A low-complexity, broad-coverage probabilistic Dependency Parser for English

Schneider, G

Abstract: Large-scale parsing is still a complex and time-consuming process, often so much that it is in-feasible in real-world applications. The parsing system described here addresses this problem by combining finite-state approaches, statistical parsing techniques and engineering knowl- edge, thus keeping parsing complexity as low as possible at the cost of a slight decrease in performance. The parser is robust and fast and at the same time based on strong linguistic foundations.

A low-complexity, broad-coverage probabilistic Dependency Parser for English

Gerold Schneider
Institute of Computational Linguistics, University of Zurich
Department of Linguistics, University of Geneva
gerold.schneider@lettres.unige.ch

Abstract

Large-scale parsing is still a complex and time-consuming process, often so much that it is infeasible in real-world applications. The parsing system described here addresses this problem by combining finite-state approaches, statistical parsing techniques and engineering knowledge, thus keeping parsing complexity as low as possible at the cost of a slight decrease in performance. The parser is robust and fast and at the same time based on strong linguistic foundations.

1 Introduction

Many extensions to text-based, data-intensive knowledge management approaches, such as Information Retrieval or Data Mining, focus on integrating the impressive recent advances in language technology. For this, they need fast, robust parsers that deliver linguistic data which is meaningful for the subsequent processing stages. This paper presents such a parsing system. Its output is a hierarchical structure of syntactic relations, functional dependency structures, which are discussed in section 2.

The parser differs on the one hand from successful Dependency Grammar implementations (e.g. (Lin, 1998), (Tapanainen and Järvinen, 1997)) by using a statistical base, and on the other hand from state-of-the-art statistical approaches (e.g. (Collins, 1999)) by carefully following an established formal grammar theory, Dependency Grammar (DG). It combines two probabilistic models of language, similar to (Collins, 1999), which are discussed in section 3. Both are supervised and based on Maximum Likelihood Estimation (MLE). The first one is based on the lexical probabilities of the heads of phrases, similar to (Collins and Brooks, 1995). It calculates the probability of finding specific syntactic relations (such as subject, sentential object, etc.) between given lexical heads. Two simple extensions for the interaction between several dependents of the same mother node are also used. The second probability model is a PCFG for the production of the VP. Although traditional CFGs are not part of DG, VP PCFG rules can model verb subcategorization frames, an important DG component.

The parser has been trained, developed and tested on a large collection of syntactically analyzed sentences, the Penn Treebank (Marcus et al., 1993). It is broad-coverage and robust and returns an optimal set of partial structures when it fails to find a complete structure for a sentence. It has been designed to keep complexity as low as possible during the parsing process in order to be fast enough to be useful for parsing large amounts of unrestricted text. This has been achieved by observing the following constraints, discussed in section 4:

- using a syntactic theory known for its relatively flat structures and lack of empty nodes (see also subsection 2.4)
- relying on finite-state preprocessing
- discarding unlikely readings with a beam search
- using the fast Cocke-Younger-Kasami (CYK) parsing algorithm
- using a restrictive hand-written linguistic grammar

The parsing system uses a divide-and-conquer approach. Low-level linguistic tasks that can be reliably solved by finite-state techniques are handed over to them. These low-level tasks are the recognition of part-of-speech by means of tagging, and the recognition of base NPs and verbal groups by means of chunking. The parser then relies on the disambiguation decisions of the tagging and chunking stage and can profit from a reduced search space, at the cost of a slightly decreased performance due to tagging and chunking errors.

The paper ends with a preliminary evaluation of this work in progress.

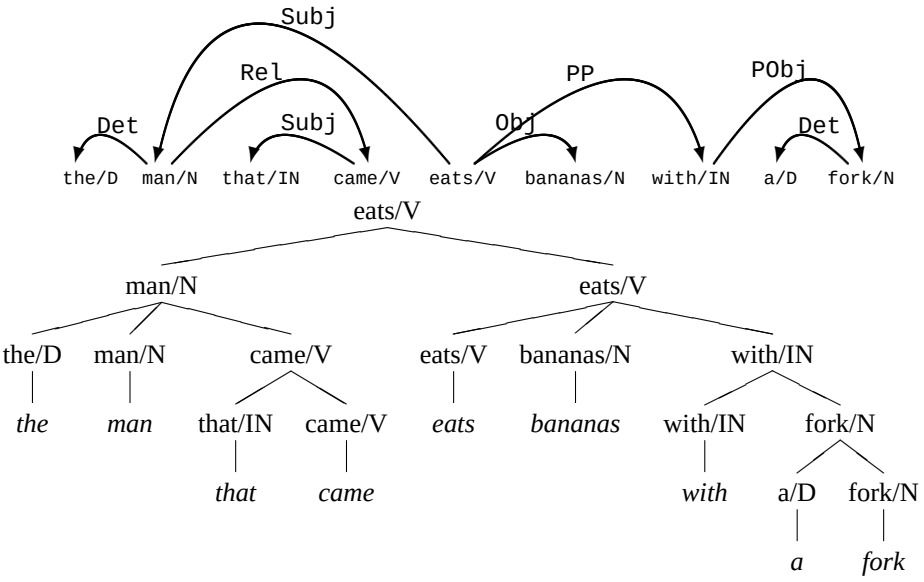

Figure 1: A dependency representation and its typically unlabeled constituency counterpart

2 Dependency Grammar

This system quite strictly follows DG assumptions. Dependency Grammar (DG) is essentially a valency grammar in which the valency concept is extended from verbs to nouns and adjectives and finally to all word classes.

2.1 Relation to Constituency

In its simplest definition, a projective DG is a binary version (except for valency, see 2.2) of a constituent grammar which only knows lexical items, which entails that

- for every mother node, the mother node and exactly one of its daughters, the so-called head, are isomorphic
- projection is deterministic, endocentric and can thus not fail, which gives DG a robustness advantage
- equivalent constituency CFG trees can be derived
- it is in Chomsky Normal Form (CNF), the efficient CYK parsing algorithm can thus be used

Any DG has an equivalent constituency counterpart (Covington, 1994). Figure 1 shows a dependency structure and its unlabeled constituency counterpart.

2.2 Valency as an isomorphism constraint

Total equivalence between mother and head daughter could not prevent a verb from taking an infinite number of subjects or objects. Therefore, valency theory is as vital a part of DG as is its constituency counterpart, subcategorization. The manually written rules check the most obvious valency constraints. Verbal valency is modeled by a PCFG for VP production.

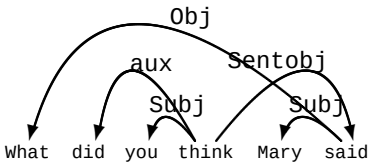

Figure 2: Non-projective analysis of a WH-question

2.3 Functionalism

DG was originally conceived to be a deep-syntactic, proto-semantic theory (Tesnière, 1959). The version of DG used here retains syntactic functions as dependency labels, like in LFG, which means that the dependency analyses returned by the parser are also a simple version of LFG f-structures, a hierarchy of syntactic relations between lexical heads which serves as a bridgehead to semantics. Functional DG only accepts content words as heads. This has the advantage that no empty heads (for example empty complementizers for zero-relatives) are needed. It also means that its syntactical structures are closer to argument-structure representation than traditional constituency-based structures such as those of GB or the Treebank. The closeness to argument-structure makes them especially useful for subsequent stages of knowledge management processing.

A restricted use of Tesnière-style translations is also made. Adjectives outside a noun chunk may function as a nominal constituent (*the poor/JJ are the saint/JJ*). Participles may function as adjectives (*Western industrialized/VBN countries*). Present participles may also function as nouns (*after winning/VBG the race*).

Traditional constituency analyses such as those in the

Treebank contain many discontinuous constituents, also known as long-distance dependencies, expressed by the use of structure-copying methods. This parser deals with them by allowing non-projectivity in specific, well-defined situations, such as in WH-questions (Figure 2). But in order to keep complexity low, discontinuity is restricted to a minimum. Many long-distance dependencies are not strictly necessary. For example, the analysis of passive clauses does not need to involve discontinuity, in which a subordinate VP whose absent object is structure-shared with the subject of the superordinate VP. Because the verb form allows a clear identification of passive clauses, a surface analysis is sufficient, as long as an appropriate probability model is used. In this parser, passive subjects use their own probability model, which is completely distinct from active subjects.

2.4 Mapping the Treebank to Functional Dependency

A popular query tool for the extraction of tree structures from Treebanks, *tgrep*, has been used for the mapping to dependencies. The mapping from a *configurational* paradigm to a *functional* one turns out to be non-trivial (Musillo and Sima'an, 2002). A relatively simple example, the verb-object (obj) relation is discussed now.

In a first approximation, a verb–object relation holds between the head of a VP and the head of the NP immediately under the VP. In most cases, the VP head is the lowest verb and the NP head is the lowest rightmost noun.

As *tgrep* seriously overgenerates, a large number of highly specific subqueries had to be used, specifying all possible configurations of arbitrarily nested NPs and VPs. Since hundreds of possible configurations are thus mapped onto one dependency relation, statistical models based on them are much less sparse than lexicalized PCFGs, which is an advantage as lexicalized models often suffer from sparseness. In order to extract relations compatible to the parser's treatment of conjunction and apposition, the queries had to be further specified, thereby missing few structures that should match.

In order to restrict discontinuity to where it is strictly necessary, copular verb complements and small clause complements are also treated as objects. Since the function of such objects can be unambiguously derived from a verb's lexical entry this is a linguistically viable decision.

The mapping from the Penn treebank to dependencies by means of *tgrep* is a close approximation but not a complete mapping. A few structures corresponding to a certain dependency are almost certain to be missed or doubled. Also, structures involving genuine discontinuity like the verb–object relation in figure 2 are not extracted.

3 Probabilistic Models of Language

Writing grammar rules is an easy task for a linguist, particularly when using a framework that is close to traditional school grammar assumptions, such as DG. Acknowledged facts such as the one that a verb has typically one but never two subjects are expressed in hand-written declarative rules. The rules of this parser are based on the Treebank tags of heads of chunks. Since the tagset is limited and dependency rules are binary, even a broad-coverage set of rules can be written in relatively little time.

What is much more difficult, also for a linguist, is to assess the scope of application of a rule and the amount of ambiguity it creates. Long real-world sentences typically have dozens to hundreds of syntactically correct complete analyses and thousands of partial analyses, although most of them are semantically so odd that one would never think of them. Here, machine-learning approaches, such as probabilizing the manually written rules, are vital to any parser, for two reasons: first, the syntactically possible analyses can be ranked according to their probabilities. For subsequent processing stages like semantic interpretation or document classification it then often suffices to take the first ranked or the n first ranked readings. Second, in the course of the parsing process, very improbable analyses can be abandoned, which greatly improves parsing efficiency (see section 4).

The parser uses two linguistic probability models. The first one is based on the lexical probabilities of the heads of phrases. Two simple extensions for the interaction between several dependents of the same mother node are also used. The second probability model is a PCFG for the expansion of the VP.

Since the parser aims at a global disambiguation, all local probabilities are stored in the parsing chart. The global probability of a parse is the product of all its local probabilities, a product of disambiguation decisions.

3.1 Lexical Dependencies

Given two adjacent lexical heads (say a and b), the probabilities of the possible dependency relations between them are calculated as Maximum Likelihood (MLE) estimates. In a binary CFG, constituents which are adjacent at some stage in the parsing process are candidates for the right-hand side (RHS) of a rewrite rule. If a rule exists for these constituents (say A and B), then in a DG or in Bare Phrase Structure, one of these is isomorphic to the LHS, i.e. the head. DG rules additionally use a syntactic relation label R, for which the probabilities are calculated in this probability model. The dependency rules used are based on Treebank tags, the relation probabilities are con-

ditioned on them and on the lexical heads.

$$p(R|A \rightarrow AB, a, b) = \frac{\#(R, A \rightarrow AB, a, b)}{\#(A \rightarrow AB, a, b)} \quad (1)$$

All that A → AB expresses is that in the dependency relation the dependency is towards the right, it is therefore rewritten as $right$.

$$p(R|right, a, b) = \frac{\#(R, right, a, b)}{\#(right, a, b)} \quad (2)$$

Such a probability model is used to model the local competition between object and adjunct relation (*he left town* vs. *he left yesterday*), in which the verb is always the left RHS constituent. But in some cases, the direction is also a parameter, for example in the subject–verb relation (*she said* versus *said she*). There, the probability space is divided into two equal sections.

$$p(R, right|a, b) = \frac{1}{2} * \frac{\#(R, right, a, b)}{\#(right, a, b) + \#(left, a, b)} \quad (3)$$

The PP-attachment model probabilities are conditioned on three lexical heads – the verb, the preposition and the description noun (Collins and Brooks, 1995). The probability model is backed off across several levels. In addition to backing off to only partly lexicalized counts (ibid.), semantic classes are also used in all the modeled relations, for verbs the Levin classes (Levin, 1993), for nouns the top Wordnet class (Fellbaum, 1998) of the most frequent sense. As an alternative to backing-off, linear interpolation with the back-off models has also been tried, but the difference in performance is very small.

A large subset of syntactic relations, the ones which are considered to be most relevant for argument structure, are modeled, specifically:

Relation	Label	Example
verb–subject	subj	*he sleeps*
verb–direct object	obj	*sees it*
verb–indirect object	obj2	*gave (her) kisses*
verb–adjunct	adj	*ate yesterday*
verb–subord. clause	sentobj	*saw (they) came*
verb–prep. phrase	pobj	*slept in bed*
noun–prep. phrase	modpp	*draft of paper*
noun–participle	modpart	*report written*
verb–complementizer	compl	*to eat apples*
noun–preposition	prep	*to the house*

Until now one relation has two distinct probability models: verb–subject is different for active and passive verbs, henceforth referred to as $asubj$ and $psubj$, where needed. The disambiguation between complementizer and preposition is necessary as the Treebank tagset unfortunately uses the same tag (IN) for both. Many relations have slightly individualized models. As an example the $modpart$ relation will be discussed in detail.

3.1.1 An Example: Modification by Participle

The noun–participle relation is also known as reduced relative clause. In the Treebank, reduced relative clauses are adjoined to the NP they modify, and under certain conditions also have an explicit RRC label. Reduced relative clauses are frequent enough to warrant a probabilistic treatment, but considerably sparser than verb–non-passive-subject or verb–object relations. They are in direct competition with the subject–verb relation, because its candidates are also a NP followed by a VP. We probably have a subject-verb relation in *the report announced the deal* and a noun-participle relation in *the report announced yesterday*. The majority of modification by participle relations, if the participle is a past participle, functionally correspond to passive constructions (*the report written* ≅ *the report which has been written*). In order to reduce data sparseness, which could lead to giving preference to a verb–non-passive-subject reading ($asubj$), the verb–passive-subject counts ($psubj$) are added to the noun–participle counts. Some past participles also express adjunct readings (*the week ended Friday*); therefore the converse, i.e. adding noun–participle counts to verb–passive-subject counts, is not recommended.

The next back-off step maps the noun a to its Wordnet-class $\mathring{a}$ and the verb b to its Levin-class $\mathring{b}$. If the counts are still zero, counts on only the verb and then only the noun are used.

$$p(modpart|a, b) = \quad (4)$$

$$\begin{cases} \frac{\#(modpart,right,a,b)+\#(psubj,left,a,b)}{\#(modpart,right,a,b)+\#(psubj,left,a,b)+\#(asubj,left,a,b)} & if > 0, else \\ \frac{\#(modpart,right,\mathring{a},\mathring{b})+\#(psubj,left,\mathring{a},\mathring{b})}{\#(modpart,right,\mathring{a},\mathring{b})+\#(psubj,left,\mathring{a},\mathring{b})+\#(asubj,left,\mathring{a},\mathring{b})} & if > 0, else \\ \frac{\#(modpart,right,b)+\#(psubj,left,b)}{\#(modpart,right,b)+\#(psubj,left,b)+\#(asubj,left,b)} & if > 0, else \\ \frac{\#(modpart,right,a)+\#(psubj,left,a)}{\#(modpart,right,a)+\#(psubj,left,a)+\#(asubj,left,a)} \end{cases}$$

As the last backoff, a low non-zero probability is assigned. In the verb–adjunct relation, which drastically increases complexity but can only occur with a closed class of nouns (mostly adverbial expressions of time), this last backoff is not used.

3.2 Interaction between Several Dependents

For the verb–prepositional-phrase relation, two models that take the interaction between the several PPs of the same verb into account have been implemented. They are based on the verbal head and the prepositions.

The first one estimates the probability of attaching a PP introduced by preposition $p2$, given that the verb to which it could be attached already has another PP introduced by the preposition $p1$. Back-offs using the verb-class $\mathring{v}$ and

then the preposition(s) only are used.

$$p(p2|v,p1) = \begin{array}{l} \frac{\#(p2,v,p1)}{\#(v,p1)} \quad if > 0, else \\ \frac{\#(p2,\hat{v},p1)}{\#(\hat{v},p1)} \quad if > 0, else \\ \frac{\#(p2,\bigcup v,p1)}{\#(\bigcup v,p1)} \quad if > 0, else \\ \frac{\#(p2,\bigcup v)}{\#(\bigcup v)} \end{array} \tag{5}$$

The second model estimates the probability of attaching a PP introduced by preposition $p2$ as a non-first PP. The usual backoffs are not printed here.

$$p(p2|v,\bigcup p1) = \frac{\#(p2,v,\bigcup p1)}{\#(v,\bigcup p1)} \tag{6}$$

As prepositions are a closed class, a zero probability is assigned if the last back-offs fail.

3.3 PCFG for Verbal Subcategoriation and VP Production

Verbs often have several dependents. Ditransive verbs, for example, have up to three NP complements, the subject, the direct object and the indirect object. An indeterminate number of adjuncts can be added. Transitivity, expressed by a verb's subcategorization, is strongly lexicalized. But because the Treebank does not distinguish arguments and complements, and because a standard lexicon does not contain probabilistic subcategorization, a probabilistic model has advantages. Dependency models as discussed hitherto fail to model complex dependencies between the dependents of the same mother, unlike PCFGs. A simple PCFG model for the production of the VP rule which is lexicalized on the VP head and has a non-lexicalized backoff, is therefore used. RHS constituents C, for the time being, are unlexicalized phrasal categories like NP,PP, $Comma$, etc. At some stage in the parsing process, given an attachment candidate C_n and a verbal head v which already has attached constituents C_1 to C_{n-1}, the probability of attaching C_n is estimated. This probability can also be seen as the probability of continuing versus ending the VP under production.

$$p(attach|C_n,v,C_1..C_{n-1}) = \begin{array}{l} \frac{\#(vp \rightarrow v,C_1,...C_n)}{\#(vp \rightarrow v,C_1,...C_{n-1})} \quad if > 0, else \\ \frac{\#(vp \rightarrow \bigcup v,C_1,...C_n)}{\#(vp \rightarrow \bigcup v,C_1,...C_{n-1})} \end{array} \tag{7}$$

4 Implementation

The parser has been implemented in Prolog, it runs in SWI-Prolog and Sicstus Prolog. For SWI-Prolog, a graphical interface has also been programmed in XPCE[1].

[1]For more information, see
`http://www.ifi.unizh.ch/CL/gschneid/parser`

If no analysis spanning the entire length of the sentence can be found, an optimal path of partial structures spanning as much of the sentence as possible is searched. The algorithm devised for this accepts the first-ranked of the longest of all the partial analyses found, say S. Then, it recursively searches for the first-ranked of the longest of the partial analyses found to the left and to the right of S, and so on, until all or most of the sentence is spanned.

The parser uses the preprocessed input of a finite-state tagger-chunker. Finite-state technology is fast enough for unlimited amounts of data, taggers and chunkers are known to be reliable but not error-free, with typical error rates between 2 and 5 %. Tagging and chunking is done by a standard tagger and chunker, *LTPos* (Mikheev, 1997). Heads are extracted from the chunks and lemmatized (Minnen et al., 2000). Parsing takes place only between the heads of phrases, and only using the best tag suggested by the tagger, which leads to a reduction in complexity. The parser uses the CYK algorithm, which has parsing complexity of $O(n^3)$, where n is the number of words in a word-based, but only chunks in a head-of-chunk-based model. The chunk to word relation is 1.52 for Treebank section 0. In a test with a toy NP and verb-group grammar parsing was about 4 times slower when using unchunked input. Due to the insufficiency of the toy grammar the lingusitic quality and the number of complete parses decreased. The average number of tags per token is 2.11 for the entire Treebank. With untagged input, every possible tag would have to be taken into consideration. Although untested, at least a similar slowdown as for unchunked input can be expected.

In a hand-written grammar, some typical parsing errors can be corrected by the grammar engineer, or rules can explicitly ignore particularly error-prone distinctions. Examples of rules that can correct tagging errors without introducing many new errors are allowing VBD to act as a participle or the possible translation of VBG to an adjective. As an example of ignoring error-prone distinctions, the disambiguation between prepositions and verbal particles is unreliable. The grammar therefore makes no distinction and treats all verbal particles as prepositions, which leads to an incorrect but consistent analysis for phrasal verbs. A hand-written grammar allows to model complex but important phenomena which overstep manageable ML search spaces, such as discontinous analysis of questions can be expressed, while on the other hand rare and marginal rules can be left out to free resources. For tagging, (Samuelsson and Voutilainen, 1997) have shown that a manually built tagger can equal a statistical tagger.

5 Preliminary Evaluation

The probabilistic language models have been trained on section 2 to 24 and the parser tested on section 0. The

	Percentage Values for			
	Subject	Object	PP-attach	IN
Precision	77	72	67	80
Recall	70	75	49	78

Table 1: Provisional precision and recall values

held out training data and the first-ranked reading for each sentence of section 0 are compared for evaluation (Lin, 1995). Parsing the 46527 words of section 0 takes 30 minutes on a 800 MHz Pentium 3 PC, including about 3 minutes for tagging and chunking. Current precision and recall values for subject, object and PP-attachment relations, and for the disambiguation between prepositions and complements are in table 1.

These results, slightly lower than state-of-the-art ((Lin, 1998), (Preiss, 2003)), are least merit figures or a proof of concept rather than accurate figures. On the one hand, the performance of the parser suffers from mistaggings and mischunkings or a limited grammar, the price for the speed increase. On the other hand, different grammatical assumptions both between the Treebank and the chunker, and between the Treebank and functional dependency, seriously affect the evaluation. For example, the chunker often recognizes units longer than base-NPs like *[many of the people]*, or smaller or longer than verbal groups *[has] for a long time [been]*, *[likely to bring]* – correct chunks which are currently considered as errors.

In addition, it is very difficult to avoid *tgrep* overgenerating or missing. It turns out that the mapping is accurate enough for a statistical model but not for a reliable evaluation. Some possible configurations are missed by the current extraction queries. For example, extraposed PPs such as the one starting this sentence, have escaped unmapped until now. For the future, the use of a standardized DG test suite is envisaged (Carroll et al., 1999).

The grammar explicitly excludes a number of grammatical phenomena which cannot currently be treated reliably. For example, since no PP-interaction model such as PCFG rules for NP-attached PPs exists yet, the current grammar does not allow a NP to take several PPs, which affects the analysis of relational nouns. The statistical models, the dependency extraction, the grammar, the tagger and chunker approach and the evaluation method will continue to be improved.

References

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL-99 Post-Conference Workshop on Linguistically Interpreted Corpora*, Bergen, Norway.

Michael Collins and James Brooks. 1995. Prepositional attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, MA.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.d. dissertation, University of Pennsylvania, Philadelphia, PA.

Michael A. Covington. 1994. An empirically motivated reinterpretation of Dependency Grammar. Technical Report AI1994-01, University of Georgia, Athens, Georgia.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

Beth C. Levin. 1993. *English Verb Classes and Alternations: a Preliminary Investigation*. University of Chicago Press, Chicago, IL.

Dekang Lin. 1995. A dependency-based method for evaluating broad-coverage parsers. In *Proceedings of IJCAI-95*, Montreal.

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*, Granada, Spain.

Mitch Marcus, Beatrice Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Andrei Mikheev. 1997. Automatic rule induction for unknown word guessing. *Computational Linguistics*, 23(3):405–423.

Guido Minnen, John Carroll, and Darren Pearce. 2000. Applied morphological generation. In *Proceedings of the 1st International Natural Language Generation Conference (INLG)*, Mitzpe Ramon, Israel.

Gabriele Musillo and Khalil Sima'an. 2002. Towards comparing parsers from different linguistic frameworks. In *Proceedings of LREC 2002 Beyond PARSEVAL Workshop*, Las Palmas, Spain.

Judita Preiss. 2003. Using grammatical relations to compare parsers. In *Proceedings of EACL 03*, Budapest, Hungary.

Christer Samuelsson and Atro Voutilainen. 1997. Comparing a linguistic and a stochastic tagger. In *Proceedings of of ACL/EACL Joint Conference*, Madrid.

Pasi Tapanainen and Timo Järvinen. 1997. A nonprojective dependency parser. In *Proceedings of the 5th Conference on Applied Natural Language Processing*, pages 64–71. Association for Computational Linguistics.

Lucien Tesnière. 1959. *Eléments de Syntaxe Structurale*. Librairie Klincksieck, Paris.