Work Sample

MS - Computational Linguistics

Mike Roylance

Introduction

For my work sample, I am submitting a coreference resolution parser I wrote as part of a team project for the graduate Natural Language Processing class at the University of Utah. The team members were myself and Ben Nelson, a computer science student. We based our Coreference Resolution parser off the paper Noun Phrase Coreference as Clustering by Cardie and Wagstaff. We collaborated together on this with the use of GitHub as our code repository. Our work, including code submission history, can be viewed here: Coreference Resolution. As a current University of Washington student, I also uploaded this to Patas under the directory /home2/roylance/cs_6340_nlp_coreference_utah/.

What This Application Does

This application takes in a document annotated with noun phrases that have been identified as having antecedents. It attempts to correctly label them. Once finished, it prints out a response document with the annotated noun phrases. For example:

The input document:

<TXT>
Mary goes on a walk.
<COREF ID="1">She</COREF> loves doing that.
</TXT>

returns the response document:

<TXT>
<COREF ID="X1">Mary</COREF> goes on a walk.
<COREF ID="1" REF="X1">She</COREF> loves doing that.
</TXT>

How It Works

The application first takes in a file with the same XML format as referenced above. It then begins to process each line, building up each word in the sentence until the end of the sentence has been reached. The application utilizes the Stanford NLP Parser to help understand when the sentence most likely has ended. Once a sentence has been created, it then utilizes the Stanford NLP Parser again, but this time to help identify the specific parts of speech. A list of all the sentences and their corresponding parts of speech are then stored in the order they appear in the document.

Once all the data for the document has been created, the application examines each noun phrase and runs through a series of heuristics to determine what the most likely

antecedent is. It evaluates a set of criteria for each possible candidate and scores them with variable weights. The criteria are:

Plurality matching - do the noun phrases match in plurality?
Gender matching - do the noun phrases match in gender?
Edit Distance - is there an acceptable edit distance score?
Proper Names - are both noun phrases proper names?
Pronoun Types - are both noun phrases matching pronouns?
I and Me - I and Me tend to relate to each other.
Appositive - is this noun phrase located inside an appositive?

The antecedent with the best score is selected.

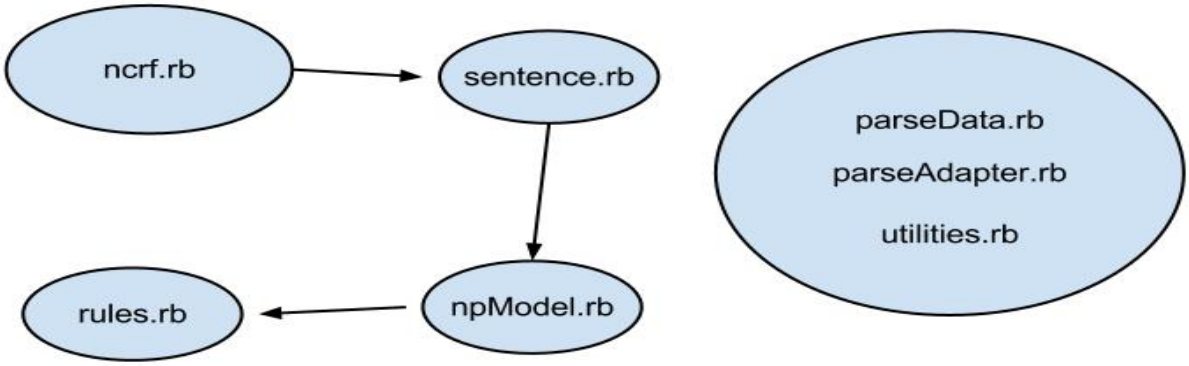
How It Performed

There were three sets given: a development set to initially develop against, a test set shortly before the deadline, and a test set used after final submissions. The application performed well and was one of the best performing in the class:

Test Set	Accuracy
Dev Set	53.50%
Test Set 1	47.35%
Test Set 2	47.20%

Technical Architecture

This was written in Ruby utilizing JRuby for the Stanford NLP Parser. The architecture is represented as the following:



ncrf.rb

This class represents the input document file. It handles all the methods for input/output, XML parsing and assigning IDs.

sentence.rb

This class is a wrapper around a collection of words and their corresponding noun phrases.

npModel.rb

This class keeps track of the identified noun phrase. It also identifies the position the phrase exists in the document, its semantic class, whether it's a proper noun and if it's a pronoun.

rules.rb

This class contains all the rules for scoring the antecedents of a noun phrase.

helper classes

parseAdapter.rb - is our adapter for functionality needed with the Standford NLP library.
parseData.rb - this parses the models from the Stanford NLP library.
utilities.rb - this has our implementation of editDistance and other algorithms.

How To Run

To run, this application can be cloned from git by typing in the following:

git clone https://github.com/roylanceMichael/cs_6340_nlp_coreference_utah.git

From there, navigate to the cs_6340_nlp_coreference_utah folder and type in

./run.sh testXml/firstSet/1.crf

to have the responseXml document printed to the command line.