Information Extraction: A Pattern Mining Approach

Chia-Hui Chang
Dept. of Computer Science and Information Engineering
National Central University, Chung-Li 320, Taiwan
chia@csie.ncu.edu.tw

Abstract

The vast amount of online information available has led to renewed interest in information extraction (IE) systems that analyze input documents to produce a structured representation of selected information from the documents. However, the design of an IE system differs greatly according to its input: from unrestricted plain text, telegraphic passage, to semi-structured Web documents. This paper gives a survey of IE systems and discusses the essentials for IE system design. A pattern mining approach is proposed as a general approach for IE system design.

Keywords: information extraction, pattern mining, background knowledge, encoding hierarchy

1 Introduction

Due to the heterogeneity and the lack of structure of Web information sources, Web mining and searching tools often require a lot of efforts for manipulation among various data formats. Hence, there presents a special need for information extraction to extract information of interest from these documents. Information Extraction (IE) is concerned with extracting relevant data from a collection of documents and converting unstructured data into structured data format, enabling data integrations or data analysis.

Tailoring an IE system to new requirements is a task that varies in scale dependent on the text type, domain, and scenario [10]. Therefore, designing a trainable IE system has been an attractive application in machine learning research and the computational linguistics forum, Message Understanding Conference (MUC) [15].

This paper gives a survey of the issues in different IE tasks and summarizes the essentials for the design of an IE system. In addition, this paper proposes a pattern mining approach as a general solution to the problem. The remainder of this paper is organized as follows: section 2 defines the IE problems for semi-structured IE, plain text IE and free-form text, respectively. Section 3 introduces related work on these three categories. Section 4 presents our pattern mining approach as a general solution for information extraction. A Chinese information extraction task is given as an example in Section 5. Finally, section 6 concludes the paper.

2 Problem Definition

The goal of information extraction is to construct IE systems that would allow users to build customized extractors, instead of manually programming them. The produced extractors accept unstructured data and convert it into structured data format. Depending on the output template of the extraction tasks, the architecture of an extractor can be a simple while-loop, a hierarchial, tree-structure, or a finite-state machine. The basic component is a pattern matching procedure, which enables/disables the extraction procedure. Therefore, the heart of an IE system is the work to generalize useful extraction rules for extractors.

2.1 Input Data

Depending on the input, information extraction can be divided into three categories: plain text, semi-structured, and free-form text. Plain text IE involves natural language written text that contains grammatical text and are fully unstructured. Semi-structured IE, on the other hand, processes documents that are marked by tags and sometimes presented in nearly structured format, e.g. tables, lists. Free-form text IE lies between these two categories and accepts inputs that are written in unofficial and telegraphic style which consist of a mixture of grammatical and ungrammatical text. For example, job postings, apartment rentals from online newsgroups [30], the family history field from life insurance applications [14] or the technician comments from tire manufacturers [23].

2.2 Output Template

For plain text IE, there are four types of IE tasks defined by MUC: name entity recognition (NE), coreference resolution (CO), template element construction (TE), and scenario template production (ST), from single-slot filling (NE) to complex template/multi-slot extraction (CO, TE, ST) [15]. For example, names of people, organizations, places, and key numeric figures, such as currency amounts and dates are termed as NE tasks. TE associates descriptive information with entities, e.g. the location of an organization. ST ties together entities into event templates and relationships. For example, the template may be composed of several objects, each containing several slots [2].

As for semi-structured IE, the extraction target is often contained in a table or a list [16]. Therefore, the template can be considered as a relation of k-tuples, where k is the number of attributes of interest in the record. An attribute may have one or multiple instantiations in a record. The extraction task is a single-slot filling when k equals 1.

The template of free-form text IE is similar to that of semi-structured IE and can be represented as a k-slot template. However, extraction target is not easy to described as semi-structured input, since free-form text IE involves telegraphic text which is a mixture of grammatical and ungrammatical text. Therefore, relevant information is often illustrated by annotated or labelled training examples as in the plain text category [25].

In addition to the specification of input data and output template, the difficulties of an IE task can also be affected by two factors. The first one is whether extraction target or answer keys are provided as training examples. For most IE tasks, extraction targets are carefully labelled by human (called `user-labelled or annotated` training examples), such that IE systems know what to extract. For some IE tasks, such as semi-structured Web pages generated by CGI scripts, the extraction target is embedded in some patterns which can be learned without user-labelled training examples [28, 6]. Most machine learning based approaches rely on user-annotated training examples, either free-text IE [27, 5] or semi-structured Web IE [19, 17, 26]. Very few systems generate extraction rules based on unlabelled text. AutoSlog-TS [28] and IEPAD [6] are two of the systems for free-text and Web IE, respectively. Annotation-free IE systems are preferable because they save users' labelling efforts, but they are much difficult to design.

The second factor is the number of records/relations in one document. Take semi-structured Web IE for example: the input pages can contain simply one record with descriptions (e.g. Amazon's book description pages) or multiple records which are presented in a table or list (e.g. Altavista's query results). The number of records in a document sometimes arouses difficulties for multi-slot extraction tasks. For example, in plain text IE, such as finding information about corporate management changes from financial news articles, the output template is a 4-slot tuple, including person, organization, position, and transition (in or out) [1]. If one person, one position and one organization is extracted by name entity identification, it is much easy to compose the management change relation. If there might be two persons, two organizations, and two positions extracted by name entity identification, it is much difficult to compose the correct relations.

3 Survey of IE Systems

As described above, rule generalization is the main focus of most IE systems. Therefore, the design of an IE systems relies on the how the extraction rules are specified, generalized and applied. Since strings are made from words/lexicons and punctuation, it is customary to divide the input strings into tokens or segments such that common features can be used for generalization. For example, some IE tasks have common delimiters around the extraction target, especially for semi-structured IE, while some extraction targets have special features for generalization. For example, semi-structured IE often contains markup/tags as common delimiters. However, plain text IE usually lacks of common delimiters, therefore it requires special analyzers such as WordNet lexical analyzers [24] or syntactic taggers to analyze common features from the contents.

The simplest form of an extraction rule is sequential patterns that are specified in forms similar to regular expressions [5, 30, 6]. Such sequential patterns contain a series of constraints on individual tokens or segments of the input text. Another form of an extraction rule is first-order logic rules [13] which consists of predicates designed to test feature-values. Other forms include classification models such as maximum entropy classifier [9], etc.

Once the extraction rules are decided, learning algorithms can be applied to generalize the extraction rules. Some generalize rules from general to specific [13] and some generalize rules from specific to general [5]. Finally, these extraction rules are applied in the extractor as enabling/disabling constraints for extraction. In this section, we introduce several IE systems and discuss how the extractors and associated patterns are constructed.

3.1 Plain Text IE

AutoSlog [27] builds a set of extraction rules that are called concept nodes. Each concept node specifies a *trigger* word, an linguistic pattern and a set of enabling conditions that guarantee the applications of the concept node. The idea of concept nodes comes from *selective concept extraction* which is a form of text skimming. The concept nodes are constructed by a sentence analyzer called CIRCUS [21]. Since the most important facts about a news event are typically reported during the initial event description, AutoSlog finds the first sentence in the text that contains the extraction targets and hand it to CIRCUS to generates a concept node. The background knowledge imposed here is a set of 13 linguistic patterns. AutoSlog has been applied in the domain of terrorist events and learns concept nodes that are further sifted by users. Therefore, this method is quite manual overall.

CRYSTAL [31] generates concepts nodes that allow both exact word match and semantic constraints on any component phrase. It applies the linguistic pattern analyzer and express the extraction rules as a set of constraints on individual constituents of sentences such as subject, verb, and preposition phrase, etc. Then features such as exact word match and semantic constraints can be imposed on each syntactic components and decides whether to extract or not. Therefore, CRYSTAL is able to generate both single-slot or multi-slot extraction rules.

3.2 Semi-structured IE

The research on IE from semi-structured Web pages can be traced to the research of information agents that integrate data sources on the World Wide Web [11, 18]. A critical problem that must be addressed for these agents is how to extract structured data from the Web. Since it is not fit to write extractors for all the Web data source, machine learning approaches are proposed to solve the problem [29]. Web IE has been quite active in recent years and a lot of literature can be found [4, 20, 17, 26, 13, 12, 7, 22, 6].

WIEN [19] was the first wrapper induction system. It assumes that there is a unique multi-slot rule for a given Web site. As a result it fails to handle attributes with multiple instances and variant attribute permutations. Softmealy [17] uses a wrapper induction algorithm that expresses the extractors as finite-state transducers. The wrapper architecture are more expressive than those of WIEN because they can handle missing items, multi-valued items, and items appeared in various permutations. STALKER [26] is a wrapper induction system that performs hierarchical information extraction. To extract data, their extractor scans a text segment several times for each item at the same level in the hierarchy. The presentation is similar to multi-pass Softmealy extractors [16].

In addition to the variance in the wrapper architecture, another difference is the rule generalization. Basically, all these systems produce extraction rules for each slot then assemble them into their wrapper architecture. Both WIEN and Softmealy use delimiters that immediately precede and follows the actual data. Though Softmealy introduced the concept of contextual rules, the generalization is limited due to its encoding scheme and alignment approach. Stalker generalizes "disjunctive landmark automata" for each attribute and organized them as a hierarchy.

Fully automatic approach to information extraction is rare and often depends on some heuristics. For example, Embley et. al. [12] describe a heuristic approach that discovers record boundaries in Web documents by identifying *candidate separator tags* using five independent heuristics and choosing a consensus separator tag based on a heuristic combination [12]. However, one serious problem in this one-tag separator approach is that their system cannot be applied to Web pages where two or more separator tags are the same within a record, because their system cannot distinguish them and will fail to extract data correctly. As a result, the applicability of their system is seriously limited.

IEPAD (an acronym for Information Extraction based on PAttern Discovery) [6] is a trainable IE system which attempts to eliminate the need of user-labelled examples. The user does not need to tell the system what information to extract, but simply choose among patterns (discovered by IEPAD from the training examples) to see if the pattern can extract the desired information. This amazing feature is based on the assumption that the input is a multi-record Web page so that sequential pattern mining can be applied to discover repeats. However, if the training page contains only one record, IEPAD fails.

3.3 Free-form Text IE

Rapier [5] specifies its extraction rules as three constraints: pre-filler, filler, post-filler. The pre-filler and post-filler specify the constraints on the surrounding delimiters, while the filler describes the constraints of the extraction target. The generalization of the extraction rules include syntactic information from part-of-speech taggers and semantic class information from WordNet. For fillers, the length of the extraction target can be added to the constraint. Rapier applies a specific-to-general search which generalizes the least general generalization (LGG) for each pair of rules. When there exists no common features, a disjunction is produced or the constraints are removed.

SRV [13] learns single-slot extraction rules that is expressed in the form of first-order logic. The extraction rules are learned from training examples that are small segments from input text. Segments that belong to extraction targets are labelled as positive examples and all other segments are negative examples. SRV includes simple features such as length, character type, orthography, part-of-speech, lexical meaning, as well as relational features that describes the texts surrounding the extraction targets. These features are presented as predicates in the extraction rule. For each slot, a covering algorithm is applied where predicates are added greedily attempting to cover as many positive and as few negative examples as possible. During extraction, all rules matching a given fragment are used to assign a confidence score and then combined by Bayesian m-estimates.

The WHISK [30] extraction rules are a special type of regular expression. The constituents include delimiters and content features. WHISK can generates single-slot and multi-slot rules. For grammatical text, the WHISK regular expressions provide a set of special constructs that are helpful when dealing with information provided by syntactic analyzer. WHISK and IEPAD are similar in their extraction rule expression, but differ in the learning procedure.

4 A Pattern Mining Approach

In this paper, we propose a pattern mining approach as a general solution to IE system design. A pattern-mining based IE system contains two parts: one is a set of encoding schemes that can translate input text into token strings or add tags to texts for common features generalization, the other is a pattern mining algorithm that is used to discover patterns. The first part is common to most IE systems. The second part is a alternative/replacement of learning algorithms for many IE systems. The advantage is that domain-specific knowledge is separate from the generalization algorithms.

4.1 Encoding Schemes

An encoding scheme is used to translate a training input into an abstracted token string in order to reveal common features among input examples. For semi-structured text, delimiter-based encoding is very useful to reveal common features. As for plain text, content-based encoding schemes are necessary.

Given a set of delimiters, the encoding procedure translates each delimiter, `X`, in the input page as a delimiter token `X` and translates any text string between two delimiters as a special token `<TEXT>`. Such delimiter-based encoding schemes not only translate an input example into a token string, but also enforce a natural segmentation of the data by the delimiters. In other words, an encoded token string of length n can divide the data into n segments, where each `<TEXT>` tokens represent a text segment between two adjacent delimiter tokens. We call this segment the *primitive data* of the `<TEXT>` token.

If no apparent delimiters can be used, content-based encoding schemes such as semantic class information, part-of-speech tagging, can be used instead. For instance, semantic class can be added to words through WordNet dictionary, date-related strings such as "2002/9/1" can be recognized as DATE token sentences can be parsed into proper grammar notations such as `<Subject><verb><dobj>`. In addition, if annotated training examples are used, the extraction targets can be encoded as special token class dentated by its slot name. For example, consider the task of extracting both the target and perpetrator in the terrorism domain, the sentence "The parliament building was bombed by Carlos." can be encoded as

```
The <Target> was bombed by <Perpetrator>,
```

if no suitable semantic class is available for extraction targets.

4.2 Pattern Mining Algorithms

The problem of mining sequential patterns can be defined as follows. The input is a set of token strings that are encoded from training examples. A sequential pattern P is a sequence of tokens. A token string T is said to match a sequential pattern P if and only if every token in P occurs in T as the order in P and the distance of two adjacent tokens is no less than a predefined threshold τ. If the distance threshold τ equal 0, we call such sequential patterns string patterns. The support of a sequential pattern P is defined as the number of token strings that match P. Given a support threshold σ, the problem is to enumerate all sequences of tokens whose support is greater than σ.

This problem definition is a variation of sequential pattern mining in data mining area [32]. Therefore, we use an Apriori-based algorithm for this problem [3]. Other algorithms can also be applied. For example, if the distance threshold τ is set to zero, suffix trees can be applied to find string repeat patterns as in IEPAD [6].

Apriori begins by enumerating patterns that contain only one token and have supports greater than σ. Call these patterns frequent 1-sequence. Next, in each pass $i, i = 2, 3, \ldots$, Apriori performs two operations: the algorithm generates all potential candidates with length i in a hash tree denoted as C; next, the input is scanned so that the supports of the

Alcedo Atthis
嘴長、身短。頭至後頸爲暗綠色佈有白斑，
眼先、耳羽及腹面爲橙紅色。體背爲水藍色
、兩翼爲翠綠色。
Extraction Targets：
（嘴，長）、(身，短)、（頭部，暗綠）、
（後頸，暗綠）、（頭部，白斑）、
（後頸，白斑）、（眼先，橙紅）、
（耳羽，橙紅）、（腹面，橙紅）、
（體背，水藍）、（兩翼，翠綠）

Figure 1: Sample text and the extracted target

candidates can be counted. In the first operation, the candidates are generated by joining two frequent $i-1$-sequences found in previous pass. In the second operation, the input is scanned and, for each token string s, the counter of each subsequence of s that presents in C is incremented. The process completes when no candidates can be generated from previous frequent sequences.

4.3 Pattern Matching for Extractors

The extractors for this pattern mining approach execute a procedure that is similar to the training process. First, the encoding schemes are applied to input text. Next, the encoded token strings are matched against discovered sequential patterns. For string patterns, inexact or approximate matching is used to allow variations in the input. For example, k-mismatch (at most k mismatches) or k-difference (at most k mismatches and spaces) can be used. Finally, the most similar sequential patterns is chosen to extract information from the input.

5 An Example

In this section, we apply the pattern mining approach to the design of a Chinese IE task. The extraction task is defined as follows. The input is a free-form text from the database of Taiwan wild birds. Each paragraph describes appearance features about one specific bird. The extraction target is the set of attribute-value pairs. One characteristic of this free-form text is that most attribute-value pairs appear in one sentence. However, mulitple pairs may co-occur in one sentence. Figure 1 shows an input example about kingfisher and the extraction target. The extraction target can be viewed as a two-slot relation. The first slot is the attribute name such as head, eye, etc. The second slot is the attribute value such as short/long, big/small, colors, etc.

5.1 Encoding Schemes

Two encoding techniques are applied here. The first encoding scheme takes the advantage of the annotated training examples where extraction targets are labelled and denoted by respective slot names. For example, “眼先,耳羽及腹面爲橙紅色” will be encoded as “<AN>,<AN>及<AN>爲<AV>” where <AN> and <AV> denote attribute name and attribute value, respectively. Meanwhile, two dictionaries are constructed for attribute names and values, respectively. These two dictionaries are used for later encoding in the test phase.

The second encoding scheme deals with a typical problem in Chinese document processing. Since a Chinese sentence is composed of characters without specific word boundary, segmentation sometimes present problems in Chinese language processing. In order to identify individual words (a character string with an independent meaning), a syntactic analysis module is needed to parse the input sentence into a syntactic structure by using a parser. However, it is also well recognized that for IE in a limited domain, a full syntactic analysis or comprehensive semantic interpretation are not necessary. An alternative is to find significant lexical phrases (SLP) as proposed in [8]. SLPs are string patterns, called repeats, that occur at least twice in the input text. However, not all repeats are SLPs, Chien proposed the computation of mutual information to decide whether a repeat is a SLP.

5.2 Pattern Matching

After these encoding, Apriori-based pattern mining is applied to the encoded token strings. The distance threshold is set to zero to discover string patterns. Each frequent pattern is then given a score defined by the number of positive examples matched to the number of negative examples matched. Two example of the discovered string patterns are as follows.

<AN>至<AN>爲<AV>色

<AN>爲<AV>色

The extractor for this extraction task executes a procedure that is similar to the training process. First, the dictionaries for attribute name and value are used for slot encoding and instances of SLPs are encoded to avoid spurious segmentation. Next, the encoded token strings are matched against discovered frequent patterns allowing $k = 2$ differences. The validated string pattern with the highest score is then chosen to extract information from the input. For example, suppose “頭” and “眼” are already in the dictionary of attribute names, “白” and “黑” are in the dictionary of attribute values. Given an input text “頭至頸爲白色、眼先爲黑色”, it can be en-

coded as ‘‘<AN>至頸爲<AV>色、<AN>先<AV>色’’. For the first clause, string alignment (as shown below) will suggest that a mismatch of “頸” and <AN> may imply “頸” a attribute value.

<AN>	至	<AN>	爲	<AV>	色
<AN>	至	頸	爲	<AV>	色

Similarly, the alignment for the second clause might suggest that “眼先” is an attribute name if “眼先” is a SLP.

<AN>	爲	<AV>	色
<AN>先	爲	<AV>	色

5.3 Experimental Results

There are 548 birds in the Taiwan Wild Bird Database. During the preprocessing, the extraction targets are labelled and divided into sentences by periods, semicolons. A total of 1981 sentences are collected, which contain 2879 extraction pairs with 319 distinct attribute names and 457 attribute names. The leaning curves in precision and recall are shown in Figure 2. With 50, 100, 150 and 200 birds as training examples, the numbers of sentences are about 8, 18, 25, 35 percentage of the total sentences. With 25% training sentences, the performance achieves 97% precision and 80% recall at the training phrase, and 70% precision and 50% recall at the testing phrase.

6 Conclusion

In this paper, we divide the IE tasks into three categories: plain text, semi-structured, and free-form text, according to the input data. The input data together with the extraction targets defines the IE tasks. Understanding the extraction tasks help us estimate the difficulty of individual IE tasks and decide whether un-labelled training examples is possible. For example, it is possible to develop an IE system with un-annotated training example for semi-structure IE tasks. However, the difficulty to design such IE systems for free-text IE tasks is comparably high.

Background knowledge is the essential part of the extraction rules since common delimiters or content features are necessary for IE systems to generalize extraction rules, no matter they are presented in forms of regular expression, first-order logic constraints, or other classification models. Finally, machine learning or pattern mining algorithms can be applied to learn the extraction rules.

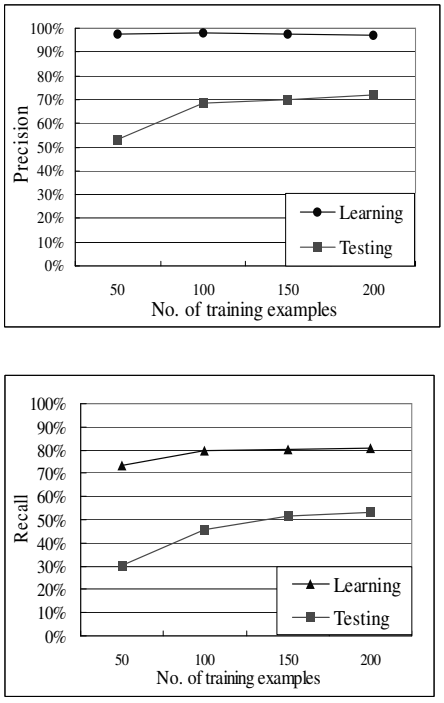

Figure 2: Performance of bird feature extraction

References

[1] Corporate management changes. http://www.cs.umanitoba.ca/ lindek/ie-ex.htm.

[2] Joint venture template fill rules. In *Plenary Session Notebook of the TIPSTER 12-month Meeting*, 1992.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th Conference on Very Large Databases*, Santiago, Chile, 1994.

[4] N. Ashish and C. Knoblock. Wrapper generation for semi-structured internet sources. *SIGMOD Record*, 26(4):8–15, 1997.

[5] M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1997.

[6] C.-H. Chang and S.-C. Lui. Iepad: Information extraction based on pattern discovery. In *Proceedings of the 10th International Conference on World Wide Web*, pages 681–688, Hong-Kong, May 2–6 2001.

[7] B. Chidlovskii, J. Ragetli, and M. Rijke. Automatic wrapper generation for web search engines. In *Proceedings of the 1st International Conference on Web-Age Information Man-*

agement (WAIM'2000), LNCS Series, Shanghai, China, 2000.

[8] L.F. Chien. Pat-tree-based keyword extraction for chinese information retrieval. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–58, Philadelphia, PA, 1997.

[9] H.-L. Chieu and H.-T. Ng. A maximum entropy approach to information extraction from semi-structured and free text. In *AAAI*, 2002.

[10] H. Cunningham. Information extraction – a user guide. Technical Report CS-97-02, Institute for Language, Speech and Hearing (ILASH) and Dept. of Computer Science, University of Sheffield, UK, 1997.

[11] R.B. Doorenbos, O. Etzioni, and D.S. Weld. A scalable comparison-shopping agent for the world-wide web. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 39–48, NewYork, USA, 1997.

[12] D. Embley, Y. Jiang, and Y.-K. Ng. Record-boundary discovery in web documents. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 467–478, Philadelphia, PA, 1999.

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

[14] B. Glasgow, A. Mandell, D. Binney, L. Ghemri, and D. Fisher. Mita: An information extraction approach to analysis of free-form text in life insurance applications. In *AAAI*, 1997.

[15] R. Grishman and B. Sundheim. Message understanding conference – 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Jun 1996.

[16] C.-N. Hsu and C.-C. Chang. Finite-state transducers for semi-structured text mining. In *Proceedings of IJCAI-99 Workshop on Text mining: Foundations, Techniques and Applications*, pages 38–49, Stockholm, Sweden, 1999.

[17] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[18] C. Knoblock, S. Minton, and et al. J. Ambite. Modeling web sources for information integration. In *Proceedings of the 15th National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*, pages 211–218, Wisconsin, USA, 1998.

[19] N. Kushmerick. Gleaning the web. *IEEE Intelligent Systems*, 14(2):20–22, Mar/Apr 1999.

[20] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, Japan, 1997.

[21] W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. University of massachusetts: Description of the circus system as used for muc-4. pages 282–288, 1992.

[22] W.-Y. Lin and W. Lam. Learning to extract hierarchical information from semi-structured documents. In *Proceedings of the 2000 ACM CIKM International Conference on Information and Knowledge Management*, pages 250–257, VA, USA, 2000.

[23] S. Manganaris and J. Ortega. Be warned – in time to take action. *DB2 Magazine*, 7(4):10–14, 2002.

[24] G. Miller. Wordnet: A lexical database for english. *Communication of ACM*, 38(11):39–41, 1995.

[25] I. Muslea. Rise: Repository of test domains for information extraction. http://www.isi.edu/ muslea/RISE/repository.html.

[26] I. Muslea, S. Minton, and C. Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the 3rd International Conference on Autonomous Agents*, pages 190–197, Seattle, WA, 1999.

[27] E. Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816, 1993.

[28] E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1996.

[29] A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *Data and Knowledge Engineering*, 36(3):283–316, 2001.

[30] S. Soderland. Learning information extraction rules for semi-structured and free text. *Journal of Machine Learning*, 34(1-3):233–272, 1999.

[31] S. Soderland, D. Fisher, J. Aseltine, and W. Lehnert. Crystal: Inducing a conceptual dictionary. pages 1314–1319, 1995.

[32] R. Srikant. *Fast algorithms for mining association rules and sequential patterns.* PhD thesis, University of Wisconsin-Madison, 1996.