A Mutually Beneficial Integration of Data Mining and Information Extraction

Un Yong Nahm and **Raymond J. Mooney**
Department of Computer Sciences,
University of Texas, Austin, TX 78712-1188
{pebronia,mooney}@cs.utexas.edu

Abstract

Text mining concerns applying data mining 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 data mining methodologies to perform text mining as well as improve the performance of the underlying extraction system. Rules mined from a database extracted from a corpus of texts are used to predict additional information to extract from future documents, thereby improving the recall of IE. Encouraging results are presented on applying these techniques to a corpus of computer job announcement postings from an Internet newsgroup.

Introduction

Data mining, a.k.a. knowledge discovery from databases (KDD), and *information extraction* (IE) are both topics of significant recent interest. KDD considers the application of statistical and machine-learning methods to discover novel relationships in large relational databases. IE concerns locating specific pieces of data in natural-language documents, thereby extracting structured information from unstructured text. However, there has been little if any research exploring the interaction between these two important areas. This paper explores the mutual benefit that the integration of IE and KDD can provide.

KDD assumes that the information to be "mined" is already in the form of a relational database. Unfortunately, for many applications, available electronic information is in the form of unstructured natural-language documents rather than structured databases. Consequently, the problem of *text mining*, i.e. discovering useful knowledge from unstructured text, is beginning to attract attention (Feldman & Dagan 1995; Hearst 1999). Information extraction can play an obvious role in text mining. Natural-language information-extraction methods can transform a corpus of textual documents into a more structured database. Standard KDD methods can then be applied to the resulting database to discover novel relationships.

In previous research, we have developed a machine-learning system, RAPIER, for inducing rules for extracting information from natural-language texts (Califf & Mooney 1999; Califf 1998). Using an IE system constructed by RAPIER, we have extracted a database of over 5,000 computer-related jobs from messages posted to the `austin.jobs` newsgroup. By applying standard rule induction methods, e.g. C4.5RULES (Quinlan 1993) or RIPPER (Cohen 1995), to the resulting database, 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." This example clearly illustrates the role that IE can play in extending KDD to textual databases.

A less obvious interaction is the benefit that KDD can in turn provide to IE. 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`$\in$`Programming-Languages` and the `graphics`$\in$`areas` have been extracted from a job posting, then an IE system might also consider extracting `PhotoShop`$\in$`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 1995; 1993), such predictive relationships might be productively used to improve recall by suggesting additional information to extract. This paper reports 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.

The remainder of the paper is organized as follows. Section 2 presents some background information on IE and RAPIER. Section 3 describes a system called DISCOTEX (DISCOvery from Text EXtraction) that improves RAPIER's performance by exploiting prediction rules. Section 4 presents and discusses experimental results obtained on a corpus of job postings from the newsgroup `austin.jobs`. Section 5 reviews some related work, section 6 discusses directions for future research, and section 7 presents our conclusions.

Sample Job Posting
Job Title: Senior DBMS Consultant
Location: Dallas,TX
Responsibilities: DBMS Applications consultant works with project teams to define DBMS based solutions that support the enterprise deployment of Electronic Commerce, Sales Force Automation, and Customer Service applications.
Desired Requirements: 3-5 years exp. developing Oracle or SQL Server apps using Visual Basic, C/C++, Powerbuilder, Progress, or similar. Recent experience related to installing and configuring Oracle or SQL Server in both dev. and deployment environments.
Desired Skills: Understanding of UNIX or NT, scripting language. Know principles of structured software engineering and project management
Salary: Up to $55K (depending on experience) plus comprehensive benefits, and ample opportunity for career growth.
A BS degree or higher in Computer Science or related field are required.

Filled Job Template
title: Senior DBMS Consultant
salary: Up to $55K
state: TX
city: Dallas
country: US
language: Powerbuilder, Progress, C, C++, Visual Basic
platform: UNIX, NT
application: SQL Server, Oracle
area: Electronic Commerce, Customer Service
required years of experience: 3
desired years of experience: 5
required degree: BS

Figure 1: Sample Text and Filled Template

Background

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 apartment rental ads (Soderland 1999), job announcements (Chai, Biermann, & Guinn 1999), and course homepages (Freitag 1998).

In this paper, we consider the task of extracting a database from postings to the USENET newsgroup, `austin.jobs`. Figure 1 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.

The RAPIER System

RAPIER is a bottom-up relational rule learner for acquiring information extraction 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 on patterns can specify the specific words, part-of-speech, 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 paper, we use the simpler version of RAPIER that employs only word and part-of-speech constraints since WordNet classes provide no additional advantage in this domain (Califf & Mooney 1999).

The learning algorithm of RAPIER was inspired by several inductive logic programming systems. First, RAPIER creates most-specific patterns for each slot in each example specifying the complete word and tag information for the filler and its full context. New rules 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 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 then be tested on novel documents. Califf (1998) and Califf & Mooney (1999) provide more information and results demonstrating that RAPIER performs well on realistic applications such as USENET job postings and seminar announcements.

The DISCOTEX System

Text Mining

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 the information in each database field given the information in all other fields. 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`$\in$`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 0 such similar terms. Trivial cases such as "Databases" $\rightarrow$ "Database" and "Client/Server" $\rightarrow$ "Client-Server" are handled by manually contrived synonym-checking rules.

Currently, DISCOTEX uses C4.5RULES (Quinlan 1993) to induce rules from the resulting binary data by learning

- Oracle∈application ∧ QA Partner∈application → SQL∈language
- $C{+}{+}$∈language ∧ C∈language ∧ CORBA∈application ∧ Title=Software Engineer → Windows∈platform
- AIX∈platform ∧ ¬(Sybase∈application) ∧ DB2 ∈application → Lotus Notes ∈application
- 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

Figure 2: Sample Mined Prediction Rules for Computer-Science Jobs

Determine T, a threshold value for rule validation
Create a database of labeled examples
    (by applying IE to the document corpus)
For each labeled example *D* do
    S := set of slot fillers of D
    Convert S to binary features
Build a prediction rule base, RB
    (by applying C4.5RULES to the binary data)
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
        Delete *R* from *RB*

Figure 3: Algorithm Specification: Rule Mining Phase

decision trees and translating them into pruned rules. We have also applied RIPPER (Cohen 1995) to learn rules, using its ability to handle *set-valued features* (Cohen 1996b) to avoid the step of explicitly translating slot fillers into binary features. The two systems produce very similar results and the experiments in this paper employ C4.5rules.

Discovered knowledge describing the relationships between slot values is written in the 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`∈`application` → `Web`∈`area`. Rules can also predict the absence of a filler in a slot; however, here we focus on rules predicting the presence of fillers. Sample rules mined from a database of 5,000 jobs extracted from the USENET newsgroup `austin.jobs` are shown in Figure 2.

Pseudocode for the text mining phase is shown in Figure 3. 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.

Using Mined Rules to Improve IE

After mining knowledge from extracted data, DISCOTEX uses the discovered rules to predict missing information

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

Figure 4: Algorithm Specification: IE Phase

during subsequent extraction. Tests of IE systems usually consider two performance measures, *precision* and *recall* defined as:

$$precision = \frac{\#ofCorrectFillersExtracted}{\#ofFillersExtracted} \quad (1)$$

$$recall = \frac{\#ofCorrectFillersExtracted}{\#ofFillersInCorrectTemplates} \quad (2)$$

Also, F-measure was introduced to combine precision and recall and is computed as follows (when the same weight is given to precision and recall):

$$F-measure = \frac{2 * Precision * Recall}{Precision + Recall} \quad (3)$$

Since the set of potential slot fillers is very large and not fixed in advance, and since only a small fraction of possible fillers is present in any given document, these performance metrics are generally more informative than the accuracy of predicting the presence/absence across all slot-value pairs.

Many extraction 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.

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 discover the rule `SQL`∈`language` → `Database`∈`area`. If the IE system extracted `SQL`∈`language` but failed to extract `Database`∈`area`, 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 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

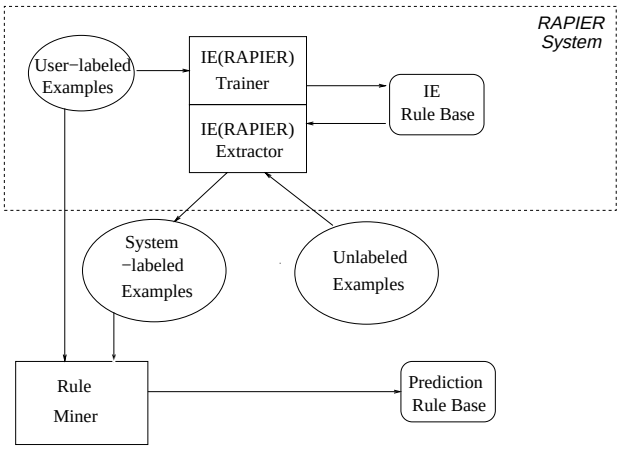

Figure 5: The System Architecture - Training

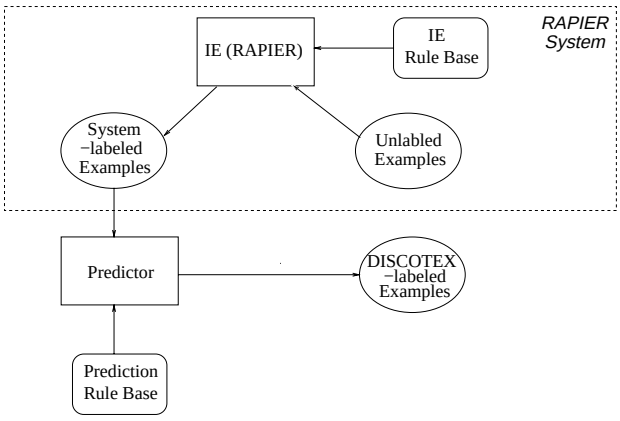

Figure 6: The System Architecture - Test

precision. This overall extraction algorithm is summarized in Figure 4.

One final issue is the order in which prediction rules are applied. When there are interacting rules, such as `HTML`$\in$`languages` $\rightarrow$ `WWW`$\in$`area` and $\neg$(`WWW`$\in$`area`) $\rightarrow$ $C++$ $\in$`languages`, different rule-application orderings can produce different results. Without the first rule, a document with `HTML`$\in$`language` but without `WWW`$\in$`area` in its initial filled template will make the second rule fire and predict $C++ \in$`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 overall architecture of the final system is shown in Figures 5 and 6. Documents which the user has annotated 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 miner 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.

Evaluation

Experimental Methodology

To test the overall system, 600 user-annotated computer-science job postings to the newsgroup `austin.jobs` were collected. 10-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 text miner. 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 `Title` slot is also used, 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.

Results

Figures 7 and 8 and show 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 the rule miner. Figures 7 and 8 show 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, each pair of systems were compared to determine if their differences in recall and F-measure 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 1 shows results on precision, recall and F-measure when additional unlabeled documents are used to 0 a larger database prior to mining for prediction rules (which are then

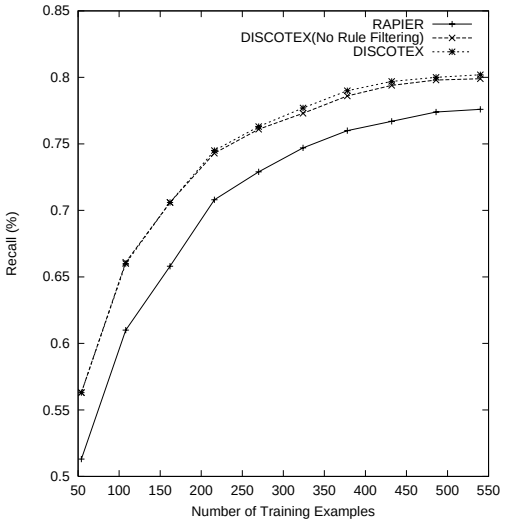

Figure 7: Recall on job postings

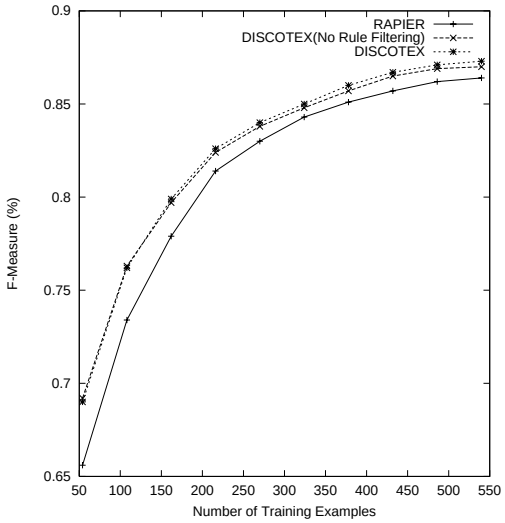

Figure 8: F-measure on job postings

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 1: Performance results of DISCOTEX with unlabeled examples

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 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 Table 1, 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. `language`. This is equivalent to replacing the mined rules in DISCOTEX with trivial rules of the form `True` $\rightarrow$ `<slot value>` 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` 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.

Related Research

There has been relatively little research exploring the integration of IE and KDD. Feldman & Dagan(1995) allude to the use of IE in text mining; however, their KDT(Knowledge Discovery in Textual Databases) system uses texts manually tagged with a limited number of fixed category labels. KDT does not actually use automated text categorization or IE and the paper does not discuss using mined knowledge to improve extraction.

There is a growing interest in the general topic of text mining (Hearst 1999); 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 text corpora for IE.

Future Work

Although our preliminary results with job postings on the newsgroup are encouraging, a fuller evaluation will apply DISCOTEX to other domains such as business news articles, medical documents such as patient records, and emails for product orders.

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," "WinNT", "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, such as character edit distance (Ristad & Yianilos 1998).

Currently, we only consider discrete-valued slots. However, real-valued slots, such as "desired years of experience," 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. By identifying and quantifying the correlations between slot values, this decision could be automated.

With regard to using KDD to improve IE, methods for using discovered predictive rules to improve precision as well as recall are needed. Simply eliminating extracted fillers that are not predicted is too course and would likely severely damage recall. Combining confidence measures for both predictive rules and extraction rules during IE could be a productive way to improve both precision and recall.

Conclusions

Information extraction and data mining can be integrated for the mutual benefit of both tasks. IE enables the application of KDD to unstructured text corpora and KDD can discover predictive rules useful for improving IE performance. This paper has presented initial results on integrating IE and KDD that demonstrate both of these advantages.

Text mining is a relatively new research area at the intersection of 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, the growing interaction between computational linguistics and machine learning (Cardie & Mooney 1999) is critical to the development of effective text-mining systems.

Acknowledgements

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

References

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

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

Califf, M. E. 1998. *Relational Learning Techniques for Natural Language Information Extraction*. Ph.D. Dissertation, 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).

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

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

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

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

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

DARPA., ed. 1993. *Proceedings of the Fifth DARPA Message Understanding Evaluation and Conference*. San Mateo, CA: Morgan Kaufman.

DARPA., ed. 1995. *Proceedings of the 6th Message Understanding Conference*. San Mateo, CA: Morgan Kaufman.

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

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

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

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

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

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

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