CoNLL-X Shared Task: Multi-lingual Dependency Parsing

Abstract

The shared task of CoNLL-X will be multi-lingual dependency parsing. Following previous CoNLL shared tasks (NP bracketing, chunking, clause identification, language independent named-entity recognition, and semantic role labeling), this task aims to define and extend the current state of the art in dependency parsing - a technology which complements the previous tasks by producing a different kind of syntactic description of input text.

Ideally, a parser should be trainable for any language, possibly by adjusting a small number of hyperparameters. The CoNLL-X shared task will provide the community with a benchmark for evaluating their parsers across different languages. Because of the variety of languages and the interest in parser performance across languages, a special focus of the CoNLL-X shared task will be on a qualitative evaluation (along with the quantitative scores as before). We will require the participants to provide an informative error analysis and will ourselves perform a cross-system comparison. This, we expect, will result in a clear picture of the problems that lie ahead for multilingual parsing and the kind of work necessary for adapting existing parsing architectures across languages.

This page provides a detailed description of the shared task and further information regarding scheduling, datasets, paper submission, etc.

Background

Head-modifier dependency relations have been employed as a useful representation in several language modelling tasks. These include unsupervised acquisition of argument structure (Briscoe and Carroll, 1997; McCarthy, 2000), generating word classes (Clark and Weir, 2000) and cooccurence probabilities (Dagan et al., 1999) for disambiguation, and extracting collocations (Lin, 1999; Pearce, 2001). Palmer et al. (1993) and Yeh (2000) used dependencies as intermediate representations for information retrieval. In addition, some of the recent parser evaluation schemes (Carroll et al., 2000; Lin, 2000; Srinivas, 2000) use dependencies instead of phrases.

Several statistical approaches to full phrase structure parsing model the probability of dependencies between pairs of words (Collins, 1997; Charniak, 1999). Brants et al. (1997) report on a markov model for adding grammatical functions to a phrase structure. Other approaches are directly aimed at unlabelled (Eisner, 1996; McDonald et al. (2005)) or labelled dependency parsing (Nivre and Scholz (2004)).

Until a few years ago, parsers have mostly been applied to only one or two languages: often English and/or the author's native language. Recently, however, several parsers have been applied to more languages, e.g. Collins' parsing model has been tested for English, Czech (Collins et al., 1999), German (Dubey and Keller, 2003), Spanish (Cowan and Collins, 2005) and French (Arun and Keller, 2005), while Nivre's parser has been tested for English (Nivre and Scholz, 2004), Swedish (Nivre et al., 2004) and Czech (Nivre and Nilsson, 2005). This has resulted in interesting insights into how the properties of a language or a treebank influence parser performance, or conversely, how one should best approach parsing for that language/treebank (see e.g. the above references for Keller). However, different parsers have been applied to different subsets of languages.

Ideally, a parser should be trainable for any language, possibly by adjusting parameters. The CoNLL 2006 shared task will provide the community members with the possibility of testing

their parsers across different languages. The training and test data for the languages differ in size, granularity and quality, but we will try to at least even out differences in the markup format.

Task definition

The shared task is to assign **labeled** dependency structures for a range of languages by means of a fully automatic dependency parser. Some gold standard dependency structures against which systems are scored will be **non-projective**. A system that produces only projective structures will nevertheless be scored against the partially non-projective gold standard. The input consists of (minimally) tokenized and part-of-speech tagged sentences. Each sentence is represented as a sequence of tokens plus additional features such as lemma, part-of-speech, or morphological properties. For each token, the parser must output its head and the corresponding dependency relation (secondary relations are not taken into consideration). Although data and settings may vary per language, the same parser should handle all languages. The parser must therefore be able to learn from training data, to generalize to unseen test data, and to handle multiple languages. Participants are expected to submit parsing results for **all** languages involved.

Clarifications (added 22 January 2006)

What is meant by "the same parser should handle all languages"?
For example, if a parsing software package allows the use of several different parsing algorithms or of several different machine learners, are those still "the same parser"? No. So in general, you should choose one parsing algorithm and one learner. However, we realize that it would still be an interesting result of this shared task if it turned out that algorithm/learner X is much better for language/treebank Y while algorithm/learner W is much better for language/treebank Z. So, in line with our focus on an informative error analysis, we allow deviation from your "default" algorithm/learner if you can explain WHY the non-default algorithm/learner is better in some cases. We hope that this will restrict deviations to the really important ones.

What about pre- and post-processing steps not integrated in the parser itself?
That is fine, provided that those steps constitute general approaches (e.g. feature construction, tree manipulation) and not just manual hacks to fix certain errors (e.g. replace relation X in context Y with relation Z).

Data format

Data adheres to the following rules:

- Data files contain sentences separated by a blank line.
- A sentence consists of one or tokens, each one starting on a new line.
- A token consists of ten fields described in the table below. Fields are separated by a single tab character. Space/blank characters are not allowed in within fields
- All data files will contains these ten fields, although only the ID, FORM, CPOSTAG, POSTAG, HEAD and DEPREL columns are guaranteed to contain non-dummy (i.e. non-underscore) values for all languages.
- Data files are UTF-8 encoded (Unicode).

Field number:	Field name:	Description:
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma or stem (depending on particular data set) of word form, or an underscore if not available.
4	CPOSTAG	Coarse-grained part-of-speech tag, where tagset depends on the language.
5	POSTAG	Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.
6	FEATS	Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.
7	HEAD	Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningfull or simply 'ROOT'.
9	PHEAD	Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).
10	PDEPREL	Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency

Field number:	Field name:	Description:
		relation may be meaningful or simply 'ROOT'.

Here is an example from the Dutch data set.