4744 Final Project Proposal:
Differences in Dependency Parsing across Languages

Isabel Sharp

April 14, 2017

1 Overview

In this project, I will implement Nivre's dependency parsing algorithm to parse sentences based on their dependencies, using an extension of the shift-reduce algorithm. I will then use this parser to analyze the differences in dependencies across languages, by parsing simple sentences from a variety of languages and examining the time (in number of steps) the parser takes to produce the dependency graph. I will obtain these language-specific dependency parses from the Universal Dependencies database, which contains specific constructions for parses from many different languages.

2 Parsing Algorithm

The first part of the project will be to implement Nivre's dependency parser, which takes a sentence and builds a set of arcs representing a dependency graph. The parser will keep track of a stack of partially processed words, along with the input sentence. The sentence needs to be annotated with parts of speech, since the dependency rules use parts of speech in their definitions. I will need to build up a grammar for each language that I intend to use, and then write a guide function that, given a grammar and a parser state, can select the next transition to make out of the four following transitions:

1. Shift: adds the next input word (from the input sentence) onto the stack
2. Reduce: takes the top word off the stack
3. Left-arc: takes the next input word, call it w, and reduces the stack, which results in removing the top word on the stack, call it w'. Adds an arc (w,w').
4. Right-arc: takes the next input word, call it w, and the top word on the stack, call it w'. Adds an arc (w',w), and shifts w onto the stack.

I will write these functions and implement the parser as specified in the reading, using a collection of dependency rules to determine when to add the arcs. The guide function links the dependency rules to the above functions, and determines which applies for the current parsing state.

One problem with the algorithm is that the transition rules are not unambiguous; at any point there could be multiple rules that apply. Since all of these could lead to a valid parse, there needs to be some kind of parsing strategy to select the appropriate transition to try. Two such strategies are given in the reading; one involves a constant hierarchy, and the other has a hierarchy and a set of conditions for when to try which function. Part of my project will be determining whether different languages parse more efficiently with using one strategy or the other.

The main focus of the project, once I have built the parser, is to determine how similar dependencies are across different languages. I will do this by writing grammars for several languages, and then using my parser to build dependency trees for simple sentences. While running the parser, I will keep track of the number of calls to the above functions occur for each sentence. This way, I will be able to determine which sentences took much more effort to parse, and thus be able to determine which have dependencies that are least similar to those that the parser is programmed to try first. I will use this information to try to tailor the parser to different languages.

3 Languages Choices

I would like to examine languages from several different language families, to see how similar dependencies are across these families. Since I am most familar with Indo-European languages, I will concentrate primarily on these. Depending on time constraints, I plan to analyze two Germanic languages (English and German), and see if these are more similar to each other than they are to a Romance language (Spanish). Time-permitting, I would also like to see whether Indo-European languages, even those from different sub-families, are more similar to each other than they are to languages from other families, for example Arabic from the Afro-Asiatic family, or Mandarin Chinese from Sino-Tibetan.

4 Research Materials

I will use chapter 13 of Nugues for implementing the parsing algorithm. The algorithm is spelled out fairly clearly here, and I will follow the implementation suggestions given. More information about this comes from Jurafsky & Martin, which I will use as well.

For the analysis of different languages, I will use the Universal Dependencies database. This will allow me to come up with simple sentences that have already been given dependency relation graphs, that I can use to make sure that my parser is producing the correct relations for me to analyze.

5 Possible Extensions (Time-Permitting)

I intend to write this parser to be able to handle only simple sentences from a small selection of languages. However, in order to obtain more substantial and interesting results, it would be necessary to broaden both the grammar that my parser works with, and the selection of languages used. I am particularly interested in determining if there is a correlation between the parsing of sentences from the same language families; intuitively, it seems that these language families would share dependency rules, but this is a statement that I would like to be able to use my parser to determine the validity of. Thus, one possible extension is simply a further development of the grammars, and a broader selection of data, both within each selected language, and across more languages. Since the Universal Dependencies website has information on so many languages, I do not see any limitations to these extensions beyond time.

I could also use search strategies to try to optimize the parser, once I determine which languages it is able to parse quickly, and which languages it has trouble with. These will come from the book *ML for the Modern Programmer*. As mentioned before, there can be indeterminacy while parsing, potentially creating many different options for the parser. I would try to incorporate search strategies into the parser in order for it to better select which parses it explores first, in an attempt to optimize the parser for different languages.

6 Bibliography

Nugues, Pierre M. *Language Processing with Perl and Prolog.* Springer Berlin Heidelberg, 2014. Accessed 14 Apr. 2017.

Jurafsky, Daniel, and James H. Martin. *Speech and Language Processing.* 2017. Accessed 14 Apr. 2017

Paulson, Larry C. *ML for the Working Programmer.* 2nd ed., Cambridge University Press, 1996. Accessed 14 Apr. 2017

Nivre, Joakim. "Universal Dependencies V2." *Universal Dependencies.* LINDAT/CLARIN, 2014. Web. 14 Apr. 2017.