Dependency Treebanks: Methods, Annotation Schemes and Tools
Tuomo Kakkonen

In this paper, current dependency-based treebanks are analyzed. The methods used for building the resources, the annotation schemes applied, and the tools used, such as part-of-speech taggers, parsers and annotation software used are discussed. The motivation for this work is to explore treebanks and methods before building a dependency-based treebank for evaluating parsers for Finnish.

Syntactically annotated corpora, *treebanks*, are needed for developing and evaluating natural language processing applications, as well as for research in empirical linguistics. The choice of type of annotation in a treebank usually boils down to two options: the linguistic resource is annotated either according to some constituent or functional structure scheme. As the name *treebank* suggests, these linguistic resources were first developed in the phrase-structure framework, usually represented as tree-shaped constructions. The most well known of such a type of treebank is the Penn treebank for English. In recent years, there has been wide interest towards functional annotation of treebanks. In particular, many dependency-based treebanks have been constructed. In addition, grammatical function annotation has been added to some constituent-type treebanks.

The most commonly used argument for selecting the dependency format for building a treebank is that the treebank is being created for a language with a relatively free word order. Such treebanks exist for Basque, Czech, Germany and Turkish. On the other hand, dependency treebanks have been developed for languages such as English, which have been usually seen as languages that can be better represented with constituent formalism. The motivations for using dependency annotation vary from the fact that the type of structure is the one needed by many, if not most, applications to the fact that it offers a proper interface between syntactic and semantic representation. Furthermore, dependency structures can be automatically converted in to phrase structures if needed, although usually not with 100% accuracy. The TIGER treebank of German, a free word order language, with 50000 sentences is an example of a treebank with both phrase structure and dependency annotations. Such hybrid treebanks are also available for Danish and Dutch.

The size of current dependency treebanks is usually quite limited. Most commonly they have a few thousand sentences, with the exception of the TIGER treebank introduced above and the Prague Dependency Treebank of Czech that has 90000 sentences. Treebank producers have in most cases aimed at creating a multipurpose resource for evaluating and developing NLP systems, and for linguistic studies. Some are built for specific purposes, e.g. the Alpino treebank for Dutch is mainly for parser evaluation. Most of the dependency treebanks consist of written text; to our knowledge there is only one that is based on a collection of spoken utterances. Annotation usually consists of part-of-speech and morphological levels accompanied by dependency-based syntactic annotation. In some cases a higher, semantic layer of annotation is also included.

Several kinds of resources and tools are needed for constructing a treebank: annotation guidelines state the conventions that guide the annotators through their work, a software tool is needed to aid the annotation work, and in the case of semi-automated treebank construction, a part-of-speech tagger, morphological analyzer and/or a syntactic parser is needed. The most commonly used method for developing a treebank is a combination of automatic and manual processing, but the practical method of implementation varies considerably. There are some treebanks that have been annotated completely manually, but with taggers and parsers available to automate some of the work; such a method of building treebanks is rarely employed in state-of-the-art treebanking. Building trees manually is very slow and error-prone process. The most efficient method currently available appears to be the method in which the part-of-speech and morphological tagging and at least some part of the syntactic parsing is done automatically and where the resulting structures are checked and corrected by human annotators. The usual procedure is to use a parser that leaves at least part of ambiguities unresolved and to use human annotators carry out the disambiguation. The main disadvantage with such a method is the amount of parses created by the parser, which in the case of a long sentence can be enormous. Thus, a parser applied for treebank building should perform at least some disambiguation.

The main problem with current treebanks in regards to use and distribution is the fact that instead of reusing existing formats, new ones are being developed. Furthermore, the schemes that have been developed usually are designed from theory and even application-specific viewpoints, and, consequently, undermine the possibility for reuse. In addition to the difficulties for reuse, creating a treebank-specific representation format requires creating a new set of tools for creating and manipulating the treebank to be developed. Yet, the existence of exchange formats such as XML-based XCES and TigerXML would allow multipurpose tools to be created and used. The use of such formats and other methods that enable conversions between treebank formats are also discussed in the paper.