A Task-based Comparison of Information Extraction Pattern Models

Mark A. Greenwood and **Mark Stevenson**
Department of Computer Science
University of Sheffield
Sheffield, S1 4DP, UK
{m.greenwood, marks}@dcs.shef.ac.uk

Abstract

Several recent approaches to Information Extraction (IE) have used dependency trees as the basis for an extraction pattern representation. These approaches have used a variety of pattern models (schemes which define the parts of the dependency tree which can be used to form extraction patterns). Previous comparisons of these pattern models are limited by the fact that they have used indirect tasks to evaluate each model. This limitation is addressed here in an experiment which compares four pattern models using an unsupervised learning algorithm and a standard IE scenario. It is found that there is a wide variation between the models' performance and suggests that one model is the most useful for IE.

1 Introduction

A common approach to Information Extraction (IE) is to (manually or automatically) create a set of patterns which match against text to identify information of interest. Muslea (1999) reviewed the approaches which were used at the time and found that the most common techniques relied on lexico-syntactic patterns being applied to text which has undergone relatively shallow linguistic processing. For example, the extraction rules used by Soderland (1999) and Riloff (1996) match text in which syntactic chunks have been identified. More recently researchers have begun to employ deeper syntactic analysis, such as dependency parsing (Yangarber et al., 2000; Stevenson and Greenwood, 2005; Sudo et al., 2001; Sudo et al., 2003; Yangarber, 2003). In these approaches extraction patterns are essentially parts of the dependency tree. To perform extraction they are compared against the dependency analysis of a sentence to determine whether it contains the pattern.

Each of these approaches relies on a *pattern model* to define which parts of the dependency tree can be used to form the extraction patterns. A variety of pattern models have been proposed. For example the patterns used by Yangarber et al. (2000) are the subject-verb-object tuples from the dependency tree (the remainder of the dependency parse is discarded) while Sudo et al. (2003) allow any subtree within the dependency parse to act as an extraction pattern. Stevenson and Greenwood (2006) showed that the choice of pattern model has important implications for IE algorithms including significant differences between the various models in terms of their ability to identify information of interest in text.

However, there has been little comparison between the various pattern models. Those which have been carried out have been limited by the fact that they used indirect tasks to evaluate the various models and did not compare them in an IE scenario. We address this limitation here by presenting a direct comparison of four previously described pattern models using an unsupervised learning method applied to a commonly used IE scenario.

The remainder of the paper is organised as follows. The next section presents four pattern models which have been previously introduced in the litera-

Proceedings of the 5th Workshop on Important Unresolved Matters, pages 81–88,
Ann Arbor, June 2005.

ture. Section 3 describes two previous studies which compared these models and their limitations. Section 4 describes an experiment which compares the four models on an IE task, the results of which are described in Section 5. Finally, Section 6 discusses the conclusions which may be drawn from this work.

2 IE Pattern Models

In dependency analysis (Mel'čuk, 1987) the syntax of a sentence is represented by a set of directed binary links between a word (the head) and one of its modifiers. These links may be labelled to indicate the relation between the head and modifier (e.g. subject, object). An example dependency analysis for the sentence *"Acme hired Smith as their new CEO, replacing Bloggs."* is shown Figure 1.

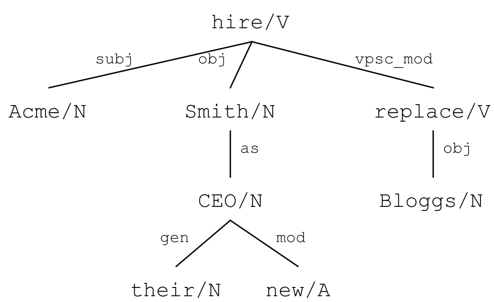

Figure 1: An example dependency tree.

The remainder of this section outlines four models for representing extraction patterns which can be derived from dependency trees.

Predicate-Argument Model (SVO): A simple approach, used by Yangarber et al. (2000), Yangarber (2003) and Stevenson and Greenwood (2005), is to use subject-verb-object tuples from the dependency parse as extraction patterns. These consist of a verb and its subject and/or direct object. Figure 2 shows the two SVO patterns[1] which are produced for the dependency tree shown in Figure 1.

This model can identify information which is expressed using simple predicate-argument constructions such as the relation between *Acme* and *Smith* in the dependency tree shown in Figure 1. However, the SVO model cannot represent information described using other linguistic constructions such as nominalisations or prepositional phrases. For example the SVO model would not be able to recognise that Smith's new job title is CEO since these patterns ignore the part of the dependency tree containing that information.

Chains: A pattern is defined as a path between a verb node and any other node in the dependency tree passing through zero or more intermediate nodes (Sudo et al., 2001). Figure 2 shows examples of the chains which can be extracted from the tree in Figure 1.

Chains provide a mechanism for encoding information beyond the direct arguments of predicates and includes areas of the dependency tree ignored by the SVO model. For example, they can represent information expressed as a nominalisation or within a prepositional phrase, e.g. "*The resignation of Smith from the board of Acme ...*" However, a potential shortcoming of this model is that it cannot represent the link between arguments of a verb. Patterns in the chain model format are unable to represent even the simplest of sentences containing a transitive verb, e.g. *"Smith left Acme"*.

Linked Chains: The linked chains model (Greenwood et al., 2005) represents extraction patterns as a pair of chains which share the same verb but no direct descendants. Example linked chains are shown in Figure 2. This pattern representation encodes most of the information in the sentence with the advantage of being able to link together event participants which neither of the SVO or chain model can, for example the relation between *"Smith"* and *"Bloggs"* in Figure 1.

Subtrees: The final model to be considered is the subtree model (Sudo et al., 2003). In this model any subtree of a dependency tree can be used as an extraction pattern, where a subtree is any set of nodes in the tree which are connected to one another. Single nodes are not considered to be subtrees. The subtree model is a richer representation than those discussed so far and can represent any part of a dependency tree. Each of the previous models form a proper subset of the subtrees. By choosing an appropriate subtree it is possible to link together any pair of nodes in a tree and consequently this model can

[1]The formalism used for representing dependency patterns is similar to the one introduced by Sudo et al. (2003). Each node in the tree is represented in the format `a[b/c]` (e.g. `subj[N/Acme]`) where `c` is the lexical item (`Acme`), `b` its grammatical tag (`N`) and `a` the dependency relation between this node and its parent (`subj`). The relationship between nodes is represented as `X(A+B+C)` which indicates that nodes `A`, `B` and `C` are direct descendents of node `X`.

SVO

```
[V/hire](subj[N/Acme]+obj[N/Smith])
[V/replace](obj[N/Bloggs])
```

Chains

```
[V/hire](subj[N/Acme])
[V/hire](obj[N/Smith])
[V/hire](obj[N/Smith](as[N/CEO]))
[V/hire](obj[N/Smith](as[N/CEO](gen[N/their])))
```

Linked Chains

```
[V/hire](subj[N/Acme]+obj[N/Smith])
[V/hire](subj[N/Acme]+obj[N/Smith](as[N/CEO]))
[V/hire](obj[N/Smith]+vpsc_mod[V/replace](obj[N/Bloggs]))
```

Subtrees

```
[V/hire](subj[N/Acme]+obj[N/Smith]+vpsc_mod[V/replace])
[V/hire](subj[N/Acme]+vpsc_mod[V/replace](obj[N/Bloggs]))
[N/Smith](as[N/CEO](gen[N/their]+mod[A/new]))
```

Figure 2: Example patterns for four models

represent the relation between any set of items in the sentence.

3 Previous Comparisons

There have been few direct comparisons of the various pattern models. Sudo et al. (2003) compared three models (SVO, chains and subtrees) on two IE scenarios using a entity extraction task. Models were evaluated in terms of their ability to identify entities taking part in events and distinguish them from those which did not. They found the SVO model performed poorly in comparison with the other two models and that the performance of the subtree model was generally the same as, or better than, the chain model. However, they did not attempt to determine whether the models could identify the relations between these entities, simply whether they could identify the entities participating in relevant events.

Stevenson and Greenwood (2006) compared the four pattern models described in Section 2 in terms of their complexity and ability to represent relations found in text. The complexity of each model was analysed in terms of the number of patterns which would be generated from a given dependency parse. This is important since several of the algorithms which have been proposed to make use of dependency-based IE patterns use iterative learning (e.g. (Yangarber et al., 2000; Yangarber, 2003; Stevenson and Greenwood, 2005)) and are unlikely to cope with very large sets of candidate patterns. The number of patterns generated therefore has an effect on how practical computations using that model may be. It was found that the number of patterns generated for the SVO model is a linear function of the size of the dependency tree. The number of chains and linked chains is a polynomial function while the number of subtrees is exponential.

Stevenson and Greenwood (2006) also analysed the representational power of each model by measuring how many of the relations found in a standard IE corpus they are expressive enough to represent. (The documents used were taken from newswire texts and biomedical journal articles.) They found that the SVO and chain model could only represent a small proportion of the relations in the corpora. The subtree model could represent more of the relations than any other model but that there was no statistical difference between those relations and the ones covered by the linked chain model. They concluded that the linked chain model was optional since it is expressive enough to represent the information of interest without introducing a potentially unwieldy number of patterns.

There is some agreement between these two studies, for example that the SVO model performs poorly in comparison with other models. However, Stevenson and Greenwood (2006) also found that the coverage of the chain model was significantly worse than the subtree model, although Sudo et al.

(2003) found that in some cases their performance could not be distinguished. In addition to these disagreements, these studies are also limited by the fact that they are indirect; they do not evaluate the various pattern models on an IE task.

4 Experiments

We compared each of the patterns models described in Section 2 using an unsupervised IE experiment similar to one described by Sudo et al. (2003).

Let D be a corpus of documents and R a set of documents which are relevant to a particular extraction task. In this context "relevant" means that the document contains the information we are interested in identifying. D and R are such that $D = R \cup \bar{R}$ and $R \cap \bar{R} = \emptyset$. As assumption behind this approach is that useful patterns will be far more likely to occur in R than D overall.

4.1 Ranking Patterns

Patterns for each model are ranked using a technique inspired by the tf-idf scoring commonly used in Information Retrieval (Manning and Schütze, 1999). The score for each pattern, p, is given by:

$$score(p) = tf_p \times \left(\frac{N}{df_p}\right)^{\beta} \tag{1}$$

where tf_p is the number of times pattern p appears in relevant documents, N is the total number of documents in the corpus and df_p the number of documents in the collection containing the pattern p.

Equation 1 combines two factors: the *term frequency* (in relevant documents) and *inverse document frequency* (across the corpus). Patterns which occur frequently in relevant documents without being too prevalent in the corpus are preferred. Sudo et al. (2003) found that it was important to find the appropriate balance between these two factors. They introduced the β parameter as a way of controlling the relative contribution of the *inverse document frequency*. β is tuned for each extraction task and pattern model combination.

Although simple, this approach has the advantage that it can be applied to each of the four pattern models to provide a direct comparison.

4.2 Extraction Scenario

The ranking process was applied to the IE scenario used for the sixth Message Understanding conference (MUC-6). The aim of this task was to identify management succession events from a corpus of newswire texts. Relevant information describes an executive entering or leaving a position within a company, for example *"Last month Smith resigned as CEO of Rooter Ltd."*. This sentence described as event involving three items: a person (*Smith*), position (*CEO*) and company (*Rooter Ltd*). We made use of a version of the MUC-6 corpus described by Soderland (1999) which consists of 598 documents.

For these experiments relevant documents were identified using annotations in the corpus. However, this is not necessary since Sudo et al. (2003) showed that adequate knowledge about document relevance could be obtained automatically using an IR system.

4.3 Pattern Generation

The texts used for these experiments were parsed using the Stanford dependency parser (Klein and Manning, 2002). The dependency trees were processed to replace the names of entities belonging to specific semantic classes with a general token. Three of these classes were used for the management succession domain (`PERSON`, `ORGANISATION` and `POST`). For example, in the dependency analysis of *"Smith will became CEO next year"*, *"Smith"* is replaced by `PERSON` and *"CEO"* by `POST`. This process allows more general patterns to be extracted from the dependency trees. For example, `[V/become](subj[N/PERSON]+obj[N/POST])`. In the MUC-6 corpus items belonging to the relevant semantic classes are already identified.

Patterns for each of the four models were extracted from the processed dependency trees. For the SVO, chain and linked chain models this was achieved using depth-first search. However, the enumeration of all subtrees is less straightforward and has been shown to be a $\#P$-complete problem (Goldberg and Jerrum, 2000). We made use of the *rightmost extension* algorithm (Abe et al., 2002; Zaki, 2002) which is an efficient way of enumerating all subtrees. This approach constructs subtrees iteratively by combining together subtrees which have already been observed. The algorithm starts with a set of trees, each of which consists of a single node. At each stage the known trees are extended by the addition of a single node. In order to avoid duplication the extension is restricted to allowing nodes only to be added to the nodes on the rightmost path of the tree. Applying the process recursively creates a search space in which all subtrees are enumerated with minimal duplication.

The rightmost extension algorithm is most suited to finding subtrees which occur multiple times and, even using this efficient approach, we were unable to generate subtrees which occurred fewer than four times in the MUC-6 texts in a reasonable time. Similar restrictions have been encountered within other approaches which have relied on the generation of a comprehensive set of subtrees from a parse forest. For example, Kudo et al. (2005) used subtrees for parse ranking but could only generate subtrees which appear at least ten times in a 40,000 sentence corpus. They comment that the size of their data set meant that it would have been difficult to complete the experiments with less restrictive parameters. In addition, Sudo et al. (2003) only generated subtrees which appeared in at least three documents. Kudo et al. (2005) and Sudo et al. (2003) both used the rightmost extension algorithm to generate subtrees.

To provide a direct comparison of the pattern models we also produced versions of the sets of patterns extracted for the SVO, chain and linked chain models in which patterns which occurred fewer than four times were removed. Table 1 shows the number of patterns generated for each of the four models when the patterns are both filtered and unfiltered. (Although the set of unfiltered subtree patterns were not generated it is possible to determine the number of patterns which would be generated using a process described by Stevenson and Greenwood (2006).)

Model	Filtered	Unfiltered
SVO	9,189	23,128
Chains	16,563	142,019
Linked chains	23,452	493,463
Subtrees	369,453	1.69×10^{12}

Table 1: Number of patterns generated by each model

It can be seen that the various pattern models generate vastly different numbers of patterns and that the number of subtrees is significantly greater than the other three models. Previous analysis (see Section 3) suggested that the number of subtrees which would be generated from a corpus could be difficult to process computationally and this is supported by our findings here.

4.4 Parameter Tuning

The value of β in equation 1 was set using a separate corpus from which the patterns were generated, a methodology suggested by Sudo et al. (2003). To generate this additional text we used the Reuters Corpus (Rose et al., 2002) which consists of a year's worth of newswire output. Each document in the Reuters corpus has been manually annotated with topic codes indicating its general subject area(s). One of these topic codes (`C411`) refers to management succession events and was used to identify documents which are relevant to the MUC6 IE scenario. A corpus consisting of 348 documents annotated with code `C411` and 250 documents without that code, representing irrelevant documents, were taken from the Reuters corpus to create a corpus with the same distribution of relevant and irrelevant documents as found in the MUC-6 corpus. Unlike the MUC-6 corpus, items belonging to the required semantic classes are not annotated in the Reuters Corpus. They were identified automatically using a named entity identifier.

The patterns generated from the MUC-6 texts were ranked using formula 1 with a variety of values of β. These sets of ranked patterns were then used to carry out a document filtering task on the Reuters corpus - the aim of which is to differentiate documents based on whether or not they contain a relation of interest. The various values for β were compared by computing the area under the curve. It was found that the optimal value for β was 2 for all pattern models and this setting was used for the experiments.

4.5 Evaluation

Evaluation was carried out by comparing the ranked lists of patterns against the dependency trees for the MUC-6 texts. When a pattern is found to match against a tree the items which match any semantic classes in the pattern are extracted. These items are considered to be related and compared against the gold standard data in the corpus to determine whether they are in fact related.

The precision of a set of patterns is computed as the proportion of the relations which were identified that are listed in the gold standard data. The recall is the proportion of relations in the gold standard data which are identified by the set of patterns.

The ranked set of patterns are evaluated incrementally with the precision and recall of the first (highest ranked) pattern computed. The next pattern is then added to the relations extracted by both are evaluated. This process continues until all patterns are exhausted.

5 Results

Figure 3 shows the results when the four filtered pattern models, ranked using equation 1, are compared.

A first observation is that the chain model performs poorly in comparison to the other three models. The highest precision achieved by this model is 19.9% and recall never increases beyond 9%. In comparison the SVO model includes patterns with extremely high precision but the maximum recall achieved by this model is low. Analysis showed that the first three SVO patterns had very high precision. These were `[V/succeed](subj[N/PERSON]+obj[N/PERSON])`, `[V/be](subj[N/PERSON]+obj[N/POST])` and `[V/become](subj[N/PERSON]+obj[N/POST])`, which have precision of 90.1%, 80.8% and 78.9% respectively. If these high precision patterns are removed the maximum precision of the SVO model is around 32%, which is comparable with the linked chain and subtree models. This suggests that, while the SVO model includes very useful extraction patterns, the format is restrictive and is unable to represent much of the information in this corpus.

The remaining two pattern models, linked chains and subtrees, have very similar performance and each achieves higher recall than the SVO model, albeit with lower precision. The maximum recall obtained by the linked chain model is slightly lower than the subtree model but it does maintain higher precision at higher recall levels.

The maximum recall achieved by all four models is very low in this evaluation and part of the reason for this is the fact that the patterns have been filtered to allow direct comparison with the subtree model. Figure 4 shows the results when the unfiltered SVO, chain and linked chain patterns are used. (Performance of the filtered subtrees are also included in this graph for comparison.)

This result shows that the addition of extra patterns for each model improves recall without effecting the maximum precision achieved. The chain model also performs badly in this experiment. Precision of the SVO model is still high (again this is due to the same three highly accurate patterns) however the maximum recall achieved by this model is not particularly increased by the addition of the unfiltered patterns. The linked chain model benefits most from the unfiltered patterns. The extra patterns lead to a maximum recall which is more than double any of the other models without overly degrading precision. The fact that the linked chain model is able to achieve such a high recall shows that it is able to represent the relations found in the MUC-6 text, unlike the SVO and chain models. It is likely that the subtrees model would also produce a set of patterns with high recall but the number of potential patterns which are allowable within this model makes this impractical.

6 Discussion and Conclusions

Some of the results reported for each model in these experiments are low. Precision levels are generally below 40% (with the exception of the SVO model which achieves high precision using a small number of patterns). One reason for this that the the patterns were ranked using a simple unsupervised learning algorithm which allowed direct comparison of four different pattern models. This approach only made use of information about the distribution of patterns in the corpus and it is likely that results could be improved for a particular pattern model by employing more sophisticated approaches which make use of additional information, for example the structure of the patterns.

The results presented here provide insight into the usefulness of the various pattern models by evaluating them on an actual IE task. It is found that SVO patterns are capable of high precision but that the

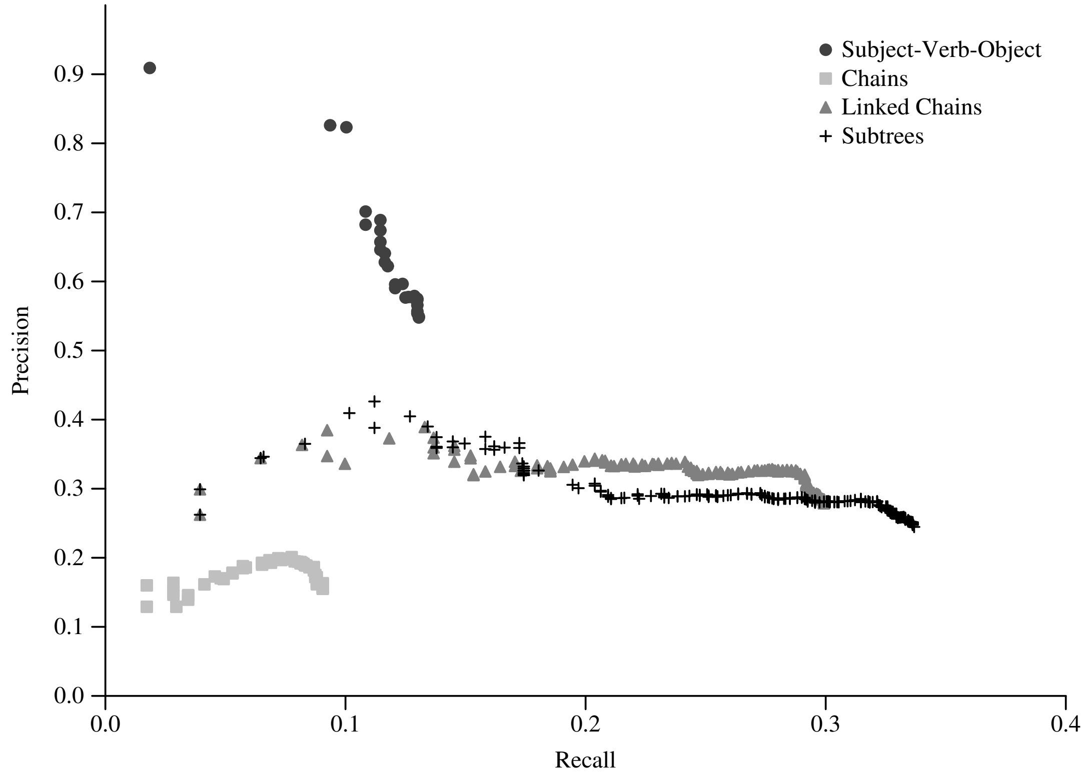

Figure 3: Comparisons of filtered pattern models.

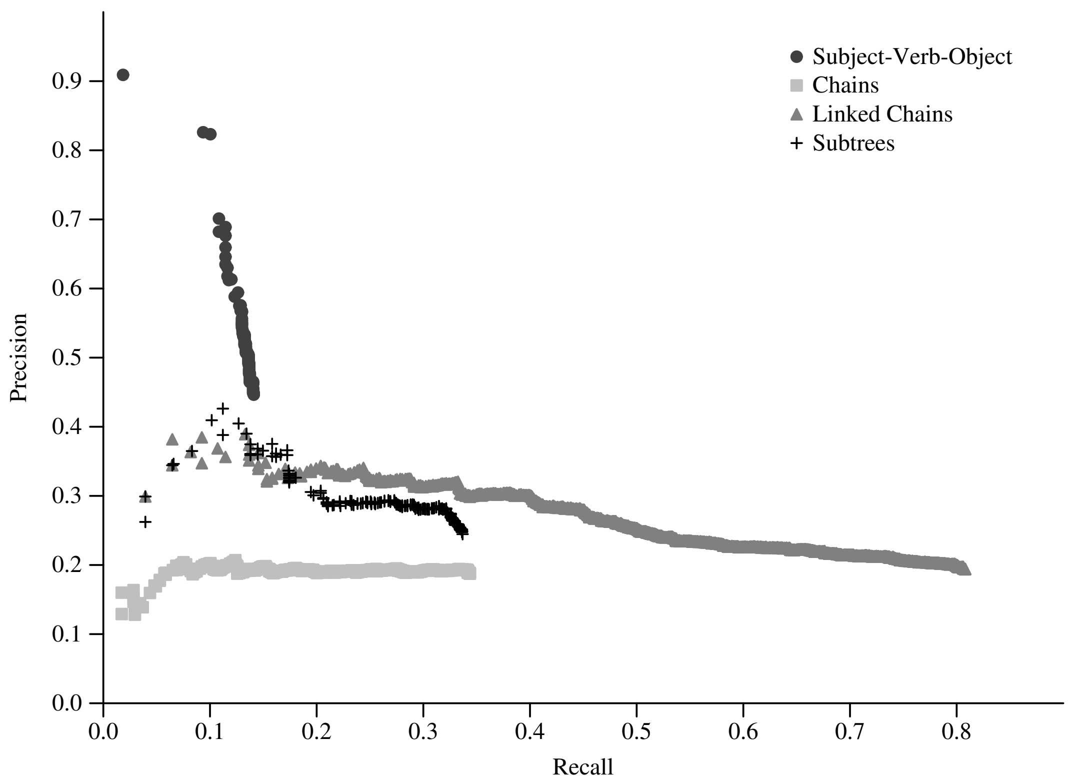

Figure 4: Comparison of unfiltered models.

restricted set of possible patterns leads to low recall. The chain model was found to perform badly with low recall and precision regardless of whether the patterns were filtered. Performance of the linked chain and subtree models were similar when the patterns were filtered but unfiltered linked chains were capable of achieving far higher recall than the filtered subtrees.

These experiments suggest that the linked chain model is a useful one for IE since it is simple enough for an unfiltered set of patterns to be extracted and able to represent a wider range of information than the SVO and chain models.

References

Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. 2002. Optimised Substructure Discovery for Semi-Structured Data. In *Proceedings of the 6th European Conference on Principles and Practice of Knowledge in Databases (PKDD-2002)*, pages 1–14.

Leslie Ann Goldberg and Mark Jerrum. 2000. Counting Unlabelled Subtrees of a Tree is $\#P$-Complete. *London Mathmatical Society Journal of Computation and Mathematics*, 3:117–124.

Mark A. Greenwood, Mark Stevenson, Yikun Guo, Henk Harkema, and Angus Roberts. 2005. Automatically Acquiring a Linguistically Motivated Genic Interaction Extraction System. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn, Germany.

Dan Klein and Christopher D. Manning. 2002. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, Vancouver, Canada.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based Parse Reranking with Subtree Features. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Ann Arbour, MI.

Chritopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

Igor Mel'čuk. 1987. *Dependency Syntax: Theory and Practice*. SUNY Press, New York.

Ion Muslea. 1999. Extraction Patterns for Information Extraction: A Survey. In *Proceedings of the AAAI-99 workshop on Machine Learning for Information Extraction*, Orlando, FL.

Ellen Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1044–1049, Portland, OR.

Tony Rose, Mark Stevenson, and Miles Whitehead. 2002. The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-02)*, pages 827–832, La Palmas de Gran Canaria.

Stephen Soderland. 1999. Learning Information Extraction Rules for Semi-structured and Free Text. *Machine Learning*, 31(1-3):233–272.

Mark Stevenson and Mark A. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 379–386, Ann Arbor, MI.

Mark Stevenson and Mark A. Greenwood. 2006. Comparing Information Extraction Pattern Models. In *Proceedings of the Information Extraction Beyond The Document Workshop (COLING/ACL 2006)*, pages 12–19, Sydney, Australia.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2001. Automatic Pattern Acquisition for Japanese Information Extraction. In *Proceedings of the Human Language Technology Conference (HLT2001)*, San Diego, CA.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An Improved Extraction Pattern Representation Model for Automatic IE Pattern Acquisition. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 224–231, Sapporo, Japan.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised Discovery of Scenario-level Patterns for Information Extraction. In *Proceedings of the Applied Natural Language Processing Conference (ANLP 2000)*, pages 282–289, Seattle, WA.

Roman Yangarber. 2003. Counter-training in the Discovery of Semantic Patterns. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, pages 343–350, Sapporo, Japan.

Mohammed Zaki. 2002. Effectively Mining Frequent Trees in a Forest. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, Edmonton, Canada.