Using Information Extraction to Aid the Discovery of Prediction Rules from Text

Un Yong Nahm
Department of Computer Sciences
University of Texas, Austin, TX 78712-1188
pebronia@cs.utexas.edu

Raymond J. Mooney
Department of Computer Sciences
University of Texas, Austin, TX 78712-1188
mooney@cs.utexas.edu

ABSTRACT

Text mining and *Information Extraction* (IE) are both topics of significant recent interest. Text mining concerns applying data mining, a.k.a. knowledge discovery from databases (KDD) techniques to unstructured text. Information extraction (IE) is a form of shallow text understanding that locates specific pieces of data in natural language documents, transforming unstructured text into a structured database. This paper describes a system called DiscoTEX, that combines IE and KDD methods to perform a text mining task, discovering prediction rules from natural-language corpora. An initial version of DiscoTEX is constructed by integrating an IE module based on Rapier and a rule-learning module, Ripper. We present encouraging results on applying these techniques to a corpus of computer job postings from an Internet newsgroup.

1. INTRODUCTION

The problem of *text mining*, i.e. discovering useful knowledge from unstructured text, is attracting increasing attention [14, 18]. This paper suggests a new framework for text mining based on the integration of Information Extraction (IE) and traditional Knowledge Discovery from Databases (KDD). 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 attracted increasing attention since the start of the Message Understanding Conferences (MUCs) [11, 12], 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. The IE module locates specific pieces of data in raw text, and the resulting database is provided to the KDD module for mining.

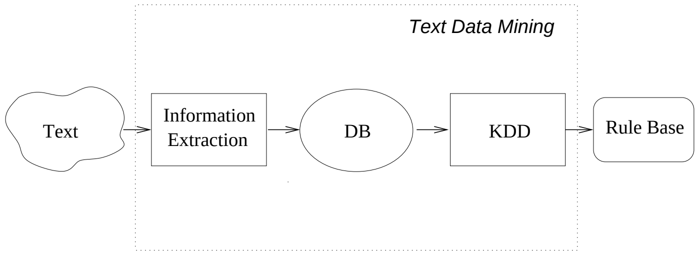

Figure 1: Overview of IE-based Text Mining Framework

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 [5, 3]. 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 [4] 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, 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."

However, the accuracy of current IE systems, whether built manually or induced from data, is limited. 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 paper we present experiments demonstrating that knowledge discovered from an automatically extracted database is close in accuracy to that discovered from a manually constructed database, demonstrating that combining IE and KDD is a viable approach to text mining.

The remainder of the paper is organized as follows. Section 2 describes a system called DiscoTEX (DISCOvery

Sample Job Posting
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 applications 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); Type of Position: Full Time

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

Figure 2: Sample Text and Filled Template

from Text EXtraction) that combines IE and KDD technologies to discover prediction rules. Section 3 presents and discusses experimental results obtained on a corpus of job postings from the newsgroup `austin.jobs`. Section 4 reviews some related work, section 5 discusses directions for future research, and section 6 presents our conclusions.

2. THE DISCOTEX SYSTEM

2.1 Information Extraction

The goal of an IE system is to locate specific data in natural-language text. 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. IE is useful for a variety of applications, particularly given the recent proliferation of Internet and web documents. Recent applications include course homepages [17], apartment rental ads [26], and job announcements [7, 4].

In this paper, we consider the task of extracting a database from postings to the USENET newsgroup, `austin.jobs`. Figure 2 shows a sample message from the newsgroup and the filled computer-science job template where several slots may have multiple fillers. For example, slots such as languages, platforms, applications, and areas usually have more than one filler, while slots related to the job's title or location have only one filler.

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 [21, 19] was trained on this data to identify relevant documents (using the Rainbow package [20]). The resulting categorizer has an accuracy of over 99% and is used to filter irrelevant documents from the original postings.

RAPIER [4], a machine-learning system for inducing rules for extracting information from natural-language texts, is used to construct an IE module for DISCOTEX. RAPIER is a bottom-up relational rule learner for acquiring IE rules from a corpus of labeled training examples. It learns patterns describing constraints on slot fillers and their surrounding context using a specific-to-general search. Constraints in patterns can specify the specific words, part-of-speech, or semantic classes of tokens. It has been demonstrated that RAPIER performs fairly well on realistic applications such as USENET job postings and seminar announcements [2, 4].

In the experiments in this paper, 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 learned extractor capable of building a database from which accurate knowledge can be discovered.

2.2 Rule Induction

After constructing an IE system that extracts the desired set of slots for a given application, a database is constructed from a corpus of texts by applying the extractor 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 classification rule-learning methods 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∈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" → "Database" and "Client/Server" → "Client-Server" are handled by manually contrived synonym-checking rules.

We have applied C4.5RULES [24] to induce rules from the

- Oracle∈application ∧ QA Partner∈application → SQL∈language
- C++∈language ∧ C∈language ∧ CORBA∈application → Windows∈platform
- HTML∈language ∧ WindowsNT∈platform ∧ Active Server Pages∈application → Database∈area
- ¬(UNIX∈platform) ∧ ¬(Windows∈platform) ∧ Games∈area → 3D∈area
- Java∈language ∧ ActiveX∈area ∧ Graphics∈area → Web∈area
- Visual Basic∈language ∧ OLE∈area → ¬(UNIX∈platform)
- 3D∈area ∧ Games∈area ∧ ¬(E-Commerce∈area) → ¬(SQL∈language)

Figure 3: Sample Mined Rules for Computer-Science Job Postings

resulting binary data by learning decision trees and translating them into pruned rules. Ripper [8] was also applied to learn prediction rules. Ripper runs significantly faster since it has an ability to handle *set-valued features* [9] 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 the in applications slot, this is represented by the production rule, `ShockWave∈application → Web∈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 3.

2.3 System Architecture

The overall architecture of DiscoTEX is shown in Figure 4. 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. C4.5 or Ripper) 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 can be found in [22].

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 job, 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 paper employ Ripper since its computational time and space complexity

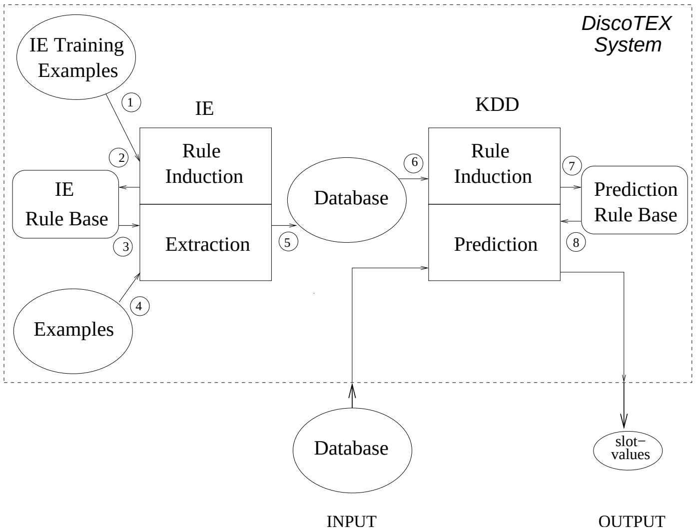

Figure 4: The System Architecture

is significantly less.

3. EXPERIMENTAL RESULTS

3.1 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 in 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. 10-fold cross validation was used to generate training and test sets. Rules were induced for predicting the fillers of the `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 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

	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

metrics are defined as follows:

$$precision = \frac{\#ofPresentSlotValuesCorrectlyPredicted}{\#ofSlotValuesPredictedToBePresent} \tag{1}$$

$$recall = \frac{\#ofPresentSlotValuesCorrectlyPredicted}{\#ofPresentSlotValues} \tag{2}$$

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):

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

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 in the training data, then this approach 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 is 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.

3.2 Results

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 in order to evaluate data mining. 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 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. Figures 5, 6 and 7 show 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. Figure 6 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 8 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 that fact that different slots show quite different learning rates.

Figures 9, 10 and 11 show 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 DiscoTEX's IE system. 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 extractor. 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 12 shows the preformance results. Increasing the number of IE training examples improves the accuracy of 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 effort devoted to labeling IE training

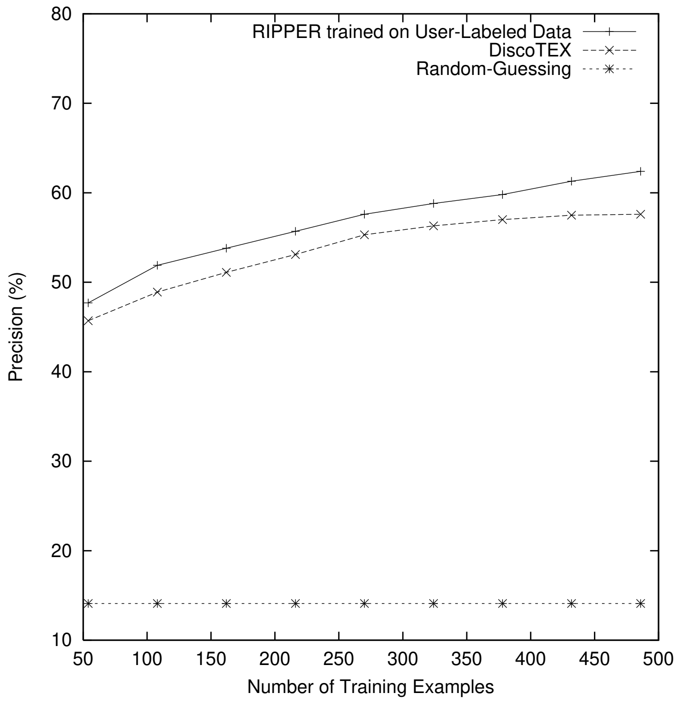

Figure 5: Precision with disjoint IE training set

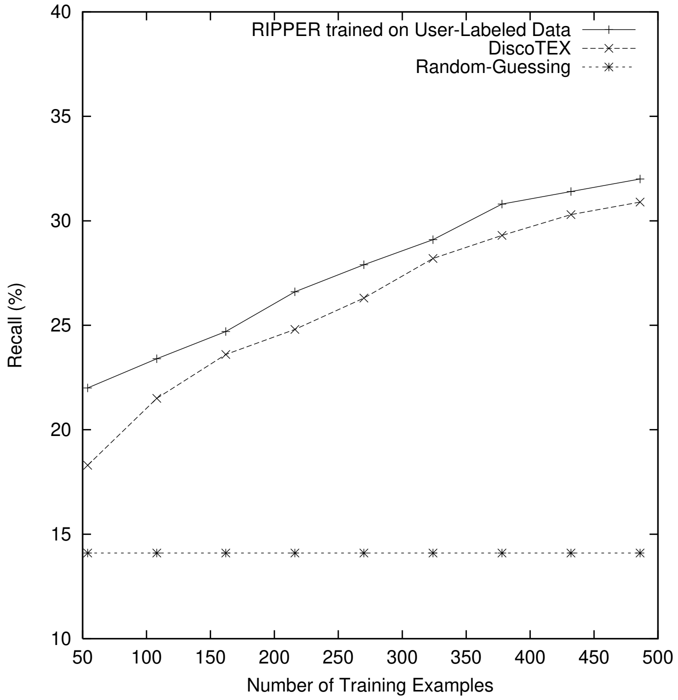

Figure 6: Recall with disjoint IE training set

examples.

4. RELATED RESEARCH

There has been relatively little research exploring the integration of IE and KDD. KDT (Knowledge Discovery in Textual Databases) alludes to the use of IE in text mining [14]; however, it uses texts manually tagged with a limited number of fixed category labels. KDT does not actually use automated text categorization or IE. Similarly, FACT [15], which discovers associations amongst keywords from text documents, does not automatically label the documents with keywords. Another approach similar to DiscoTEX is a method used to discover semantic relationships between terms in a collection of documents [16]. In this work, a natural-language parser is used in place of information extraction since the extraction process is one of finding syntactic or semantic relations between terms.

5. FUTURE WORK

Although our preliminary results with job postings are encouraging, a fuller evaluation will apply DiscoTEX to larger job corpora and to other realistic domains. For instance, we plan to use DiscoTEX to discover knowledge from a database extracted from business news articles. Mining a database extracted from doctors' English notes and records about medical patients is another promising application area.

One step in DiscoTEX that is currently performed manually is collapsing similar slot-fillers in the extracted data into a canonical form, e.g. mapping "NT," "Windows NT," and "Microsoft Windows NT" all to a unique term. In many cases, such collapsing could be automated by clustering slot fillers using a distance metric based on textual similarity,

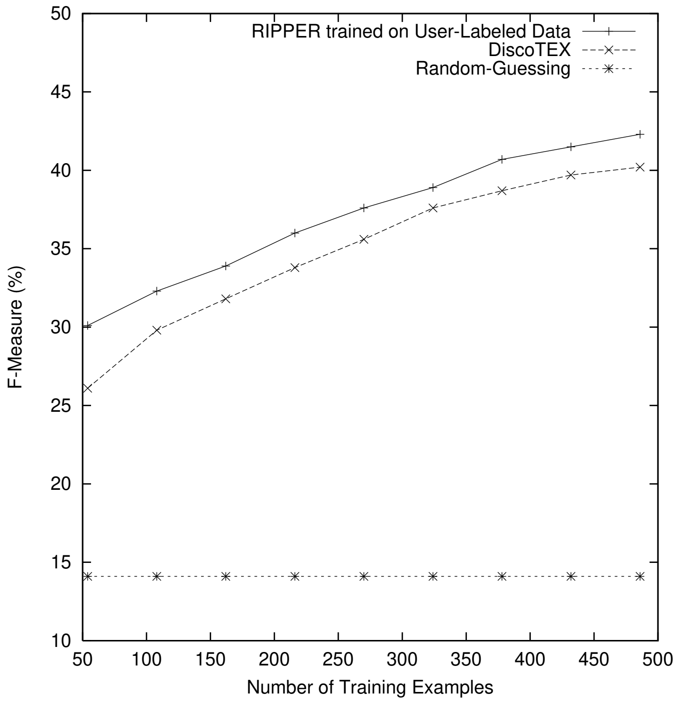

Figure 7: F-measure with disjoint IE training set

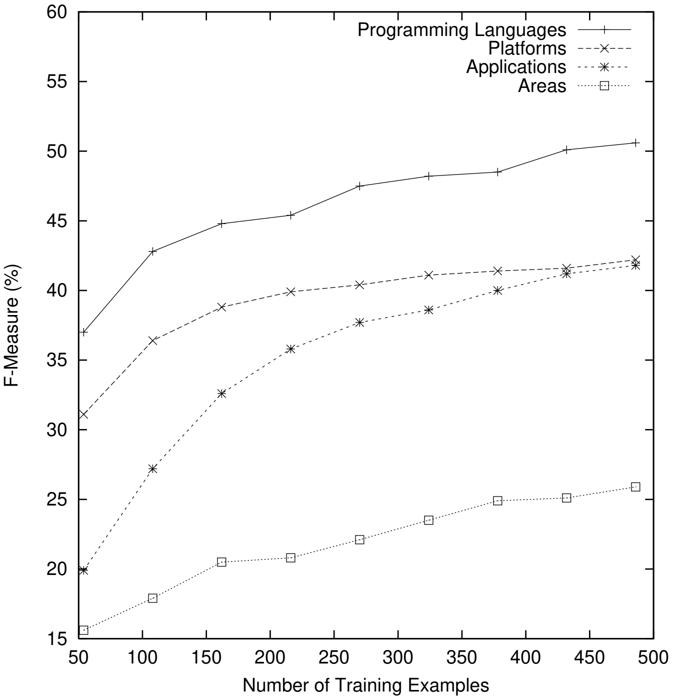

Figure 8: F-measure for DiscoTEX by slots

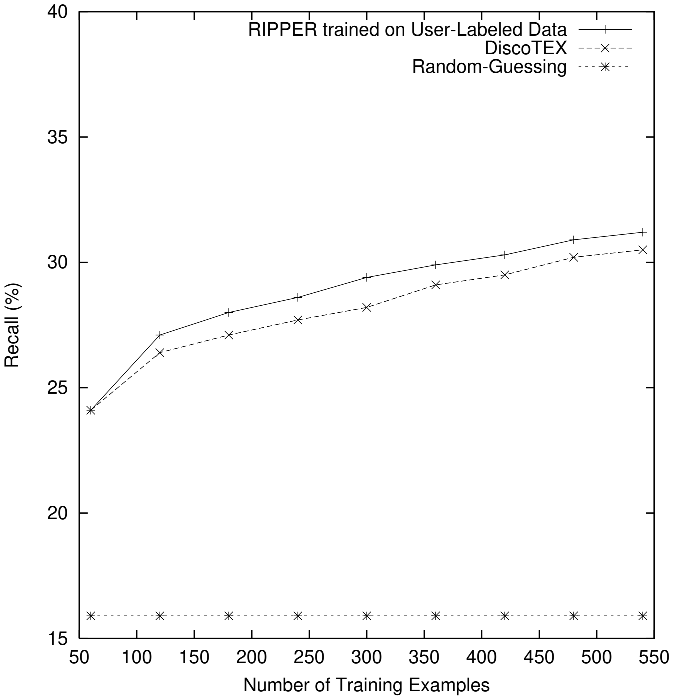

Figure 10: Recall with reused IE training set

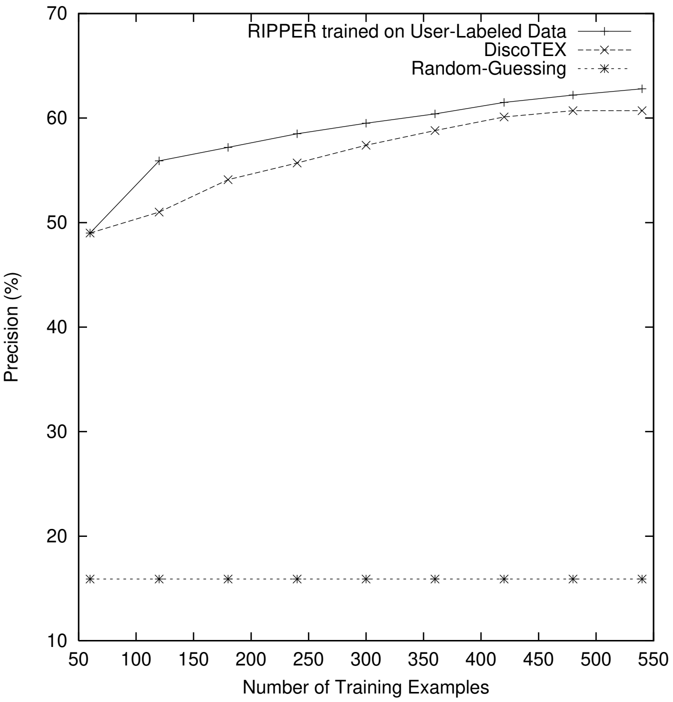

Figure 9: Precision with reused IE training set

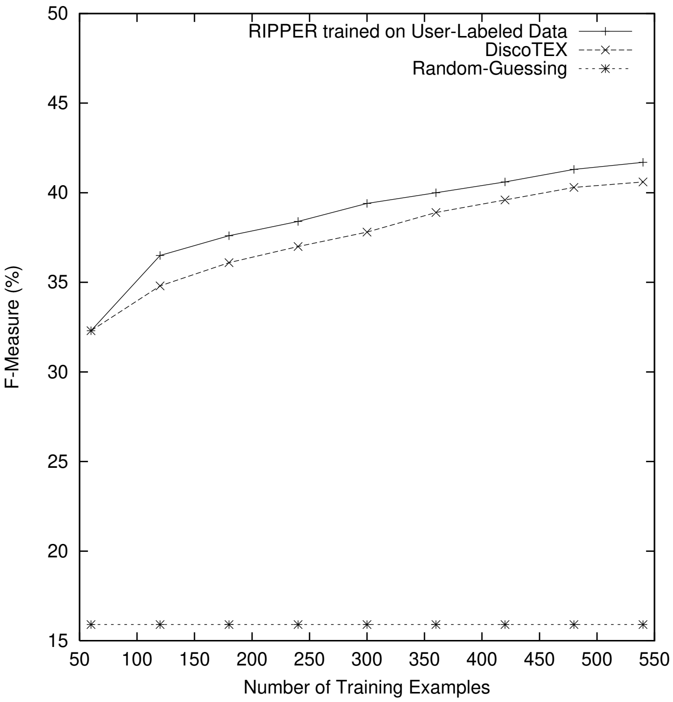

Figure 11: F-measure with reused IE training set

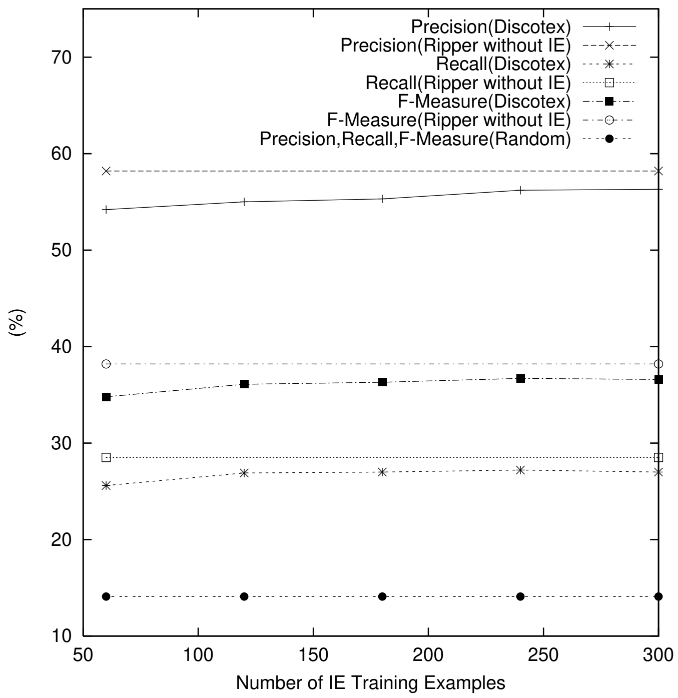

Figure 12: Performances with varied sizes of IE training set

such as character edit distance [25], or semantic similarity as determined by context-based word clustering [23].

Instead of requiring or creating canonical slot-fillers that must match exactly, an alternative approach is to allow partial matching of slot-fillers during the discovery process. The need for soft matching of text strings in discovered rules is an important aspect of text mining that requires changes to existing rule induction methods. Since text strings in traditional databases also contain typos, misspellings, and non-standardized variations, this is also an important aspect of traditional KDD that has not been adequately addressed. We are currently exploring the discovery of rules that allow soft matching of slot-fillers by adapting the RISE approach to unifying rule-based and instance-based learning methods [13]. Like WHIRL [10], our approach uses a TFIDF text-similarity metric from information retrieval [1] to find examples that are close but not exact matches to the conditions of a rule. We have applied a preliminary version of this system to a database of science-book descriptions extracted from Amazon.com and discovered rules such as: "If a 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 a synopsis of a book contains History and Life, then title of that book includes a substring, Origin."

Currently, we only consider discrete-valued slots. However, real-valued slots, such as "required years of experience" or "salary" could also be provided to the rule miner as additional input features when predicting other slots. Predicting such continuous values using regression methods instead of categorization techniques is another area for future research.

The procedure for selecting slots to be used in rule mining also needs to be automated. In the current experiments, we manually chose five slots from the computer-science job template. For example, title slots for job postings is not used because it has many possible values and is difficult to predict. By identifying and quantifying the correlations between slot values, this decision could be automated.

6. CONCLUSIONS

There is a growing interest in the general topic of text mining [18]; however, there are few working systems or detailed experimental evaluations. By utilizing existing IE and KDD technology, text-mining systems can be developed relatively rapidly and evaluated on existing IE corpora. In this paper, 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.

Text mining is a relatively new research area at the intersection of data mining, 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 [6] is critical to the development of effective text-mining systems.

Acknowledgements

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

7. REFERENCES

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press, New York, 1999.

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

[3] M. E. Califf, editor. *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, Orlando, FL, 1999. AAAI Press.

[4] M. E. Califf and R. J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, Orlando, FL, July 1999.

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

[6] C. Cardie and R. J. Mooney. Machine learning and natural language (introduction to special issue on natural language learning). *Machine Learning*, 34:5–9, 1999.

[7] J. Y. Chai, A. W. Biermann, and C. I. Guinn. Two dimensional generalization in information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 431–438, Orlando, FL, July 1999.

[8] W. W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, 1995.

[9] W. W. Cohen. Learning trees and rules with set-valued features. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 709–716, Portland, OR, August 1996.

[10] W. W. Cohen and H. Hirsh. Joins that generalize: Text classification using WHIRL. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 169–173, New York, NY, 1998.

[11] DARPA, editor. *Proceedings of the Fifth DARPA Message Understanding Evaluation and Conference*, San Mateo, CA, 1993. Morgan Kaufman.

[12] DARPA, editor. *Proceedings of the 6th Message Understanding Conference*, San Mateo, CA, 1995. Morgan Kaufman.

[13] P. Domingos. Unifying instance-based and rule-based induction. *Machine Learning*, 24:141–168, 1996.

[14] R. Feldman and I. Dagan. Knowledge discovery in textual databases (KDT). In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, 1995.

[15] R. Feldman and H. Hirsh. Mining associations in text in the presence of background knowledge. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 343–346, Portland, OR, August 1996.

[16] M. Finkelstein-Landau and E. Morin. Extracting semantic relationships between terms: Supervised vs. unsupervised methods. In *Proceedings of International Workshop on Ontological Engineering on the Global Information Infrastructure*, pages 71–80, Dagstuhl Castle, Germany, May 1999.

[17] D. Freitag. Information extraction from HTML: Application of a general learning approach. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 517–523, 1998.

[18] M. Hearst. Untangling text data mining. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 3–10, University of Maryland, 1999.

[19] A. McCallum and K. Nigam. A comparison of event models for naive Bayes text classification. In *Papers from the AAAI 1998 Workshop on Text Categorization*, pages 41–48, Madison, WI, 1998.

[20] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classfication and clustering, 1996. `http://www.cs.cmu.edu/~mccallum/bow`.

[21] T. Mitchell. *Machine Learning*. McGraw-Hill, New York, NY, 1997.

[22] U. Y. Nahm and R. J. Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, Austin, TX, July 2000.

[23] F. Pereira, N. Tishby, and L. Lee. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Columbus, Ohio, 1993.

[24] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo,CA, 1993.

[25] E. S. Ristad and P. N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5), 1998.

[26] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34:233–272, 1999.