Generating Typed Dependency Parses from Phrase Structure Parses

Marie-Catherine de Marneffe,[†*] **Bill MacCartney,**[*] and **Christopher D. Manning**[*]

[†] Department of Computing Science, Université catholique de Louvain
B-1340 Louvain-la-Neuve, Belgium
[*] Computer Science Department, Stanford University
Stanford, CA 94305, USA
{mcdm,wcmac,manning}@stanford.edu

Abstract

This paper describes a system for extracting typed dependency parses of English sentences from phrase structure parses. In order to capture inherent relations occurring in corpus texts that can be critical in real-world applications, many NP relations are included in the set of grammatical relations used. We provide a comparison of our system with Minipar and the Link parser. The typed dependency extraction facility described here is integrated in the Stanford Parser, available for download.

1. Introduction

We describe a system for automatically extracting typed dependency parses of English sentences from phrase structure parses. Typed dependencies and phrase structures are different ways of representing the structure of sentences: while a phrase structure parse represents nesting of multi-word constituents, a dependency parse represents dependencies between individual words. A *typed* dependency parse additionally labels dependencies with grammatical relations, such as *subject* or *indirect object*. There has been much linguistic discussion of the two formalisms. There are formal isomorphisms between certain structures, such as between dependency grammars and one bar-level, headed phrase structure grammars (Miller, 2000). In more complex theories there is significant debate: dominant Chomskyan theories (Chomsky, 1981) have defined grammatical relations as configurations at phrase structure, while other theories such as Lexical-Functional Grammar has rejected the adequacy of such an approach (Bresnan, 2001). Our goals here are more practical, though in essence we are following an approach where structural configurations are used to define grammatical roles.

Recent years have seen the introduction of a number of treebank-trained statistical parsers [Collins (Collins, 1999), Charniak (Charniak, 2000), Stanford (Klein and Manning, 2003)] capable of generating parses with high accuracy. The original treebanks, in particular the Penn Treebank, were for English, and provided only phrase structure trees, and hence this is the native output format of these parsers. At the same time, there has been increasing interest in using dependency parses for a range of NLP tasks, from machine translation to question answering. Such applications benefit particularly from having access to dependencies between words typed with grammatical relations, since these provide information about predicate-argument structure which are not readily available from phrase structure parses. Perhaps partly as a consequence of this, several more recent treebanks have adopted dependency representation as their primary annotation format, even if a conversion to a phrase structure tree form is also provided (e.g., the Dutch Alpino corpus (van der Beek et al., 2002) and the Danish Dependency Treebank (Kromann, 2003)). However, existing dependency parsers for English such as Minipar (Lin, 1998) and the Link Parser (Sleator and Temperley, 1993) are not as robust and accurate as phrase-structure parsers trained on very large corpora. The present work remedies this resource gap by facilitating the rapid extraction of grammatical relations from phrase structure parses. The extraction uses rules defined on the phrase structure parses.

2. Grammatical relations

This section presents the grammatical relations output by our system.

The selection of grammatical relations to include in our schema was motivated by practical rather than theoretical concerns. We used as a starting point the set of grammatical relations defined in (Carroll et al., 1999) and (King et al., 2003). The grammatical relations are arranged in a hierarchy, rooted with the most generic relation, *dependent*. When the relation between a head and its dependent can be identified more precisely, relations further down in the hierarchy can be used. For example, the *dependent* relation can be specialized to *aux* (auxiliary), *arg* (argument), or *mod* (modifier). The *arg* relation is further divided into the *subj* (subject) relation and the *comp* (complement) relation, and so on. The whole hierarchy of our grammatical relations is given in Figure 2.

Altogether, the hierarchy contains 48 grammatical relations. While the backbone of the hierarchy is quite similar to that in (Carroll et al., 1999), over time we have introduced a number of extensions and refinements to facilitate use in applications. Many NP-internal relations play a very minor role in theoretically motivated frameworks, but are an inherent part of corpus texts and can be critical in real-world applications. Therefore, besides the commonest grammatical relations for NPs (*amod* - adjective modifier, *rcmod* - relative clause modifier, *det* - determiner, *partmod* - participial modifier, *infmod* - infinitival modifier, *prep* - prepositional modifier), our hierarchy includes the following grammatical relations: *appos* (appositive modifier), *nn* (noun compound), *num* (numeric modifier), *number* (element of compound number) and *abbrev* (abbreviation). The example sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican

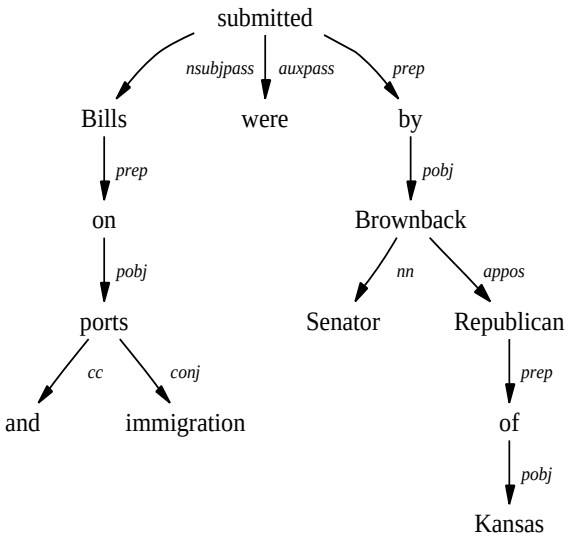

Figure 1: An example of a typed dependency parse for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas."

of Kansas" in Figure 1 illustrates the *appos* relation between "Brownback" and "Republican" and the *nn* relation between "Brownback" and "Senator". The *num* relation qualifies a number that serves to modify the meaning of a NP: *num*(sheep, 3) in "Sam ate 3 sheep", whereas the *number* relation captures the internal structure of multi-word numbers like *number*(5, million) in "I lost 5 million dollars". The *abbrev* relation indicates that MIT is the abbreviation for "Massachusetts Institute of Technology" in the following sentence: "The Massachusetts Institute of Technology (MIT) is located in Boston". Such information can be useful in the context of a textual inference application, as explained below.

3. Extraction method

Our technique for producing typed dependencies is essentially based on rules – or patterns – applied on phrase structure trees. The method is general, but requires appropriate rules for each language and treebank representation. Here we present details only for Penn Treebank English, but we have also developed a similar process for Penn Treebank Chinese. The method for generating typed dependencies has two phases: *dependency extraction* and *dependency typing*. The dependency extraction phase is quite simple. First, a sentence is parsed with a phrase structure grammar parser. Any Penn Treebank parser could be used for the process described here, but in practice we are using the Stanford parser (Klein and Manning, 2003), a high-accuracy statistical phrase structure parser trained on the Penn *Wall Street Journal* Treebank. The head of each constituent of the sentence is then identified, using rules akin to the Collins head rules, but modified to retrieve the semantic head of the constituent rather than the syntactic head. While heads chosen for phrase structure parsing do not really matter, retrieving sensible heads is crucial for extracting semantically appropriate dependencies. For example, in relative clauses, the Collins rule will choose as head the

dep - dependent
- aux - auxiliary
  - auxpass - passive auxiliary
  - cop - copula
- conj - conjunct
- cc - coordination
- arg - argument
  - subj - subject
    - nsubj - nominal subject
      - nsubjpass - passive nominal subject
    - csubj - clausal subject
  - comp - complement
    - obj - object
      - dobj - direct object
      - iobj - indirect object
      - pobj - object of preposition
    - attr - attributive
    - ccomp - clausal complement with internal subject
    - xcomp - clausal complement with external subject
    - compl - complementizer
    - mark - marker (word introducing an advcl)
    - rel - relative (word introducing a rcmod)
    - acomp - adjectival complement
  - agent - agent
- ref - referent
- expl - expletive (expletive *there*)
- mod - modifier
  - advcl - adverbial clause modifier
  - purpcl - purpose clause modifier
  - tmod - temporal modifier
  - rcmod - relative clause modifier
  - amod - adjectival modifier
  - infmod - infinitival modifier
  - partmod - participial modifier
  - num - numeric modifier
  - number - element of compound number
  - appos - appositional modifier
  - nn - noun compound modifier
  - abbrev - abbreviation modifier
  - advmod - adverbial modifier
    - neg - negation modifier
  - poss - possession modifier
  - possessive - possessive modifier (*'s*)
  - prt - phrasal verb particle
  - det - determiner
  - prep - prepositional modifier
- sdep - semantic dependent
  - xsubj - controlling subject

Figure 2: The grammatical relation hierarchy.

pronoun introducing the relative clause. As all the other words in the relative clause will depend on the head, it makes more sense to choose the verb as head when determining dependencies. In general, we prefer content words as heads, and have auxiliaries, complementizers, etc. be dependents of them. Another example concerns NPs with ambiguous structure or multiple heads which are annotated

with a flat structure in the Penn Treebank:

(NP the new phone book and tour guide)

Using the Collins rule, the head for this example is the word "guide", and all the words in the NP depend on it. In order to find semantically relevant dependencies, we need to identify two heads, "book" and "guide". We will then get the right dependencies (the noun "book" still has primacy as a governing verb will link to it, but this seems reasonable):

nn(book, phone)
nn(guide, tour)
CC_and(book, guide)
amod(book, new)
det(book, the)

It is essential in such cases to determine heads that will enable us to find the correct dependencies.

In the second phase, we label each of the dependencies extracted with a grammatical relation which is as specific as possible. For each grammatical relation, we define one or more patterns over the phrase structure parse tree (using the tree-expression syntax defined by `tregex` (Levy and Andrew, 2006)). Conceptually, each pattern is matched against every tree node, and the matching pattern with the most specific grammatical relation is taken as the type of the dependency (in practice, some optimizations are used to prune the search).

Up until this point, if one assumes an extra "root" for the sentence, then each word token is the dependent of one thing, and the number of typed dependencies in the representation is the same as the number of words in the sentence. The dependency graph is a tree (a singly rooted directed acyclic graph with no re-entrancies). However, for some applications, it can be useful to regard some words, such as prepositions and conjunctions, as themselves expressing a grammatical relation. This is achieved by collapsing a pair of typed dependencies into a single typed dependency, which is then labeled with a name based on the word between the two dependencies (the word itself being excised from the dependency graph). This facility is provided by our system, primarily targeted at prepositions, conjunctions, and possessive clitics. As already mentioned, Figure 1 shows the typed dependency parse obtained for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas." Figure 5 gives the typed dependency parse for the same sentence after the "collapsing" process, where the dependencies related to the prepositions "on" and "of" have been collapsed, as well as the conjunct dependencies for "ports and immigration". Our system optionally provides another layer of processing of conjunct dependencies which aims to produce a representation closer to the semantics of the sentence. In our example, this processing will add a *PREP_on* dependency between "Bills" and "immigration" as shown in Figure 6. An additional example of dependency structure modification is in a relative clause such as "I saw the man who loves you", the dependencies *ref*(man, who) and *nsubj*(loves, who) will be extracted, as shown in Figure 3. However it might be more useful to get *nsubj*(loves, man)

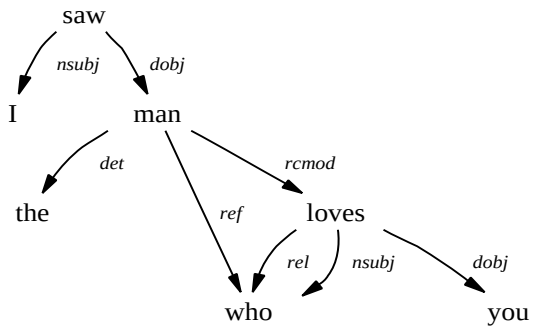

Figure 3: An example of a typed dependency parse for the sentence "I saw the man who loves you".

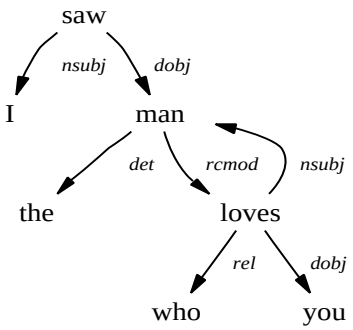

Figure 4: An example of a typed dependency parse for the sentence "I saw the man who loves you", with "collapsing" turned on.

where the relative pronoun is replaced by its actual referent. In such case the output will be the one in Figure 4. Note that as a result of this structure modification, a dependency graph may actually become cyclic, as shown in Figure 4. The usefulness of such structures depends on downstream software being able to correctly handle cyclic directed graphs.

4. Comparison

Direct comparison between our system and other dependency parsers like Minipar and the Link Parser is compli-

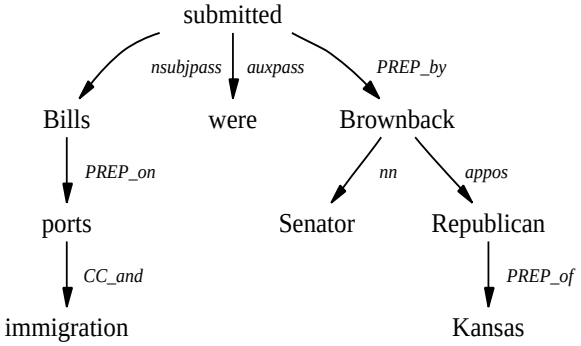

Figure 5: A dependency parse for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas", with "collapsing" turned on.

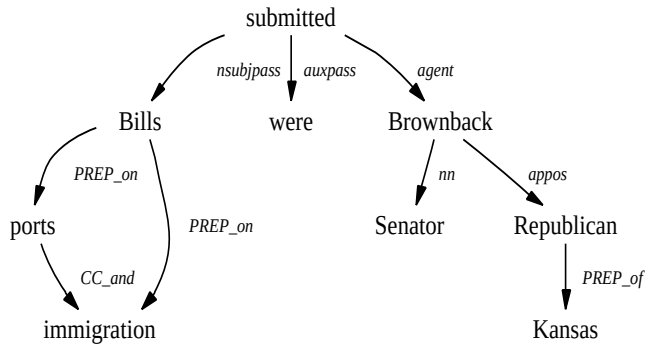

Figure 6: A dependency parse for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas", with "collapsing" turned on and processing of the conjunct dependencies.

cated by differences between the annotation schemes targeted by each system, presumably reflecting variation in theoretical and practical motivations. The differences fall into two main categories: dependency structure (which pairs of words are in a dependency relation) and dependency typing (what the grammatical relation for a particular dependency is).

First, the systems do not always agree about which words should be counted as the dependents of a particular governor. For example, the Link Parser has a dependency type *C* which is described as follows: "C links conjunctions to *subjects* of subordinate clauses ("He left WHEN HE saw me"). It also links certain verbs to *subjects* of embedded clauses ("He SAID HE was sorry")."[1] This leads the Link Parser to link "that" with "irregularities" and "said" with "investigation" in sentence 1 of table 2. In contrast, our system links subordinating conjunctions with the *verb* of the clause and main verbs to the *verb* of an embedded clause: in sentence 1, "that" is linked with "took_place" (*compl*(took_place, that)) and "said" with "produced" (*ccomp*(said, produced)). Another example regards the word "below" in sentence 6: the Link parser connects it with "he", whereas our system links it with "see" (*advmod*(see, Below)).

Moreover, there are differences among the systems with regard to the "collapsing" of prepositions and coordination; as discussed above in section 3, we have tried to handle these in a way that facilitates semantic analysis.

Even where the systems agree about whether two words are in a dependency relation, they may diverge about the type of the dependency. Each system assigns dependency types from a different set of grammatical relations, and it is not straightforward to establish mappings between these sets. Of course, the names used for relations vary considerably, and the distinctions between different relations may vary as well. But the most salient difference between the schemes is the level of granularity. As indicated in table 1, the set of relations defined by Carroll is comparatively coarse-grained. Carroll's scheme makes a distinction between verb or noun arguments, but doesn't further distinguish among these. A mapping of our grammatical relations into Carroll's scheme in order to evaluate our system using Carroll's Greval test suite[2] would not reflect the finer distinctions we make. But often these finer distinctions drive success in applications. For example, our PASCAL Recognizing Textual Entailment (see Section 5) derives considerable value from relations such as *appos* and *abbrev*.

In contrast, the Link Parser uses a very fine-grained set of relations, which often makes distinctions of a structural rather than a semantic nature, as for example the *MX* relation which "connects modifying phrases with commas to preceding nouns ("The DOG, a POODLE, was black"; "JOHN, IN a black suit, looked great")." The Link Parser has specific relations for idiomatic expressions. It also has three different relations for an adverb modifying another adverb, or an adjective, or a comparative adjective. The Link Parser uses a different set of dependency types for dependencies appearing in questions and relative clauses. We suggest that many of these distinctions are too fine to be of practical value, and in our system we have aimed for an intermediate level of granularity, motivated by the needs of practical applications.

Such differences make it difficult to directly compare the quality of the three systems. Lin (Lin, 1998) proposes two ways to evaluate the correctness of a dependency parse against a gold standard. In the first method, one simply examines whether each output dependency also occurs in the gold standard, while ignoring the grammatical type of the dependency; this method is therefore sensitive only to the structure of the dependency tree. The second method also considers whether the type of each output dependency matches the gold standard. But because the correctness of a dependency parser must be evaluated according to the annotation scheme it targets, and because each parser targets a different scheme, quantitative comparison is difficult.

However, a qualitative comparison may be of value. Figures 6, 7, and 8, show a comparison of the outputs of the Stanford parser, MiniPar and the Link Parser respectively on the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas". We chose this sentence as an illustrative example because it is short but shows typical structures like prepositional phrases, coordination, and noun componding. The graph representing Minipar output collapses directed paths through preposition nodes. It also adds antecedent links to 'clone' nodes between brackets. The graph for the Link Parser presents the same collapsing of directed paths through preposition nodes.

To provide a qualitative comparison, we parsed, with the three parsers, ten sentences randomly chosen from the Brown Corpus. The sentences we examined are given in table 2. Globally, the Stanford parser and the Link parser lead to more accurate structure trees than Minipar. However all parsers are misled by sentence 10 where "ride" is analyzed as a noun.

The Stanford parser trained on the Penn *Wall Street Journal*

[1] A complete summary of the grammatical relations used by the Link parser can be found at http://bobo.link.cs.cmu.edu/link/dict/summarize-links.html.

[2] Carroll's evaluation software is available at http://www.informatics.susx.ac.uk/research/nlp/carroll/greval.html

Scheme	# GR
Carroll	23
MiniPar	59
Link	106
Stanford	47

Table 1: Number of grammatical relations of four different annotation schemes.

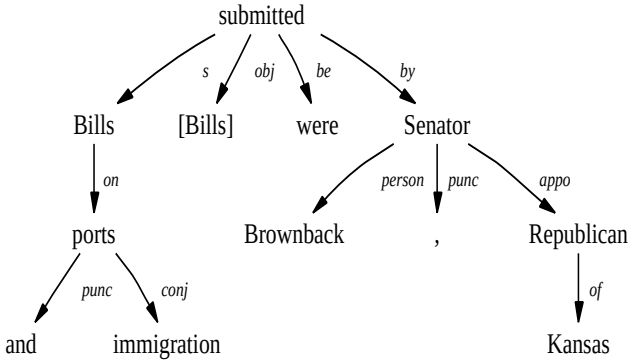

Figure 7: Minipar dependency parse for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas".

Treebank does a poor job at parsing questions (sentences 7 and 9) and the dependencies outputted are therefore wrong or not specific enough. This is easily explained by the fact that the parser is trained on the Wall Street Journal section of the Penn Treebank in which not many questions occur. For use in other projects, we have augmented the training data with a modest number of additional questions. In sentence 8, we got *dep*(chair, out) while "out" should be connected to "sat". This link is correctly identified by both Minipar and the Link parser.

Minipar is confused by punctuation (this fact has already be mentioned in (Lin, 1998)): e.g., in sentence 5 no subject of the verb "had suggested" is found, and the parser outputs only chunks of the sentence not related to one another. Minipar is also confused by conjunction: in sentence 3, "awarding" is connected with "administrators", while it should be related to "appointment". An advantage of Minipar is its capacity to identify collocations as "comment on" in sentence 3 or "how many" in sentence 7.

As already mentioned, the *MX* relation of the Link parser leads to weird dependencies: in sentence 9, "smoking" and "waiting" are dependents of "tree". They should however be related to "Rector". The Link parser has trouble with conjunction: the parse of sentence 3 is wrong. Question 9 is also wrongly parsed.

We evaluated our system on this sample of 10 sentences, with the "collapsing" option turned on. A dependency tagged as *dep* is considered to be wrong if a more specific dependency type should have been used. We obtained a per-dependency accuracy of 80.3%. However it can be only considered as a rough estimate because the sample size is very small.

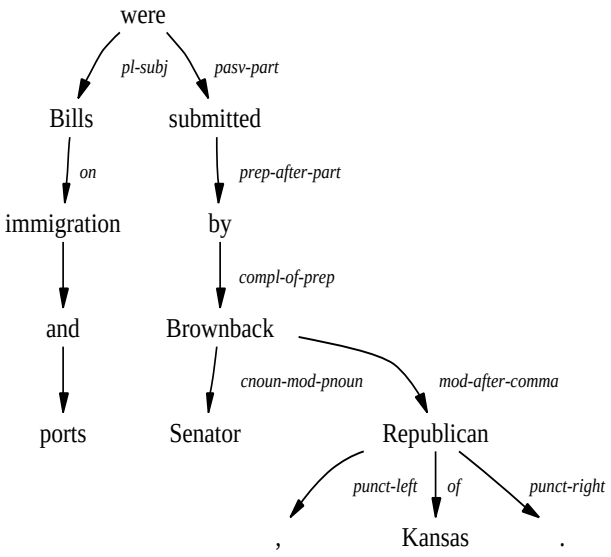

Figure 8: Link Parser dependency parse for the sentence "Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas".

5. Application

The typed dependency trees generated by this system have been used as the foundation for systems (Raina et al., 2005; de Marneffe et al., 2006) which were Stanford's entry in the PASCAL Recognizing Textual Entailment (RTE) challenges. Here the task is to determine whether one sentence can reasonably be inferred from another sentence. The Stanford system exploits the information about predicate-argument structure encoded in the generated typed dependency trees in three ways: in generating a quasi-logical representation of the event structure represented by each sentence (following the work of Moldovan and Harabagiu in question answering (Moldovan et al., 2003)), in finding a good alignment between the structures of the two sentences, and in generating features used as input to a learning module. The Stanford system, which used the information supplied by our typed dependency extractor, attained the highest confidence-weighted score of all entrants in the 2005 competition by a significant margin.

The typed dependency generation facility described in this paper has been integrated into the Stanford parser, which is available for download at http://www-nlp.stanford.edu/software/lex-parser.shtml.

6. References

Joan Bresnan. 2001. *Lexical-Functional Syntax*. Blackwell, Oxford.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC)*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL-2000*.

Noam Chomsky. 1981. *Lectures on Government and Binding*. Foris, Dordrecht.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

1	The Fulton_County_Grand_Jury said Friday an investigation of Atlanta 's recent primary_election produced "no evidence" that any irregularities took_place.
2	However, the jury said it believes "these two offices should be combined to achieve greater efficiency and reduce the cost of administration".
3	The jury also commented on the Fulton ordinary's court which has been under_fire for its practices in the appointment of appraisers, guardians and administrators and the awarding of fees and compensation.
4	When the larvae hatch, they feed_on the beebread, although they also receive extra honey meals from their mother.
5	In her letter to John_Brown, "E._B.", the Quakeress from Newport, had suggested that the American people owed more honor to John_Brown for seeking to free the slaves than they did to George_Washington.
6	Below he could see the bright torches lighting the riverbank.
7	"How many pamphlets do we have in_stock?", Rector said.
8	Then Rector, attired in his best blue serge suit, sat in a chair out on the lawn, in the shade of a tree, smoking a cigarette and waiting.
9	Have you any objection to the following plan?
10	She was watching a tree ride wildly down that roiling current.

Table 2: 10 sentences from the Brown Corpus, to compare outputs of Minipar, the Link Parser and the Stanford parser.

Marie-Catherine de Marneffe, Bill MacCartney, Trond Grenager, Daniel Cer, Anna Rafferty, and Christopher D. Manning. 2006. Learning to distinguish valid textual entailments. To appear in PASCAL RTE-2 Challenge workshop.

Tracy H. King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald Kaplan. 2003. The PARC 700 dependency bank. In *4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics*.

Matthias T. Kromann. 2003. The Danish Dependency Treebank and the underlying linguistic theory. In Joakim Nivre and Erhard Hinrichs, editors, *Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003)*. Vaxjo University Press.

Roger Levy and Galen Andrew. 2006. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *LREC 2006*. http://www-nlp.stanford.edu/software/tregex.shtml.

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

Philip H. Miller. 2000. *Strong Generative Capacity: The Semantics of Linguistic Formalism*. Number 46 in Lecture Notes. CSLI Publications, Stanford, CA.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. Cogex: A logic prover for question answering. In *HLT-NAACL*.

Rajat Raina, Andrew Y. Ng, and Christopher D. Manning. 2005. Robust textual inference via learning and abductive reasoning. In *Proceedings of AAAI 2005*. AAAI Press.

Daniel D. Sleator and Davy Temperley. 1993. Parsing English with a link grammar. In *Third International Workshop on Parsing Technologies*.

Leonoor van der Beek, Gosse Bouma, and Robert Malouf andGertjan van Noord. 2002. The Alpino Dependency Treebank. In *Computational Linguistics in the Netherlands CLIN 2001*.