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 have 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, rcomod - relative clause modifier, det - determiner, partmod - participial modifier, inffmod - 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.”

Figure 2: The grammatical relation hierarchy.
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)

np(guidetour)

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 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.

1A 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.

2Carroll’s evaluation software is available at http://www.informatics.susx.ac.uk/research/nlp/carroll/greval.html.
Table 1: Number of grammatical relations of four different annotation schemes.

<table>
<thead>
<tr>
<th>Scheme</th>
<th># GR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carroll</td>
<td>23</td>
</tr>
<tr>
<td>MiniPar</td>
<td>59</td>
</tr>
<tr>
<td>Link</td>
<td>106</td>
</tr>
<tr>
<td>Stanford</td>
<td>48</td>
</tr>
</tbody>
</table>

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 been 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.

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.

Acknowledgements

Thanks to Roger Levy and Galen Andrew for developing tregex, which is centrally used in this system, and to Galen for restructuring the grammatical relations code. This material is based upon work funded in part by the U.S. Government under the Advanced Research and Development Activity (ARDA’s) Advanced Question Answering for Intelligence (AQUAINT) Program. Any opinions, finding, conclusion, or recommendation expressed in this material are those of the author(s) and do not necessarily reflect the views of the U.S. Government.
The Fulton County Grand Jury said Friday an investigation of Atlanta’s recent primary election produced “no evidence” that any irregularities took place.

However, the jury said it believes “these two offices should be combined to achieve greater efficiency and reduce the cost of administration”.

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.

When the larvae hatch, they feed on the beebread, although they also receive extra honey meals from their mother.

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.

Below he could see the bright torches lighting the riverbank.

"How many pamphlets do we have in stock?", Rector said.

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.

Have you any objection to the following plan?

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.

6. References


It is widely accepted that POS tagging and dependency parsing are closely related. In our transition system, there are three types of conflicts: 1. Tag conflict among all possible POS tags \( \{ \text{TAG} | t \in T \} \), 2. Shift/reduce conflict between SHIFT, LEFT, and RIGHT. Generating typed dependency parses from phrase structure parses. In Proceedings of LREC, 2006. [dos Santos and Zadrozny, 2014] Cícero Nogueira dos Santos and Bianca Zadrozny. Phrase Structure Parsing. à£¢ Structural decomposition of a sentence à£¢ as a sequence of words. à£¢ Into constituents or brackets à£¢ Generally applicable data structure: nested trees à£¢ Enormous amount of ambiguity. à£¢ Even small grammars (10 rules) + Any lexicons generate 100à£™s of parses per sentence. Syntactic ambiguities. Æ Yet Another Parse. Unsupervised Learning of Dependency Grammar. à£¢ Production rules for DGs: à£¢ Same set of nonterminals and terminals as PSGà£™s. à£¢ SO, depends in a minor way on which PSG youà£™re dealing with. à£¢ Note: ROOT usually replaces S as distinguished start symbol. à£¢ X À XY or X À YX à£¢ And thatà£™s it: all dependencies are binary à£¢ Oxen in learning there are no lexical items; the.