Dependency Pattern Models for Information Extraction

Mark Stevenson and Mark A. Greenwood
Department of Computer Science
The University of Sheffield
Regent Court, 211 Portobello
Sheffield, S1 4DP
United Kingdom
{M.Stevenson, M.Greenwood}@dcs.shef.ac.uk

Abstract

Several techniques for the automatic acquisition of Information Extraction (IE) systems have used dependency trees to form the basis of an extraction pattern representation (Yangarber, Grishman, Tapanainen, and Huttunen 2000a; Yangarber 2003; Sudo, Sekine, and Grishman 2003; Bunescu and Mooney 2005; Stevenson and Greenwood 2005). These approaches have used a variety of pattern models (schemes for representing IE patterns based on particular parts of the dependency analysis). An appropriate pattern model should be expressive enough to represent the information which is to be extracted from text without being overly complex. Previous investigations into the appropriateness of the currently proposed models have been limited. This paper compares a variety of pattern models, including ones which have been previously reported and variations of them. Each model is evaluated using existing data consisting of IE scenarios from two very different domains (newswire stories and biomedical journal articles). The models are analysed in terms of their ability to represent relevant information, number of patterns generated and performance on an IE scenario. It was found that the best performance was observed from two models which use the majority of relevant portions of the dependency tree without including irrelevant sections.

Keywords: complexity, dependency analysis, evaluation, expressivity, Information Extraction, parsing, relation extraction

1 Introduction

The problem of achieving an appropriate balance between the complexity and expressive completeness of a system which models natural language is well known, for example (Pullum and Gazdar 1982; Fox and Lappin 2005). Representation schemes should be rich enough to capture phenomena accurately but will be less useful if more complex than necessary.

Recent research in Information Extraction (IE) has used semantic patterns based on dependency trees to identify items of interest in text (Yangarber 2003; Sudo, Sekine, and Grishman 2003; Bunescu and Mooney 2005). These semantic patterns are based on a variety of pattern models, specific portions of the dependency analysis which can be used as patterns. Previous comparison of these pattern models is limited. This paper provides an analysis of various pattern models, including those previously introduced in the literature and variations of them, in terms of their complexity, expressivity and performance in an extraction scenario. In this context complexity is analysed in terms of the number of possible patterns which could be generated from a dependency tree and expressivity as the portion of relations which the pattern models are able to represent in standard IE corpora from two very different textual domains, management succession and biomedical text.

The analysis shows that there is a wide variation between the models' performance. We find that the best models include enough information from the dependency tree to identify relevant information while only requiring a fraction of the patterns which may be generated for other models.

The remainder of this article is organised as follows: the next section provides an overview of the use of semantic patterns based on dependency trees in IE. Six pattern models are described (Section 3) and their complexity analysed (Section 4). Section 5 describes experiments comparing the expressivity of each model while the performance of the most promising models on an extraction scenario are described in Section 6. Implications of this work are discussed in Section 7.

2 Background

A variety of dependency parsers are now readily available (Tapanainen and Järvinen 1997; Lin 1999; Briscoe and Carroll 2002; Nivre and Scholz 2004) and dependencies can often be generated from the analyses generated by parsers based on other formalisms, including phrase structure (Klein and Manning 2003) and Combinatory Categorical Grammar (Clark and Curran 2008). Dependency trees are considered to be a suitable basis for semantic patterns because they abstract away from the surface structure to represent the relations between elements of the sentence. This has been demonstrated by several recent applications to a diverse set of problems including the discovery of inference rules for question answering (Lin and Pantel 2001) and paraphrase identification (Szpektor, Tanev, Dagan, and Coppola 2004). They have also been widely applied in IE, for example (Yangarber 2003; Sudo, Sekine, and Grishman 2001; Sudo, Sekine, and Grishman 2003; Bunescu and Mooney 2005; Stevenson and Greenwood 2005; Romano, Kouylekov, Szpektor, Dagan, and Lavelli 2006). These systems have concentrated on relation extraction, the identification of connections between pairs of items in the text which may be words or short phrases. For example, the sentence *"Professor Smith has worked for Stanford University since 1985."* describes the relation between a person (*Professor Smith*) and their employer (*Stanford University*). A major difference between these approaches is that they rely on a diverse set of pattern models. For example, Yangarber (2003) uses subject-verb-object tuples as patterns while Sudo et al. (2003) allow any subpart of the analysis to be a pattern. These approaches also use machine leaning to generate a set of patterns that can identify instances of relations. Approaches include iterative semi-supervised algorithms (Yangarber, Grishman, Tapanainen, and Huttunen 2000b; Yangarber, Grishman, Tapanainen, and Huttunen 2000a; Yangarber 2003; Stevenson and Greenwood 2005; Greenwood and Stevenson 2006) where the algorithm is provided with a small set of useful patterns and then attempts to create an IE system by finding similar ones. Various methods have been used to determine pattern similarity including comparison of pattern distribution in corpora (Yangarber, Grishman, Tapanainen, and Huttunen 2000b; Yangarber, Grishman, Tapanainen, and Huttunen 2000a; Yangarber 2003) and examination of the patterns' structure to identify ones with similar meanings (Stevenson and Greenwood 2005; Greenwood and Stevenson 2006). Sudo et al. (2001)(2003) also used information about the distribution of patterns in corpora although they used an unsupervised algorithm which ranked patterns according to whether they tended to occur in documents which were known to contain information of interest. Support Vector Machines have also been used (Zelenko, Aone, and Richardella 2003; Culotta and Sorensen 2004; Bunescu and Mooney 2005). In this supervised approach the algorithm learns to classify example patterns as useful or otherwise based on the training examples.

The task of all learning algorithms becomes less practical if there are a large number of patterns. For approaches which make use of the structure of patterns, including (Stevenson and Greenwood 2005; Greenwood and Stevenson 2006; Zelenko, Aone, and Richardella 2003; Culotta and Sorensen 2004), the task may also be made more difficult as the structure of the patterns becomes more complex. In addition, overly complex pattern models which contain more information than necessary are likely to lead to overfitting of supervised algorithms, such as Support Vector Machines.

Sudo et al. (2003) indirectly compared three models by determining how well they could identify the participants of events but did not explore whether they were capable of identifying the relations between them. However, there has been no direct analysis of the various pattern models and, in particular, there has been no investigation into their expressive completeness (ability to accurately represent relations) and complexity (for example the number of patterns generated). We address this here by analysing six pattern models, five of which have been previously introduced in the literature and another which is an extension of one of those models. The models are analysed in terms of their ability to represent the relations which are found in a variety of IE scenarios, the number of patterns which are generated (which may determine whether the model can be practically applied within a learning system) and their performance in an IE scenario.

3 Pattern Models

This section provides details of the six pattern models which are compared in this paper. Figure 1 shows a dependency tree for the sentence *"Acme Inc. hired Mr Smith as their new CEO, replacing Mr Bloggs."* and will be used as a running example. This sentence describes several relations such as the ones between a person and their employer (*Mr. Smith* - *Acme Inc.*; *Mr. Bloggs* - *Acme Inc.*), a person and their job title (*Mr. Smith* - *CEO*; *Mr. Bloggs* - *CEO*) and two people changing jobs (*Mr. Smith* - *Mr. Bloggs*).

Predicate-Argument (SVO): A simple approach, used by 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[1]. An SVO pattern is extracted for each verb in a sentence so this model produces

[1]Yangarber et al. (2000a) and Sudo et al. (2003) use a slightly extended version of this model in

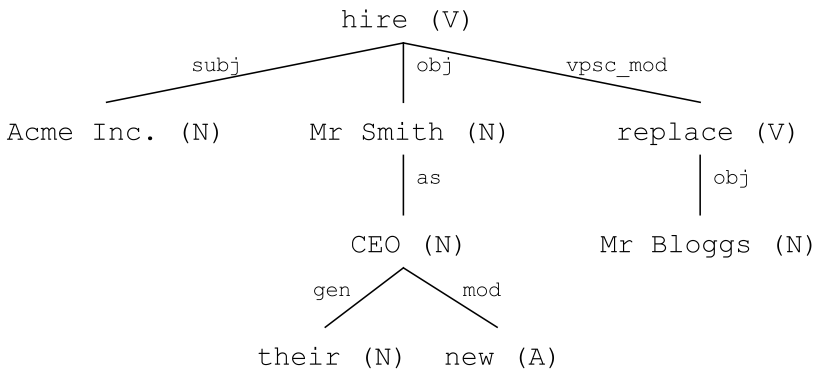

Figure 1: Example dependency tree.

two patterns for the dependency tree shown in Figure 1:

`[V/hire](subj[N/Acme Inc.]+obj[N/Mr Smith])`

`[V/replace](obj[N/Mr Bloggs])`[2].

This model may be motivated by the assumption that many IE scenarios involve the extraction of participants in specific events. For example, the management succession scenario used in the Sixth Message Understanding Conference (MUC-6) (MUC 1995) concerns the identification of individuals who are changing job including details such as the organisation and job title. This information is often described using a simple predicate argument structure, e.g. *"Acme Inc. fired Smith"*. However, the SVO model cannot represent information described using a wide variety of linguistic constructions such as nominalisations and prepositional phrases. For example, in the MUC-6 texts it is common for job titles to be mentioned within prepositional phrases (e.g. *"Smith joined Acme Inc. as CEO"*) so the relation between the person (*Smith*) and their job title (*CEO*) cannot be represented by this model.

Previous work which used this model has largely relied on indirect evaluation, such as document and sentence filtering (Yangarber, Grishman, Tapanainen, and Huttunen 2000a; Stevenson and Greenwood 2005). This paper fully analyses the suitability of this model for IE.

Chain: A pattern is defined as the direct path between a verb node and any of

which the pattern also includes certain phrases "which referred to either the subject or object."

[2]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/bomber]`) where `c` is the lexical item (`bomber`), `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 descendants of node `X`.

its descendants, passing through zero or more intermediate nodes (Sudo, Sekine, and Grishman 2001).

Eight chains can be extracted from the tree in Figure 1:

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

The chain model provides 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, for example the relation between *Smith* and *Acme* in "*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 such as when the same relation is expressed as *"Smith left Acme Inc."*

Linked Chain: The linked chain model, introduced by Greenwood et al. (2005), represents extraction patterns as a pair of chains which share the same verb as their root but do not share any direct descendants. This pattern model encodes much of the information in the sentence with the advantage of being able to link together event participants which the SVO or chain models are unable to, for example the relation between *"Smith"* and *"Bloggs"* in Figure 1.

This model generates 14 patterns for the verb *hire* in Figure 1, examples of which include:

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

Unconstrained Linked Chain (ULC): Greenwood et al. (2005) stipulate that the root of each pair of chains must be a verb. A simple extension of this model is to remove this restriction and allow patterns which are rooted at any node. A total of 32 unconstrained linked chain patterns would be generated for the tree in Figure 1 including:

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

Shortest Path: An alternative model, introduced by Bunescu and Mooney (2005), allows a pattern to be formed by following the shortest route in the dependency tree between any pair of nodes. The shortest path model includes all ULC and chain patterns. The 28 shortest path patters which could be generated from the example dependency tree include:

```
[V/hire](obj[N/Mr Smith](as[N/CEO]))
[N/Mr Smith](as[N/CEO](mod[A/new]))
[V/hire](subj[N/Acme Inc.]+obj[N/Mr Smith](as[N/CEO]))
[N/CEO](gen[N/their]+mod[A/new])
```

Subtree: The final pattern model to be considered is the subtree model, introduced by Sudo et al. (2003). In this model any subtree of a dependency tree can be used as an extraction pattern, where a subtree defined as any connected subset (possibly improper) of nodes in the tree. Single nodes are not considered to be subtree patterns. The subtree model is a richer representation than those discussed so far and can represent any part of a dependency tree. The patterns generated by each of the previous models form proper subsets of those generated by the subtree model. The 43 subtree patterns which could be generated from the example dependency tree include the following patterns which would not be generated by any of the other models:

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

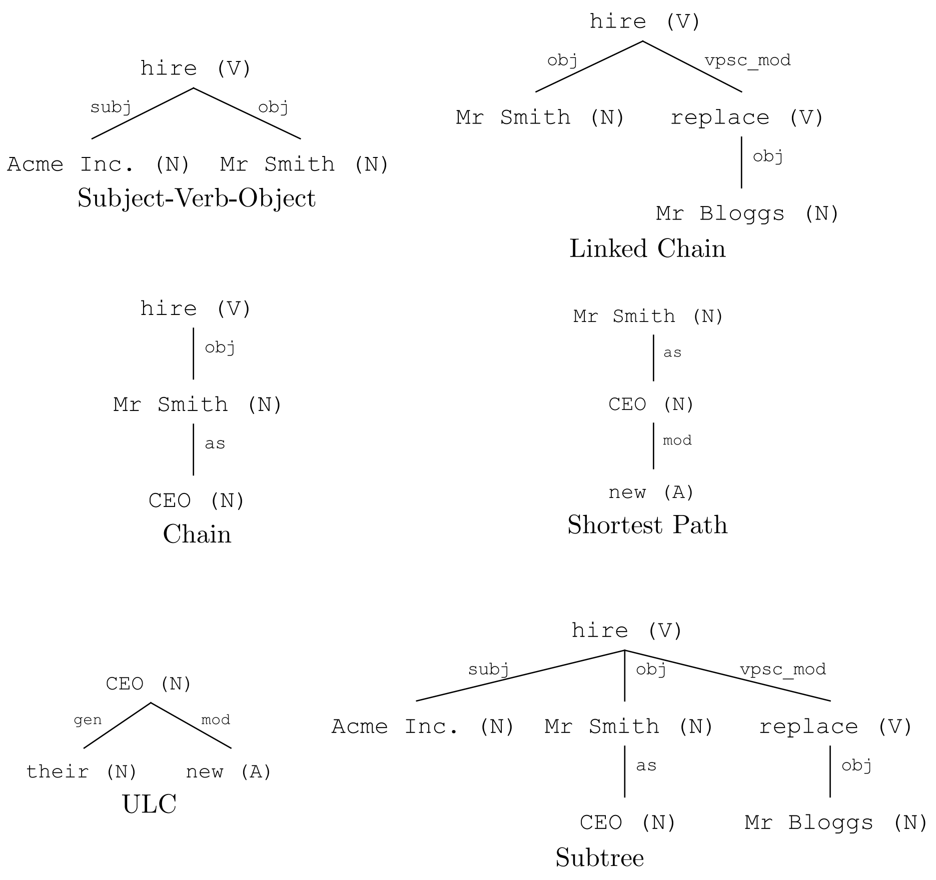

Figure 2: Examples of six pattern models

Figure 2 shows an example pattern for each of the six models.

When patterns are used within an IE system the patterns are generalised by replacing the named entities participating in the relation with a semantic class indicating the type of the entity. For example, the sentences ‘‘Acme Inc. hired Smith” and *IBM hired Jones last week.”* will be matched by the generalised pattern:

`[V/hire](subj[Organisation]+obj[Person])`

4 Pattern Enumeration and Complexity

In addition to encoding different parts of the dependency analysis, each model will also generate a different number of potential IE patterns.

A dependency tree, T, can be viewed as a set of N connected nodes. Assume that V, such that $V \subseteq N$, is the set of nodes in the dependency tree which are labeled as a verb.

Predicate-Argument Model (SVO): The number of SVO patterns identifiable within a dependency tree T, $N_{svo}(T)$, is bounded by $|V|$.

Chain Model: A chain can be created between any verb and a node it dominates (either directly or indirectly). Now assume that $d(v)$ denotes the total number of nodes produced by taking v, a node marked as a verb, and all its descendants. The number of possible chains in T is given by:

$$N_{chain}(T) = \sum_{v \in V} (\, d(v) - 1\,)$$

Linked Chain: Let $C(n)$ denote the set of direct child nodes of node n (i.e. $C(n)$ is the set of nodes which are immediately dominated by n). Let n_i denote the i-th child, so $C(n) = \left\{n_1, n_2, ...n_{|C(v)|}\right\}$. The number of possible linked chain patterns in T is given by:

$$N_{linked\,chain}(T) = \sum_{v \in V} \sum_{i=1}^{|C(v)-1|} \sum_{j=i+1}^{|C(v)|} d(v_i)\, d(v_j)$$

Similarly, the number of ULC patterns can be calculated as follows:

$$N_{ULC}(T) = \sum_{t \in T} \sum_{i=1}^{|C(t)-1|} \sum_{j=i+1}^{|C(t)|} d(t_i)\, d(t_j)$$

Shortest Path: The the number of shortest paths in T is given by:

$$N_{shortest\,path}(T) = \binom{|N|}{2} = \frac{|N|^2 - |N|}{2}$$

Subtree: Now assume that $sub(n)$ is a function denoting the number of subtrees, including single nodes, rooted at node n. This function can be defined recursively as follows:

$$sub(n) = \begin{cases} 1 & \text{if}\, n \,\text{is a leaf node} \\ \prod_{i=1}^{|C(n)|} (sub(n_i) + 1) & \text{otherwise} \end{cases}$$

The total number of subtree patterns (which do not include single nodes) is now:

$$N_{subtree}(T) = \left(\sum_{n \in N} sub(n) \right) - |N|$$

The number of SVO patterns is constant on the number of verbs in the dependency tree. The number of patterns generated by the chain model is generally linear on the size of the dependency tree, although in the worst case it can be polynomial. The linked chain, ULC and shortest path models produce a polynomial number of patterns while the number for the subtree model is exponential.

5 Pattern Expressiveness

The various pattern models introduced in Section 3 represent different portions of dependency trees. At one extreme the SVO model is limited to verbs and their direct arguments while the subtree model can represent arbitrary parts of the dependency tree. In order to determine what effect these differences make we carried out an experiment to compare how suitable the various pattern models are for encoding the information of interest to IE systems. To do this we chose a set of corpora annotated with the relations to be extracted, generated sets of patterns using a variety of dependency parsers that were then examined to discover how much of the target information each could capture.

5.1 Corpora

Corpora representing different genres of text were chosen for these experiments; one containing newspaper text and another composed of biomedical abstracts. The first corpus consisted of Wall Street Journal texts from the Sixth Message Understanding Conference (MUC 1995). These are reliably annotated with information about the movement of executives between jobs. Events in this corpus identify relations between up to four types of entity: `PersonIn` (the person starting a new job), `PersonOut` (the person leaving a job), `Post` (the job title) and `Organisation` (the employer). For example, the sentence *"Smith was recently made chairman of Acme."* contains information about the new employee (*Smith*) of a company (*Acme*) and the post they are moving to (*chairman*). We make use of a version of the corpus produced by Soderland (1999) in which only events described within a single sentence were annotated.

The second corpus uses documents taken from the biomedical domain, specifically the training corpus used in the LLL-05 challenge task (Nédellec 2005) and a pair of corpora (Craven and Kumlien 1999) which were derived from the Yeast Proteome Database (YPD) (Hodges, McKee, Davis, Payne, and Garrels 1999) and the Online Mendelian Inheritance in Man database (OMIM) (Hamosh, Scott, Amberger, Bocchini, Valle, and McKusick 2002). Each of these corpora are annotated with relations between pairs of items. The LLL-05 corpora contains interactions between genes and proteins. For example the sentence *"Expression of the sigma(K)-dependent cwlH gene depended on gerE"* contains relations between *sigma(K)* and *cwlH* and between *gerE* and *cwlH*. The YPD corpus is concerned with the subcellular compartments in which particular yeast proteins localize. An example sentence *"Uba2p is located largely in the nucleus"* relates *Uba2p* and *the nucleus*. The relations in the OMIM corpora are between genes and diseases, for example *"Most sporadic colorectal cancers also have two APC mutations"* contains a relation between *APC* and *colorectal cancer*.

The various corpora used for these experiments encode events in different formats. In the MUC-6 corpus each event may connect either two or three items while the biomedical corpora contain events consisting of two items. For consistency we treat each event as a set of binary relations and events in the MUC-6 corpus were converted appropriately. For example, the sentence *"Smith was recently made chairman of Acme."* describes an event involving three named entities: a `PersonIn` (*Smith*), `Post` (*chairman*) and `Organisation` (*Acme*). For evaluation we convert this event structure into a set of three relations: `PersonIn-Post` (*Smith-chairman*), `PersonIn-Organisation` (*Smith-Acme*) and `Post-Organisation` (*chairman-Acme*).

The MUC-6 corpus contains a total of six different relation types and each of the three biomedical corpora contain a single relation type, giving a total of nine for the experiments. There are a total of 3911 instances of relations across all corpora. Table 1 details the distribution of relations.

Table 1: Distribution of relations within the evaluation corpora.

Relation Type	Corpus	Instances
`PersonOut-Company`	MUC-6	186
`PersonIn-Company`	MUC-6	133
`PersonOut-Post`	MUC-6	334
`PersonIn-Post`	MUC-6	308
`PersonIn-PersonOut`	MUC-6	103
`Post-Company`	MUC-6	258
All Management Succession Relations	–	1322
Genic Interaction	LLL05	103
`Protein-Location`	YPD	1372
`Gene-Disease`	OMIM	1114
All Biomed Relations	–	2589
All Relations	–	3911

5.2 Generating Dependency Patterns

Three dependency parsers were used to generate patterns[3].

1. MINIPAR[4] uses an internal constituency grammar before converting the result to a dependency tree (Lin 1999). An evaluation over the SUSANNE corpus (Sampson 1995) reported that 89% of the dependency relationships output by MINIPAR are correct.

2. The Stanford[5] parser is a lexicalised probabilistic CFG parser (Klein and Manning 2003) which incorporates lexical dependency information derived from treebanked corpora (Klein and Manning 2002). In addition to generating phrase structure analyses this parser can also be used to produce dependency trees.

3. MaltParser[6] is a deterministic dependency parser that employs an algorithm which combines top-down and bottom-up parse strategies and can process a sentence in linear time (Nivre and Scholz 2004).

Before these parsers were applied to the evaluation corpora (Section 5.1) the items participating in relations were replaced by special tokens indicating their role. For example, *"Smith was recently made chairman of Acme."* becomes "`PersonIn` *was recently*

[3]We compared a variety of dependency parsers. Results are reported for the three which we found to perform best on our data sets.
[4]`http://www.cs.ualberta.ca/~lindek/minipar.htm`
[5]`http://www-nlp.stanford.edu/software/lex-parser.shtml`
[6]`http://w3.msi.vxu.se/~nivre/research/MaltParser.html`

made `Post` *of* `Organisation`". Each parser was adapted to deal with these tokens. The parsers were applied to each corpus and patterns for each of the four models extracted from the dependency trees they produced.

The parser output was post-processed to maximise the information which could be derived from their analyses. It was found that the parsers were often unable to generate a single dependency tree which spanned the entire sentence and would instead generate an analysis in which fragments of the sentence are represented as independent tree structures. Many of these fragments did not include a verb and consequently no patterns could be extracted for the SVO, chain or linked chain models, each of which requires a verb to be the root of the pattern. Consequently we allowed the root node of any tree fragment to take the place of a verb in a pattern (see Section 3). This allows more chain and linked chain patterns to be produced but has no effect on the number of SVO, ULC, shortest path or subtree patterns generated. (SVO patterns require specific dependency relations between verbs and their arguments while there is no restriction on the root node of the other three models.) The effect of parser fragmentation is discussed in Section 5.6.

The dependency graphs produced by the parsers were left unaltered with the exception of MINIPAR which is unique in allowing some nodes to have more than one governing node. MINIPAR uses this feature to cope with phenomena such as conjunction, anaphora and VP-coordination. For example, in the sentence *"The bomb caused widespread damage and killed three people" the bomb* is the subject of the verbs *cause* and *kill.* MINIPAR represents this by generating a dependency analysis in which the *bomb* node is dominated by both *cause* and *kill.* Consequently, the graph produced by this parser is a Directed Acyclic Graph rather than a tree structure. For ease of processing we preferred to represent each analysis as a tree structure. This is achieved in the MINIPAR output by identifying nodes with more than one head and duplicating the subtree for which they are the root node. In this way the output is converted to a tree structure, albeit one in which a lexical item may be represented by multiple nodes.

5.3 Number of Patterns Generated

Table 2 shows the number of patterns which would be generated for each pattern model given the analyses generated by the three parsers. The number of subtree patterns was determined using the formula in Section 4[7], while all patterns were generated for the five other models.

	Parser		
Model	MINIPAR	Stanford	MALT
SVO	16,915	11,271	11,528
Chain	196,887	208,063	302,559
Linked chain	715,503	788,542	1,268,676
ULC	845,429	1,079,659	1,657,934
Shortest path	1,194,910	1,604,426	2,326,893
Subtree	1.64×10^{64}	1.70×10^{12}	4.56×10^{16}

Table 2: 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 figures are consistent with the analysis in Section 4.

There is variation between the number of patterns generated by the various parsers. This is mainly caused by differences in the ways parsers choose to deal with linguistic phenomena and whether they are able to generate a single dependency analysis which spans the entire sentence. (This point is discussed further in Section 5.6.) The number of subtrees generated from MINIPAR parses is many orders of magnitude higher than the other two parsers due to duplication of subtrees with a root node governed by more than one node (see Section 5.2)[8].

It is worth noting that these figures are computed only over sentences which are known to contain relations. In the majority of scenarios, where the sentences containing relations would not be known a priori, there would be a significant increase in the number of patterns which would have to be generated.

5.4 Evaluating Expressiveness

Patterns generated by each of the models are examined to check whether they cover the information which should be extracted. In this context a pattern is said to cover

[7] Section 6.1 discusses why this was necessary.

[8] One dependency tree generated by MINIPAR contained approximately 1×10^{64} subtrees when expanded in this way.

a relation if it contains the two items which participate in the relation. These items can appear anywhere within the pattern. For example, an SVO pattern extracted from the dependency parse of *"Smith was recently made chairman of Acme."* would be `[V/make](subj[N/PersonIn]+obj[N/Post])` which covers the relation between *Smith* and *chairman* but not the relations between *Smith* and *Acme* or *chairman* and *Acme*.

In general we are interested in determining the proportion of relations in a corpus which are covered by the patterns generated for a particular model. We do this by introducing the notion of coverage for a pattern model. This is calculated by counting the number of relations in a corpus which are covered by the patterns the model generates and dividing this by the total number of relations. (Coverage can be computed for a single type of relation or over a set of relation types.)

More formally, let R be the set of relations in corpus C. Also, let M be the patterns generated from the analyses produced by some parser, P, when applied to C. Then,

$$Coverage(M, C, P) = \frac{|\{r \epsilon R : \exists m \epsilon M \; such \; that \; m \; covers \; r\}|}{|R|}$$

The coverage score of a pattern model indicates the proportion of the relations in the corpus which it is capable of representing. Consequently, coverage places an upper bound on the recall of an IE system which uses that model. The recall figure cannot be any higher than the coverage since the model cannot represent any more of the relations.

In practical applications parsers may not be able to provide a complete analysis for every sentence, if it cannot produce a single tree which spans the whole sentence. So, given the output of a parser there may be pairs of related items which are not connected in the analysis. In this case it will not be possible to construct a pattern which covers that relation and we refer to it as an inexpressible relation, given the dependency analysis. Pairs of items which are connected are said to be expressible. The proportion of relations in a corpus which fall into the expressible and inexpressible categories, and therefore the parser used, has an effect on a pattern model's coverage.

To account for this we also introduce an additional measure for the evaluation of pattern models; bounded coverage. This is computed, for a given dependency analysis of the corpus produced by a given parser, by dividing all relations the model covers by the total number of expressible relations in the corpus. So,

$$Bounded \; Coverage(M, C, P) = \frac{|\{r \epsilon R : \exists m \epsilon M \; such \; that \; m \; covers \; r\}|}{|\{r \epsilon R : r \; is \; expressible \; given \; P\}|}$$

Three of the models (ULC, shortest path and subtree) cover any event whose participating items are included in the dependency tree, and their bounded coverage is always 100%. In discussion of the results of these experiments we refer to these models as the "comprehensive models". Since the coverage figures of these models are always the same we report only a single figure for the group of models and do not report the bounded coverage.

The results of these experiments were analysed in a number of ways. Firstly, performance of each of the patterns models was compared (Section 5.5). It was found that the choice of parser used to generate the patterns had an effect on performance and this is discussed in Section 5.6. Finally, the results of each pattern model are analysed for individual relation types (e.g. `PersonIn-PersonOut`) and it was found that certain pattern models are more suitable for the extraction of particular types of relations (Section 5.7).

5.5 Pattern Models

Coverage and bounded coverage results for each pattern model and parser combination are shown in Table 3 in the columns labeled "%C" and "%B-C" respectively[9]. Both sets of scores are listed for the SVO, chain and linked chain models. Bounded coverage for the comprehensive models is always 100% and is not listed in the table. Coverage results for the management succession and biomedical corpora are quite different so both are listed in this table, in the rows labeled "Management" and "Biomed". The row labeled "Combined" shows the coverage and bounded coverage score for each parser across the combination of all corpora used for these experiments.

The simplest representation, SVO, does not perform well in this evaluation. The highest bounded coverage score is 15.1% (Management corpus, Stanford parser) but the coverage over the combined corpora is less than 6% for any parser. The SVO model performs noticeably worse on the biomedical text. Our analysis suggests that this is because the items of interest are commonly described in ways which the SVO model is unable to represent (see Section 5.7).

[9]In previous experiments we allowed single chain patterns to be considered as linked chain patterns (Stevenson and Greenwood 2006). We do not do this here, preferring to separate these models in a stricter way, although this leads to lower coverage (and bounded coverage) figures for the linked chain model.

The more complex chain model covers a greater percentage of the relations. However its bounded coverage is still less than half of the relations in either the management succession or biomed texts; the best bounded coverage score achieved is 49.73% (Management corpus, MINIPAR parser). Results for the linked chain model are more promising, with bounded coverage in excess of 90% across the combined corpora for two of the three parsers.

A one-way repeated measures ANOVA was carried out to analyse the difference in each model's performance. It was found that there were significant differences between the models' bounded coverage scores ($p < 0.01$). Two-tailed pared t-tests (with Bonferroni correction) were used to compare the differences between the results obtained for individual pairs of models. This revealed that the SVO and chain models were significantly different ($p < 0.01$) from the linked chain and comprehensive models. The SVO and chain models were not found to be significantly different nor were the linked chain and comprehensive models.

These results suggest that the linked chain and comprehensive models are more expressive than either the SVO or chain models but that their accuracy is comparable.

5.6 Parsers

Table 3 shows a wide variation in the bounded coverage scores of the three parsers, although a one-way repeated measures ANOVA did not indicate that these were significant.

The results shown in Table 3 also show that there is a strong tendency for the

Table 3: Coverage and bounded coverage scores for different pattern models using a variety of parsers.

		SVO		Chain		Linked Chain		Comprehensive
Parser	Corpus	%C	%B-C	%C	%B-C	%C	%B-C	%C
MINIPAR	Management	7.49	9.07	41.07	49.73	76.78	92.95	82.60
	Biomed	0.93	1.30	17.38	24.44	62.53	87.93	71.11
	Combined	3.14	4.19	25.39	33.86	67.35	89.81	74.99
Stanford	Management	15.05	15.10	41.07	41.20	93.65	93.93	99.70
	Biomed	0.46	0.49	16.53	17.39	88.53	93.13	95.06
	Combined	5.40	5.58	24.83	25.69	90.26	93.42	96.62
Malt	Management	5.98	6.61	33.96	37.57	79.35	87.87	90.39
	Biomed	0.23	0.26	11.63	13.07	73.04	82.11	88.60
	Combined	2.17	2.43	19.18	21.44	75.17	84.05	89.44

parser to perform better on the management succession corpus than the biomedical text. For each parser the coverage scores for all models is greater on the management succession corpus than the biomedical text. A two-tailed paired t-test indicated that the difference between the performance of the parsers on the two genres of text was significant ($p < 0.01$). The most likely reason for this difference is that parsers are commonly trained and evaluated on newswire text, such as the Wall Street Journal (Marcus, Marcinkiewicz, and Santorini 1993). The MUC-6 corpus comprises of newswire article but the biomedical corpora represent a very different genre of text. Grover et al. (2005) reported that biomedical text contain a wide range of constructions which current parsers find difficult to analyse correctly (e.g. coordination, ellipsis and complex nominals).

It has already been mentioned, Section 5.2, that some of the parsers tended to fragment rather than generate a single dependency analysis which spanned the sentence. There is a very strong negative correlation between the coverage of the comprehensive models, for a particular parser, and the average number of fragments per sentence in the analyses it produces ($r = -0.92$). In many ways it is not surprising that a parser with a tendency to fragment is unlikely to generate analyses which connect the items of interest. However, the result is useful since it is simple to measure the average number of fragments produced by a parser but more difficult to compute the coverage for a particular IE task since the latter requires an annotated corpus. Consequently this result is useful since it provides a simple way of determining the suitability of a particular parser, for a particular corpus, with minimal need for annotated resources.

5.7 Analysis of Individual Relations

The results in Section 5.5 show performance of the SVO model to be very poor in comparison to the others so we do not discuss it in detail here. Although we note that the SVO model does perform well (around 54% bounded coverage) for the `PersonIn-PersonOut` relation when the MINIPAR parser is used. Corpus analysis revealed several examples where this relation was expressed using a simple predicate argument structure containing the verb *succeeds*, for example "`PersonIn` *succeeds* `PersonOut`", "`PersonOut` *will be succeeded by* `PersonIn`" and "`PersonIn` *is expected to succeed* `PersonOut`". A single SVO

Table 4: Bounded coverage for each relation using chain model

Relation	Parser MINIPAR	Stanford Parser	Malt	Average
`PersonOut-Company`	48.13	40.86	18.29	35.76(15.56)
`PersonIn-Company`	22.55	18.94	18.10	19.86 (2.36)
`PersonOut-Post`	75.43	58.89	48.54	60.95(13.56)
`PersonIn-Post`	30.60	25.32	25.27	27.06 (3.06)
`PersonIn-PersonOut`	11.11	17.65	40.74	23.17(15.57)
`Post-Company`	63.60	58.37	58.47	60.15 (2.99)
`Genic Interaction`	7.53	4.85	1.01	4.46 (3.28)
`Protein-Location`	17.52	12.18	5.51	11.74 (6.02)
`Gene-Disease`	35.23	25.72	23.24	28.09 (6.33)

pattern, `[V/succeed](subj[N/PersonIn]+obj[N/PersonOut])`, covered these examples.

Tables 4 and 5 show the bounded coverage for each relation for the chain and linked chain models. The result for each of the parsers is presented and the average score shown in the rightmost column of each table followed by the standard deviation in brackets. (The number of instances of each relation was shown in Table 1.)

The chain model shows a varied performance across the various relations (Table 4). In the management succession domain this model performs best on the `PersonOut-Post`, `PersonOut-Company` and `Post-Company` relations. Analysis of the corpus showed that the `PersonOut-Company` and `PersonOut-Post` relations were often expressed as relative clauses or appositions both of which tended to be non-restrictive. For example, "`PersonOut`*, who was* `Post`*,*", "`PersonOut`*, who recently left* `Company` *to ...*", "*current acting* `Post`*,* `PersonOut`*,*" and "`PersonOut` *, a former* `Post` *of* `Company`". The `Post-Company` relations had a strong tendency to be expressed as prepositional phrases with the most frequent being "`Post` *of* `Company`". This relation was also expressed using a possessive: "`Company`*'s* `Post`". In the biomedical domain the highest bounded coverage was recorded for the `Gene-Disease` relation. It was found that these were frequently expressed using appositions. Examples include "*mutations in* `Gene` *, the candidate gene for* `Disease`" and "*the gene for* `Disease` *,* `Gene`*,*".

Results for the linked chain model are shown in Table 5. This model performs well for all relations, particularly those in the management succession corpus. Performance of the linked chain model is consistently better than the combined performance of the SVO and chain models. For every pattern model and parser combination performance of the linked chain model is greater than the sum of the performance of the other two

Table 5: Bounded coverage for each relation using linked chain model

Relation	MINIPAR	Stanford Parser	Malt	Average
`PersonOut-Company`	91.87	89.78	76.83	86.16 (8.15)
`PersonIn-Company`	96.08	95.46	81.90	91.14 (8.01)
`PersonOut-Post`	90.30	94.59	89.97	91.62 (2.58)
`PersonIn-Post`	98.29	95.78	93.50	95.86 (2.40)
`PersonIn-PersonOut`	100.00	95.09	90.12	95.07 (4.94)
`Post-Company`	87.72	92.61	87.91	89.41 (2.77)
`Genic Interaction`	89.25	89.32	75.76	84.78 (7.81)
`Protein-Location`	90.16	93.73	82.46	88.78 (5.76)
`Gene-Disease`	84.97	92.71	82.33	86.67 (5.40)

models. This indicates that the linked chain model is more than a simple concatenation of the set of patterns which form the other two models.

Analysis of relations covered by linked chain but not the SVO or chain models showed that the related items tend to be connected within a predicate-argument structure in which the two related items do not both function as the subject and object. At least one of them is connected to the head of the verb's argument by some other relation. For example, *"*`PersonIn` *succeeds Jones as head of mortgaged-based securities at* `Company`*"*, *"*`Company` *announced a new CEO,* `PersonIn`*,"*, *"*`Protein` *encodes a homolog of vertebrate synaptic* `Location` *membrane proteins"* and *"mutations of the* `Gene` *tumore suppressor gene predisppose women to* `Disease`*"*.

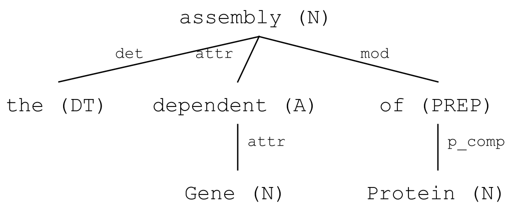

Figure 3: Example dependency analysis of a nominalisation

The linked chain model covers the majority of relations which could be identified given a particular parse. However, the bounded coverage is not 100% which indicates that there are still some relations which can be captured by the three comprehensive models and not the linked chain model. Examination of these suggested that there are certain constructions which cause difficulties. For example, the relation between `Gene` and `Protein` in the nominalisation *"the* `Gene`*-dependent assembly of* `Protein`*"*. The dependency analysis which would be generated for this phrase is shown in Figure 3.

It can be seen that the node which dominates the two related items is marked as a noun. Consequently this relation cannot be covered since each of the chains forming a linked chain must be anchored at the same verb (or the root of a tree fragment) and cannot share any other nodes. A similar problem was observed for certain appositive constructions, for example, *"*`Organisation`*'s chairman,* `PersonOut`*, resigned".* The pattern where the lowest node in the dependency tree dominating the related items is marked as a noun was observed for all relations where a pattern was covered by the comprehensive models but not linked chain.

This analysis has revealed a difference in the bounded coverage of each model across the various relations examined here. In general the linked chain model performs well for all relations, particularly those in the management succession domain. It is likely that this is caused by the parsers being more suited to newswire text and the larger number of problematic constructions, such as nominalisations, in the biomedical text.

The analysis described in this section suggests that two of the dependency pattern models which have been previously used in the literature, SVO and chain, are inadequate due to their limited ability to represent the information which should be extracted from text. The experiments show that the coverage scores for these models is less than 50% over all the corpora we evaluated against, indicating that the models are unable to capture the information in over half the instances of the relations of interest in these texts.

The coverage measure effectively places an upper bound on the recall which we could expect from an IE system based on that model. However, there are limitations to the conclusions which can be drawn from an analysis of pattern models in terms of coverage. The coverage measure is not suited to evaluating the usefulness of individual pattern for IE since it does not take account of any incorrect matches. For example, the pattern `[V/succeed](subj[N/PersonIn]+obj[N/PersonOut])` will match the sentence "Smith succeeded Jones", which contains a relation, but will also match "Smith never succeeded Jones" and "It is not known whether Smith succeeded Jones." We go on to address these limitations in the next section which compares pattern models more directly.

6 Extraction Scenario Experiment

In order to directly compare the various pattern models described in this paper we carried out an experiment to compare them on a practical IE scenario. Five of the models introduced in Section 3 were compared using an unsupervised approach to IE similar to the one described by Sudo, Sekine, and Grishman (2003). (SVO patterns were not included in light of performance reported in the previous section.)

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.

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

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

where tf_p is the number of times pattern p appears in a relevant document 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. The process of setting the value of β in these experiments is described in Section 6.2.

This ranking process was applied to the IE scenario from the Sixth Message Understanding Conference described in Section 5.1. This is a standard scenario which has been widely used to evaluate IE systems. 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.

6.1 Pattern Generation

Results in Section 5.5 show that the Stanford parser is the most accurate when applied to the management succession texts and was used to generate patterns for each of the various models in this experiment.

Patterns for each of the models were extracted from the processed dependency trees. For the majority of the pattern 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, Kawasoe, Asai, Arimura, and Arikawa 2002; Zaki 2002) which is an efficient way of enumerating all subtrees. This approach constructs subtrees iteratively by combining together previously observed subtrees. 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 filtered versions of the sets of patterns extracted for the chain, linked chain, ULC and shortest path models in which patterns which occurred fewer than four times were removed.

6.2 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, Sekine, and Grishman (2003). To generate this additional text we used the Reuters Corpus (Rose, Stevenson, and Whitehead 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 MUC-6 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.

6.3 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.

6.4 Results

Figure 4 shows the results when the filtered pattern models, ranked using equation 1, are compared.

A first observation is that the chain model performs poorly in comparison to the other models. The highest precision achieved by this model is 19.9% and recall never increases beyond 9%. Differences between the remaining pattern models are less pronounced. Performance of the linked chain, ULC and shortest path patterns is difficult to distinguish; the highest precision, 45.7% with 11.5% recall, is obtained by both models. The highest precision obtained by linked chain is slightly lower, 44.5% with 12.9% recall, while the best precision obtained by the subtree model is 42.6% (with 11.2% recall). Linked chain achieve a maximum recall of 23.6% while the ULC and shortest path models both reach 25.7%. Maximum recall for the subtree model is higher, at 33.7%, but this is to be expected since the number of subtrees is so much higher than the other patterns. However, it is likely that the large number of patterns also contribute to the fact that the maximum precision for this model is lower than for linked chain, ULC and shortest path since the large number of patterns make it difficult to identify those which are useful for IE. Performance of the subtree model also seems less robust than the other models since precision drops quickly as recall increases.

The maximum recall achieved by any model 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 5 shows the results using the unfiltered linked chain, ULC and shortest path patterns. Chain patterns are not included because of their poor performance using the filtered patterns, although filtered subtree patterns are included for comparison.

For two of the models, ULC and shortest path, the best precision and recall scores improves when the unfiltered patterns are used. The ULC model achieves precision of 50% at 11.8% recall while the best precision for shortest path is 48.7% at 14.2% recall. All three models outperform the scores obtained by the filtered subtree patterns. Performance of the linked chain, ULC and shortest path models is also difficult to distinguish when the patterns are filtered. The ULC and shortest path models achieve higher precision scores, possibly due to the linked chain model's incomplete coverage, but precision of the linked chain model drops off more slowly as patterns are added, perhaps because a larger proportion of this model's patterns are useful for IE.

The extra patterns lead to maximum recall scores for the linked chain, ULC and shortest path models which are more than double the best filtered scores without overly degrading precision. It is likely that the subtree 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.

Precision levels for all models are low, generally below 50%, for all models in this experiment. One reason for this is that the the patterns were ranked using a simple unsupervised learning algorithm which allowed direct comparison of 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 patterns (Zelenko, Aone, and Richardella 2003; Bunescu and Mooney 2005; Greenwood and Stevenson 2006).

The results presented here provide insight into the usefulness of the various pattern models by evaluating them on an actual IE task. The best performance was obtained using the ULC and shortest path models. Performance of the linked chain model is comparable. The chain model was found to perform badly with low recall and precision regardless of whether the patterns were filtered. The most likely reason for the poor performance of this model is its inability to represent the necessary information in text, as shown by the analysis of this model in terms of coverage described in Section 5. Performance of the subtree model is also limited. It was not possible to generate all possible patterns, although this could be done for the other models. In addition, the subtree model was not as accurate as the linked chain, ULC or shortest path models, presumably because the large number of potential patterns make it difficult for the learning algorithm to identify the ones which are useful for extraction.

7 Summary and Discussion

This article compares six IE pattern models: SVO, chain, linked chain, ULC, shortest path and subtree. It was found that the first two of these were limited by the fact they are unable to represent the relations found in corpora representing very different domains (newswire stories and biomedical journal articles). The linked chain model could represent up to 93% of the expressible relations in these corpora while, if provided with a suitable dependency analysis, the other three can represent them all. Some of the models were directly compared on an IE task. Performance of the linked chain, ULC and shortest path models was found to be better than other models. The chain model was hampered by limited coverage while the large number of possible patterns appeared to make learning difficult when subtree patterns were used.

The linked chain, ULC and shortest path models represent similar parts of dependency trees. It seems likely that these areas contain the most relevant information for IE. The SVO and chain models omit much of this information while the subtree model contains more information than necessary.

Techniques have been developed for comparing parse trees efficiently and without the need to generate all subtrees, such as the work on convolution kernels developed for parse ranking (Collins and Duffy 2001), and these have been applied to IE (Zelenko, Aone, and Richardella 2003; Bunescu and Mooney 2005; Zhang, Zhang, Su, and Zhou 2006). However, our results suggest that there is little to be gained from considering all possible subtrees and a more suitable approach would be to focus on the portions of the dependency tree most likely to contain relevant information, an approach several studies have shown to be effective. For example, Bunescu and Mooney (2005) use the shortest path between pairs of entities in the dependency graph. Zhang et al. (2006) restrict attention to the subtree enclosed by the shortest path and refer to this as the "Path-enclosed Tree". They found that this approach was more effective than those using other portions of the tree, although they chose to parse text using a CFG formalism and their work cannot be directly compared with ours. Jiang and Zhai (2007) systematically compared a wide variety of features that have been used for relation extraction, including parse trees. They used subtrees containing three or fewer nodes and also found that restricting attention to the Path-enclosed Tree improved performance. However, there are obvious cases where information outside the Path-enclosed Tree is necessary to identify a relation. Zhou et al. (2007) provide the example sentence *"John and Mary got married"* and point out that the information contained in the path-enclosed tree ("John and Mary") is not sufficient to determine their relationship. They propose heuristics that include portions of the tree linked via a predicate in certain situations and show that this improves performance. These studies support our finding that subtrees contain more information than is necessary to identify relations and show that results improve when attention is focused on the portions of the tree most likely to contain useful information.

Results of the experiments described here should be borne in mind during the design and implementation of IE systems. It is important for the choice of representation to be expressive enough to represent the information which is to be extracted. This may seem obvious but our analysis has shown that the SVO and linked chain models, which have been used by a number of researchers (Yangarber, Grishman, Tapanainen, and Huttunen 2000b; Sudo, Sekine, and Grishman 2001; Yangarber 2003; Stevenson and Greenwood 2005), are severely limited. Models which allow patterns to be formed from pairs of paths in the dependency tree (such as linked chain, ULC and shortest path) seem to be the most useful and including additional parts of the dependency tree may not improve performance and create a far larger space of potential patterns than other models. However, the choice of pattern model may ultimately depend on the application. For example, we observed that the simple models such as SVO and chain have reasonable coverage for some relations (see Section 5.7) while generating far fewer possible patterns than alternative models. It is possible that these simple representations may be suitable for identifying specific relations in situations where achieving complete recall is not necessary.

Our experiments considered binary relations which are described within a single sentence. We do not believe these restrictions limit the conclusions which can be drawn from the work described here. Recent work on the use of dependency tree patterns for IE (Yangarber 2003; Sudo, Sekine, and Grishman 2001; Sudo, Sekine, and Grishman 2003; Bunescu and Mooney 2005; Stevenson and Greenwood 2005) has made the same assumptions and the aim of this work is to evaluate and compare the models they propose. The fact that the relations being considered are binary does not limit the usefulness of this work since others, including (Chieu and Ng 2002; McDonald, Pereira, Kulick, Winters, Jin, and White 2005), have shown that n-ary relationships can be recovered from sets of binary relations to generate complex template structures.

These experiments also provide insights into the more general question of how suitable dependency trees are as a basis for extraction patterns. Dependency parsers have the advantage of generating analyses which abstract away from the surface realisation of text to a greater extent than phrase structure grammars tend to. This leads to the semantic information being more accessible in the representation of the text which can be useful for IE. For practical applications this approach relies on the ability to accurately generate dependency analyses. The results presented here suggest that at least one parser (the Stanford parser) is capable of generating analyses from which useful semantic patterns can be constructed for almost all sentences within corpora from two very different domains.

Acknowledgments

The authors would like to acknowledge the support of the UK EPSRC under the grants RESuLT (GR/T06391) and CASTLE (EP/D069548). This paper greatly benefited from comments provided by the anonymous reviewers . Any mistakes are our own.

References

Abe, K., S. Kawasoe, T. Asai, H. Arimura, and S. 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)*, pp. 1–14.

Briscoe, T. and J. Carroll (2002). Robust accurate statistical annotation of general text. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, Las Palmas, Gran Canaria, pp. 4–8.

Bunescu, R. and R. Mooney (2005). A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, Vancouver, B.C., pp. 724–731.

Chieu, H. and H. Ng (2002). A Maximum Entroy Approach to Information Extraction from Semi-structured and Free Text. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence (AAAI-02)*, Edmonton, Canada, pp. 768–791.

Clark, S. and J. Curran (2008). Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. *Computational Linguistics*, (to appear).

Collins, M. and N. Duffy (2001). Convolution kernels for natural language processing. In *Adnavces in Neural Information Processing Systems (NIPS 2001)*, Vancouver, Canada, pp. 625–632.

Craven, M. and J. Kumlien (1999). Constructing Biological Knowledge Bases by Extracting Information from Text Sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, Heidelberg, Germany, pp. 77–86. AAAI Press.

Culotta, A. and J. Sorensen (2004). Dependency Tree Kernels for Relation Extraction. In *42nd Annual Meeting of the Association for Computational Linguistics*, Barcelona, Spain, pp. 423–429.

Fox, C. and S. Lappin (2005). Achieving expressive completeness and computational efficiency for underspecified semantic representations. In *Proceedings of the Fifteenth Amsterdam Colloquium*, pp. 77–82.

Goldberg, L. A. and M. Jerrum (2000). Counting Unlabelled Subtrees of a Tree is $\#P$-Complete. *London Mathmatical Society Journal of Computation and Mathematics 3*, 117–124.

Greenwood, M. A. and M. Stevenson (2006). Improving Semi-supervised Acquisition of Relation Extraction Patterns. In *Proceedings of the Information Extraction Beyond The Document Workshop (COLING/ACL 2006)*, Sydney, Australia, pp.

29–35.

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

Grover, C., A. Lascarides, and M. Lapata (2005). A Comparison of Parsing Technologies for the Biomedical Domain. *Natural Language Engineering 11*(1), 27–65.

Hamosh, A., A. F. Scott, J. Amberger, C. Bocchini, D. Valle, and V. A. McKusick (2002). Online Mendelian Inheritance in Man (OMIM), a knowledge base of human genes and genetic disorders. *Nucleic Acids Research 30*(1), 52–55.

Hodges, P. E., A. H. Z. McKee, B. P. Davis, W. E. Payne, and J. I. Garrels (1999). The Yeast Proteome Database (YPD): a model for the organization and presentation of genome-wide functional data. *Nucleic Acids Research 27*(1), 69–73.

Jiang, J. and C. Zhai (2007). A systematic exploration of the feature space for relation extraction. In *Proceeding of the Conference of the North American Chapter of the Association for Computational Linguistics*, Rochester, New York, pp. 113–120.

Klein, D. and C. 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, pp. 3–10.

Klein, D. and C. D. Manning (2003). Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, Sapporo, Japan, pp. 423–430.

Kudo, T., J. Suzuki, and H. Isozaki (2005). Boosting-based Parse Reranking with Subtree Features. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbour, MI, pp. 189–196.

Lin, D. (1999). MINIPAR: A Minimalist Parser. In *Maryland Linguistics Colloquium*, University of Maryland, College Park.

Lin, D. and P. Pantel (2001). Discovery of inference rules for question answering. *Natural Language Engineering 7*(4), 343–360.

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

Marcus, M. P., M. A. Marcinkiewicz, and B. Santorini (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics 19*(2), 313–330.

McDonald, R., F. Pereira, S. Kulick, S. Winters, Y. Jin, and P. White (2005). Simple Algorithms for Complex Relation Extraction with Applications to Biomedical IE. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbour, MI, pp. 491–498.

MUC (1995). *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, San Mateo, CA. MUC: Morgan Kaufmann.

Nédellec, C. (2005). Learning Language in Logic - Genic Interaction Extraction Challenge. In *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn, Germany.

Nivre, J. and M. Scholz (2004). Deterministic Dependency Parsing of English Text. In *Proceedings of the Twentieth International Conference on Computational Linguistics (COLING-04)*, Geneva, Switzerland, pp. 64–70.

Pullum, G. and G. Gazdar (1982). Natural Languages and Context-Free Languages. *Linguistics and Philosophy 4*, 471–504.

Romano, L., M. Kouylekov, I. Szpektor, I. Dagan, and A. Lavelli (2006). Investigating a generic paraphrase-based approach for relation extraction. In *Proceeings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, Trento, Italy, pp. 401–408.

Rose, T., M. Stevenson, and M. 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)*, La Palmas de Gran Canaria, pp. 827–832.

Sampson, G. (1995). *English for the Computer*. Oxford: Oxford University Press.

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

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

Stevenson, M. and M. A. Greenwood (2006). Comparing Information Extraction Pattern Models. In *Proceedings of the workshop "Information Extraction Beyond The Document" at COLING/ACL 2006*, Sydney, Australia, pp. 12–19.

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

Sudo, K., S. Sekine, and R. 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)*, Sapporo, Japan, pp. 224–231.

Szpektor, I., H. Tanev, I. Dagan, and B. Coppola (2004). Scaling web-based acquisition of entailment relations. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain, pp. 41–48.

Tapanainen, P. and T. Järvinen (1997). A Non-Projective Dependency Parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, Washington, DC, pp. 64–74.

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

Yangarber, R., R. Grishman, P. Tapanainen, and S. Huttunen (2000a). Automatic Acquisition of Domain Knowledge for Information Extraction. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken, Germany, pp. 940–946.

Yangarber, R., R. Grishman, P. Tapanainen, and S. Huttunen (2000b). Unsupervised Discovery of Scenario-level Patterns for Information Extraction. In *Proceedings of the Applied Natural Language Processing Conference (ANLP 2000)*, Seattle, WA, pp. 282–289.

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

Zelenko, D., C. Aone, and A. Richardella (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research 3*, 1083–1106.

Zhang, M., J. Zhang, J. Su, and G. Zhou (2006). A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of the 21st International Conference on Computational Linguistcs and 44th Annual Meeting of the ACL*, Sydney, Australia, pp. 825–832.

Zhou, G., M. Zhang, D. Ji, and Q. Zhu (2007). Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, Prague, Czech Republic, pp. 728–736.

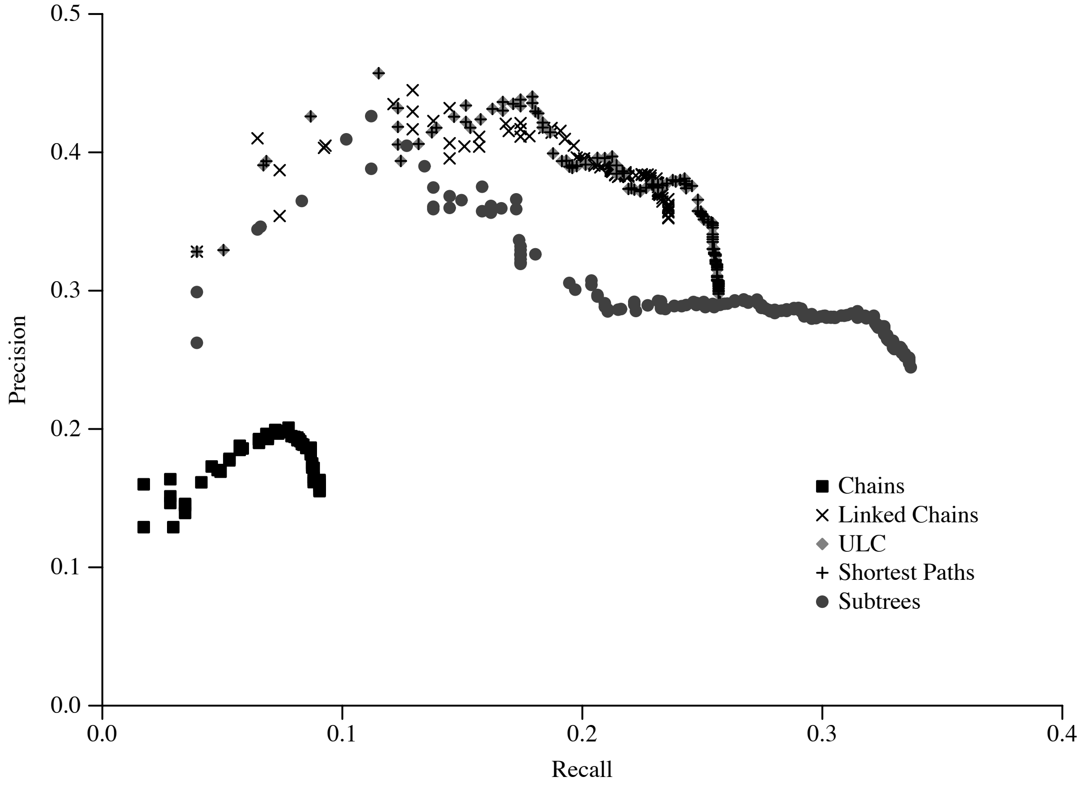

Figure 4: Comparisons of filtered pattern models.

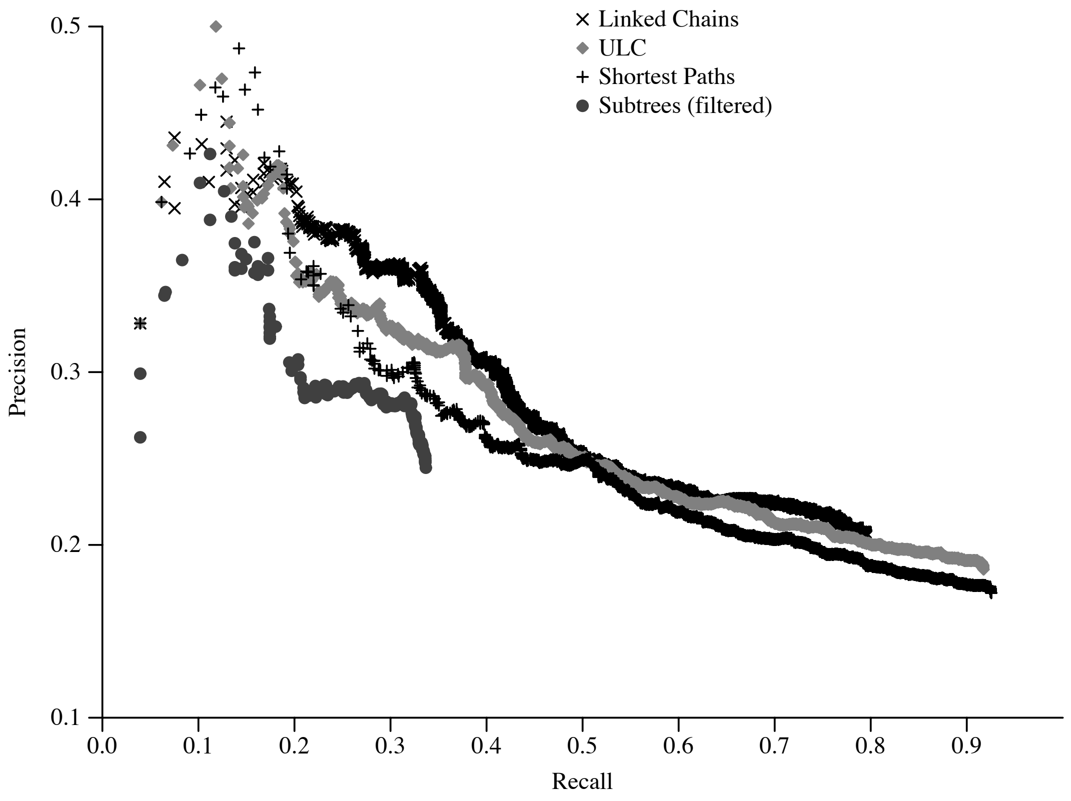

Figure 5: Comparison of unfiltered models.