Dependency Parsing

Outline

Two formalisms for syntactic structure: Phrase structure and dependencies

Two algorithms for dependency parsing

- Transition based dependency parsing
- Graph based dependency parsing

Evaluating dependencies

Dependency parsing

- Input: Sentence, tokenized + a dummy ROOT word
- Output: A dependency tree
  - Each word in the sentence is a node
  - Every word (except ROOT) should have an incoming edge indicating its head word
  - Only one word should be a dependent of ROOT
  - There are no cycles

Dependency parsing

- Input: Sentence, tokenized + a dummy ROOT word

- Output: A dependency tree
  - Each word in the sentence is a node
  - Every word (except ROOT) should have an incoming edge indicating its head word
  - Only one word should be a dependent of ROOT
  - There are no cycles

- Dependency theory also allows arrows to cross
  - Trees no arrows cross are called projective
  - Otherwise, they are called non-projective

Projective parse tree: No crossing dependency arcs when the words are laid out in their linear order, with all arcs above the word

Two families of parsing algorithms

Transition-based parsing

Graph based parsing

Two families of parsing algorithms

Transition-based parsing

- A generalization of the idea of shift-reduce parsing
- Greedily build attachments, using classifiers to decide which attachments to perform next
- Before neural networks: MaltParser (Nivre et al 2008)
- After neural networks: Chen and Manning (2014), Kipperwaser and Goldberg (2017)

Graph based parsing

Two families of parsing algorithms

Transition-based parsing

- A generalization of the idea of shift-reduce parsing
- Greedily build attachments, using classifiers to decide which attachments to perform next
- Before neural networks: MaltParser (Nivre et al 2008)
- After neural networks: Chen and Manning (2014), Kipperwaser and Goldberg (2017)

Graph based parsing

- Score all possible pairs of dependencies using a classifier
- Use a minimum spanning tree algorithm to find the best labeled tree
- Before neural networks: MSTParser (McDonald et al, 2005)
- With neural networks: Dozat and Manning (2017)

Two families of parsing algorithms

Transition-based parsing

- A generalization of the idea of shift-reduce parsing
- Greedily build attachments, using classifiers to decide which attachments to perform next
- Before neural networks: MaltParser (Nivre et al 2008)
- After neural networks: Chen and Manning (2014), Kipperwaser and Goldberg (2017)

Graph based parsing

- Score all possible pairs of dependencies using a classifier
- Use a minimum spanning tree algorithm to find the best labeled tree
- Before neural networks: MSTParser (McDonald et al, 2005)
- With neural networks: Dozat and Manning (2017)

Other algorithms exist as well. E.g. Eisner's algorithm is a dynamic programming approach

Outline

Two formalisms for syntactic structure: Phrase structure and dependencies

Two algorithms for dependency parsing

- Transition based dependency parsing
- Graph based dependency parsing

Evaluating dependencies

Dependency parses = spanning trees

A dependency tree is a tree whose words are nodes. Every word in a sentence has to be reachable from the root of the tree

- This means that the dependency tree is a spanning tree over the words

Dependency parses = spanning trees

A dependency tree is a tree whose words are nodes. Every word in a sentence has to be reachable from the root of the tree

- This means that the dependency tree is a spanning tree over the words

Key idea of graph-based parsing: Given an input sentence S, the goal is to find the “best” tree

- Define “best” using a scoring function called score

Dependency parses = spanning trees

A dependency tree is a tree whose words are nodes. Every word in a sentence has to be reachable from the root of the tree

- This means that the dependency tree is a spanning tree over the words

Key idea of graph-based parsing: Given an input sentence S, the goal is to find the "best" tree

- Define "best" using a scoring function called score

The goal of parsing:

$$T^* = \arg\max_{t \in G(S)} \text{score}(t, S)$$

Dependency parses = spanning trees

A dependency tree is a tree whose words are nodes. Every word in a sentence has to be reachable from the root of the tree

- This means that the dependency tree is a spanning tree over the words

Key idea of graph-based parsing: Given an input sentence S, the goal is to find the "best" tree

- Define "best" using a scoring function called score

The goal of parsing:

$$T^* = \arg\max_{t \in G(S)} \mathrm{score}(t, S)$$

The best tree is the one that maximizes the score of the tree among all spanning trees for that sentence

Dependency parses = spanning trees

A dependency tree is a tree whose words are nodes. Every word in a sentence has to be reachable from the root of the tree

- This means that the dependency tree is a spanning tree over the words

Key idea of graph-based parsing: Given an input sentence S, the goal is to find the "best" tree

- Define "best" using a scoring function called score

The goal of parsing:

$$T^* = \arg\max_{t \in G(S)} \mathrm{score}(t, S)$$

The best tree is the one that maximizes the score of the tree among all spanning trees for that sentence

Dependency parses = spanning trees

A dependency tree is a tree whose words are nodes. Every word in a sentence has to be reachable from the root of the tree

- This means that the dependency tree is a spanning tree over the words

Key idea of graph-based parsing: Given an input sentence S, the goal is to find the "best" tree

- Define "best" using a scoring function called score

The goal of parsing:

$$T^* = \arg\max_{t \in G(S)} \text{score}(t, S)$$

The best tree is the one that maximizes the score of the tree among all spanning trees for that sentence

Dependency parses = spanning trees

A dependency tree is a tree whose words are nodes. Every word in a sentence has to be reachable from the root of the tree

- This means that the dependency tree is a spanning tree over the words

Key idea of graph-based parsing: Given an input sentence S, the goal is to find the "best" tree

- Define "best" using a scoring function called score

The goal of parsing:

$$T^* = \arg\max_{t \in G(S)} \text{score}(t, S)$$

The best tree is the one that maximizes the score of the tree among all spanning trees for that sentence

Scoring trees: The edge factored model

Trees can be arbitrarily large because sentences can be arbitrarily large

We need a scoring scheme that can account for this

The edge factored model: The score for a tree is the sum of scores of all its edges

$$score(t, S) = \sum_{e \in t} edge\text{-}score(e)$$

The modeling problem: Scoring edges

Any edge in the dependency tree is the tripe (head, dependent, label)

The scoring function can be any neural network.

Dozat and Manning (2017) trained two neural networks:

1. One computes the probability that there is an edge from word i to word j i.e., $P(i \rightarrow j)$
2. Another assigns a label to an edge, i.e., $P(label \mid i \rightarrow j)$

The inference problem

Given some edge scoring function, we need to search over all trees to find the highest scoring one

This is the well studied problem of finding a maximum spanning tree

The inference problem

Given some edge scoring function, we need to search over all trees to find the highest scoring one

This is the well studied problem of finding a maximum spanning tree

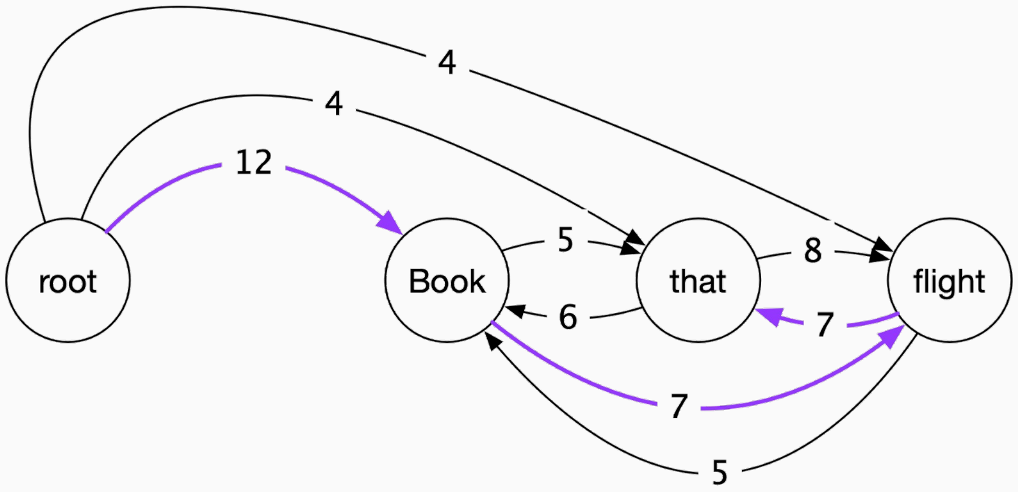
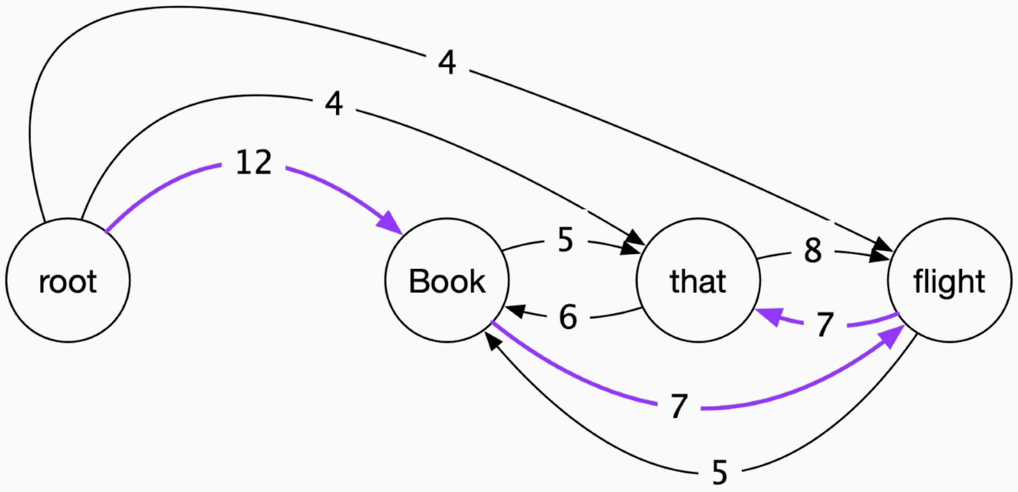


Image from Jurafsky and Martin’s book

The inference problem

Given some edge scoring function, we need to search over all trees to find the highest scoring one

This is the well studied problem of finding a maximum spanning tree



The standard solution: The Chu-Liu Edmonds algorithm

Image from Jurafsky and Martin’s book

Comparing transition- and graph-based parsing

- Computational Complexity
  - Graph based parsing is quadratic in the input length
  - Transition based parsing is linear in sentence length
- Projectivity
  - Graph based parsing can produce non-projective parse trees
  - Transition based parsing can only produce projective trees without additional cleanup
  - Tends to be a smaller issue English than for many languages
- Accuracy
  - Graph based parsers can be better on dependency arcs where the head is far from the dependent