Intrinsic versus Extrinsic Evaluations of Parsing Systems

Diego Mollá
Centre for Language Technology
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia
diego@ics.mq.edu.au

Ben Hutchinson
Division of Informatics
University of Edinburgh
Edinburgh EH8 9LW, United Kingdom
B.Hutchinson@sms.ed.ac.uk

Abstract

A wide range of parser and/or grammar evaluation methods have been reported in the literature. However, in most cases these evaluations take the parsers independently (intrinsic evaluations), and only in a few cases has the effect of different parsers in real applications been measured (extrinsic evaluations). This paper compares two evaluations of the Link Grammar parser and the Conexor Functional Dependency Grammar parser. The parsing systems, despite both being dependency-based, return different types of dependencies, making a direct comparison impossible. In the intrinsic evaluation, the accuracy of the parsers is compared independently by converting the dependencies into grammatical relations and using the methodology of Carroll et al. (1998) for parser comparison. In the extrinsic evaluation, the parsers' impact in a practical application is compared within the context of answer extraction. The differences in the results are significant.

1 Introduction

Parsing is a principal stage in many natural language processing (NLP) systems. A good parser is expected to return an accurate syntactic structure of a sentence. This structure is typically forwarded to other modules so that they can work with unambiguous and well-defined structures representing the sentences. It is to be expected that the performance of an NLP system quickly degrades if the parsing system returns incorrect syntactic structures, and therefore an evaluation of parsing coverage and accuracy is important.

According to Galliers and Sparck Jones (1993), there are two main criteria in performance evaluation: "*Intrinsic* criteria are those relating to a system's objective, *extrinsic* criteria those relating to its function i.e. to its role in relation to its setup's purpose." (Galliers and Sparck Jones, 1993, p22). Thus, an intrinsic evaluation of a parser would analyse the accuracy of the results returned by the parser as a stand-alone system, whereas an extrinsic evaluation would analyse the impact of the parser within the context of a broader NLP application.

There are currently several parsing systems that attempt to achieve a wide coverage of the English language (such as those developed by Collins (1996), Järvinen and Tapanainen (1997), and Sleator and Temperley (1993)). There is also substantial literature on parsing evaluation (see, for example, work by Sutcliffe et al. (1996), Black (1996), Carroll et al. (1998), and Bangalore et al. (1998)). Recently there has been a shift from constituency-based (e.g. counting crossing brackets (Black et al., 1991)) to dependency-based evaluation (Lin, 1995; Carroll et al., 1998). Those evaluation methodologies typically focus on comparisons of stand-alone parsers (intrinsic evaluations). In this paper we report on the comparison between an intrinsic evaluation and an evaluation of the impact of the parser in a real application (an extrinsic evaluation).

We have chosen answer extraction as an example of a practical application within which to test the parsing systems. In particular, the extrinsic evaluation uses ExtrAns, an answer extraction system that operates over Unix manual pages (Mollá et al., 2000). The two grammar systems to compare are Link Grammar (Sleator and Temperley, 1993) and the Conexor Functional Dependency Grammar parser (Tapanainen and Järvinen, 1997) (henceforth referred to as Conexor FDG). These parsing systems were chosen because both include a dependency-based parser and a comprehensive grammar of English. However, the structures returned are so different that a direct comparison between them is not straightforward. In Section 2 we review the main differences between Link Grammar and Conexor FDG. In Section 3 we present the intrinsic comparison of parsers, and in Section 4 we comment on the extrinsic comparison within the context of answer extraction. The results of the evaluations are discussed in Section 5.

2 Link Grammar and Conexor FDG

Link Grammar (Sleator and Temperley, 1993) is a grammar theory that is strongly dependency-based. A freely available parsing system that implements the Link Grammar theory has been developed at Carnegie Mellon University. The parsing system includes an extensive grammar and lexicon and has a wide coverage of the English language. Conexor FDG (Tapanainen and Järvinen, 1997) is a commercial parser and grammar, based on the theory of Functional Dependency Grammar, and was originally developed at the University of Helsinki.

Despite both being dependency-based, there are substantial differences between the structures returned by the two parsers. Figure 1 shows Link Grammar's output for a sample sentence, and Figure 2 shows the dependency structure returned by Conexor FDG for comparison. Table 1 explains the dependency types used in the dependency structures of the figures.

The differences between the dependency structures returned by Link Grammar 2.1 and Conexor FDG 3.6 can be summarised as follows.

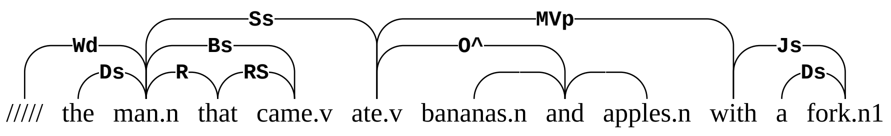
Direction of dependency: Link Grammar's 'links', although similar to true dependencies, do not state which participant is the head and which is the dependent. However, Link Grammar uses different link types for head-right links and head-left links, so this information can be recovered. Conexor FDG always indicates the direction of the dependence.

Clausal heads: Link Grammar generally chooses the front-most element to be the head of a clause, rather than the main verb. This is true of both matrix and subordinate clauses, as exemplified by the `Wd` and `R` links in Figure 1. Conexor FDG follows the orthodox convention of choosing the main verb as the head of the clause.

Graph structures: Link Grammar's links combine dependencies at the surface-syntactic and deep-syntactic levels (e.g., the link `Bs`, which links a noun modified by a subject-type relative clause to the relative clause's head verb, in Figure 1 indicates a deep-syntactic dependency). The resulting structures are graphs rather than trees. An example is shown in Figure 1, where the noun *man* modified by a relative clause is linked to both the complementiser and the head verb of the relative clause.

Conjunctions: Our version of Link Grammar analyses a coordinating conjunction as the head of a coordinated phrase (Figure 1). This is a modification of Link Grammar's default behaviour which returns a list of parses, one parse per conjunct. However in Conexor FDG's analyses the head will be either the first or the last conjunct, depending on whether the coordinated phrase's head lies to the left or to the right (Figure 2).

Dependency types: Link Grammar uses a set of about 90 link types and many subtypes, which address very specific syntactic constructions (e.g. the link type `EB` connects adverbs to forms of *be* before a noun phrase or prepositional phrase: *He is APPARENTLY a good programmer*). On the other hand, Conexor FDG uses a set of 32 de-



Ss
MVp
Wd
Bs
O^
Js
Ds
R
RS
Ds
///// the man.n that came.v ate.v bananas.n and apples.n with a fork.n1

Figure 1: Output of Link Grammar.

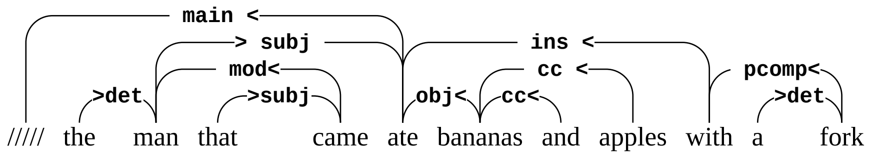


Figure 2: Dependency structure returned by Conexor FDG.

pendency relations, ranging from traditional grammatical functions (e.g. subject, object), to specific types of modifiers (e.g. frequency, duration, location).

Both Conexor FDG and Link Grammar also return non-dependency information. For Link Grammar, this consists of some word class information, shown as suffixes in Figure 1. For Conexor FDG, the base form morphological information of each word is returned, along with a "functional" tag or morpho-syntactic function and a "surface syntactic" tag for each word.[1]

3 Intrinsic Evaluations

Given that both parses are dependency-based, intrinsic evaluations that are based on constituency structures (e.g. (Black et al., 1991)) are hard to perform. Dependency-based evaluations are not easy either: directly comparing dependency graphs (as suggested by Lin (1995), for example) becomes difficult given the differences between the structures returned by the Link Grammar parser and Conexor FDG. We therefore need an approach that is independent from the format of the parser output. Following Carroll et al. (1998) we use *grammatical relations* to compare the accuracy of Link Grammar and Conexor FDG. Carroll et al. (1998) propose a set of twenty parser-independent grammatical relations arranged in a hierarchy representing different degrees of specificity. Four relations from the hierarchy are shown in Table 2. The arguments to each relation specify a head, a dependent, and possibly an initial grammatical relation (in the case of `SUBJ` in passive sentences, for example) or the 'type', which specifies the word introducing the dependent (in the case of `XCOMP`).

[1]See (Järvinen and Tapanainen, 1997) for more information on the output from Conexor FDG.

For example, the grammatical relations of the sentence *the man that came ate bananas and apples with a fork without asking* has the following relations:

```
SUBJ(eat,man,_),
OBJ(eat,banana),
OBJ(eat,apple),
MOD(fork,eat,with),
SUBJ(come,man,_),
MOD(that,man,come),
XCOMP(without,eat,ask)
```

The terms 'head' and 'dependent' used by Carroll et al. (1998) to refer to the arguments of grammatical relations should not be confused with the similar terms in the theory of dependency grammar. Grammatical relations and dependency arcs represent different phenomena. An example should suffice to illustrate the difference; consider *The man that came ate bananas and apples with a fork.* In dependency grammar a unique head is assigned to each word, for example the head of *man* is *ate*. However *man* is the dependent of more than one grammatical relation, namely `SUBJ(eat,man,_)` and `SUBJ(come,man,_)`. Furthermore, in dependency grammar a word can have at most one dependent of each argument type, and so *ate* can have at most one object, for example. But

Link Grammar		**Conexor FDG**	
Name	*Description*	*Name*	*Description*
`Bs`	Singular external object of relative clause	`cc`	Coordination
`Ds`	Singular determiner	`det`	Determiner
`Js`	Singular object of a preposition	`ins`	*<not documented>*
`MVp`	Verb-modifying preposition	`main`	Main element
`O^`	Object	`mod`	General post-modifier
`R`	Relative clause	`obj`	Object
`RS`	Part of subject-type relative clause	`pcomp`	Prepositional complement
`Ss`	Singular subject	`subj`	Subject
`Wd`	Declarative sentence		

Table 1: Some of the dependency types used by Link Grammar and Conexor FDG.

Relation	*Description*
`SUBJ(head, dependent, initial_gr)`	Subject
`OBJ(head, dependent)`	Object
`XCOMP(type, head, dependent)`	Clausal complement without an overt subject
`MOD(type, head, dependent)`	Modifier

Table 2: Grammatical relations used in the intrinsic evaluation.

the same is not true for grammatical relations, and we get both `OBJ(eat,banana)` and `OBJ(eat,apple)`.

3.1 Accuracy

Our intrinsic evaluation began on the assumption that grammatical relations could be deduced from the dependency structures returned by the parsers. In practise, however, this deduction process is not always straightforward; for example complexity arises when arguments are shared across clauses. In addition, Link Grammar's analysis of the front-most elements as clausal heads complicates the grammatical relation deduction when there are modifying clauses.

An existing corpus of 500 sentences/10,000 words annotated with grammatical relations was used for the evaluation (Carroll et al., 1999). We restricted the evaluation to just the four relations shown in Table 2. This decision had two motivations. Firstly, since the dependency parsers' output did not recognise some distinctions made in the hierarchy of relations, it did not make sense to test these distinctions. Secondly, we wanted the deduction of grammatical relations to be as simple a process as possible, to minimise the chance of introducing errors. This second consideration also led us to purposefully ignore the sharing of arguments induced by control verbs, as this could not always be deduced reliably. Since this was done for both parsers the comparison remains meaningful.

Algorithms for producing grammatical relations from Link Grammar and Conexor FDG output were developed and implemented. The results of parsing the corpus are shown in Table 3. Since Conexor FDG returns one parse per sentence only and Link Grammar returns all parses ranked, the first (i.e. the best) parse returned by Link Grammar was used in the intrinsic evaluation.

The table shows significantly lower values of recall and precision for Link Grammar. This is partly due to the fact that Link Grammar's links often do not connect the head of the clause, as we have seen with the `Wd` link in Figure 1.

3.2 Speed

Link Grammar took 1,212 seconds to parse the 10,000 word corpus, while Conexor FDG took 20.5 seconds. This difference is due partly to the fact that Link Grammar finds and returns multiple (and often many) alternative parses. For example,

		With Link Grammar	*With Conexor FDG*
Precision	SUBJ	50.3%	73.6%
	OBJ	48.5%	84.8%
	XCOMP	62.2%	76.2%
	MOD	57.2%	63.7%
	Average	***54.6%***	***74.6%***
Recall	SUBJ	39.1%	64.5%
	OBJ	50%	53.4%
	XCOMP	32.1%	64.7%
	MOD	53.7%	56.2%
	Average	***43.7%***	***59.7%***

Table 3: Accuracy of identification of grammatical relations.

Link Grammar found a total of 410,509 parses of the 505 corpus sentences.

4 Extrinsic Evaluations

It is important to know not only the accuracy of a parser but how possible parsing errors affect the success of an NLP application. This is the goal of an extrinsic evaluation, where the system is evaluated in relation to the embedding setup. Using answer extraction as an example of an NLP application, we compared the performance of the Link Grammar system and Conexor FDG.

4.1 Answer Extraction and ExtrAns

The fundamental goal of Answer Extraction (AE) is to locate those exact phrases of unedited text documents that answer a query worded in natural language. AE has received much attention recently, as the increasingly active Question Answering track in TREC demonstrates (Voorhees, 2001b; Voorhees, 2001a).

ExtrAns is an answer extraction system that operates over UNIX manual pages (Mollá et al., 2000). A core process in ExtrAns is the production of semantic information in the shape of logical forms for each sentence of each manual page, as well as the user query. These logical forms are designed so that they can be derived from *any* sentence (using robust approaches to treat very complex or ungrammatical sentences), and they are optimised for NLP tasks that involve the semantic comparison of sentences, such as AE.

ExtrAns' logical forms are called *minimal logical forms* (MLFs) because they encode the minimum information required for effective answer extraction. In particular, only the main dependencies between the verb and arguments are expressed, plus modifier and adjunct relations. Thus, complex quantification, tense and aspect, temporal relations, plurality, and modality are not expressed.

The MLFs use *reification* to achieve flat expressions, very much in the line of Davidson (1967), Hobbs (1985), and Copestake et al. (1997). In the current implementation only reification to objects, eventualities (events or states), and properties is applied. For example, the MLF of the sentence *cp will quickly copy files* is:

```
holds(e4),
object(cp,o1,[x1]),
object(s_command,o2,[x1]),
evt(s_copy,e4,[x1,x6]),
object(s_file,o3,[x6]),
prop(quickly,p3,[e4]).
```

In other words, there is an entity `x1` which represents an object of type `command`;[2] there is an entity `x6` (a file); there is an entity `e4`, which represents a copying event where the first argument is `x1` and the second argument is `x6`; there is an entity `p3` which states that `e4` is done quickly, and the event `e4`, that is, the copying, holds.

ExtrAns finds the answers to the questions by converting the MLFs of the questions into Prolog queries and then running Prolog's default resolution mechanism to find those MLFs that can prove the question.

This default search procedure is called the *synonym mode* since ExtrAns uses a small WordNet-style thesaurus (Fellbaum, 1998) to convert all the synonyms into a synonym representative. ExtrAns also has an *approximate mode* which, besides normalising all synonyms, scores all document sentences on the basis of the maximum number of predicates that unify between the MLFs of the query and the answer candidate (Mollá et al., 2000). If all query predicates can be matched then

[2]ExtrAns uses additional domain knowledge to infer that *cp* is a command.

the approximate mode returns exactly the same answers as the synonym mode.

4.2 The Comparison

Ideally, answer extraction systems should be evaluated according to how successful they are in helping users to complete their tasks. The use of the system will therefore depend on such factors as how many potential answers the user is presented with at a time, the way these potential answers are ranked, how many potential answers the user is prepared to read while searching for an actual answer, and so on. These issues, though important, are beyond the scope of the present evaluation. In this evaluation we focus solely on the relevance of the set of results returned by ExtrAns.

4.2.1 Method

Resources from a previous evaluation of ExtrAns (Mollá et al., 2000) were re-used for this evaluation. These resources were: a) a collection of 500 man pages, and b) a test set of 26 queries and relevant answers found in the 500 manual pages. The careful and labour-intensive construction of the test set gives us confidence that practically all relevant answers to each query are present in the test set. The queries themselves were selected according to the following criteria:

- There must be at least one answer in the manual page collection.
- The query asks how to perform a particular action, or how a particular command works.
- The query is simple, i.e. it asks only one question.

The manual pages were parsed using Conexor FDG and Link Grammar. The latter has a parameter for outputting either all parses found, or just the best parse found, and both parameter settings were used. The queries were then parsed by both parsers and their logical forms were used to search the respective databases. The experiment was repeated using both the synonym and approximate search modes.

Parser	*Precision*[4]	*Recall*	*F-score*
Conexor FDG	55.8%	8.9%	0.074
LG–best	49.7%	11.4%	0.099
LG–all	50.9%	13.1%	0.120

Table 4: Averages per query in synonym mode.

Parser	*Precision*[4]	*Recall*	*F-score*
Conexor FDG	28.3%	21.9%	0.177
LG–best	31.8%	15.8%	0.150
LG–all	40.5%	20.5%	0.183

Table 5: Averages per query in approximate mode.

4.2.2 Results

Precision, Recall and the F-score (with Precision and Recall equally weighted) for each query were calculated.[3] When no results were returned for a query the precision could not be calculated, but the F-score is equal to zero. The results are shown in Tables 4 and 5. The number of times the results for a query contained no relevant answers are shown in Table 6.

The tables show that the approximate mode gives better results than the synonym mode. This is to be expected, since the synonym mode returns exact matches only and therefore some questions may not produce any results. For those questions, recall and F would be zero. In fact, the number of questions without answers in the synonym mode is so large that the comparison between Conexor FDG and Link Grammar becomes unreliable in this mode. In this discussion, therefore, we will focus on the approximate mode.

The results returned by Link Grammar when all parses are considered are significantly better than when only the first (i.e. the best) parse is consid-

[3] F was calculated using the expression

$$F = 2 \times \frac{|\text{returned and relevant}|}{|\text{returned}| + |\text{relevant}|}$$

which is equivalent to the usual formulation (with $\beta = 1$):

$$F = (\beta^2 + 1) \times \frac{\text{Precision} \times \text{Recall}}{\beta^2\text{Precision} + \text{Recall}}$$

[4] Average over queries for which precision is defined, i.e. when the number of returns is non-zero.

Parser	*Search mode*	*No results returned*	*Nothing relevant returned*
Con. FDG	Synonym	20	20
Con. FDG	Approximate	0	8
LG–best	Synonym	16	18
LG–best	Approximate	1	11
LG–all	Synonym	15	18
LG–all	Approximate	4	12

Table 6: Numbers of times no relevant answers were found.

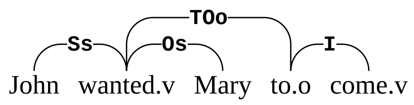
ered. This shows that, in the answer extraction task, it is better to use the logical forms of all possible sentence interpretations. Recall increases and, remarkably, precision increases as well. This means that the system is more likely to include new relevant answers when all parses are considered.

In many applications it is more practical to consider one parse only. Conexor FDG, for example, returns one parse only, and the parsing speed comparison (Section 3.2) shows an important difference in parsing time. If we compare Conexor FDG with Link Grammar set to return just the best parse — since Conexor FDG returns one parse only, this is the fairest comparison — we can see that recall of the system using Conexor FDG is higher than that of the system using Link Grammar, while retaining similar precision.

5 Discussion

The fairest extrinsic comparison between Conexor FDG and Link Grammar is the one that uses the best parse returned by Link Grammar, and the answer extraction method follows the approximate mode. With these settings, Conexor FDG produces better results than Link Grammar. However, the results of the extrinsic comparison are far less dramatic than those of the intrinsic comparison, specially in the precision figures.

One reason for the difference in the results is that the intrinsic evaluation compares grammatical relation accuracy, whereas the answer extraction system used in the extrinsic evaluation uses logical forms. A preliminary inspection of the grammatical relations and logical forms of questions and correct answers shows that high overlap of grammatical relations does not translate into high overlap of logical forms. A reason for this difference is that the semantic interpreters used in the extrinsic evaluation explore exhaustively the dependency structures returned by both parsing systems and they try to recover as much information as possible. In contrast with this, the generators of grammatical relations used in the intrinsic evaluation provide the most direct mapping from dependency structures to grammatical relations. For example, typically a dependency structure would not show a long dependency like the subject of *come* in the sentence *John wanted Mary to come*:



As a result, the grammatical relations would not show the subject of *come*. However, the subject of *come* can be traced by following several dependencies (`I`, `TOo` and `Os` above) and ExtrAns' semantic interpreters *do* follow these dependencies. In other words, the semantic interpreters use more information than what is directly encoded in the dependency structures. Therefore, the logical forms contain richer information than the grammatical relations. We decided not to optimise the grammatical relations used in our evaluation because we wanted to test the expressivity of the inherent grammars. It would be questionable whether we should recover more information than what is directly expressed. After all, provided that the parse contains all the words in the original order, we can theoretically ignore the sentence structure and still recover *all* the information.

6 Summary and Further Work

We have performed intrinsic evaluations of parsers and extrinsic evaluations within the context of answer extraction. These evaluations strengthen Galliers and Sparck Jones (1993)'s claim that intrinsic evaluations are of very limited value. In particular, our evaluations show that intrinsic evaluations may provide results that are distorted with respect to the most intuitive purpose of a parsing system: to deliver syntactic structures to subsequent modules of practical NLP systems. There is a clear need for frameworks for extrinsic evaluations of parsers for different NLP applications.

Further research to confirm this conclusion will be to try and minimise the occurrence of variables in the experiments by using the same corpus for both the intrinsic and the extrinsic evaluations and/or by using an answer extraction system that operates on the level of grammatical relations instead of MLFs. Additional further research will be the use of other intrinsic evaluation methodologies and extrinsic evaluations within the context of various other embedding setups.

Acknowledgement

This research is supported by the Macquarie University New Staff grant MUNS–9601/0069.

References

Srinivas Bangalore, Anoop Sarkar, Christine Doran, and Beth Ann Hockey. 1998. Grammar & parser evaluation in the XTAG project. In *Proc. Workshop on the Evaluation of Parsing Systems, LREC98*.

Ezra Black, S.P. Abney, D. Flickinger, C. Gdaniec, R. Grisham, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M.P. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proc. DARPA Speech and Natural Language Workshop*, pages 306–311, Pacific Grove, CA. Morgan Kaufmann.

Ezra Black. 1996. Evaluation of broad-coverage natural-language parsers. In Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, and Victor Zue, editors, *Survey of the State of the Art in Human Language Technology*, pages 488–490. CSLU, Oregon Graduate Institute.

John Carroll, Ted Briscoe, and Antonio Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proc. LREC98*.

John Carroll, G. Minnen, and T. Briscoe. 1999. Corpus annotation for parser evaluation.

Michael John Collins. 1996. A new statistical parser based on bigram lexical dependencies. In *Proc. ACL*. Santa Cruz.

Ann Copestake, Dan Flickinger, and Ivan A. Sag. 1997. Minimal recursion semantics: an introduction. Technical report, CSLI, Stanford University, Stanford, CA.

Donald Davidson. 1967. The logical form of action sentences. In Nicholas Rescher, editor, *The Logic of Decision and Action*, pages 81–120. Univ. of Pittsburgh Press.

Christiane Fellbaum. 1998. Wordnet: Introduction. In Christiane Fellbaum, editor, *WordNet: an electronic lexical database*, Language, Speech, and Communication, pages 1–19. MIT Press, Cambrige, MA.

Julia R. Galliers and Karen Sparck Jones. 1993. Evaluating natural language processing systems. Technical Report TR-291, Computer Laboratory, University of Cambridge.

Jerry R. Hobbs. 1985. Ontological promiscuity. In *Proc. ACL'85*, pages 61–69. University of Chicago, Association for Computational Linguistics.

Timo Järvinen and Pasi Tapanainen. 1997. A dependency parser for english. Technical Report TR-1, Department of Linguistics, University of Helsinki, Helsinki.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proc. IJCAI-95*, pages 1420–1425, Montreal, Canada.

Diego Mollá, Rolf Schwitter, Michael Hess, and Rachel Fournier. 2000. Extrans, an answer extraction system. *T.A.L.*, 41(2):495–522.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Proc. Third International Workshop on Parsing Technologies*, pages 277–292.

Richard F. E. Sutcliffe, Heinz-Detlev Koch, and Annette McElligott, editors. 1996. *Industrial Parsing of Software Manuals*. Rodopi, Amsterdam.

Pasi Tapanainen and Timo Järvinen. 1997. A non-projective dependency parser. In *Procs. ANLP-97*. ACL.

Ellen M. Voorhees. 2001a. Overview of the TREC 2001 question answering track. In Ellen M. Voorhees and Donna K. Harman, editors, *Proc. TREC-10*, number 500-250 in NIST Special Publication. NIST.

Ellen M. Voorhees. 2001b. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378.