Stanford CS224N NLP with Deep Learning

Winter 2021

Lecture 4 – Dependency Parsing

Syntactic Structure

There are two main views of the structure of sentences – constituency-based (examples of which are phrase structure grammars and context-free grammars) and dependency-based approaches.
Constituency-based approaches group words into larger chunks (constituents), and build hierarchical structures of sentences. Dependency grammars, on the other hand, show which words depend on others. "Dependence" here may refer to modifying, attaching to or being an argument of a word.

Dependency Structure

Dependency syntax describe syntactic structure in terms of (binary asymmetric) relations, called dependencies, between lexical items.

Conventionally, a dummy `ROOT` node is added during parsing to ensure that every word is dependent on some other word.

Dependency data is obtained from human-annotated *treebanks*, or collections of sentences paired with their dependency trees. Methods making use of treebanks stand in contrast to those using handcrafted rule systems or grammars, which have reduced in popularity in recent years.

Dependency Parsing

There are a number of sources of information that systems can use to identify a dependency parse. These include bilexical affinities, valencies, intervening material, and word distance.

A sentence is parsed by choosing, for each word, which word it is dependent on. This is done under certain constraints: only one word can be the dependent of `ROOT`, there should be no cycles, etc.

Projectivity

A *projective* parse is one in which there are no crossing dependency arcs when the words are presented in the same order as in the sentene. Dependencies generated from a CFG parse are always projective.

Methods

There are various methods to identify the dependency parse of sentences:

- dynamic programming: Eisner (1996)
- graph algorithms: finding the minimum spanning tree for a set of edge scores
- constraint satisfaction: Karlsson (1990)
- transition-based parsing: greedily choosing attachment using ML-based classifiers

Greedy transition-based parsing proceeds by manipulating a stack of input tokens with three actions: `shift`, `leftArc-reduce` and `rightArc-reduce`.
Choosing the next action given the state of the stack is usually carried out using a classifier (among 3 classes for unlabelled parsing, and $2c+1$ classes for labelled parsing with c relations). Features like the word on top of the stack, the first word in the buffer, POS tags, and so on, are also used.
This model provides performance fractionally below SoTA, in linear time.

Evaluation

Dependency parsers are evaluated according to two common metrics – unlabelled and labelled attachment scores (UAS and LAS). UAS measures the percentage of words that have been assigned their correct heads, while LAS measures the percentage of words that have been assigned their correct heads *with the correct relation.*