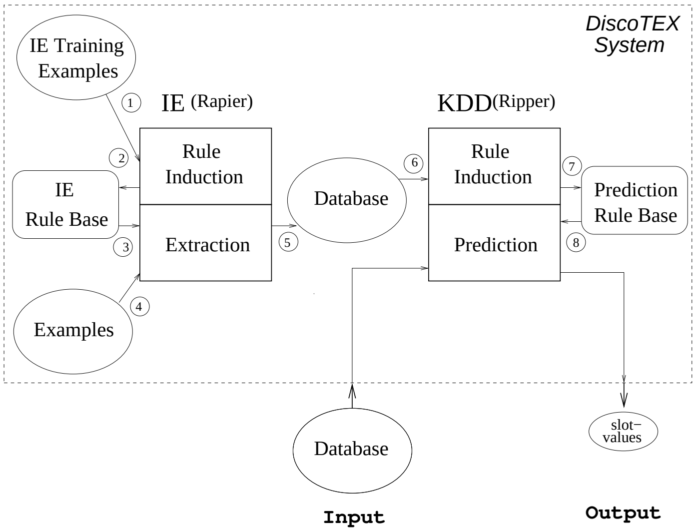


Figure 7: The DiscoTEX system architecture

$\rightarrow$ `Web` $\in$ `area`". Rules can also predict the absence of a filler in a slot. Sample rules mined from a database of 600 jobs extracted from the USENET newsgroup `austin.jobs` are shown in Figure 6.

3.3 System Architecture

The overall architecture of DiscoTEX is shown in Figure 7. First, documents annotated by the user are provided to Rapier as training data. IE rules induced from this training set are stored in the IE rule base and subsequently used by the extraction module. The learned IE system then takes unlabeled texts and transforms them into a database of slot-values, which is provided to the KDD component (i.e. Ripper or C4.5rules) as a training set for constructing a knowledge base of prediction rules. The training data for KDD can include the user-labeled documents used for training IE, as well as a larger IE-labeled set automatically extracted from raw text. DiscoTEX also includes a capability for improving the recall of the learned IE system by proposing additional slot fillers based on learned prediction rules. More details on this aspect of the system are described in Section 3.5

In order to test the accuracy of the discovered rules, they are used to predict the information in a disjoint database of user-labeled examples. For each test document, each possible slot-value is predicted to be present or absent given information on all of its other slot-values. Average performance across all features and all test examples is then computed. The rules produced by Ripper and C4.5rules were found to be of similar accuracy, and the experiments in this proposal employ Ripper except those in Section 3.6. The computational time and space complexity for Ripper is significantly less than those of C4.5rules.

3.4 Experimental Results

3.4.1 Experimental Methodology

Discovered knowledge is only useful and informative if it is accurate. Discovering fluke correlations in data is not productive, and therefore it is important to measure the accuracy of discovered knowledge on independent test data. The primary question we address in the experiments of this section is whether knowledge discovered from automatically extracted data (which may be quite noisy) is relatively reliable compared to knowledge discovered from a manually constructed database.

To test the overall system, 600 computer-science job postings to the newsgroup `austin.jobs` were collected and manually annotated with correct extraction templates. Ten-fold cross validation was used to generate training and test sets. Rules were induced for predicting the fillers of the `programming-languages`, `platforms`, `applications`, and `areas` slots, since these are usually filled with multiple discrete-valued fillers and have obvious potential relationships between their values. The total number of slot-values used in the experiment is 476; 48 slot-values are for the `programming-languages` slot, 59 for `platforms`, 159 for `applications`, and 210 for `areas`.

The classification accuracy for predicting absence or presence of slot fillers is not a particularly informative performance metric since high accuracy can be achieved by simply assuming every slot filler is absent. For instance, with 60 user-labeled examples, DiscoTEX gives a classification accuracy of 92.7% while the all-absent strategy has an accuracy of 92.5%. This is because the set of potential slot fillers is very large and not fixed in advance, and only a small fraction of possible fillers is present in any given example. Therefore, we evaluate the performance of DiscoTEX using the IE performance metrics of precision, recall, and F-measure with regard to predicting slot fillers. These metrics are defined as follows:

$$precision = \frac{NumberOfPresentSlotValuesCorrectlyPredicted}{NumberOfSlotValuesPredictedToBePresent} \tag{1}$$

$$recall = \frac{NumberOfPresentSlotValuesCorrectlyPredicted}{NumberOfPresentSlotValues} \tag{2}$$

F-measure is the harmonic mean of precision and recall and is computed as follows (when the same weight is given to precision and recall):

$$F-measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

In order to obtain non-trivial bounds on precision and recall, a simple random guessing method is used as a benchmark. This approach guesses a slot-value based on its frequency of occurrence in the training data. For instance, if `Java` occurs as a `programming-language` in 29% of jobs

	Present	Absent
Predicted To Be Present	$m \times p$	$(n-m) \times p$
Predicted To Be Absent	$m \times (1-p)$	$(n-m) \times (1-p)$

Table 1: The expected outcome for random guessing

in the training data, then this method guesses that it occurs 29% of the time for the test data. Instead of simulating this method, we analytically calculated its expected precision and recall for each slot-value. The expected outcome for this strategy for a given slot-value is summarized in Table 1, where p is the percentage of times the slot-value appears the training examples, n is the total number of the test examples and m is the number of times the slot-value occurs in the test data.

Using the information in the table, the precision and the recall for random-guessing are determined as follows:

$$precision = \frac{m \times p}{(m \times p) + ((n-m) \times p)} = m/n \tag{4}$$

$$recall = \frac{m \times p}{(m \times p) + (m \times (1-p))} = p \tag{5}$$

Therefore, the benchmark precision for a slot-value is its probability of occurrence as estimated from the test data and the recall is its probability of occurrence as estimated from the training data. The only difference between the two is due to sampling error.

3.4.2 Results

Since `austin.jobs` is not a moderated newsgroup, not all posted documents are relevant to our task. Some of them are resumes posted by job-seekers, advertisements, or non-computer-science job postings. Therefore, before constructing a database using an IE system, we filtered out irrelevant documents from the newsgroup using a trained text categorizer. First, 1,000 postings were collected and classified by a human expert as relevant or irrelevant. Next, a bag-of-words Naive-Bayes text categorizer (McCallum & Nigam, 1998) was trained on this data to identify relevant documents (using the RAINBOW package (McCallum, 1996)). The resulting categorizer has an accuracy of over 99% and is used to filter irrelevant documents from the original postings.

In the experiments in this section, RAPIER was trained on only 60 labeled documents, at which point its accuracy at extracting information is somewhat limited; extraction precision (percentage of extracted slot fillers that are correct) is about 91.9% and extraction recall (percentage of all of the correct fillers extracted) is about 52.4% . We purposely trained RAPIER on a relatively small corpus in order to demonstrate that labeling only a relatively small number of documents can result in a good set of extraction rules that is capable of building a database from which accurate knowledge can be discovered.

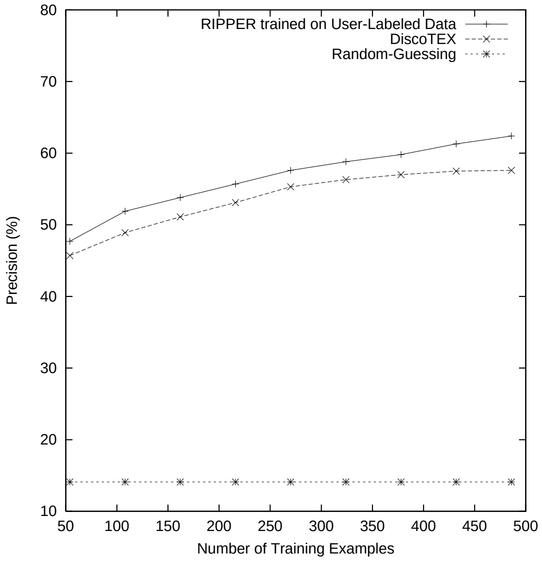
Because of the two different training phases used in DiscoTEX, there is a question of whether or not the training set for IE should also be used to train the rule-miner. In realistic situations, there is no reason not to use the IE training data for mining since the human effort has already been expended to correctly extract the data in this text. However, to clearly illustrate the difference between mining human-labeled and IE-labeled data, we first show a comparison with a disjoint IE training set. In this experiment, the IE training data are thrown away once they have been used to train Rapier, since the extractor is unlikely to make the normal number of extraction errors on this data. Ten-fold cross-validation is performed on the remaining 540 examples for evaluation of the data mining part. In order to clearly illustrate the impact of mining automatically extracted data, the same set of training examples was provided to both KDD systems. The only difference between them is that the training data for the rule-miner of DiscoTEX is automatically extracted by Rapier after being trained on a disjoint set of 60 user-labeled examples. Both systems are tested on user-labeled data to identify the quality of the rules produced. Figure 8 shows the learning curves for precision, recall, and F-measure, respectively.

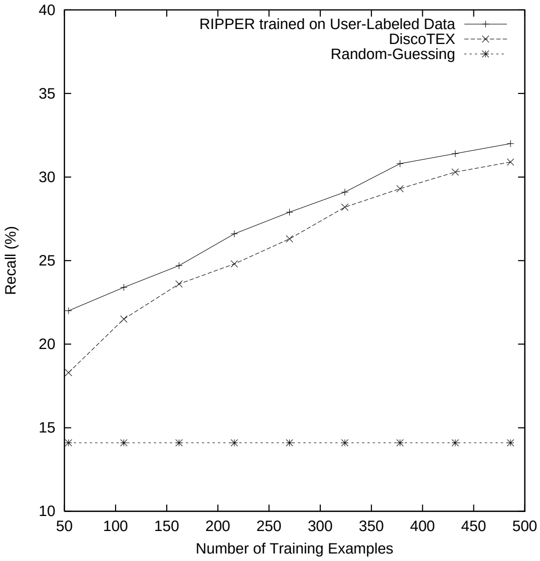
Even with a small amount of user-labeled data, the results indicate that DiscoTEX achieves a performance fairly comparable to the rule-miner trained on a manually constructed database, while random-guessing does quite poorly. The recall curves in Figure 8 indicates that DiscoTEX does relatively worse with the first 60 training examples, but quickly improves with 60 additional examples. The results also show that the precision of DiscoTEX seems to start leveling off a bit sooner, this is presumably due to the fact that extraction errors put a somewhat lower ceiling on the performance it can eventually achieve.

Figure 9 presents F-measures for DiscoTEX's performance on individual slots. Not surprisingly, the `programming-languages` slot with the least number of possible values shows the best performance, and the `area` slot with as many as 210 values does poorly. More interesting is the fact that different slots show quite different learning rates.

Figure 10 shows the learning curves for precision, recall, and F-measure under the "more natural" scenario in which the training set provided to Rapier, consisting of 60 user-labeled examples, is also provided to the rule-miner as a part of its training set. In this case, both approaches start with the same 60 user-labeled examples, which have already been used to train the IE part of DiscoTEX. However, as DiscoTEX proceeds to discover knowledge from data it automatically extracts from raw text, it fairly closely tracks the performance of a system trained on additional data laboriously extracted by a human expert. Since in this case DiscoTEX has the advantage of a small set of relatively noise-free data to start with, its performance is even somewhat closer to that achieved by mining a hand-built database.

All of the results presented above employed 60 labeled examples to train the IE system. In a followup experiment, we examined the effect of increasing the number of IE training examples to obtain a more accurate extraction module. We varied the number of training examples given to Rapier (trying 60, 120, 180, 240, and 300 examples), always using 240 examples in the database to be mined by Ripper. The size of the test set is 60 as in the previous experiment. Figure 11 shows the performance results. Increasing the number of IE training examples improves the accuracy of





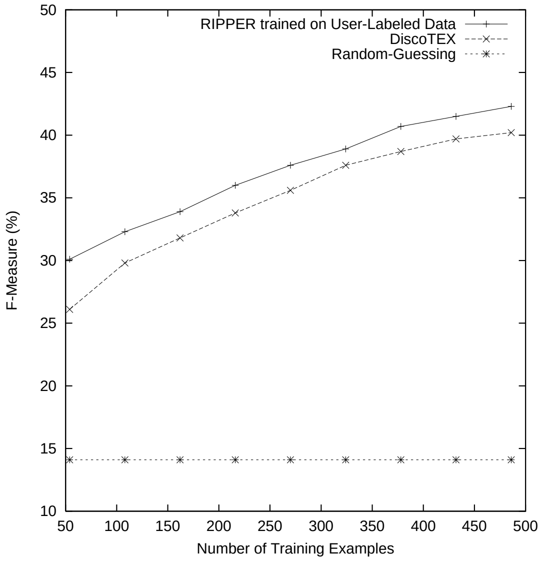


Figure 8: Precision, Recall, F-Measure with disjoint IE training set

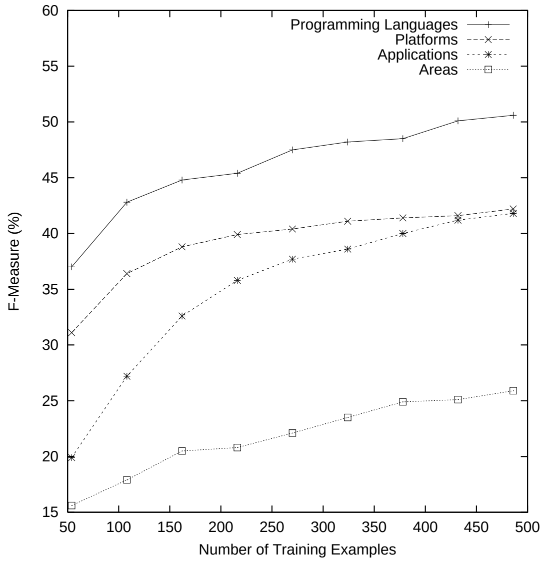


Figure 9: F-measure for DiscoTEX by slots

the mined rules a bit, further approaching the accuracy of Ripper trained on user-labeled data. However, accuracy improves slowly with additional IE training data. This result indicates that if the training set for data mining to be automatically labeled by an IE module is large enough (240 in this experiment), DiscoTEX is able to achieve a fairly good performance with only a small amount of effort devoted to labeling IE training examples.

3.5 Using Mined Rules to Improve IE

After mining knowledge from extracted data, DiscoTEX is able to use the discovered rules to predict missing information during subsequent extraction. In this subsection, we present an algorithm to use mined rules for the purpose of improving the underlying IE system.

Tests of IE systems usually consider two performance measures again, precision and recall. For extraction they are defined as:

$$precision = \frac{NumberOfCorrectFillersExtracted}{NumberOfFillersExtracted} \tag{6}$$

$$recall = \frac{NumberOfCorrectFillersExtracted}{NumberOfFillersInCorrectTemplates} \tag{7}$$

F-measure is also computed to combine precision and recall by Equation 3. These performance

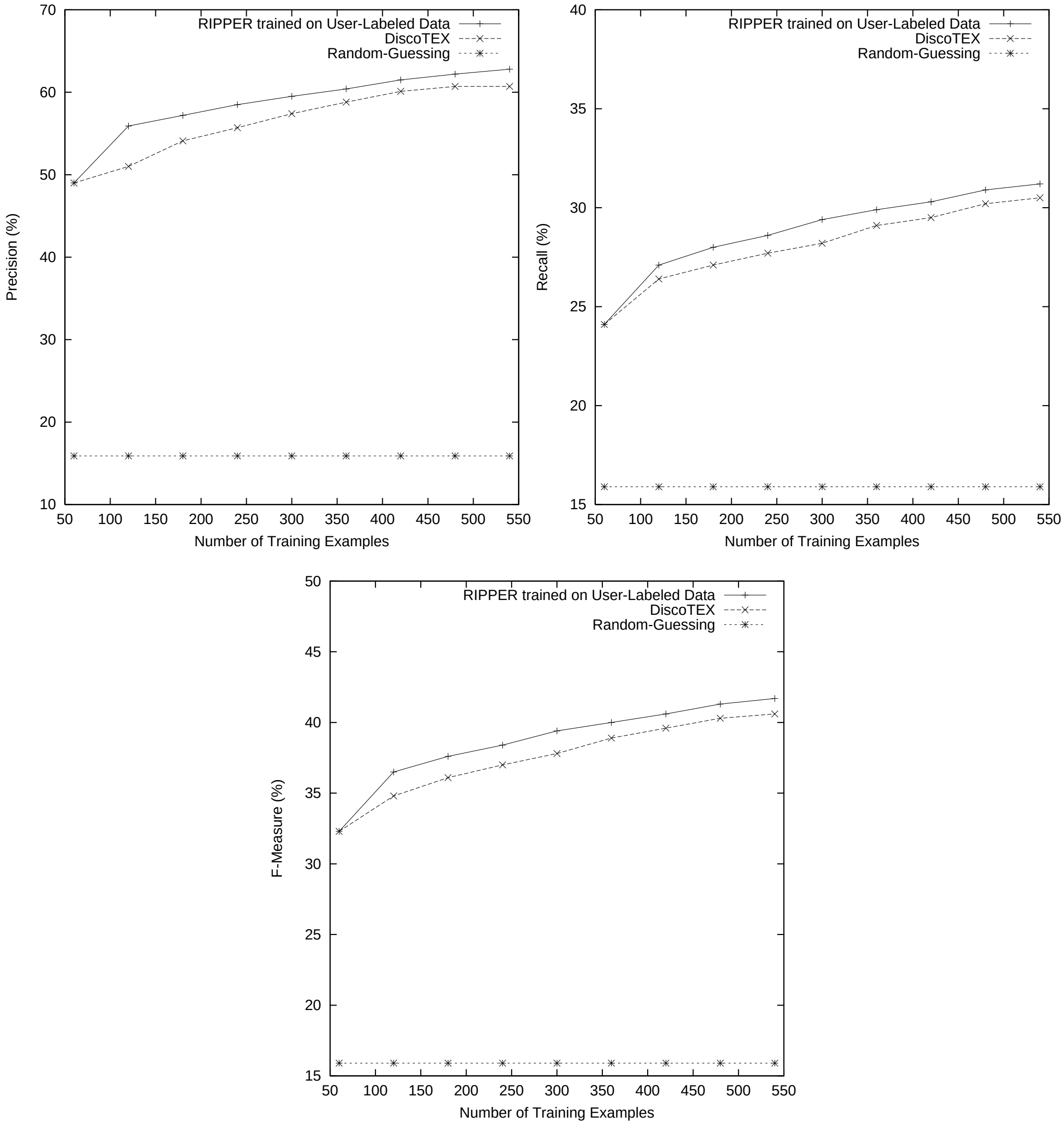


Figure 10: Precision, recall, F-measure with reused IE training set

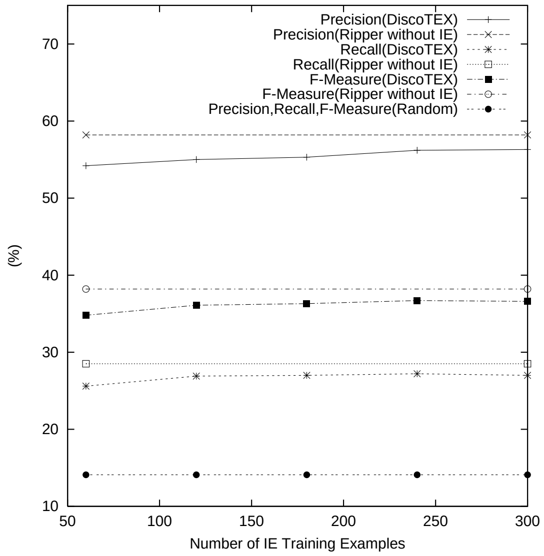


Figure 11: Performances with varied sizes of IE training set

metrics are more informative than the accuracy of predicting the presence/absence across all slot-value pairs due to the large size of the potential slot filler set and its sparsity.

Many IE systems provide relatively high precision, but recall is typically much lower. Previous experiments in the job postings domain showed RAPIER's precision (e.g. low 90%'s) is higher than its recall (e.g. mid 60%'s) (Califf, 1998). Currently, RAPIER's search focuses on finding high-precision rules and does not include a method for trading-off precision and recall. Although several methods have been developed for allowing a rule learner to trade-off precision and recall (Cohen, 1996a), this typically leaves the overall F-measure unchanged.

Pseudocode for the rule mining phase is shown in Figure 12. A final step shown in the figure is filtering the discovered rules on both the training data and (optionally) a disjoint set of labeled validation data in order to retain only the most accurate of the induced rules. Currently, rules that make *any* incorrect predictions on either the training or validation extracted templates are discarded.

By using additional knowledge in the form of prediction rules mined from a larger set of data automatically extracted from additional unannotated text, it may be possible to improve recall without unduly sacrificing precision. For example, suppose we discovered a rule:

- SQL $\in$ programming-languages $\rightarrow$ database $\in$ area

If the IE system extracted `SQL` in the `programming-languages` slot but failed to extract `database` in `area` slot, we may want to assume there was an extraction error and add `database` to the `area` slot, potentially improving recall. Therefore, after applying extraction rules to a document, DISCOTEX applies its mined rules to the resulting initial data to predict additional potential extractions. The final decision whether or not to extract a predicted filler is based on whether

Input: D is the training set.

Procedure DiscoTEX-Mining (D)

Determine T, a threshold value for rule validation
/ Information extraction */*
Create a database of labeled examples
    (by applying IE to the document corpus)
/ Conversion to binary features for data mining to be applied */*
For each labeled example D **do**
  $S :=$ Set of slot fillers of D
  Convert S to binary features, namely S'
/ Data mining */*
Build a prediction rule base, RB
    (by applying RIPPER or C4.5RULES to S')
/ Rule Validation */*
For each prediction rule $R \in RB$ **do**
  Verify R on training data and validation data
  If the accuracy of R is lower than T
  Then delete R from RB

Figure 12: Algorithm specification for DISCOTEX: rule mining phase

Input: D is the set of unlabeled examples.

Procedure DiscoTEX-Extraction (D)

For each example D do
  Extract fillers from D using extraction rules
  For each rule R in the prediction rule base RB **do**
    / Using mined rules */*
    If R fires on the current extracted fillers
      If the predicted filler is a substring of D
      / Information extraction */*
      Then Extract the predicted filler

Figure 13: Algorithm specification for DISCOTEX: IE Phase

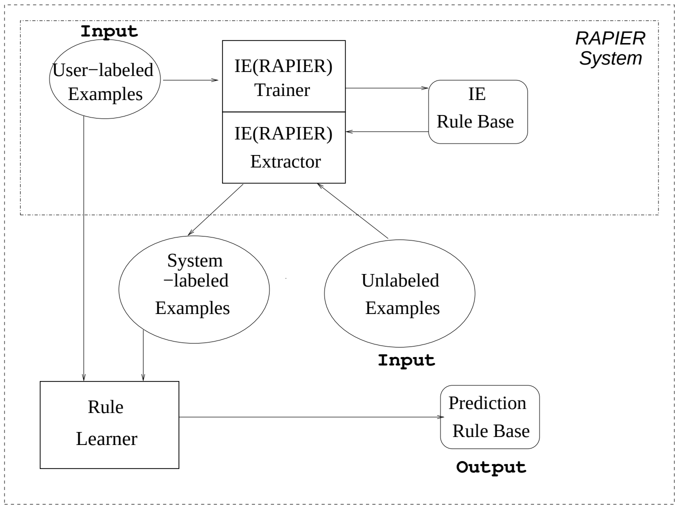


Figure 14: The system architecture - Training

the filler (or any of its synonyms) occurs in the document as a substring. If the filler is found in the text, the extractor considers its prediction confirmed and extracts the filler. Mined rules that predict the absence of a filler are not used to remove extracted information since there is no analogous confirmation step for improving precision. This overall extraction algorithm is summarized in Figure 13.

One final issue is the order in which prediction rules are applied. When there are interacting rules, such as $\texttt{HTML} \in \texttt{programming-languages} \rightarrow \texttt{WWW} \in \texttt{area}$ and $\texttt{WWW} \notin \texttt{area} \rightarrow C{+}{+} \in \texttt{programming-languages}$, different rule-application orderings can produce different results. Without the first rule, a document with $\texttt{HTML} \in \texttt{programming-language}$ but without $\texttt{WWW} \in \texttt{area}$ in its initial filled template will make the second rule fire and predict $C{+}{+} \in \texttt{programming-languages}$. However, if the first rule is executed first and its prediction is confirmed, then WWW will be extracted and the second rule can no longer fire. In DiscoTEX, all rules with negations in their antecedent conditions are applied first. This ordering strategy attempts to maximally increase recall by making as many confirmable predictions as possible.

The training and testing phase of the final system are shown separately in two Figures 14 and 15. Documents annotated by user with extracted information, as well as unsupervised data which has been processed by the initial IE system (which Rapier has learned from the supervised data) are all used to create a database. The rule mining module then processes this database to construct a knowledge base of rules for predicting slot values. These prediction rules are then used during testing to improve the recall of the existing IE system by proposing additional slot fillers whose presence in the document are confirmed before adding them to final extraction template.

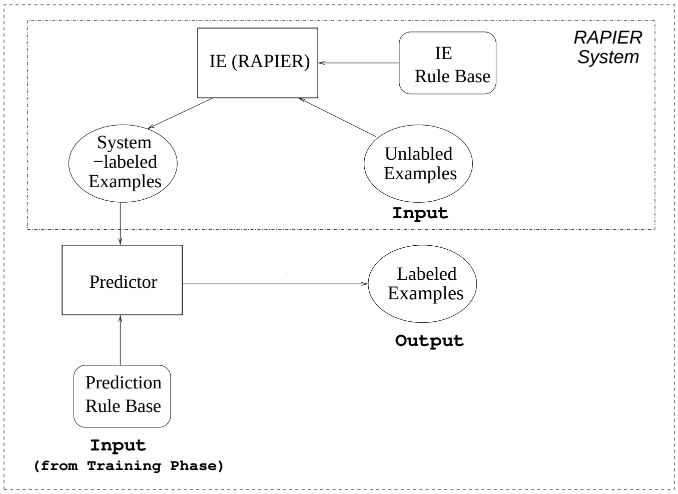


Figure 15: The system architecture - Test

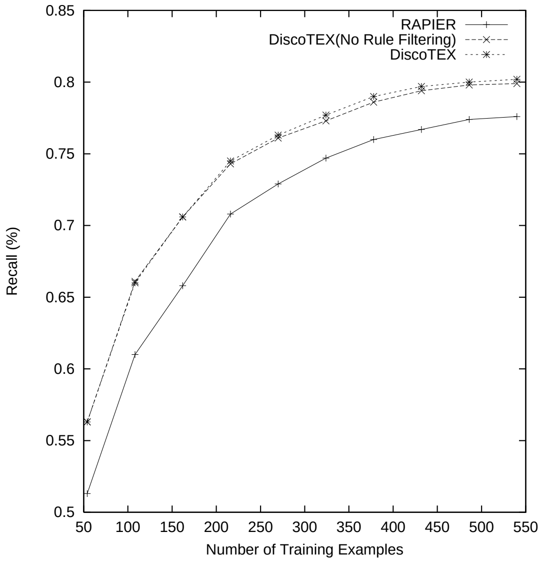
3.6 Evaluation of Improving IE

3.6.1 Experimental Methodology

600 user-annotated computer-science job postings from the newsgroup `austin.jobs`, collected for the experiments in Section 3.4 were used to test the overall system. Ten-fold cross validation was used to generate training and test sets. In addition, 4,000 unannotated documents were collected as additional optional input to the data mining module. Rules were induced for predicting the fillers of four slots, `programming-languages`, `platforms`, `applications`, and `areas` slots as in Section 3.4. The `title` slot is also used in this experiment, but only as a possible antecedent condition of a production rule, not as a consequent. `Title` has many possible values and is difficult to predict; however, may be useful as a predictor since fillers such as "Database Administrator" can help determine other values. C4.5RULES (Quinlan, 1993) is used instead of RIPPER for prediction rule learning in this experiment. C4.5RULES induces rules by learning decision trees and translating them into pruned rules.

3.6.2 Results

Figure 16 shows the learning curves for recall and and F-measure. Unlabeled examples are not employed in these results. In order to clearly illustrate the impact of the amount of training data for both extraction and prediction rule learning, the same set of annotated data was provided to both RAPIER and C4.5RULES. Figure 16 shows a comparison between the performance of RAPIER alone, DISCOTEX without filtering rules on independent data, and DISCOTEX with fully filtered rules. As previously stated, there are two ways of filtering rules before they are added to the prediction rule base; simply training and testing induced rules on the entire training set, or holding out part of the original training set as a disjoint validation set. The results were statistically evaluated by a two-tailed, paired t-test. For each training set size, the performances



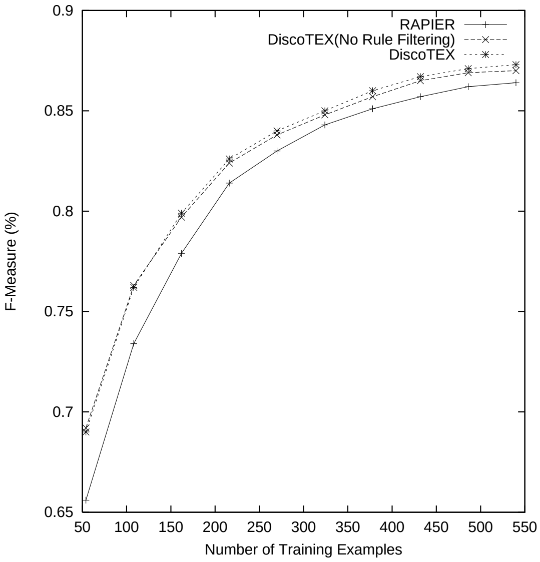


Figure 16: Recall and F-Measure on job postings

of Rapier and DiscoTEX (either with or without rule filtering) were compared to determine if their differences were statistically significant ($p < 0.05$).

DiscoTEX using fully filtered rules performs the best, although DiscoTEX without filtering on disjoint data does almost as well. As hypothesized, DiscoTEX provides higher recall, and although it does decrease precision somewhat, overall F-measure is moderately increased. One interesting aspect is that DiscoTEX retains a fixed recall advantage over Rapier as the size of the training set increases. This is probably due to the fact that the increased amount of data provided to the text miner also continues to improve the quality of the acquired prediction rules. Overall, these results demonstrate the role of data mining in improving the performance of IE.

Table 2 shows results on precision, recall and F-measure when additional unlabeled documents are used to construct a larger database prior to mining for prediction rules (which are then filtered on a validation set). The 540 labeled examples used to train the extractor were always provided to the rule miner, while the number of additional unsupervised examples were varied from 0 to 4,000. The results show that the more unsupervised data supplied for building the prediction rule base, the higher the recall and the overall F-measure. Although precision does suffer, the decrease is not as large as the increase in recall.

Although adding information extracted from unlabeled documents to the database may result in a larger database and therefore more good prediction rules, it may also result in noise in the database due to extraction errors and consequently cause some inaccurate prediction rules to be discovered as well. The average F-measure without prediction rules is 86.4%, but it goes up to 88.1% when DiscoTEX is provided with 540 labeled examples and 4,000 unlabeled examples. Unlabeled examples do not show as much power as labeled examples in producing good prediction rules, because only 540 labeled examples boost recall rate and F-measure more than 4,000

Number of Examples for Rule Mining	Precision	Recall	F-measure
0	97.4	77.6	86.4
540(Labeled)	95.8	80.2	87.3
540+1000(Unlabeled)	94.8	81.5	87.6
540+2000(Unlabeled)	94.5	81.8	87.7
540+3000(Unlabeled)	94.2	82.4	87.9
540+4000(Unlabeled)	93.5	83.3	88.1
Matching Fillers	59.4	94.9	73.1

Table 2: Performance results of DiscoTEX with unlabeled examples

unlabeled examples. However, unlabeled examples are still helpful since recall and F-measure do slowly increase as more unlabeled examples are provided.

As a benchmark, in the last row of the Table 2, we also show the performance of a simple method for increasing recall by always extracting substrings that are known fillers for a particular slot. This version remembers all strings that appear at least once in each slot in the database. Whenever a known filler string, e.g. `Java`, is contained in a test document, it is extracted as a filler for the corresponding slot, e.g. `programming-language`. This is equivalent to replacing the mined rules in DiscoTEX with trivial rules of the form "`True` $\rightarrow$ `<slot-value>` $\in$ `<slot>`" for every known slot value. The reason why this works poorly is that a filler string contained in a job posting is not necessarily the correct filler for the corresponding slot. For instance, `database` or `Web` can appear in a newsgroup posting, not in the list of required skills of that particular job announcement, but in the general description for the company. The fact that the precision and F-measure of this strawman are much worse than DiscoTEX's demonstrates the additional value of rule mining for improving extraction performance.

4 TextRISE: Mining Soft-Matching Rules from Textual Data

As discussed in Section 3.2, one step in DiscoTEX that is performed manually is collapsing similar slot-fillers in the extracted data into a canonical form. For example, "NT," "Windows NT," and "Microsoft Windows NT" are typically extracted fillers for the `platform` slot in the USENET job announcement domain. All of those are mapped to a unique term by a synonym-checking dictionary before the rule mining step and treated as the same attribute afterwards. Such collapsing could be automated by clustering slot fillers using a distance metric based on textual similarity, such as character edit distance used by Cora research paper search engine (McCallum, Nigam, Rennie, & Seymore, 2000), or semantic similarity as determined by context-based word clustering (Pereira, Tishby, & Lee, 1993). An alternative approach we adopted in this proposal is to allow partial matching of slot-fillers during the discovery process, instead of requiring or creating canonical slot-fillers that must match exactly.

The need for soft matching of text strings in discovered rules is an aspect of text mining that requires changes to existing rule induction methods. In this section, we explore the discovery of rules that allow soft matching of slot-fillers by adapting the RISE algorithm of unifying rule-based and instance-based learning methods (Domingos, 1996). Like WHIRL (Cohen, 1998), our approach uses a TFIDF text-similarity metric from information retrieval (Baeza-Yates & Ribeiro-Neto, 1999) to find examples that are close but not exact matches to the conditions of a rule. In Section 4.4, we presents results on applying a preliminary version of the TEXTRISE system to two text databases, one of book information extracted from `Amazon.com` and another of patent applications from `Getthepatent.com`. We evaluate the quality of the discovered rules on independent data by measuring the similarity of predicted text and actual text. By comparing results to the predictions made by nearest-neighbor and mined association rules, we demonstrate the advantage of mining soft-matching rules.

4.1 The RISE Algorithm

The RISE (Rule Induction from a Set of Exemplars) algorithm was proposed to overcome the well-known *small disjuncts problem* and *splintering problem* of the rule induction, and mitigating the instance-based learning's vulnerability to noise and irrelevant features at the same time. Unlike other combined model approaches, RISE is a unified single algorithm which is able to behave both as an instance-based classifier and a rule induction system. In extensive experiments, RISE was fairly consistently more accurate than alternative methods, including standard rule-based and instance-based algorithms (Domingos, 1996).

Instead of requiring rules to match exactly in order to make a prediction, RISE makes predictions by selecting the closest matching rule according to a standard distance metric typically used in nearest-neighbor methods (a modified Euclidian distance). By generating generalized rules instead of remembering specific instances and by using a similarity metric rather than exact matching to make predictions, it elegantly combines the properties of rule induction and instance-based learning.

Flexible matching rules are acquired using a specific-to-general (bottom-up) induction algorithm that starts with maximally specific rules for every example and then repeatedly minimally generalizes each rule to cover the nearest example it does not already cover, unless this results in a decrease in the performance of the overall rule base. Performance of a rule base is measured by conducting leave-one-out testing on the training data using the closest-matching rule for making predictions. This process repeats until any additional generalization does not increase performance. When classifying test examples, the nearest rule to each example is found, and the rule's class is assigned to the example. The RISE algorithm for learning rules is summarized in Figure 17.

For a walk-though example, consider the following training set for voting records.

```
1) y  y  n  n  y  -->   republican
2) n  y  y  n  n  -->   republican
3) n  n  n  y  n  -->   republican
4) y  n  n  n  n  -->   democrat
5) n  n  y  y  n  -->   democrat
```

Input: ES is the training set.
Output: RS is the rule set.

Function RISE (ES)

$RS := ES$.
Compute $Accuracy(RS)$.
Repeat
  For each rule $R \in RS$,
    Find the nearest example E to R (not covered, but of the same class)
    $R' :=$ Most_Specific_Generalization(R, E)
    $RS' := RS$ with R replaced by R'
    If $Accuracy(RS', ES) \geq Accuracy(RS, ES)$
    Then $RS := RS'$
    Delete R' from RS if it is a duplicate.
Until no increase in $Accuracy(RS, ES)$ is obtained.
Return RS.

Figure 17: The RISE rule-learning algorithm

These input/output pairs associate patterns of voting decision with party allegiance. The output variable shows whether the politician that cast the votes was Republican or Democrat. Within the input, a 'y' indicates a vote in favor while a 'n' indicates a vote against.

We assume that the Manhattan distance is used to find the closest rule. The initial rule set for the RISE algorithm is the same as the training set since RISE starts with maximally specific rules. The initial accuracy of the rule set is 0 because no example is correctly classified by this rule set with the leave-one out method. For instance, example 1) is incorrectly classified as `democrat` since the closest rule for this example is rule 5) and its class is `democrat`.

After calculating the initial accuracy, we locate the most similar example with the first rule in the current rule base (among those that have the same class). Example 1) is not considered as a candidate because it is already covered by this rule. The closest example in the training examples is 2), that is to be generalized with rule 1) in the current rule set. The generalized rule between these two is

```
1') ?  y  ?  n  ?   -->  republican
```

By replacing rule 1) with 1)$'$, the rule set to be compared with the original one is updated as follows.

```
1') ?  y  ?  n  ?   -->  republican
2)  n  y  y  n  n   -->  republican
3)  n  n  n  y  n   -->  republican
4)  y  n  n  n  n   -->  democrat
5)  n  n  y  y  n   -->  democrat
```

We calculate the global accuracy of the updated rule set in order to determine if the old rule should be replaced with the generalized one. Every example in the training set is matched successively to rules in those rule sets. Since the generalized rule increases the accuracy by classifying the second example (which used to be classified incorrectly as `democrat` by the old rule set) correctly, we accept this updated rule base and repeat this process to all the remaining rules. As a result, we get the second-stage rule set like following (after eliminating duplicated rules).

```
1) ?  y  ?  n  ?  -->   republican
2) n  ?  ?  ?  n  -->   republican
3) ?  n  ?  ?  n  -->   democrat
```

The algorithm is terminated at this point because no rules can be generalized with any examples in the training set to increase the overall accuracy. Generalizations of rule 1) or rule 2) with the closest training example,

```
?  ?  ?  ?  ?  -->   republican
```

cause false predictions (classifying `demoract` as `republican`), resulting in a decrease in the global accuracy.

If we are now presented with the following test case,

```
y  y  y  y  y  -->
```

the output value is `republican` because the closest rule to this test example is rule 1) in the final rule base and its class is `republican`.

Domingos (1996) empirically shows that the RISE algorithm can create synergistic effects between the rule induction method and instance-based learning scheme. Results indicate that the classifier generated by RISE is better than those produced by its parent algorithms, rule induction and nearest-neighbor. Compared to nearest-neighbor which is sensitive to irrelevant features, RISE has a superior ability to choose relevant features for different regions of the instance space. RISE has two major advantages over rule induction. First, RISE is better at dealing with exceptions while rule induction suffers from small disjuncts problem. Second, it mitigates the splintering problem of having dwindling number of available examples during the induction process, by evaluating each rule set with respect to the accuracy on the entire training set. Training is reasonably computationally efficient, requiring time $O(e^2, a^2)$ where e is the number of examples, and a the number of attributes.

4.2 The TextRISE algorithm

RISE is not directly applicable to mining rules from extracted text because: 1) its similarity metric is not text-based and 2) it learns rules for classification rather than text prediction. TextRISE addresses both of these issues. A standard vector-space metric from information retrieval (IR) (Baeza-Yates & Ribeiro-Neto, 1999) is currently used to provide an appropriate similarity metric for TextRISE. Classification accuracy as a measure of performance is replaced with the average similarity of the text predicted to fill a slot and the actual filler.

Book Description

Title: Harry Potter and the Goblet of Fire (Book 4)
Author: Joanna K. Rowling
Comments: This book was the best book I have ever read.
If you are in for excitement this book is the one you want to read.
Subject: Fiction, Mystery, Magic, Children, School,
Juvenile Fiction, Fantasy, Wizards

Representation

Author = {“joanna”, “rowling”}
Title = {“harry”, “potter”, “goblet”, “fire”, “book”}
Comments = {“book”, “book”, “read”, “excitement”, “read”}
Subject = {“fiction”, “mystery”, “magic”, “children”, “school”,
“juvenile”, “fiction”, “fantasy”, “wizards”}

Figure 18: An example of representation for a book document

4.2.1 Representation

We represent an IE-processed document as a list of bags of words (BOWs), one bag for each slot filler. We currently eliminate 524 commonly-occurring stop-words (e.g. “the”, “is”, and “you”) but do not perform stemming. Figure 18 shows an example for a book description and its BOW representation. Standard set-operations are extended to bags in the obvious way (Peterson, 1976). The definition of “bag” and several operators for bags are given in Appendix A. A learned rule is represented as an antecedent that is a conjunction of BOWs for some subset of slots and a conclusion that is a predicted BOW for another slot.

The standard cosine measure between two vectors is adopted for the similarity metric, while edit distance metric could be used for cases in which the extracted slot-fillers contain short strings rather than a series of words. The similarity of two slot-fillers is calculated by the *vector space model* for text. In this model, a text is represented as a vector of real numbers, where each component corresponds to a word that appears in the set of all documents and the value is its frequency in the document. The similarity of two documents x and y is the *cosine of the angle* between two vectors $\vec{x}$ and $\vec{y}$ representing x and y respectively, and calculated by the following formula:

$$Similarity(x, y) = \frac{\vec{x} \cdot \vec{y}}{\mid \vec{x} \mid \times \mid \vec{y} \mid} \tag{8}$$

where $\mid \vec{x} \mid$ and $\mid \vec{y} \mid$ are the norms of each document vector.

The TFIDF (Term Frequency times Inverse Document Frequency) weighting scheme (Salton, 1989) has been used to assign higher weights to distinguished terms in the given document. TFIDF maintains two assumptions about the importance of a term. First, the more a term appears in

Inputs: $R = (A_1, A_2, ..., A_n, C_R)$ is a rule
$E = (E_1, E_2, ..., E_n, C_E)$ is an example.
A_i, E_i, C_R, and C_E are bags-of-words, possibly empty.
Output: R' is the generalized rule.
Function Most_Specific_Generalization (R, E)
For $i := 1$ to n do
$A_i{}' := A_i \cap E_i$
$R' := (A_1{}', A_2{}', ..., A_n{}', C_R \cap C_E)$
Return R'.

Figure 19: Generalization of a rule to cover an example

the document, the more important it is (*term frequency* or *tf factor*). The other assumption is that the more it appears through the entire collection of documents, the less important it is since it does not characterize the particular document well (*inverse document frequency* or *idf factor*). In the TFIDF framework, the weight for term t_j in a document d_i, w_{ij} is defined as follows.

$$w_{ij} = tf_{ij} \times \log_2 \frac{N}{n} \tag{9}$$

where tf_{ij} is the frequency of term t_j in document d_i, N the total number of documents in collection, n the number of documents where term t_j occurs at least once, and $\log_2 \frac{N}{n}$ corresponds to the inverse frequency measure.

The standard TFIDF-weighted cosine metric is used to compute the similarity of two BOWs. The *similarity* of two examples (i.e. extracted documents) or rules is the average similarity of the BOWs in their corresponding slots.

4.2.2 Algorithm

Bag intersection is used to compute the minimal generalization of two BOWs. For instance, the generalization of the `title` slot in Figure 18 and that of "Harry Potter and the Prisoner of Azkaban (Book 3)"(`Title` = {"harry", "potter", "prisoner", "azkaban", "book"}) is a simple intersection of those two, which is `Title` = {"harry", "potter", "book"}. The minimal generalization of two examples or rules is the minimal generalization of the BOWs in each of their corresponding slots.

RISE finds the nearest example to generalize a given rule, satisfying two requirements: that the example should not be already covered by that rule, and the class assigned to the example is the one predicted by that rule. Since TEXTRISE does not learn simple categorization rules, the second requirement must be changed: the similarities between the slot-filler of an example and the consequent of the rule should be maximized. To combine this requirement with the goal of the original task which is to find the nearest example of a given rule, we calculate the similarity between each examples and the given rule to find an example with minimal distance to the rule.

A rule is said to *cover* an example if all of its antecedents are subbags of the example's corresponding BOWs. To extend the algorithm from classification to text prediction, we define

Input: ES is the training set.
Output: RS is the rule set.
Function TextRISE (ES)
$RS := ES$.
Compute $TextAccuracy(RS, ES)$.
Repeat
  For each rule $R \in RS$,
    $\hat{E} := \underset{E \in ES'}{arg\,max}\ Similarity(E, R)$
      where $ES' = \{E'$: $E' \in ES$ and
        E' is not covered by $R\}$
    R' := Most_Specific_Generalization(R, $\hat{E}$)
    RS' := RS with R replaced by R'
    If $TextAccuracy(RS', ES) \geq TextAccuracy(RS, ES)$
    Then $RS := RS'$
    If R' is identical to another rule in RS,
    Then delete R' from RS.
Until no increase in $TextAccuracy(RS, ES)$ is obtained.
Return RS.

Figure 20: The TEXTRISE rule-learning algorithm

a new measure for the accuracy of a rule set on an example set: $TextAccuracy(RS, ES)$ is the average cosine similarity of the predicted fillers for the examples in ES to the corresponding fillers predicted by a rule set RS. The algorithms for generalizing a rule to cover an example and for learning rules are described in Figure 19 and Figure 20 respectively. The algorithm is a straightforward modification of RISE using the new similarity and predictive-accuracy metrics, and is used to induce soft-matching rules for predicting the filler of each slot given the values of all other slots.

4.2.3 Finding Intra-slot Relationships

Although all the rules shown so far are for prediction of one slot from all other slots, the learning algorithm of TEXTRISE is not restricted to this fixed form. Rules predicting two slots from all the other slots, e.g. "Organism in the book `title` implies life in `synopses` *and* biology in `subjects`.", can be mined without fundamental modifications to the algorithm by slightly changing the representation for rules. This raises the issue of finding intra-slot relationships in addition to the inter-slot relationships we have been exploring. Why not learn a rule with intra-slot specifications such as "Shakespeare in `subject` implies classics and literature in *subject.*"?

We can learn such rules by modifying the representation of examples and rules in TEXTRISE. Specifically, the difinitions for a rule R and an example E are modified as following:

$$R = (A_1, A_2, ..., A_n) \rightarrow (A_1, A_2, ..., A_n)$$
$$E = (E_1, E_2, ..., E_n, E_1, E_2, ..., E_n)$$

where A_i and E_i are BOWs for each slot, possibly empty. With this representation, all slots are predicted from all slots while only one slot is predicted from all the other slots in the original version. In this version of TEXTRISE, a rule covers an example if all of its slots are more general than the example's corresponding slots ($A_i \subseteq E_i$ for each i).

Rules produced by this algorithm specify relationships between terms implicitly. For instance, a rule of "`subject` = {"shakespeare", "classics", "literature"}" can be interepreted as ""Shakespeare", "classics", and "literature" frequently occur together in the `subject` slot". This version of TEXTRISE works like clustering algorithms because the resultings rules are generalizations of rules that are close to each other. Each rule produced by this algorithm can be viewed as a *centroid* of a cluster consisting of all examples closer to that rule than any other rules.

4.3 Interestingness Measures

The output of TEXTRISE is an unordered set of soft matching rules. Ranking rules based on an interestingness metric can help a human user focus attention on the most promising relationships. Several metrics for evaluating the "interestingness" or "goodness" of mined rules, such as *confidence* and *support*, have been proposed (Bayardo Jr. & Agrawal, 1999).

However, the traditional definitions for confidence and support assume exact matches for conditions. For instance, the support of a rule "C∈languages → WindowsNT∈platform" is defined as the number of examples in the database in which `C` ∈ `languages` and `WindowsNT` ∈ `platform` occur together. By this definition, an example with `C` ∈ `languages` and `WinNT` ∈ `platform` is not counted when support and confidence are computed even though `WindowsNT` and `WinNT` could be treated as a unique term based on high similarity of those terms. Consequently, we modify two common metrics, confidence and support, for judging the goodness of the soft matching rules.

A *rule* consists of two conditions called antecedent and consequent, and is denoted as $A \rightarrow C$ where A is equal to $A_1 \wedge A_2 \wedge ... \wedge A_i$. The *similarity-support* of an antecedent A, denoted as $simsup(A)$ is the number of examples in the data set, that is to be soft-matched by A. In other words, $simsup(A)$ is the number of examples to which A is the closest rule in the rule base. The similarity-support of rule $A \rightarrow C$, denoted as $simsup(A \rightarrow C)$, is defined as the sum of similarities between C and the consequents of the examples soft-matched by A in the data set. In these definitions, we replace the traditional hard-matching constraints for a rule with weaker constraints determined relative to all the other rules in the rule base. Similarity-confidence of a rule $A \rightarrow C$, denoted by $simconf(A \rightarrow C)$, is computed as below.

$$simconf(A \rightarrow C) = \frac{simsup(A \rightarrow C)}{simsup(A)}$$

These measures are used to rank the rules generated by TEXTRISE to show users more interesting rules first. Users can specify the minimum value of similarity support(confidence) for rules to be displayed in order to filter out rules with limited coverages(accuracies) by setting a cutoff level, although a rule pruning mechanism based on minimum confidence or support is not incorporated in the rule learning algorithm of TEXTRISE as in association rule mining.

4.4 Experimental Results

To conduct preliminary experiments on text, we implemented TEXTRISE using the BOW library (McCallum, 1996) for the bag-of-words text processing.

4.4.1 The Domains

Two domains are employed in our evaluation of TEXTRISE: book data from `Amazon.com` and patent data downloaded from the publicly available US patent database, `Getthepatent.com`.

The book data-set is composed of 6 subsets, science fiction, literary fictions, mystery, romance, science, and children's books. 1,500 titles were randomly selected for each genre to make the total size of the book data-set to be 9,000. From the web pages downloaded for these titles, a simple hand-built pattern-based information extraction system or "wrapper" extracts 6 slots, `titles`, `authors`, `subject` terms, `synopses`, published `reviews`, and customer `comments`. This data-set was originally constructed for LIBRA, a content-based book recommending system (Mooney & Roy, 2000). Other slots such as related titles, related authors, publisher, date, ISBN, price are also extracted, but these are not used by TEXTRISE.

3,000 AI-related patent data are collected from dynamically generated web pages returned by a keyword search ("artificial intelligence"). Four slots of `titles`, `abstracts`, `claims`, and `descriptions` are identified in the collection of patent data. Patent documents also have slots for application numbers and names of assignee, attorney, and examiner, but they are discarded in our experiments.

The text extracted for each slot is then processed into a BOW after removal of stop-words and remaining HTML commands. Documents represented as vectors of BOWs for each slot constitute a set of training examples for TEXTRISE.

4.4.2 Results

With initial experiments with TEXTRISE, we discovered rules such as "If the `title` of a science-book has a substring of "gender", then "men", "women", and "differences" are found in the `review` for that book." and "If the `title` and the `synopses` of a book contains "Confucius", then the `subjects` of that book includes "philosophy"." More sample rules discovered in this domain are given in Figure 21. The number associated to each word denotes the number of occurrences in the bag. The similarity-confidence and similarity-support values for each rule are also given.

Figure 22 shows prediction rules for the `title` and the `abstracts` slot for patent data. For patent documents, the `claims` and the `description` slots generally contain much longer text than the `title` and the `abstracts` slot.

Sample rules produced from the modified version of TEXTRISE (Section 4.2.3) are presented in Figure 23. These rules are learned from 1,000 documents for children's books. For example, the first rule does not only say "american" in the `title` slot implies "native", "american" and "present" in `synopses`, but it also means "children", "native" and "american" in the `subject` slot often appear with "north", "america", and "indians".

title	nancy(1), drew(1)	
synopses	nancy(1)	
subject	children(2), fiction(2), mystery(3), detective(3), juvenile(1), espionage(1)	[49.71%, 1.99]
→		
author	keene(1), carolyn(1)	
synopses	role(1), protein(1), absorption(1), metabolism(4), vitamins(1), minerals(1)	
reviews	health(1)	[27.87%, 0.56]
subject	science(1), human(1), physiology(1)	
→		
title	nutrition(1)	
author	beatrice(1), gormley(1)	
synopses	witness(1), ufo(1), landing(1)	
subject	science(1), fiction(2)	[13.79%, 0.34]
→		
reviews	aliens(1), ufo(1), book(2)	
title	charlotte(1), perkins(1), gilman(1)	
synopses	work(1), utopias(1), herland(1), ourland(1)	
reviews	gilman(1), author(1)	
subject	literature(2), criticism(2), classics(1), women(1), literary(1)	[36.46%, 0.73]
→		
comments	utopia(1), feminist(1)	
title	dance(1)	
→		[31.68%, 0.98]
subject	romance(2), fiction(2)	

Figure 21: Sample rules from 2,500 book descriptions

As stated in Section 4.2.3, this version of TextRISE works like a clustering algorithm and the resulting rules represent centroids of clusters found. It becomes clear when we apply it to a collection of book descriptions documents with all the genres. Many of the rules learned from 6,000 book descriptions, 1,000 for each genre, put constraints only on the `subject` slot, e.g. "`subject` = {"children", "juvenile", "fiction"}", "`subject` = {"literature", "classics", "criticism"}", "`subject` = {"fiction", "mystery", "detective"}", or "`subject` = {"engineering", "science"}". The `subject` slot fillers can be viewed as pre-assigned labels for each cluster in the collection, and the output of this version resembles the one obtained from a clustering algorithm.

4.4.3 Evaluation

Unlike a standard rule learner that predicts the presence or absence of a specific slot value, TextRISE predicts a BOW for each slot. Therefore, we evaluate the performance of TextRISE by measuring the average cosine similarity of the predicted slot values to the actual fillers for each

abstract	device(2), images(1)	
claims	invention(4), system(5)	
description	information(5), data(2), control(2), stored(2), point(1), user(3), application(2), flow(1), objects(1), operation(1), software(1), storage(1)	[9.44%, 0.54]
$\rightarrow$		
title	video(1)	
title	automated(1)	
claims	device(4), based(1), determining(3), comprising(4), input(3), plurality(2), comprises(5), claim(7)	[7.25%, 0.42]
$\rightarrow$		
abstract	apparatus(1)	

Figure 22: Sample rules from 3,000 patent documents

author	**title**	**synopses**	**review**	**comments**	**subject**
	american(1)	native(1) american(1) present(1)			children(1) native(1) american(1) north(1) america(1) indians(1)
	space(1) camp(1)	space(1) shuttle(1) astronauts(1) program(1)	space(1) nasa(1)		children(1) science(1) technology(1) nonfiction(1) juvenile(1)
lynne(1) reid(1) banks(1)		life(1) family(1)	life(1) family(1) dramatic(1)	parents(1) living(1) indian(1) reading(1) wonderful(1) children(1)	children(2) fiction(1) juvenile(1)

Figure 23: Sample rules from 1,000 book description

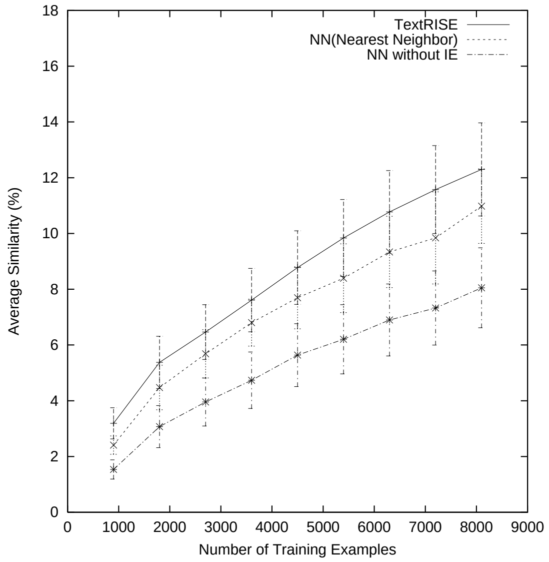


Figure 24: Average similarities for book data (`title`)

slot. We compare the system to a standard nearest-neighbor method to show that TEXTRISE's compressed rule base is superior at predicting slot-values. In both methods, prediction is made by selecting the closest rule/example using only the text in the antecedent slots. We also tested nearest-neighbor without using information extraction to show the benefit of IE-based text mining. To clearly show IE's role, the only change made to nearest-neighbor was to treat the set of BOW's for the antecedent slots as a single, larger BOW.

The experiments were performed on the 9,000 book descriptions using ten-fold cross validation. Learning curves for predicting the `title` slot are shown in Figure 24. The graph shows 95% confidence intervals for each point. All the results on average similarities, precisions, and F-measures were statistically evaluated by a one-tailed, paired t-test. For each training set size, two pairs of systems (TEXTRISE versus nearest-neighbor and nearest-neighbor versus nearest-neighbor without information extraction) were compared to determine if their differences were statistically significant ($p < 0.05$).

The results indicate that TEXTRISE does best, while nearest-neighbor without IE does worst. This shows TEXTRISE successfully summarizes the input data in the form of prediction rules. The rule-compression rate of TEXTRISE is about 80%, which means the number of rules TEXTRISE produces is 80% of the number of examples originally stored in the initial rule base. We conducted the same experiments for other slots, and found similar results except for predicting the `author` slot. In predicting the `author` slot, neither information extraction nor TEXTRISE improves performance over simple nearest-neighbor.

In addition to the textual similarity metric, we developed analogs for precision and recall. Precision and recall were defined as follows, where C is the correct BOW and P is the predicted

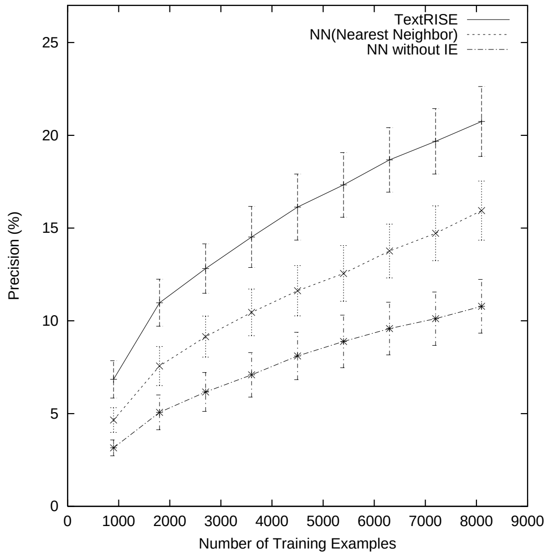


Figure 25: Precisions for book data (`title`)

one.

$$Precision = Similarity(C \cap P, P) \tag{10}$$

$$Recall = Similarity(C \cap P, C) \tag{11}$$

F-measure is defined as the harmonic mean for precision and recall by Equation 3.

Learning curves for precision and F-measure are presented in Figure 25 and Figure 26. TEXTRISE provides higher precision, since the conclusions of many of its rules are smaller generalized BOWs, and overall F-measure is moderately increased.

To compare TEXTRISE with traditional rule mining methods, we generated association rules using the APRIORI algorithm (Agrawal & Srikant, 1994) and a publicly available implementation (Borgelt, 2000). We treated each word in each slot as a separate item and generated associations between them. Among all the generated rules, those with words for the slot to be predicted are selected. For each test example, a prediction is made by building a BOW using the conclusions of all matching association rules. With the default parameter setting (minimum support of 10% and minimum confidence of 80%), the average similarity of predictions is almost 0%. We lowered the minimum support and the confidence until memory problems occurred on a SUN Ultra Sparc 1 (120MB). With the lowest minimum setting for support (3%) and confidence (30%), the average similarity remains very low: 0.0005% for 900 training examples and 0.0014% for 8,100 training examples. These results strongly suggest the usefulness of soft-matching rules in prediction tasks for textual data.

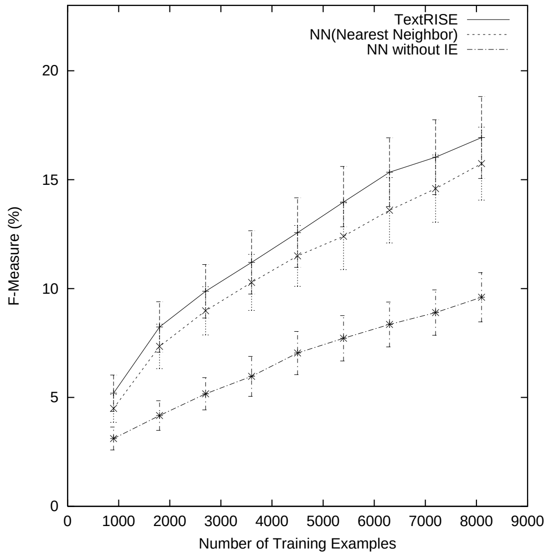


Figure 26: F-measures for book data (`title`)

5 Future Research

We plan to address a number of issues in future research. These fall into three primary areas. First we plan to do more extensive evaluations of our system on additional domains. There are also several enhancements which we plan to make to the current system. Finally, we intend to explore the applicability of our text mining framework with information extraction to other text mining tasks. The first two of these areas are more definite future works we plan to examine soon, while the last one is the longer-term issue. Each of the three areas of future research is discussed in some detail below.

5.1 Full Evaluation

Currently, TextRISE is applied on two domains, book descriptions and patent documents for preliminary experimentations. We plan to evaluate TextRISE on additional domains in comparison with other methods such as standard rule induction algorithm.

5.1.1 Additional Domains

Although our preliminary results with two domains are encouraging, a fuller evaluation will apply our system to larger corpora and to other realistic domains in order to demonstrate that our text mining framework is sufficiently general and easy to be applied to multiple domains. First, we plan to download more patent documents and construct a larger data-set to test our system on. Currently we have only 3,000 patent documents about Artificial Intelligence.

For other domains, business-related documents, medical records, and research texts are considered. For instance, we plan to use the DiscoTEX framework to discover knowledge from a database extracted from business news articles. Mining a database constructed from doctors' English notes or records about medical patients is another promising application area. We are also considering e-mail exchanges related to product orders, abstracts of National Science Foundation (NSF) grant proposals, and a collection of more than 50,000 computer science research papers gathered by Cora search interface (`www.cora.justresearch.com`) (McCallum et al., 2000) or ResearchIndex (`citeseer.nj.nec.com`), for additional domains.

5.1.2 Comparison with Other Methods

We intend to further enhance our system as will be discussed in Section 5.2 and compare its performance to other methods such as standard rule induction algorithm (C4.5rules (Quinlan, 1993) or Ripper (Cohen, 1995)) and probabilistic classifier (Naive-Bayes (Mitchell, 1997)). As TextRISE predicts a BOW from a set of BOWs, we must apply these classification-driven techniques to all terms that appear in the conclusion part of the TextRISE rules. To make the rule induction process computationally feasible, terms might need to be selected based on frequencies from the dictionary of all terms seen in the training examples.

For the first step of comparing the accuracy of TextRISE rules with that of Ripper rules, a dictionary of the entire terms seen in the predicted slots of TextRISE rules are constructed. For each term in the dictionary, we learn a set of classification rules for predicting the presence or the absence of the corresponding term from the same training set. Since we believe the best evaluation metric is similarity of actual and predicted text, we cast the problem of simple prediction of presence/absence of each terms in slots to measuring similarity so that an experiment comparing TextRISE and other rule induction methods can be conducted.

5.2 Enhancing the Current Algorithm

The version of DiscoTEX and TextRISE described thus far are preliminary implementations, and we plan several enhancements on them.

5.2.1 Alternative Generalization Schemes

A potential extension of TextRISE is to replace the current generalization scheme, taking the intersection of two BOWs. First, we could generalize to a k-nearest-neighbor method that uses the k closest rules or examples rather than just the single nearest one. The predictions of these k rules could be combined by taking the average of the BOW vectors in their consequents. Likewise during learning, rules could be generalized to the k nearest uncovered examples using a similar averaging technique, possibly rounding values to maintain integer counts and simplify the resulting rules.

Another option is to use the WordNet (Fellbaum, 1998) hierarchy of *hypernyms* to generate generalizations that takes semantics into account. For example, the two rules:

- thermodynamics $\in$ *subject* $\rightarrow$ heat, modern, theory, waves $\in$ *synopses*
- optics $\in$ *subject* $\rightarrow$ electromagnetics, laser, waves $\in$ *synopses*

might be minimally generalized to the rule

- physics $\in$ *subject* $\rightarrow$ waves $\in$ *synopses*

if a semantic lexicon provided by WordNet is utilized. "Thermodynamics" and "optics" have the common parent, "physics", in the hypernym tree. The current implementation of TEXTRISE generates an empty filler slot for the `subject` slot in this case because it is not able to recognize the semantic relationships between the two slot fillers. This change requires a redefinition of distance between words in terms of the WordNet hierarchy.

5.2.2 Alternative Similarity Metrics

Alternative metrics for measuring textual similarity will also be explored. For short extracted strings, string edit distance (Wagner & Fisher, 1974) might be a more useful measure of textual similarity than the cosine measure. String edit distance can be calculated by dynamic programming using different costs associated with various transformation rules (McCallum et al., 2000).

5.2.3 Ranking Rules based on Interestingness

Even though we currently rank rules before presenting them to users, this task bears further investigation. The interestingness measures we used currently, confidence and support, are common measures in the KDD community. However, the quality or goodness of rules specific for textual databases could be defined in different ways. For instance, trivial rules such as "If the `title` of a book contains "physics", both the `synopses` and the `subject` have "physics", too." are very often found in the rule set generated by TEXTRISE. This kind of rules can be left out via simple identity checking filters.

Another idea is to use a semantic network like WordNet (Fellbaum, 1998) to measure the semantic distance between the words in the antecedent and the consequent of a rule, preferring more "surprising" rules where this distance is larger. For example, this would allow ranking the rule "beer $\rightarrow$ diapers" above "beer $\rightarrow$ pretzels" since beer and pretzels are both food products and therefore closer in WordNet. Although WordNet provides rich information for a given word such as synonyms, antonyms, hypernyms, and meronyms, the interestingness measure might mostly concern synonyms and hypernyms. We could define the semantic distance between two words as the length of the path between those in the WordNet hierarchy.

The interestingness of rules are sometimes very subjective. For instance, a discovered rule, "isaac and asimov in the `author` slot often implies science and fiction in the `subject` slot.", is intriguing to someone who does not know the fact "Isaac Asimov wrote many science fictions", but not to others. To measure this subjective interestingness, we could evaluate rules based on rankings made by human judges. These human ratings could then be compared to various interestingness metrics.

5.2.4 Feature Selection Prior to Learning

Domain-specific dictionaries constructed in the process of building word vectors for document collections can be used to eliminate uninteresting terms before learning. For instance, in the `synopses` slot of book data, many examples contain "table", and "contents", since a "table of contents" is usually included in the synopses of a book. The rules learned by TEXTRISE therefore contain both "table" and "contents" in their `synopses` slots in many cases, although they are neither informative nor interesting. "Copyright" in the `reviews` slot or any words related to books such as "book" or "read" are another typical examples for this problem. They do not provide any clue to the specific book they describe. Although we already used the TFIDF weighting scheme to prefer words that are dominant in particular document, eliminating such domain-specific frequent words spread through a collection of documents could help finding more interesting rules.

5.2.5 Beyond Words

Instead of taking individual words, n-grams could be used as additional terms for the vector representation. When applying TEXTRISE to book data, we found that it recognizes "Juvenile Fiction" and "Science Fiction" in the `subject` slot as a BOW with "juvenile(1)", "science(1)" and "fiction(2)", thus missing useful information which could have been maintained if they are treated as a single term. For example, we could construct a bag-of-uni/bigrams with "juvenile(1)", "science(1)", "fiction(2)", "juvenile-fiction(1)", and "science-fiction(1)" from the same slot-filler.

A related issue is to combine words for a named entity, such as an author's name or a company name. "Stephen" or "King" by themselves in "Stephen King" might not be very useful for mining interesting rules about this author, but "stephen-king" as a whole makes more sense to users looking at this domain. Named entity recognition task, introduced in the MUC-6 (DARPA, 1995), was found to be a relatively easy task because the best system submitted to the competition scored 96.4% in F-measure (business news domain). Automatic extraction of named entities with RAPIER or other machine learning techniques such as (Bikel, Schwartz, & Weischedel, 1999) could be considered for a preprocessing step of building a training set with bags-of-terms.

5.2.6 Automated Slot Selection

The procedure for selecting slots to be used in rule mining will be automated. In the current experiments in Section 3 and 4, we manually chose slots from the templates of computer-science job, books, and patent data. For example, the `title` slot for job postings is not used in the experiments of Section 3.4 because it has many possible values and is difficult to predict. In Section 4, some of the extracted slots such as ISBN or publishing date of a book are not included in rule mining for the same reason.

By computing the correlations between slot values, this decision could be automated. Prior information on the correlations between slot values can be also used for weighting the antecedents slots in TEXTRISE. For instance, if the `reviews` slot turns out not to be useful for predicting the `author` of a book while `title` is very informative, different weights can be put to each slot in analogy of weighting words by their frequencies. One possibility is to apply the mining algorithm

```
<book>
<title>The Complete Works of William Shakespeare</title>
<author>William Shakespeare</author>
<reviews>
   <review> <reviewer> <name>John Doe</name>
                       <email>johndoe@email.com</email>
            </reviewer>
            <rating>3/5</ranking>
            <review_txt>There are pros and cons to buying this.</review_txt>
   </review>
   <review> <reviewer> <name>Jane Doe</name>
                       <email>janedoe@email.com</email>
            </reviewer>
            <rating>5/5</rating>
            <review_txt>His writing is one that can't be compared.</review_txt>
   </review>
</reviews>
</book>
```

Figure 27: An example of the XML document for book description

to a small subset of the training examples, and evaluate the resulting rule base to determine if the predicted slot is viable for further mining or not. In this strategy, some parameters must be empirically tuned, e.g. how many of the training examples are to be used in the pre-mining step and how to set a threshold for slot selection.

5.2.7 Extension of TEXTRISE for Semi-Structured Documents

There is a growing need to handle semi-structured documents written in mark-up languages. The current version of TEXTRISE does not have any special ability to cope with web pages written in HTML or XML (EXtensible Markup Language) (Bray, Paoli, Sperberg-McQueen, & Maler, 2000). Automatically generated web pages in `Amazon` and `Getthepatent` contain HTML tags, but they are simply removed before documents are processed by an IE module. Structural hints given by such tags could be utilized in mining web documents.

One option for this task is to take the XML document structure tree generated by an XML parser as the representation for a document. A shortened example for XML documents for book description is shown in Figure 27. This type of document is easy to convert to a tree structure where the leaf nodes contain BOWs. In that case, the generalization of two set of bags in the current system is replaced by the generalization of two trees of bags. The similarity measures should be also redefined appropriately for this representation.

5.3 Extension to Other Text Mining Tasks

We also believe that additional text mining tasks can be found for which our framework may prove useful. Since IE can be useful for many kinds of text processing, using extracted information for other text mining tasks might be an interesting issue. However, little research has been done in this field. We present here possible extensions of our framework to four other text mining tasks, document clustering, soft association mining, relationship mining, and topic detection and tracking.

5.3.1 Document Clustering

The use of TextRISE and its representation for document clustering task is a natural extension to TextRISE. As mentioned in Section 4.2, the rule-learning algorithm of TextRISE shows a clustering-like behavior.

Clustering algorithms partition a set of objects into k groups (Manning & Schütze, 1999). The goal of clustering algorithms is to put similar objects together in the same group and to place dissimilar objects in different groups. Clustering is often called unsupervised learning because the partition of objects is not known unlike the classification problem. Hierarchical clustering with bottom-up generalization is called agglomerative clustering. Agglomerative clustering begins with the separate clusters for all the objects, and continues to merge similar clusters until it reaches to k clusters. Clustering for text documents or web pages can be useful for many applications such as thesaurus construction and browsing by providing an overview of the content of a large document collection.

Except that TextRISE learns rules with associated conclusions, the agglomerative clustering is essentially similar to the learning algorithm of TextRISE. An alternative view of TextRISE as a clustering algorithm could be examined.

5.3.2 Soft Association Mining

Since the idea of finding association such as "A customer who buys `bread` will also buy `milk` with probability 80%."originated in the analysis of market-basket data, association rule mining has been a popular mining method. In this proposal, we proposed a text-mining system for prediction rules by applying both standard rule-learning methodologies (Section 3) and a hybrid one (Section 4). However, an association-rule mining algorithm can also discover *frequent item sets* from a collection of documents. Despite the fact that the problem of finding frequent item sets is known to be NP-complete, researchers have developed algorithms for this problem that work well in practice. However, existing association-rule mining algorithms are not tailored to non-standardized textual databases since they assume a "clean" databases. The use of hard-matching association rule discovery methods such as Apriori (Agrawal & Srikant, 1994) could be further investigated and the soft matching mechanism used in TextRISE could be incorporated to the existing association rule mining algorithms. Specifically, we believe the join step of the Apriori algorithm can be replaced with a "soft join" step based on similarity operators such as the one used in Whirl (Cohen, 1998).

5.3.3 Relationship Mining

Locating more complicated relations between entities is another possible extension. Currently our systems only find simple relationships, e.g. frequency of co-occurrence. One option for mining other types of relationship is to discover *causal relationships* among items. Instead of merely saying that the presence of A implies the presence of B, causal relationships determine whether the existence of A causes the existence of B, vice versa, or the other item C causes both to occur together.

There is ongoing research in Bayesian analysis for causality detection. Silverstein, Brin, Motwani, and Ullman (2000) applied constraint-based causal discovery techniques (Cooper, 1997) to market-based data with boolean features. Using the same Bayesian learning framework, we might be able to extend our system to discover not only associations but also causal relationships from text documents. Even though Silverstein et al. (2000) analyzed news articles to find causal relationships between words, they simply treated each article as an entry (or a *basket* in market-based data mining terminology) in a database, and each word as an item. Since the size of Bayesian structure to be searched for determining the presence or absence of causality relations is huge, we believe our IE-based compression of textual information could be useful for practical discovery of causality in a large corpora of text.

5.3.4 Topic Detection and Tracking

Another appropriate application for the DiscoTEX framework might be Topic Detection and Tracking (TDT). TDT, or novelty detection, is a variant of traditional document classification that allows new classes over a time-series text corpora. Unlike query-based retrieval task or simple categorization task, users do not know in advance what they want to retrieve or what categories there are. Information retrieval and machine learning techniques have been used to identify new events from streams of articles, concentrating on news stories (Yang, Carbonell, Brown, Pierce, Archibald, & Liu, 1999).

As in TopCat (Clifton & Cooley, 1999), we consider a TDT task in a data mining context: preprocessing a collection of articles for identifying key concepts in individual document, and applying data mining techniques such as frequent item set generation and clustering. Frequent item sets, defined as all item sets that often occur together, identify correlated topics while clustering tracks natural boundaries for neighboring topic periods to detect changes of topic items over time. TopCat uses a named entity recognizer which is limited to extracting information related to "who?" and "where?" questions (e.g. `location`, `organization`, and `person`). We could tag news articles, which is a typical kind of time-series text corpora, using information extraction, and to apply TextRISE to the textual databases constructed by IE. An example of an entry for such databases is shown in Figure 28. This example is from a CNN article of November 2000 and the title of it is "Bush maintains slim lead over Gore as Florida count continues".

Subject	Politics Presidential Election
Keyword	Ballot Recount
Date	November 10, 2000
Location	Tallahassee, Florida
Related-Location	Austin, Texas Palm Beach County
Person	Al Gore George W. Bush Katherine Harris Pat Buchanan Warren Christopher James Baker
Title	Vice President Texas Governor Florida Gov. Secretary of State

Figure 28: An example for set of topics identified by IE: from a CNN article

6 Related Work

There has been relatively little research exploring the use of Information Extraction in text mining. In this section, we will present earlier works devoted to using existing data mining techniques on unstructured text and to mining knowledge from the Web, and explain our novelty in using IE for the task of knowledge discovery from text. We briefly review the use of information extraction in other natural language processing tasks such as text summarization, building a translingual information access system, and question-answering. Soft matching used in the rule-learning algorithm of TextRISE has barely been applied to text-mining systems either. Whirl (Cohen, 1998) with soft matching operators is described to show its similarities and differences with TextRISE.

6.1 Using Data Mining Techniques on Text

The most relevant system to our approach might be KDT (Knowledge Discovery in Textual Databases) (Feldman & Dagan, 1995). KDT alludes to the use of IE in text mining; however, it uses texts manually tagged with a limited number of fixed category labels instead of actually using automated text categorization or IE. Similarly, FACT (Feldman & Hirsh, 1996), which finds associations amongst keywords from text documents, does not have an automated routine to label the documents with keywords, which can be viewed as the basic level of information extraction. Document Explorer extracts terms to label a document in a more automatic manner, but it is still restricted to highlighting selective terms based on predetermined syntactic patterns such as "noun-noun" or "adjective-noun" (Feldman et al., 1998). Recently Loh, Wives, and de Oliveira

(2000) followed up KDT by suggesting a concept-based knowledge discovery from the Web. In this research, concepts are identified through a text categorization algorithm for the purpose of listing key-concepts and finding correlation between concepts. One of the limitations for these approaches is that they require a substantial amount of background knowledge provided by a domain expert in advance.

Text analysis tasks such as descriptive phrase extraction and co-occurring phrase extraction have been conducted by applying existing data mining techniques to text (Ahonen et al., 1998; Ahonen-Myka et al., 1999). In these works, frequent set of items (i.e. words) are located, but no preprocessing of texts to be analyzed is employed except simple morphological tagging. Another approach related to linguistic issues is a method used to discover semantic relationships between terms in a collection of documents (Finkelstein-Landau & Morin, 1999). In their "term level text mining", a natural-language parser is used in place of information extraction since the extraction process is followed by finding syntactic or semantic relations between the extracted terms. Learned rules by these systems are hard matching rules unlike those produced by TEXTRISE.

6.2 Web Mining

Information Retrieval (IR) from the Web, e.g. building Web search engines, has been a hot topic particularly given the growing number of web documents. However, more organized and deeper understanding of the text documents such as discovery of prediction rules from text is not a much studied area yet in the field of web mining. Web mining beyond search technology is lately beginning to draw great attentions in the intersection of various research fields. Recent web mining systems have been focusing on either information extraction itself (Soderland, 1997, 1999), mining simple relationships such as acronyms or author-book pairs (Brin, 1998; Larkey, Ogilvie, Price, & Tamilio, 2000) traditional text categorization tasks (Weiss, Apté, Damerau, Johnson, Oles, Goetz, & Hampp, 1999), event detection/tracking (Yang et al., 1999) or trends discovery (Lent, Agrawal, & Srikant, 1997). None of these works use explicit information extraction in a general way for further utilization of the extracted concepts.

Etzioni (1996) discusses applying data mining techniques to web resources available on the Internet. It identifies the significance of using information extraction in building a web mining system with an emphasis on the scalability problem. However, information extraction systems surveyed in this article are very application-oriented and domain-specific, e.g. extraction of answers for frequently asked questions (FAQ) in FAQ-FINDER (Hammond, Burke, Martin, & Lytinen, 1995) or extraction of product information from web vendors for a shopping agent, SHOPBOT (Doorenbos, Etzioni, & Weld, 1997). We proposed a more generalized framework incorporating domain-independent information extraction and standard data mining methodologies.

Ghani et al. (2000) applied several data mining methodologies to a knowledge base constructed from the Web as a part of the WEBKB project (Craven et al., 2000). Data mining techniques used by this system include finding association rules (APRIORI), inducing decision trees (C5.0), and learning rules with FOIL. Interesting regularities such as "Advertising agencies tend to be located in New York" are discovered from a knowledge base about corporations; however, such learned rules must exactly match extracted text.

6.3 Use of IE for Another Natural Language Processing Tasks

Several natural language processing systems use information extraction. One of those earlier efforts is found in (Riloff, 1996a). It uses a dictionary of extraction patterns, learned originally for IE, to classify text documents. This can be viewed as an indirect use of IE, since it does not employ a full IE system as a component for a larger system but utilizes by-products of an IE learning process.

Information extraction has been considered as a front-end of text summarization or abstracting system consisting of sentence extraction and summary generation modules. Radev and McKeown (1998) developed a text summarization system called SUMMONS by combining a message understanding system which can be viewed as an IE component with a postprocessing module, summary generator. It demonstrated that filled IE templates for terrorism domain can be helpful for generating abstracts for news articles given successful postprocessing on them. More recently, IE began to be used for applications such as machine translation (MT) or question-answering (QA). White, Cardie, Han, Kim, Lavoie, Palmer, Rambow, and Yoon (2000) helps analysts perform information filtering tasks on foreign language documents, by making IE techniques based on English transferable to other languages. AutoSlog (Riloff, 1993) is combined with a machine translation system to develop an English information access gateway to newspapers published electronically in foreign countries. QA is another complicated task consisting of understanding questions, locating possible answer from databases (entire Web if view the Web as an immense knowledge base possibly containing answers for every imaginable question), and presenting the most reasonable answer. Textract (Srihari & Li, 1999), presented in the QA track of the TREC (Text REtreival Conference)-8 test, answers for natural language questions such as "Who won the 2000 Nobel Peace Prize?" by combining a named entity recognizer with other necessary components for QA, e.g. question processors and answer search engines.

Although all of the above research works acknowledged the use of information extraction as an essential component for doing other natural-language understanding tasks, none of these concerned on an important aspect of IE, constructing structured textual databases from raw text, for use in text mining.

6.4 Soft Retrieval

The TextRISE approach introduced in this proposal uses a flexible matching mechanism both in its rule-learning algorithm and in its classification scheme. Whirl is a logic that combined traditional databases with retrieval methods from information retrieval, by introducing a "soft join" operation (Cohen, 1998). In Whirl, all information is assumed to be represented in a relational model in which every element of every tuple contains free text.

As an example, Figure 29 shows two relational databases of free text, one containing book listings, and the other containing book reviews. Whirl is able to "soft join" these two relations by:

$$\begin{aligned}&bookListing\ (Book1,\ Author,\ SalesRank)\\&\wedge\ bookReview\ (Book2,\ Review)\\&\wedge\ Book1\ \sim\ Book2\end{aligned}$$

Book	Author	Sales Rank
Machine Learning	Tom Mitchell	58,454
Statistical Learning Theory	Vladimir Vapnik	38,619
Reinforcement Learning	Richard S. Sutton	43,767

Book	Review
Neuro-Dynamic Programming, 1996	Devoted to the theory that explores...
Reinforcement Learning: An Introduction, 1998	This is a ground breaking work, dealing with a subject that you would have expected...
Bioinformatics, 1998	This book covers the field of Bioinformatics,...

Figure 29: Data representation in WHIRL

where $\sim$ is a similarity operator. This query binds two given relations, in a preference to the binding with the most similar pair of $Book1$ and $Book2$. In this example, "Reinforcement Learning" and "Reinforcement Learning, An Introduction, 1998" can be considered *similar*. As a result of the join operation, separate information about this book (`author` and `sales-rank` in *bookListing* and `review` in *bookReview*) are able to be combined together.

WHIRL and TEXTRISE share a focus on soft-matching rules for text processing; however, rules in WHIRL must be written by the user while TEXTRISE tries to discover such rules automatically.

7 Conclusion

There is a growing interest in the general topic of knowledge discovery from text (Hearst, 1999; Feldman, 1999; Mladenić, 2000); however, there are few working systems or detailed experimental evaluations. By utilizing existing Information Extraction (IE) and KDD (Knowledge Discovery from Databases) technology, text-mining systems can be developed relatively rapidly and evaluated on existing IE corpora.

In this proposal, we presented an approach to using an automatically learned IE system to extract a structured databases from a text corpus, and then mining this database with traditional KDD tools. Our preliminary experimental results demonstrate that the knowledge discovered from such an automatically extracted database is close in accuracy to the knowledge discovered from a manually constructed database. In addition to showing our system performs quite well with a relatively small amount of human-labeled data, we also demonstrated the integration of information extraction and data mining can be mutually beneficial for both tasks. IE enables the application of KDD to unstructured text corpora and KDD can discover predictive rules useful for improving IE performance.

Existing text-mining systems discover rules that require exactly matching substrings; however, due to variability and diversity in natural-language data, some form of soft matching based on textual similarity is needed. We have also presented a system called TEXTRISE that uses a hybrid of rule-based and instance-based learning methods to discover soft-matching rules from textual databases automatically constructed from document corpora via information extraction. With encouraging results of preliminary experiments, we showed how this approach can induce accurate predictive rules despite the heterogeneity of automatically extracted textual databases.

Text mining is a relatively new research area at the cross road of databases, natural-language processing, machine learning, and information retrieval. By appropriately integrating techniques from each of these disciplines, useful new methods for discovering knowledge from large text corpora can be developed. In particular, we believe that the growing interaction between computational linguistics and machine learning (Cardie & Mooney, 1999) is critical to the development of effective text-mining systems.

A A Brief Theory of Bags

Bag theory is a natural extension of set theory. A *bag* is similar to a set in that it is defined as a collection of elements over a domain. However, unlike a set, bags allow *multiple* occurrences of elements. For an element x and a bag B, we denote the number of occurrences of x in B by $count(x, B)$.

An element x is a *member* of a bag B if $count(x, B) > 0$. This is denoted $x \in B$. Similarly, if $count(x, B) = 0$, then $x \notin B$. The empty bag, $\emptyset$, has no members: For all x, $count(x, \emptyset) = 0$. The *cardinality* $|B|$ of a bag B is the total number of occurrences of elements in the bag:

$$|B| = \sum_x count(x, B)$$

A bag A is a *subbag* of a bag B if every element of A is also an element of A at least as many times:

$$A \subseteq B \text{ iff } count(x, A) \leq count(x, B) \text{ for all } x$$

Two bags A and B are equal ($A = B$) if $count(x, A) = count(x, B)$ for all x. Operations for sets such as union, intersection, sum, and difference can be redefined for bags.

$$\text{Union: } count(x, A \cup B) = max(count(x, A), count(x, B))$$
$$\text{Intersection: } count(x, A \cap B) = min(count(x, A), count(x, B))$$
$$\text{Sum: } count(x, A + B) = count(x, A) + count(x, B)$$
$$\text{Difference: } count(x, A - B) = count(x, A) - count(x, A \cap B)$$

Let $D = \{a, b, c, d, e\}$ be a domain. Then for the following bags,

$$A = \{a, b\}$$
$$B = \{a, a, b, c, e\}$$
$$C = \{a, a, a, c, c, e\}$$

we have

$$
\begin{gathered}
|A| = 2 \\
|C| = 6 \\
A \cap B = \{a, a, b, c\} = A \\
A \cup C = \{a, a, a, b, c, c, e\} \\
A \cap C = \{a\} \\
B \cap C = \{a, a, c, e\} \\
A + B = \{a, a, a, b, b, c, e\} \\
A - B = \emptyset
\end{gathered}
$$

Acknowledgements

This research was supported by the National Science Foundation under grant IRI-9704943.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases(VLDB-94)*, pp. 487–499 Santiago, Chile.

Ahonen, H., Heinonen, O., Klemettinen, M., & Verkamo, A. I. (1998). Applying data mining techniques for descriptive phrase extraction in digital document collections. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries*, pp. 2–11 Santa Barbara, CA.

Ahonen-Myka, H., Heinonen, O., Klemettinen, M., & Verkamo, A. I. (1999). Finding co-occurring text phrases by combining sequence and frequent set discovery. In Feldman, R. (Ed.), *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99) Workshop on Text Mining: Foundations, Techniques and Applications*, pp. 1–9 Stockholm, Sweden.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval.* ACM Press, New York.

Bayardo Jr., R. J., & Agrawal, R. (1999). Mining the most interesting rules. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pp. 145–154 San Diego, CA.

Berger, A., & Mittal, V. O. (2000). Ocelot: A system for summarizing web pages. In *Proceedings of 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2000)*, pp. 144–151 Athens, Greece.

Bikel, D. M., Schwartz, R., & Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine Learning*, *34*, 211–232.

Borgelt, C. (2000). Apriori version 2.6. http://fuzzy.cs.Uni-Magdeburg.de/~borgelt/.

Bray, T., Paoli, J., Sperberg-McQueen, C. M., & Maler, E. (2000). EXtensible Markup Language (XML) 1.0 (second edition). http://www.w3.org/TR/2000/REC-xml-20001006.

Brill, E. (1994). Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 722–727 Seattle, WA.

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)*, pp. 172–183 Valencia, Spain.

Califf, M. E. (1998). *Relational Learning Techniques for Natural Language Information Extraction.* Ph.D. thesis, Department of Computer Sciences, University of Texas, Austin, TX. Also appears as Artificial Intelligence Laboratory Technical Report AI 98-276 (see http://www.cs.utexas.edu/users/ai-lab).

Califf, M. E. (Ed.). (1999). *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, FL. AAAI Press.

Califf, M. E., & Mooney, R. J. (1999). Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 328–334 Orlando, FL.

Cardie, C. (1997). Empirical methods in information extraction. *AI Magazine*, *18*(4), 65–79.

Cardie, C., & Mooney, R. J. (1999). Machine learning and natural language (Introduction to special issue on natural language learning). *Machine Learning*, *34*, 5–9.

Clifton, C., & Cooley, R. (1999). TopCat: Data mining for topic identification in a text corpus. In *Proceedings of Third European Conference of Principles and Practice of Knowledge Discovery in Databases(PKDD-99)*, pp. 174–183 Prague, Czech Replublic.

Cohen, W. W. (1995). Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)*, pp. 115–123 San Francisco, CA.

Cohen, W. W. (1996a). Learning to classify English text with ILP methods. In De Raedt, L. (Ed.), *Advances in Inductive Logic Programming*, pp. 124–143. IOS Press, Amsterdam.

Cohen, W. W. (1996b). Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 709–716 Portland, OR.

Cohen, W. W. (1998). Providing database-like access to the web using queries based on textual similarity. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD-98)*, pp. 558–560 Seattle, WA.

Cohen, W. W. (1999). What can we learn from the web?. Invited talk, The Sixteenth International Conference on Machine Learning (ICML-99).

Cooper, G. (1997). A simple constraint-based algorithm for efficiently mining observational databases for causal relationships. *Data Mining and Knowledge Discovery*, *1*(2), 203–224.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A. K., Mitchell, T., Nigam, K., & Slattery, S. (2000). Learning to construct knowledge bases from the World Wide Web. *Artificial Intelligence*, *118*(1-2), 69–113.

DARPA (Ed.). (1993). *Proceedings of the Fifth DARPA Message Understanding Evaluation and Conference (MUC-93)*, San Mateo, CA. Morgan Kaufman.

DARPA (Ed.). (1995). *Proceedings of the Sixth Message Understanding Conference(MUC-95)*, San Mateo, CA. Morgan Kaufman.

DARPA (Ed.). (1998). *Proceedings of the Seventh Message Understanding Conference (MUC-98)*, Fairfax, VA. Morgan Kaufman.

Domingos, P. (1996). Unifying instance-based and rule-based induction. *Machine Learning*, *24*, 141–168.

Doorenbos, R. B., Etzioni, O., & Weld, D. S. (1997). A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the First International Conference on Autonomous Agents (Agents-97)*, pp. 39–48 Marina del Rey, CA.

Dörre, J., Gerstl, P., & Seiffert, R. (1999). Text mining: Finding nuggets in mountains of textual data. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pp. 398–401 San Diego, CA.

Etzioni, O. (1996). The World-Wide Web: Quagmire or gold mine?. *Communications of the Association for Computing Machinery*, *39*(11), 65–68.

Feldman, R. (Ed.)., text mining (1999). *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99) Workshop on Text Mining: Foundations, Techniques and Applications*, Stockholm, Sweden.

Feldman, R., & Dagan, I. (1995). Knowledge discovery in textual databases (KDT). In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95)*, pp. 112–117 Montreal, Canada.

Feldman, R., Fresko, M., Hirsh, H., Aumann, Y., Liphstat, O., Schler, Y., & Rajman, M. (1998). Knowledge management: A text mining approach. In Reimer, U. (Ed.), *Proceedings of Second International Conference on Practical Aspects of Knowledge Management*, pp. 9.1–9.10 Basel, Switzerland.

Feldman, R., & Hirsh, H. (1996). Mining associations in text in the presence of background knowledge. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pp. 343–346 Portland, OR.

Fellbaum, C. D. (1998). *WordNet: An Electronic Lexical Database.* MIT Press, Cambridge, MA.

Finkelstein-Landau, M., & Morin, E. (1999). Extracting semantic relationships between terms: Supervised vs. unsupervised methods. In *Proceedings of International Workshop on Ontological Engineering on the Global Information Infrastructure*, pp. 71–80 Dagstuhl Castle, Germany.

Freitag, D. (1997). Using grammatical inference to improve precision in information extraction. In *Working Notes of the ICML-97 Workshop on Automata Induction, Grammatical Inference and Language Acquisition* Nashville, TN.

Freitag, D. (1998a). Information extraction from HTML: Application of a general learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 517–523 Madison, WI.

Freitag, D. (1998b). Multi-strategy learning for information extraction. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pp. 161–169 Madison, WI.

Freitag, D., & McCallum, A. K. (1999). Information extraction with HMMs and shrinkage. In *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, pp. 31–36 Orlando, FL.

Ghani, R., Jones, R., Mladenić, D., Nigam, K., & Slattery, S. (2000). Data mining on symbolic knowledge extracted from the web. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining*, pp. 29–36 Boston, MA.

Hammond, K., Burke, R., Martin, C., & Lytinen, S. (1995). FAQ-Finder: A case-based approach to knowledge navigation. In *Working Notes of AAAI Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments*, pp. 69–73 Stanford University. AAAI Press.

Hearst, M. (1999). Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL99)*, pp. 3–10 University of Maryland.

Honkela, T., Kaski, S., Lagus, K., & Kohonen, T. (1997). WEBSOM - self-organizing maps of document collections. In *Proceedings of the Workshop on Self-Organizing Maps*, pp. 298–303 Espoo, Finland.

Hsu, C.-N. (1999). Finite-state transducers for semi-structured text mining. In Feldman, R. (Ed.), *Proceedings of Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99) Workshop on Text Mining: Foundations, Techniques and Applications* Stockholm, Sweden.

Huffman, S. B. (1996). Learning information extraction patterns from examples. In Wermter, S., Riloff, E., & Scheler, G. (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pp. 246–260. Springer, Berlin.

Joachims, T., Freitag, D., & Mitchell, T. (1997). WebWatcher: A tour guide for the world wide web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pp. 770–775 Nagoya, Japan.

Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *SIGKDD Explorations*, *2*(1), 1–15.

Larkey, L. S., Ogilvie, P., Price, M. A., & Tamilio, B. (2000). Acrophile: an automated acronym extractor and server. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pp. 205–214 San Antonio, TX.

Lent, B., Agrawal, R., & Srikant, R. (1997). Discovering trends in text databases. In Heckerman, D., H. Mannila, D. P., & Uthrysamy, R. (Eds.), *Proceedings of the Third International*

Conference on Knowledge Discovery and Data Mining (KDD-97), pp. 227–230 Newport Beach, CA.

Loh, S., Wives, L. K., & de Oliveira, J. P. M. (2000). Concept-based knowledge discovery in texts extracted from the web. *SIGKDD Explorations*, *2*(1), 29–39.

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

McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/˜mccallum/bow.

McCallum, A. K., & Nigam, K. (1998). A comparison of event models for naive Bayes text classification. In *Papers from the AAAI-98 Workshop on Text Categorization*, pp. 41–48 Madison, WI.

McCallum, A. K., Nigam, K., Rennie, J., & Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, *3*(2), 127–163.

Merkl, D. (1998). Text data mining. In Dale, R., Moisl, H., & Somers, H. (Eds.), *A Handbook of Natural Language Processing: Techniques and Applications for the Processing of Language as Text.* Marcel Dekker, New York.

Mitchell, T. (1997). *Machine Learning.* McGraw-Hill, New York, NY.

Mladenić, D. (Ed.)., text mining (2000). *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD-2000) Workshop on Text Mining*, Boston, MA.

Mooney, R. J., & Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pp. 195–204 San Antonio, TX.

Pazzani, M. J., Muramatsu, J., & Billsus, D. (1996). Syskill & Webert: Identifying interesting web sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 54–61 Portland, OR.

Pereira, F., Tishby, N., & Lee, L. (1993). Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics (ACL-93)*, pp. 183–190 Columbus, Ohio.

Peterson, J. L. (1976). Computation sequence sets. *Journal of Computer and System Sciences*, *13*(1), 1–24.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo,CA.

Radev, D. R., & McKeown, K. R. (1998). Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, *24*(3), 469–500.

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pp. 811–816 Washington, D.C.

Riloff, E. (1996a). Using learned extraction patterns for text classification. In Wermter, S., Riloff, E., & Scheler, G. (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pp. 275–289. Springer, Berlin.

Riloff, E. (1996b). Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1044–1049 Portland, OR.

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

Silverstein, C., Brin, S., Motwani, J., & Ullman, J. (2000). Scalable techniques for mining causal structures. *Data Mining and Knowledge Discovery*, *4*(2/3), 163–192.

Soderland, S. (1997). Learning to extract text-based information from the World Wide Web. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pp. 251–254 Newport Beach, CA.

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

Soderland, S., Fisher, D., Aseltine, J., & Lehnert, W. (1995). CRYSTAL: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pp. 1314–1319 Montreal, Canada.

Srihari, R., & Li, W. (1999). Information extraction supported question answering. In *Proceedings of the Eighth Text REtrieval Conference(TREC-8)* Gaithersburg, MD. NIST Special Publication.

Wagner, R. A., & Fisher, M. J. (1974). The string to string correction problem. *Journal of the Association for Computing Machinery*, *21*, 168–173.

Weiss, S. M., Apté, C., Damerau, F., Johnson, D. E., Oles, F. J., Goetz, T., & Hampp, T. (1999). Maximizing text-mining performance. *IEEE Intelliget Systems*, *14*(4), 63–69.

Weizenbaum, J. (1966). ELIZA – A computer program for the study of natural language communications between men and machines. *Communications of the Association for Computing Machinery*, *9*, 36–45.

White, M., Cardie, C., Han, C.-H., Kim, N., Lavoie, B., Palmer, M., Rambow, O., & Yoon, J. (2000). Towards translingual information access using portable information extraction. In *Proceedings of the Applied Natural Language Processing and the North American Chapter of the Association for Computational Linguistics (ANLP/NACAC-2000) Workshop on Embedded MT Systems*, pp. 31–37 Seattle, WA.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, *1*(1/2), 67–88.

Yang, Y., Carbonell, J., Brown, R., Pierce, T., Archibald, B., & Liu, X. (1999). Learning approaches for detecting and tracking news events. *IEEE Intelligent Systems*, *14*(4), 32–43.