International Journal of Medical Informatics (2006) **75**, 430—442

ELSEVIER

International Journal of Medical Informatics

www.intl.elsevierhealth.com/journals/ijmi

Evaluation of two dependency parsers on biomedical corpus targeted at protein—protein interactions

Sampo Pyysalo*, Filip Ginter, Tapio Pahikkala, Jorma Boberg, Jouni Järvinen, Tapio Salakoski

Turku Centre for Computer Science (TUCS), Department of Computer Science, University of Turku, Lemminkäisenkatu 14A, 20520 Turku, Finland

Received 15 April 2005; accepted 30 June 2005

KEYWORDS
Natural language processing;
Evaluation;
Parser comparison;
Dependency syntax;
Protein—protein interactions

Summary We present an evaluation of Link Grammar and Connexor Machinese Syntax, two major broad-coverage dependency parsers, on a custom hand-annotated corpus consisting of sentences regarding protein—protein interactions. In the evaluation, we apply the notion of an interaction subgraph, which is the subgraph of a dependency graph expressing a protein—protein interaction. We measure the performance of the parsers for recovery of individual dependencies, fully correct parses, and interaction subgraphs. For Link Grammar, an open system that can be inspected in detail, we further perform a comprehensive failure analysis, report specific causes of error, and suggest potential modifications to the grammar. We find that both parsers perform worse on biomedical English than previously reported on general English. While Connexor Machinese Syntax significantly outperforms Link Grammar, the failure analysis suggests specific ways in which the latter could be modified for better performance in the domain.

1. Introduction

The challenges of processing the vast amounts of biomedical publications available in databases such as PubMed[1] have recently attracted a considerable interest in the natural language processing (NLP) research community. A wide variety of NLP methods have been applied to the domain, ranging from purely statistical methods (e.g. [1,2]) to full syntactic analysis as employed, for example, by Craven and Kumlien [3], Yakushiji et al. [4], Temkin and Gilder [5], and Daraselia et al. [6]. A task of particular interest is information extraction (IE). IE systems are applied to free text, extracting from it factual information such as, for example, all protein—protein interactions stated in a given research article. A typical IE system performs the

* Corresponding author. Tel.: +358 2 333 8648
E-mail addresses: sampo.pyysalo@it.utu.fi (S. Pyysalo), filip.ginter@it.utu.fi (F. Ginter), tapio.pahikkala@it.utu.fi (T. Pahikkala), jorma.boberg@it.utu.fi (J. Boberg), jouni.jarvinen@it.utu.fi (J. Järvinen), tapio.salakoski@it.utu.fi (T. Salakoski).
[1] http://www.pubmed.com.

1386-5056/$ — see front matter © 2005 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.ijmedinf.2005.06.009

extraction in the following steps. Named entity recognition (NER) identifies relevant named entities in the text, e.g. proteins, genes, mRNAs, and organisms. Then syntactic analysis, also termed parsing, is performed to obtain a structure that specifies how words and phrases are related in the sentence. Finally, a domain analysis is performed which, given the named entities and the syntactic structure of the sentence, identifies the related entities and the nature of their relationship. In addition, other steps, such as coreference resolution are also often applied.

In this paper, we focus on the syntactic analysis component of IE systems. There are several approaches to parsing, which can be roughly categorized into partial parsing and full parsing, depending on how complete an analysis they produce. Parsers can also be divided with respect to their underlying grammatical formalism into two broad categories: constituency and dependency. Constituency parsers produce syntactic analysis in the form of a tree that specifies the phrases that constitute the sentence (e.g. noun phrase and verb phrase) and the hierarchy in which these phrases are nested. In contrast, dependency parsers analyse the sentence as a set of pairwise word-to-word dependencies, each dependency having a type that specifies its grammatical function (e.g. subject and object).

While both constituency and dependency approaches have been applied to mining biomedical literature, our purpose here is to assess the applicability of dependency parsers to the IE task in the biomedical domain. Due to the specific nature of the language used in scientific articles, the evaluation needs to be performed using a corpus that captures the domain language, rather than general language. For this purpose, we have developed a corpus that consists of sentences from scientific article abstracts. The sentences were selected using a procedure that results in a corpus with a distinct focus on protein—protein interactions. The corpus was hand-annotated, giving each sentence a full dependency analysis. Bioentities and their interactions stated in the sentences were identified as well. The manual annotation provides gold-standard data for the evaluation of the parsers.

We study the performance of two major broad-coverage full dependency parsers, the Link Grammar[2] (LG) of Sleator and Temperley [7] and the Connexor Machinese Syntax (CMS) parser[3]. The Link Grammar parser, in particular, has recently received significant attention in the Bio-NLP research literature [8—10]. The work of Szolovits [8] proposes a fully automated method to extend the dictionary of the LG parser with terms from the UMLS Specialist[4] lexicon, and Ding et al. [9] perform a basic evaluation of LG performance on biomedical text. Szolovits does not attempt to evaluate parser performance at all, and Ding et al. provide only an informal evaluation on manually simplified sentences. CMS has been applied to the biomedical domain as well, for example, in the named entity recognition system YAPEX of Franzén [11]. However, we are not aware of any study regarding the performance of CMS in the biomedical domain.

Here, we perform a more formal evaluation of the performance of the parsers. We use the hand-annotated corpus to evaluate the two parsers with respect to the domain language and we also employ certain IE-specific criteria. We further assess the effect of protein name recognition as a preprocessing step. Moreover, as LG is an open source system that can be investigated in detail, we perform a thorough failure analysis of LG and propose modifications in order to improve its applicability to biomedical IE. We also measure the effect of the LG dictionary extension proposed by Szolovits.

This work is a continuation of our previous work on the analysis of Link Grammar [12]. We now consider CMS in the evaluation and compare the two parsers. The corpus used here represents a validated version of the previously used corpus, further extended for CMS evaluation.

In the following sections, we describe the two dependency parsers and discuss issues in parser comparison. The corpus and the evaluation criteria are then introduced, followed by the evaluation results. We also present an analysis of LG failures, evaluate the effect of two potential improvements to the parsers, and conclude the results.

2. Dependency parsers

A dependency parse of a sentence consists of a set of pairwise word-to-word relationships referred to as dependencies. A dependency specifies an asymmetric relationship between words, where one word is a *dependent* of the other word, which is called its *governor*. Each dependency is given a type that describes its grammatical function. A dependency parse is usually presented using a diagram where the dependencies are drawn as labeled arcs connecting the words of the sentence to each other.

[2] http://www.link.cs.cmu.edu/link.
[3] http://www.connexor.com.
[4] http://www.nlm.nih.gov/research/umls.

Due to the ubiquitous ambiguity present in natural language, essentially all non-trivial sentences can be assigned several syntactically plausible parses. Consider, for instance, the phrase *staining with antibodies to desmin*. Both *antibodies-to-desmin* and *staining-to-desmin* are syntactically correct parses. The former is, however, the semantically correct interpretation of this phrase.

A number of grammatical phenomena can be given different, yet equally plausible, dependency structures according to convention. In the following, we refer to the set of design decisions that specify the structures to produce for each of these phenomena as the *dependency scheme*.

2.1. Link grammar parser

The Link Grammar and its parser represent an implementation of a dependency-based computational grammar. In LG terminology, a dependency is termed *link* and a parse is termed *linkage*. In the following, we use these terms interchangeably. The links in the linkage may not cross when drawn above the words of the sentence, and the types of the links must satisfy the linking constraints specified for each word in the grammar. A linkage where all words, including punctuation, are linked is called *complete*. The links are not directional, that is, they do not explicitly specify which word is the dependent and which word is the governor.

In cases where, due to ambiguity, several linkages can be constructed for an input sentence, the LG parser enumerates all linkages allowed by the grammar. A post-processing step is then employed to enforce a number of additional constraints. Rarely, the number of linkages for a sentence can be very high, in the order of hundreds of millions, which would make post-processing and storage prohibitively resource-intensive. This problem is addressed in the LG parser by defining $k_{\max}$, the maximal number of linkages to be post-processed. If the parsing algorithm produces more than $k_{\max}$ linkages, the output is reduced to $k_{\max}$ linkages by random sampling. The linkages are then ordered from best to worst using heuristic goodness criteria.

In order to be usable in practice, a parser is typically required to provide a partial parse of sentences for which it cannot construct a full parse. If the LG parser cannot construct a complete linkage for a sentence, the connectedness requirement is relaxed so that some words do not belong to the linkage at all. The LG parser is also time-limited. If the full set of linkages cannot be constructed in a given time $t_{\max}$, the parser enters the *panic mode*, in which it performs an efficient but considerably restricted parse, resulting in reduced performance. The parameters $k_{\max}$ and $t_{\max}$ set the trade-off between the qualitative performance and the resource efficiency of the parser.

2.2. Connexor Machinese Syntax parser

Connexor Machinese Syntax (CMS) is a commercial parser based on the Functional Dependency Grammar [13] dependency formalism, which in turn builds on the English Constraint Grammar (EngCG) tagger [14]. In contrast to LG, the parser builds on the detailed morphological analysis provided by EngCG. The CMS dependency scheme is more traditional than the LG scheme in using explicitly directional dependencies and being head- and verb-driven.

CMS also adopts a different approach than LG to ambiguous sentences. Instead of enumerating the set of all possible parses allowed by the grammar, the parser returns only a single, preferred, parse. Heuristics are applied to resolve some phenomena, such as prepositional phrase attachment ambiguity (e.g. *antibodies-to-desmin* versus *staining-to-desmin*). This strategy avoids the performance implications of generating a potentially large amount of alternative parses and does not involve the costly post-processing phase of LG. In our experiments, the CMS parser produced parses very efficiently for even the most complex sentences, and while we are not aware of the detailed operation of the parser, there was nothing to suggest that the parser applies time-limits or different parsing modes for parses that take longer than a given time.

Similarly to LG, CMS is capable of constructing partial parses for sentences for which a fully connected parse is not found.

3. Challenges in parser comparison

One of the issues complicating parser comparison is that different parsers implement different schemes and thus have different notions of what constitutes a correct parse. In the following, we describe some systematic differences between the parsers, discuss ways of performing a balanced evaluation

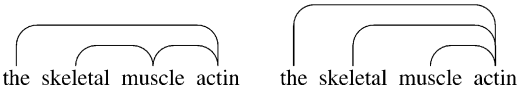

Fig. 1 Serial (left) and parallel (right) attachment of pre-modifiers to a noun.

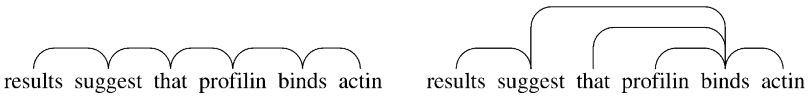

Fig. 2 Dependency structures for subordination. LG structure on the left, CMS structure on the right.

despite such differences, and present the solutions adopted in this paper.

There is no single generally accepted dependency scheme, and many grammatical phenomena can be given several different, yet equally plausible dependency structures. Two examples of such differences between the schemes followed by LG and CMS are illustrated in Figs. 1 and 2. The most frequently occurring difference is in the attachment of pre-modifiers to nouns, where LG attaches pre-modifiers in parallel, while CMS attaches pre-modifiers serially (Fig. 1). The parsers do not attempt to resolve the semantics of the attachment, but instead follow their respective schemes systematically.

Another example of a frequently occurring systematic difference is the dependency structure given to subordinate clauses. Fig. 2 illustrates the structures assigned by the two parsers to the sentence *results suggest that profilin binds actin*. Several other grammatical phenomena, including infinitival and preposed complements as well as coordination, are also analysed differently by the parsers.

Different dependency schemes can differ in their expressive capacity. For example, in the dependency structure used by LG for coordination[5] the coordinated elements are linked to the coordinator, which in turn represents the functional role of each of the coordinates. In contrast, the CMS parser chains the coordinated elements, and the head of the chain shows the functional role of the coordinated units, while the coordinator is a mere dependent of one of the elements in the chain.

Here, the LG approach is more expressive than the CMS chaining approach, allowing, for example, a single pre-modifier to apply to all of the coordinated elements (Fig. 3). Thus, the LG scheme is capable of expressing more detail in the resolution of modifier attachment than can be expressed in the CMS scheme. Note also that the more expressive LG structure can be transformed to the less expressive chaining structure, but not vice versa.

Even when dependency schemes agree on the dependency structure, the dependency types may differ. Indeed, the LG and CMS parsers use very different sets of dependency types, as illustrated in Fig. 4.

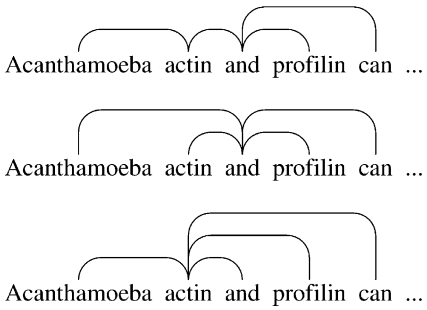

Fig. 3 LG dependency structure for the coordination [[*Acanthamoeba actin*] *and* [*profilin*]] (top) and [*Acanthamoeba* [*actin and profilin*]] (center). The CMS structure (bottom) is not capable of expressing the distinction and the annotation is thus the same for both cases.

There are several alternative solutions available for comparing parsers despite differences such as those discussed above. One possibility is to only compare those structures that are analysed in the same way by the different parsers. However, in our case, even if dependency types are disregarded, the two parsers share only about 74% of the dependencies in the corpus. Restricting the comparison to these dependencies would mean giving up the ability to evaluate the performance of the parsers with respect to several key phenomena such as coordination.

The opposite alternative would be to produce full dependency annotation separately for both parsers. While this approach is straightforward, it has the significant drawback of demanding essentially twice the effort required to produce annotation for a single parser.

Here, we adopt an approach that allows us to have full dependency annotation for both parsers

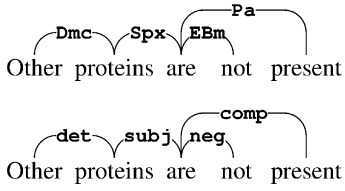

Fig. 4 Different dependency types for the same structure. Above, LG; below, CMS.

[5] When producing output, LG splits sentences containing coordination into several sub-sentences. Instead of the split output, we consider the internal representation used by LG.

without doing twice the work. We use an annotation formalism that allows automatic transformations between the dependency schemes for the main differences between dependency structures, only requiring manual modifications to the annotation for less common phenomena. Due to the significant differences in dependency types used by the two parsers, we do not assign or compare the types of the dependencies.

4. Corpus annotation and interaction subgraphs

To compile a corpus of sentences describing protein–protein interactions, we first selected pairs of proteins that are known to interact from the Database of Interacting Proteins[6] [15]. We entered these pairs as search terms into the PubMed retrieval system. We then split the publication abstracts returned by the searches into sentences and included titles. These were again searched for the protein pairs. Domain experts annotated these sentences for protein names and for words stating their interactions. Thereafter, we performed a syntactic analysis and produced an annotation of dependencies.

The dependency structure annotation of the corpus was produced in a way that allows generating separate versions of the corpus corresponding either to the LG or to the CMS schemes. Most notably, automatic processing was applied to noun pre-modifier attachment and coordination. Noun phrases were marked with annotation that can be expanded to correspond either to the parallel or to the serial modifier attachment scheme. Annotation of coordinations was produced according to the LG scheme, which can be automatically translated to the corresponding CMS annotation using a set of translation rules. This annotation system allows us to avoid producing separate parser-specific annotation manually for all but 6% of the dependencies.

To minimize the amount of mistakes, each sentence was independently annotated by two annotators and differences were then resolved by discussion. Dependency types were not included in the annotation, and no cycles were introduced in the dependency graphs. All ambiguities the parsers attempt to resolve, such as prepositional phrase attachment, were resolved in the corpus annotation as well.

A random sample consisting of 300 sentences has been annotated for both parsers, giving over 7500 word-to-word dependencies. This set of sentences is the corpus we refer to in the following sections.

An information extraction system targeted at protein–protein interactions and their types needs to identify the constituents that express an interaction in a sentence: the proteins involved and the word or phrase that states their interaction and suggests the type of this interaction. To extract this information from a full dependency parse, the dependencies connecting these items must be recovered correctly by the parser. The following definition formalizes this notion considering the dependency parse as a graph with the words as its nodes and the dependencies as its edges.

Definition *(Interaction subgraph)*. The interaction subgraph for an interaction between two proteins A and B in a dependency parse P is the minimal connected subgraph of P that contains A, B, and the word or phrase that states their interaction.

For each interaction stated in a sentence, the corpus annotation specifies the proteins involved and the interaction word. The interaction subgraph for each interaction can thus be extracted automatically from the corpus. Because the corpus does not contain cyclic dependencies, the interaction subgraphs are unique. In total, 401 interaction subgraphs were identified from the corpus, one for each stated interaction. The interaction subgraphs can be partially overlapping, because a single dependency can be part of more than one interaction subgraph. Fig. 5 shows an example of an annotated sentence with an interaction subgraph.

5. Evaluation criteria

We evaluated the performance of the parsers according to the following three quantitative criteria:

- Number of dependencies recovered,
- Number of fully correct parses,
- Number of interaction subgraphs recovered.

The number of recovered dependencies gives an estimate of the probability that a dependency will be correctly identified by the parser (this criterion is also employed by e.g. Collins et al. [16]). The number of fully correct parses, that is, the parses where all annotated dependencies are recovered, measures the fraction of sentences that are parsed without error. However, a fully correct parse is not necessary to extract protein–protein interactions from a sentence; to estimate how many interactions can potentially be recovered, we measure

[6] http://dip.doe-mbi.ucla.edu.

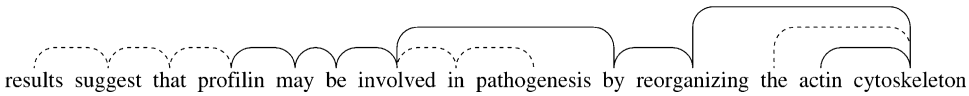

Fig. 5 Annotation example. The interaction of two proteins, *profilin* and *actin*, is stated by the word *reorganizing*. The dependencies joining these words form the interaction subgraph (drawn with solid lines). The annotation follows the LG scheme.

the number of interaction subgraphs for which all dependencies were recovered.

CMS produces exactly one parse for each sentence and we measure the performance of the parser with respect to this analysis. The LG parser, however, enumerates many alternative linkages, heuristically ordered from most likely to least likely. Apart from measuring the performance for the *first linkage*, we also measure the performance for the *best linkage*. The best linkage is the linkage that maximizes the criterion among all the alternative linkages produced by the parser, disregarding the heuristics that the parser uses to order the linkages. The best linkage performance is thus the maximal performance achievable by the parser, providing ideal heuristics for parse ordering. Comparison of the first linkage and the best linkage performance then reveals the effect of the ordering heuristics on the overall performance of the LG parser.

To study the effect of the resource constraints in the LG parser, we divide the parsed sentences into three categories: (1) sentences for which the time t_{max} for producing a normal parse was exhausted and the parser entered the *panic* mode, (2) sentences where linkages were *sampled* because more than k_{max} linkages were produced, and (3) *stable* sentences for which neither of these occurred. We report LG performance for each of the three categories separately, as well as the overall performance. For CMS, where such a division is not applicable, we only give the overall performance.

To assess the statistical significance of differences in parsing performance, we apply the two-tailed paired *t*-test on the fraction of recovered dependencies in each sentence.

6. Evaluation results

In the following, we present the evaluation results for the two parsers. We present and compare the results for LG and CMS. We also consider parser efficiency and resource exhaustion.

6.1. Link grammar results

To evaluate the ability of the LG parser to produce correct linkages, we increased the number of stable sentences by setting the t_{max} parameter to 10 min and the k_{max} parameter to 10000 instead of using the defaults $t_{max} = 30$ s and $k_{max} = 1000$. When parsing the corpus using these parameters, 21 sentences fell into the panic category, 66 into the sampled category, and 213 were stable. The measured

Table 1 LG performance

Criterion	Linkage	Category						Overall	
		Stable		Sampled		Panic			
Dependency	First linkage	3399	(77.3%)	1574	(73.1%)	522	(52.7%)	5495	(72.9%)
	Best linkage	3771	(85.8%)	1790	(83.1%)	571	(57.6%)	6132	(81.3%)
	Total	4397		2157		991		7541	
Fully correct	First linkage	20	(9.4%)	1	(1.5%)	0	(0.0%)	21	(7.0%)
	Best linkage	73	(34.3%)	8	(12.1%)	0	(0.0%)	81	(27.0%)
	Total	213		66		21		300	
Interaction subgraph	First linkage	86	(32.1%)	22	(25.0%)	0	(0.0%)	108	(26.9%)
	Best linkage	172	(64.2%)	57	(64.8%)	4	(8.9%)	233	(58.1%)
	Total	268		88		45		401	

The fraction of fulfilled criteria is shown by category (the criteria and categories are explained in Section 5). The total rows give the number of criteria for each category, and the overall column gives combined results for all categories.

parser performance for the corpus is presented in Table 1.

The overall fraction of individual dependencies recovered in the first linkage is 73%. Dividing this overall result into the three categories of sentences, we find remarkable differences. For example, for the panic category sentences, the parser recovered 25% units fewer dependencies than for the stable category sentences. The difference between the stable and sampled categories is a more modest 4% units. Coupled with the fact that the sentences in the sampled category also tend to be more ambiguous, the result indicates that sampling does not degrade the performance of the parser significantly.

The fraction of sentences that have a fully correct parse as the first linkage is very low (7%). For 27% of sentences the parser is capable of producing a fully correct parse, yet the parse ordering heuristics fail to identify it in the majority of cases. Performance was especially poor for the publication titles in the corpus. Titles are typically fragments not containing a verb and LG is designed to model full clauses. The parser failed to produce a fully correct linkage for any of the 27 fragments in the corpus.

The performance for recovered interaction subgraphs is more encouraging, as 27% of the subgraphs were recovered in the first linkage. Yet many interaction subgraphs remain unrecovered by the parser even in the best linkage. The results suggest an upper limit of 58% to the fraction of protein—protein interactions that can be recovered from any linkage produced by the unmodified LG. The results indicate that two important factors for LG failures to recover all dependencies are resource exhaustion and ordering heuristics.

The heuristics that LG applies to order linkages are based on examination and intuitions on general English, and may not be optimal for biomedical text. Note in Table 1 that both for recovered full linkages and interaction subgraphs, the number of items that were recovered in the best linkage is more than twice the number recovered in the first linkage, suggesting that a better ordering heuristic could dramatically improve the performance of the parser. Such improvements could perhaps be achieved by tuning the heuristics to the domain or by adopting a probabilistic ordering model.

6.2. Connexor Machinese Syntax results and comparison

The evaluation results for CMS are presented in Table 2. As the division into the stable, sampled and panic categories as well as the output of several parses are specific to LG, we only give overall performance for CMS.

Table 2 CMS performance

Criterion	Overall	
Dependency	6031/7540	(80.0%)
Fully correct	21/300	(7.0%)
Int. subgraph	145/401	(36.2%)

Even with 80% of dependencies recovered, the CMS parser correctly identifies only 36% of the interaction subgraphs. This figure suggests an upper limit on the fraction of protein—protein interactions that can be recovered from full dependency parses produced by CMS. The fraction of fully correct sentences is 7%.

When comparing the results of CMS with LG, the difference in the expressive power of their respective dependency schemes should be taken into consideration. As illustrated in Fig. 3, LG is capable of expressing more detail for some coordinations than CMS. Consequently, in these cases LG is required to resolve more ambiguity than CMS and is thus expected to perform slightly worse. However, an analysis of the corpus reveals that such attachments account for only 0.7% of individual dependencies, thus rendering the effect on the parser comparison negligible.

With 80% of individual dependencies recovered in the single parse generated, CMS performs significantly better ($p < 0.001$) than LG (73% in the first linkage). The performance of CMS is competitive even with the overall best linkage performance of LG (81%) that ignores the errors introduced by the LG parse ordering heuristics.

CMS also notably outperforms LG (first linkage) on the recovery of interaction subgraphs. Interestingly, in contrast to the individual dependencies, the CMS performance on interaction subgraph recovery does not compare to LG best linkage performance. Further, for the fully correctly parsed sentences measure, both parsers perform at the same, low, level of 7%.

To relate these numbers to performance on non-biomedical text, the Functional Dependency Grammar parser, on which CMS builds, has been reported to achieve performance between 87.9% and 88.6% for recovery of individual dependencies on three different genres on Bank of English data [13]. Another study of a broad-coverage dependency parser, MINIPAR [17], on the SUSANNE Corpus reported 79% recovery of individual dependencies. The performance of the parsers on biomedical text is thus lower than what has been reported

for broad-coverage dependency parsers on general English.

6.3. Efficiency and resource exhaustion

The CMS parser is particularly efficient—the components that form the core of the parser have been estimated to have average running times of $O(n \lg n)$, where n is the length of the sentence [18]. In practice, CMS averaged roughly 0.5 s per sentence in our experiments. In general, it appears that resource exhaustion is not an issue for the CMS parser.

In contrast to CMS, resource exhaustion is a significant issue for the LG parser. Our experiments indicate than on a 1 GHz personal computer sentences that can be fully parsed in under one second constitute 40% of the corpus. Similarly, 80% require less than 10 s and 90% less than 10 min each. Yet approximately 5% of sentences would require more than an hour each to fully parse. With t_{max} set to 10 min, the total parsing time was 165 min, averaging approximately 30 s per sentence.

No fully correct linkages and very few interaction subgraphs were found in the panic mode, which the parser enters when resources are exhausted, that is, when the current parsing time reaches t_{max}. The effect of panics on performance can be directly estimated by forcing the parser to bypass standard parsing and to directly enter the panic mode. We found that the fraction of recovered dependencies then falls by approximately 10% units, and the number of interaction subgraphs recovered in the first linkage decreases by approximately 30% units. The panic mode parsing is thus a significant source of errors.

For LG, long parsing times and subsequent resource exhaustion are mainly caused by ambiguous sentences for which the parser may create thousands or even millions of alternative linkages. In addition to simply increasing the time limit, the panic mode can thus be avoided by reducing the ambiguity of the sentences, for example, by extending the dictionary of the parser (see Section 8.2).

7. Failure analysis of Link Grammar

A significant fraction of dependencies were not recovered in any linkage by LG, even in sentences where resources were not exhausted. In order to identify reasons for the parser failing to recover the correct dependencies, we perform a manual failure analysis of sentences where it is certain that LG cannot produce a fully correct linkage, that is, for sentences in the stable category. For sentences in the other two categories, random effects may affect the results: sentences for which more than k_{max} linkages are produced are subject to randomness in sampling, and in our experiments sentences where the parser enters panic mode were always subject to subsequent sampling.

Table 3 Results of failure analysis

Reason for failure	Cases	
Unknown grammatical structure	72	(34.4%)
Dictionary issue	54	(25.8%)
Unknown word handling	35	(16.7%)
Sentence fragment	27	(12.9%)
Ungrammatical sentence	17	(8.1%)
Other	4	(1.9%)

Due to the closed, proprietary nature of the CMS parser, it would be difficult to identify the core reasons for parser errors, and hence we do not perform failure analysis for CMS. Further, as it is not possible for CMS users to alter the syntactic rules of the parser, a detailed failure analysis would be of limited utility.

For LG, we attempt to identify the reason for the failure of the parser separately for each sentence. For each identified reason, we manually edit the sentence to remove the source of failure. We repeat this procedure until the parser is capable of producing a correct parse for the sentence. Note that this implies that also the interaction subgraphs in the sentence are correctly recovered, and therefore the reasons for failures to recover interaction subgraphs are a subset of the identified issues. The results of the analysis are summarized in Table 3. In many of the sentences, more than one reason for parser failure was found; in total 209 issues were identified in the 132 analyzed sentences[7].

The results are described in more detail in the following sections.

7.1. Fragments and ungrammatical sentences

As some of the analysed sentences were taken from publication titles, not all of them were full clauses. To identify further problems when parsing fragments not containing a verb, the phrase ''is/are explained'' and required determiners were added to these fragments, a technique used also by Ding et

[7] The failure analysis was performed based on our earlier evaluation results. The number of sentences for which the analysis is performed thus slightly differs from the most recent results presented here.

al. [9]. The completed fragments were then analysed for potential further problems.

A number of other ungrammatical sentences were also encountered. The most common problem was the omission of determiners, but some other issues, such as missing possessive markers and errors in agreement (e.g. *expressions. . .has*) were also encountered.

Ungrammatical sentences pose interesting challenges for parsing. Because many authors are not native English speakers, a greater tolerance for grammatical mistakes should allow LG to identify the intended parse for more sentences. Similarly, the ability to parse publication titles would extend the applicability of the parser; in some cases it may be possible to extract information concerning the key findings of a publication from the title. However, while relaxing completeness and correctness requirements, such as mandatory determiners and subject-predicate agreement, would allow LG to create a complete linkage for more sentences, it would also be expected to lead to increased ambiguity for all sentences, and subsequent difficulties in identifying the correct linkage. If the ability to parse titles is considered important, a potential solution not incurring this cost would be to develop a separate version of the grammar for parsing titles. In contrast to LG, fragments pose no issues for the CMS parser.

7.2. Unknown grammatical structures

The method of the LG implementation for parsing coordinations was found to be a frequent cause of failures. A specific coordination problem occurs with multiple noun-modifiers: the parser assumes that coordinated constituents can be connected to the rest of the sentence through exactly one word, and the grammar attaches all noun-modifiers to the head. Biomedical texts frequently contain phrases that cause these requirements to conflict. For example, in the phrase *capping protein and actin genes* (where [[[*capping protein*] *and* [*actin*]] *genes*] is the intended interpretation), the parser allows only one of the words *capping* and *protein* to connect to the word *genes*, and is thus unable to produce the correct linkage (for illustration, see Fig. 6).

This particular multiple modifier coordination issue could be addressed by modifying the grammar to chain modifiers. This alternative model is adopted by CMS. The problem could also potentially be addressed by altering the coordination system in the parser.

Other identified grammatical structures not known to the parser were numeric postmodifiers

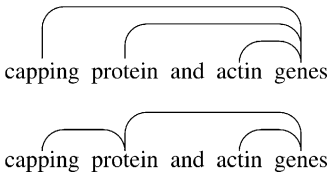

Fig. 6 Multiple modifier coordination problem. Above: correct linkage disallowed by the LG parser. Below: solution by chaining modifiers.

to nouns (e.g., *serine 38*), some instances of specifiers in parentheses (e.g. *myosin heavy chain* (*MHC*) *expression*), coordination with the phrase *but not*, and various unknown uses of punctuation. Single instances of several distinct unknown grammatical structures were also noted (e.g. *5—10*, *as expected from*, *most concentrated in*). Most of these issues can be addressed by local modifications to the grammar.

7.3. Unknown word handling

The LG parser assigns unknown words to categories based on morphological or other surface clues when possible. For remaining unknown words, parses are attempted by assigning the words to the generic noun, verb, and adjective types in all possible combinations.

Some problems with the unknown word processing method were encountered during the analysis. For example, the assumption that unknown capitalized words are proper nouns often caused failures, especially in sentences beginning with an unknown word. Similarly, the assumption that words containing a hyphen behave as adjectives was violated by a number of unknown verbs (e.g., *cross-links*).

Another problem that was noted occurred with lowercase unknown words that should be treated as proper nouns: because LG does not allow unknown lowercase words to act as proper nouns, the parser assigns incorrect structure to a number of phrases containing words such as *actin*. Improving unknown word handling requires some minor modifications to the LG parser.

7.4. Dictionary issues

Cases where the LG dictionary contains a word, but not in the sense in which it appears in a sentence, almost always lead to errors. For example, the LG dictionary does not contain the word *assembly* in the sense *construction*, causing the parser to erroneously require a determiner for *protein*

Table 4 Overall performance with identified biomedical entities

Criterion	LG first linkage		Δ	CMS		Δ
Dependency	5569/7251	(76.8%)	3.9%	5870/7250	(81.0%)	1.0%
Fully correct	26/300)	(8.7%	1.7%	24/300	(8.0%)	1.0%
Interaction subgraph	139/401	(34.7%)	7.8%	167/401	(41.6%)	5.4%

The Δ columns give the difference to parser performance without prerecognition of entity names. Note that the total number of dependencies in the corpus has changed, since multi-word entity names have been replaced with single tokens.

assembly[8]. A related frequent problem occurred with proper names headed by a common noun, where the parser expects a determiner for such names (e.g. *myosin heavy chain*), and fails when one is not present. Such dictionary issues are mostly straightforward to address in the grammar, but difficult to identify automatically.

8. Evaluation of potential improvements

In this section, we consider and evaluate two approaches for improving parser performance: applying the recognition of bioentity names as a preprocessing step and extending the LG dictionary.

8.1. Biomedical entity names

We have noted that many of the causes for parser failure are related to the presence of biomedical entity names. For LG, perhaps the most common class of such failure stems from the common assumption that proper names are consistently recognizable by capitalization, which often does not hold for entity names such as *profilin*. While the causes for failures related to names can be addressed in the grammar, the existence of biomedical named entity recognition (NER) systems suggests an alternative solution: named entities could be identified in preprocessing and treated as single (proper noun) tokens during the parse. To test the effect of such a preprocessing step on parser performance, we replaced annotated bio-entity names, such as proteins, including multi-word names, with single words that can be recognized by both parsers as proper names. The sentences were then parsed and we re-evaluated parser performance. The overall results for the two parsers are given in Table 4.

For the number of recovered dependencies, an increase of 3.9% units is observed for LG ($p < 0.001$). The respective increase for CMS is 1.0% units ($p < 0.01$). Interaction subgraph performance also shows a substantial increase for both parsers. While the positive effect was greater for LG than for CMS in all criteria, CMS performance remains significantly better ($p < 0.001$). Thus, as expected, NER preprocessing results in a systematic improvement in all criteria for both parsers. However, the performance of current automatic NER systems is not perfect. In a recent competitive evaluation of NER methods, the best-performing system achieved 70% precision and 76% recall [19]. State-of-the-art NER systems would thus not be expected to achieve the full benefits observed here.

8.2. Dictionary extension for link grammar

Szolovits describes an automatic method for mapping lexical information from one lexicon to another, and applies this method to augment the LG dictionary with terms from the extensive UMLS Specialist lexicon [8]. The extension introduces more than 125,000 new words into the LG dictionary, more than tripling its size. We repeated the performance evaluation for LG with the extended dictionary. The dictionary extension is only available for LG and thus we do not perform a similar evaluation for CMS.

The proportion of distinct corpus words recognized by LG increased from 52% to 72% with the dictionary extension, representing a significant reduction in uncertainty due to unknown words. This reduction was coupled with a 32% decrease in total parsing time.

The effect of the dictionary extension was negative for publication titles, which are often fragments not containing a verb. Since LG requires a verb to be present, it can only parse a fragment if it can incorrectly analyse some unknown word in the fragment as a verb. Since extending the dictionary reduces the number of unknown words, the fraction of individual dependencies recovered by LG in fragments decreased. The results in Table 5 are thus given separately for fragments and non-fragments. None of the differences resulting from the dictionary extension in the number of recovered dependencies were statistically significant.

[8] Thirty distinct problematic word definitions were identified, including *breakdown*, *composed*, *factor*, *half*, *independent*, *localized*, *parallel*, *promoter*, *segment* and *upstream*.

Table 5 LG performance with the dictionary extension

Criterion	LG first linkage		Δ Fragment	Δ Non-fragment	Δ Overall
Dependency	5489/7541	(72.8%)	−8.8%	+0.4%	−0.1%
Fully correct	22/300	(7.3%)	0.0%	+0.4%	+0.3%
Interaction subgraph	110/401	(27.4%)	+4.6%	+0.3%	+0.5%

The Δ columns give the difference to parser performance without the extension.

For non-fragments, the benefits of the dictionary extension were most notable for sentences that were in the panic category when using the unextended LG dictionary; seven of these 21 sentences could be parsed without panic with the dictionary extension. In the first linkage of these 21 sentences, the fraction of recovered dependencies increased by 4% units, however, the difference was not statistically significant. In our previous experiments with stricter resource limits we observed greater gains from adopting the dictionary extension. However, the new results confirm that the greatest effect of the extension is in reducing ambiguity, which in turn is especially beneficial when computational resources are limited.

The overall effect of the dictionary extension was negligible, despite the three-fold increase in dictionary size. This result agrees with the failure analysis: most problems cannot be removed by extending the dictionary and must instead be addressed by modifications of the grammar or parser.

9. Conclusion

In this paper, we have studied the applicability of full dependency parsers to information extraction in the biomedical domain, with a focus on protein–protein interactions. For this purpose, we have developed a custom hand-annotated dependency corpus. Here, we present an evaluation of two major broad-coverage dependency parsers, Link Grammar and Connexor Machinese Syntax. The performance of the parsers was evaluated with respect to the standard criteria of recovered individual dependencies and fully correct parses. To further focus the evaluation on the IE task at hand, we have introduced the concept of the interaction subgraph, which captures the structures in the sentence that are relevant to protein–protein interactions. The interaction subgraphs were used as an additional criterion in the parser evaluation.

LG was able to recover 73% of dependencies in the first linkage, compared to 80% for CMS. CMS thus significantly outperforms LG on biomedical text ($p < 0.001$). The performance of both parsers nevertheless remains below what has previously been reported for broad-coverage parsers on general English.

For the recovery of interaction subgraphs, the performance figures were 27% and 36% for LG and CMS, respectively. The low fraction of recovered interaction subgraphs indicates that neither of the parsers in its current form is well applicable to biomedical information extraction tasks. However, analysis of best linkage performance suggests that LG has remarkable potential for improvement in interaction subgraph recovery through better heuristics.

We also evaluated the effect of using named entity recognition as a preprocessing step and found a systematic improvement for both parsers and all the criteria. The improvement was particularly pronounced for the recovery of interaction subgraphs, which increased to 35% and 42% for LG and CMS, respectively.

In contrast to CMS, LG is an open system that can be investigated in great detail and freely adapted, and therefore we undertook a more detailed analysis of the common causes of LG failures. In addition to errors caused by resource exhaustion and ordering heuristics, several distinct causes of parser failure were determined by manual analysis. We carefully examined the sentences and were able to identify five problem types. For each identified type, we discussed potential modifications for addressing the problems.

We also evaluated the effect of the LG dictionary extension proposed by Szolovits [8] and found that while it significantly reduced ambiguity, overall performance improvement was negligible. This indicates that extending the dictionary is not sufficient to address the performance problems of LG and that modifications to the grammar and parser itself are necessary.

Since both parsers perform below their generally expected performance and fail to capture more than half of the interaction subgraphs in the corpus, domain adaptation is necessary to increase the applicability of these full dependency parsers to the information extraction tasks in the biomedical domain. In the failure analysis, we proposed concrete adaptations for LG, and one would expect

Contribution of the study
Previous knowledge:

- Informal evaluation for Link Grammar parser, no evaluation of Connexor machinese
- No formal evaluation for any dependency parser in the domain
- Lexicon extension for Link Grammar proposed but not evaluated with respect to parser performance

Knowledge added by this study:

- Both Link Grammar and Connexor machinese formally evaluated and compared, using standard criteria and hand-annotated corpus
- Detailed Link Grammar evaluation, discussing issues such as sampling and panic mode
- Primary sources of parser failure in Link Grammar identified and improvements proposed
- Previously proposed extension to Link Grammar lexicon evaluated with respect to parser performance

that similar adaptations could be identified and implemented for CMS as well. The adaptation of LG according to the results of the failure analysis is a natural follow-up of this study. Indeed, our initial experiments suggest that it is possible to implement many of these, increasing the applicability of the parser.

Acknowledgments

We are grateful to Meelis Kolmer for consultation and wish to thank the annotators Jari Björne, Juho Heimonen, Jeppe Koivula, Lilli Nurmi, and Suvi Laukkanen for their efforts in producing the corpus. This work has been supported by Tekes, the Finnish National Technology Agency.

References

[1] E.M. Marcotte, I. Xenarios, D. Eisenberg, Mining literature for protein-protein interactions, Bioinformatics 17 (2001) 359—363.
[2] F. Ginter, T. Pahikkala, S. Pyysalo, J. Boberg, J. Järvinen, T. Salakoski, Extracting protein—protein interaction sentences by applying rough set data analysis, in: S. Tsumoto, R. Slowinski, J. Komorowski, J. Grzymala-Busse (Eds.), Lecture Notes in Artificial Intelligence 3066, Springer, Heidelberg, 2004.
[3] M. Craven, J. Kumlien, Constructing biological knowledge bases by extracting information from text sources, in: T. Lengauer, R. Schneider, P. Bork, D. Brutlag, J. Glasgow, H-W. Mewes, R. Z (Eds.), Proceedings of the Seventh International Conference on Intelligent Systems in Molecular Biology, AAAI Press, Menlo Park, CA, 1999, pp. 77—86.
[4] A. Yakushiji, Y. Tateisi, Y. Miyao, J. Tsujii, Event extraction from biomedical papers using a full parser, in: R.B. Altman, A.K. Dunker, L. Hunter, K. Lauderdale, T. Klein (Eds.), Proceedings of the Sixth Pacific Symposium on Biocomputing (PSB 2001), World Scientific Press, Singapore, 2001, pp. 408—419.
[5] J.M. Temkin, M.R. Gilder, Extraction of protein interaction information from unstructured text using a context-free grammar, Bioinformatics 19 (2003) 2046—2053.
[6] N. Daraselia, A. Yuryev, S. Egorov, S. Novichkova, A. Nikitin, I. Mazo, Extracting human protein interactions from MEDLINE using a full-sentence parser, Bioinformatics 20 (2004) 604—611.
[7] D. Sleator, D. Temperley, Parsing English with a Link Grammar, Tech. Rep. CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1991.
[8] P. Szolovits, Adding a medical lexicon to an English parser, in: M. Musen (Ed.), Proceedings of the 2003 AMIA Annual Symposium, American Medical Informatics Association, Bethesda, MD, 2003, pp. 639—643.
[9] J. Ding, D. Berleant, J. Xu, A.W. Fulmer, Extracting biochemical interactions from Medline using a Link Grammar parser, in: B. Werner (Ed.), Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society, Los Alamitos, CA, 2003, pp. 467—471.
[10] E. Alphonse, S. Aubin, P. Bessiéres, G. Bisson, T. Hamon, S. Lagarrigue, A. Nazarenko, A.-P. Manine, C. Nédellec, M.O.A. Vetah, T. Poibeau, D. Weissenbacher, Event-based information extraction for the biomedical domain: the Caderige project, in: N. Collier, P. Ruch, A. Nazarenko (Eds.), Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), Geneva, Switzerland, 2004, pp. 43—49.
[11] K. Franzén, G. Eriksson, F. Olsson, L. Asker, P. Lidén, J. Cöster, Protein names and how to find them, Int. J. Med. Inform. 67 (2002) 49—61.
[12] S. Pyysalo, F. Ginter, T. Pahikkala, J. Boberg, J. Järvinen, T. Salakoski, J. Koivula, Analysis of Link Grammar on biomedical dependency corpus targeted at protein—protein interactions, in: N. Collier, P. Ruch, A. Nazarenko (Eds.), Proceedings of the International Joint workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), Geneva, Switzerland, 2004, pp. 15—21.
[13] P. Tapanainen, T. Järvinen, A non-projective dependency parser, in: P. Jacobs (Ed.), Proceedings of the Fifth Conference on Applied Natural Language Processing, Association for Computational Linguistics, Somerset, NJ, 1997, pp. 64—71.
[14] P. Tapanainen, The Constraint Grammar Parser CG-2. Number 27 in Publications of the Department of General Linguistics, University of Helsinki, 1996.
[15] I. Xenarios, L. Salwinski, X.J. Duan, P. Higney, S-M. Kim, D. Eisenberg, DIP, the database of interacting proteins: a research tool for studying cellular networks of protein interactions, Nucl. Acids Res. 30 (2002) 303—305.
[16] M. Collins, J. Hajič, L. Ramshaw, C. Tillmann, A statistical parser for Czech, in: Robert Dale, Ken Church (Eds.), Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Somerset, NJ, 1999, pp. 505—512.

[17] D. Lin, Dependency-based evaluation of MINIPAR, in: J. Carroll (Ed.), Workshop on the Evaluation of Parsing Systems, European Language Resources Association, Paris, France, 1998.
[18] P. Tapanainen, Parsing in two frameworks: finite-state and functional dependency grammar, Ph.D. Thesis, University of Helsinki, 1999.
[19] J.-D. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, N. Collier, Introduction to the bio-entity recognition task at JNLPBA, in: N. Collier, P. Ruch, A. Nazarenko (Eds.), Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA), Geneva, Switzerland, 2004, pp. 70—75.