Dependency grammar and dependency parsing

Syntactic analysis (5LN455)

2012-12-11

Joakim Nivre
Department of Linguistics and Philology

Based on slides by Marco Kuhlmann
Dependency grammar
Dependency grammar

- The term ‘dependency grammar’ does not refer to a specific grammar formalism.
- Rather, it refers to a specific way to describe the syntactic structure of a sentence.
The notion of dependency

- The basic observation behind **constituency** is that groups of words may act as one unit.
  
  *Example:* noun phrase, prepositional phrase

- The basic observation behind **dependency** is that words have grammatical functions with respect to other words in the sentence.
  
  *Example:* subject, modifier
Dependency grammar

Phrase structure trees

S
  NP  VP
    |    |
  Pro  Verb  NP
    |    |
    I  booked  Det  Nom
      |    |
      |  a  Nom  PP
      |    |
      |  Noun  from LA
      |    |
      |  flight
• In an arc $h \rightarrow d$, the word $h$ is called the head, and the word $d$ is called the dependent.

• The arcs form a rooted tree.
Heads in phrase structure grammar

• In phrase structure grammar, ideas from dependency grammar can be found in the notion of heads.

• Roughly speaking, the head of a phrase is the most important word of the phrase: the word that determines the phrase function.

Examples: noun in a noun phrase, preposition in a prepositional phrase
Heads in phrase structure grammar

Dependency grammar

S
  └── NP
      ├── Pro
      │    └── I
      └── VP
          └── NP
              └── Verb
t
Det
  └── Nom
      └── PP
          └── from LA

NP
  └── Nom
      └── Noun
          └── flight

Nom
  └── a
  └── booked

NP
  └── a
  └── I
  └── from LA

NP
  └── I
  └── from LA
The history of dependency grammar

• The notion of dependency can be found in some of the earliest formal grammars.

• Modern dependency grammar is attributed to Lucien Tesnière (1893–1954).

• Recent years have seen a revived interest in dependency-based description of natural language syntax.
Head-dependency relations

- Verb + arguments
  - Subject: *Sandy writes poetry*
  - Object: *Sandy writes poetry*

- Noun + modifiers
  - Determiner: *the little black cat*
  - Adjectival modifier: *the little black cat*
Some tricky cases

- Coordination
  - *Sandy and Kim write* poetry

- Verb groups
  - *Sandy could have written* poetry

- Prepositional phrases
  - *Sandy went to London*
Examples

• What dependency relations do you find in the following sentences?

  Her mother sent her a letter.

  Economic news had little effect on financial markets.
Linguistic resources

- Descriptive dependency grammars exist for some natural languages.
- Dependency treebanks exist for a wide range of natural languages.
- These treebanks can be used to train accurate and efficient dependency parsers.
Overview

- Arc-factored dependency parsing
  - Collins’ algorithm
  - Eisner’s algorithm
- Transition-based dependency parsing
  - The arc-standard algorithm
- Evaluation of dependency parsers
Arc-factored dependency parsing
Just like phrase structure parsing, dependency parsing has to deal with ambiguity.
Just like phrase structure parsing, dependency parsing has to deal with ambiguity.

I booked a flight from LA.
Disambiguation

- We need to **disambiguate** between alternative analyses.
- We develop mechanisms for scoring dependency trees, and disambiguate by choosing a dependency tree with the highest score.
Scoring models and parsing algorithms

Distinguish two aspects:

• **Scoring model:**
  How do we want to score dependency trees?

• **Parsing algorithm:**
  How do we compute a highest-scoring dependency tree under the given scoring model?
The arc-factored model

• Split the dependency tree $t$ into parts $p_1, ..., p_n$, score each of the parts individually, and combine the score into a simple sum.

$$\text{score}(t) = \text{score}(p_1) + \ldots + \text{score}(p_n)$$

• The simplest scoring model is the arc-factored model, where the scored parts are the arcs of the tree.
To score an arc, we define features that are likely to be relevant in the context of parsing.

We represent an arc by its feature vector.
Examples of features

- ‘The head is a verb.’
- ‘The dependent is a noun.’
- ‘The head is a verb and the dependent is a noun.’
- ‘The head is a verb and the predecessor of the head is a pronoun.’
- ‘The arc goes from left to right.’
- ‘The arc has length 2.’
Arc-factored dependency parsing

Feature vectors

- Feature: ‘The head is a verb.’
- Feature: ‘The dependent is a noun.’

```
booked → flight
flight → from LA
flight → a
booked → 1
```

Feature vectors: (0, 0), (0, 1), (1, 0), (1, 1)
Implementation of feature vectors

- We assign each feature a unique number.
- For each arc, we collect the numbers of those features that apply to that arc.
- The feature vector of the arc is the list of those numbers.

*Example:* $[1, 2, 42, 313, 1977, 2008, 2010]$
Arc-factored dependency parsers require a training phase.

During training, our goal is to assign, to each feature $f_i$, a feature weight $w_i$.

Intuitively, the weight $w_i$ quantifies the effect of the feature $f_i$ on the likelihood of the arc.

How likely is it that we will see an arc with this feature in a useful dependency tree?
We define the **score** of an arc \( h \rightarrow d \) as the weighted sum of all features of that arc:

\[
\text{score}(h \rightarrow d) = f_1w_1 + \ldots + f_nw_n
\]
Training using structured prediction

- Take a sentence $w$ and a gold-standard dependency tree $g$ for $w$.
- Compute the highest-scoring dependency tree under the current weights; call it $p$.
- Increase the weights of all features that are in $g$ but not in $p$.
- Decrease the weights of all features that are in $p$ but not in $g$.
Training involves repeatedly parsing (treebank) sentences and refining the weights.

Hence, training presupposes an efficient parsing algorithm.

Next time we will look at parsing algorithms for the arc-factored model.
Higher-order models

- The arc-factored model is a first-order model, because scored subgraphs consist of a single arc.
- An nth-order model scores subgraphs consisting of (at most) n arcs.
  - Second-order: siblings, grand-parents
  - Third-order: tri-siblings, grand-siblings
- Higher-order models capture more linguistic structure and give higher parsing accuracy.
The term ‘arc-factored dependency parsing’ refers to dependency parsers that score a dependency tree by scoring its arcs. Arcs are scored by defining features and assigning weights to these features. The resulting parsers can be trained using structured prediction. More powerful scoring models exist.