HW7
Mike Roylance
roylance@uw.edu

Describe and discuss your work in a write-up file

I completed this assignment using Python with nltk. My solution is organized into the following files and directories:

Location Description	**Purpose**
docs/	folder that contains all the documents needed for this assignment: grammar.cfg and coref_sentences.txt.
docs/grammar.cfg	file that contains the context free grammar for the sentences to be parsed
docs/coref_sentences.txt	file that contains tuples of sentences, with the first containing noun phrases as antecedents and the second with pronouns to be resolved
source/	folder that contains all the source code
source/main.py	entry point script that reads in the files from the user and prints out the result to the console.
source/tests.py	tests around functionality
source/hobbs.py	class that takes in a couple of parsed sentences (as nltk trees) and runs the hobbs algorithm to determine the pronouns and antecedents.
source/utils.py	utility module to load in the nltk trees and return a tuple
source/itemIndex.py	model class to hold information about pronouns and potential antecedents. this class also determines the gender and plurality of itself for agreement.
source/rules.py	class that calculates all the rules associated with selected a pronoun or a potential antecedent. also calculates if two itemIndexes are in agreement with each other (and therefore acceptable).
hw7.cmd	command file used by condor. this calls hw7.sh with the parameters of docs/grammar.cfg docs/coref_sentences.txt and results
hw7.sh	file to handle calling the Python file main.py with specific

	parameters

Include problems you came across and how (or if) you were able to solve them, any insights, special features, and what you learned. Give examples if possible.

I have done coreference resolution before, but I didn't focus specifically on Hobb's algorithm. I spent an amount of time making sure that I wasn't oversimplifying what the algorithm should be doing by what I had done previously.

I feel I created a good infrastructure around how to handle the algorithm. The driver class, Hobbs.py, takes in two nltk Trees that represent the first and second sentence. It also instantiates a singleton Rules.py class that contains the rules for identifying antecedents. It loops through the different lists and selectively filters them based on the criteria.

The rules.py class identifies what type of parts of speech are acceptable for pronouns (PRP, PossPro) and potential antecedents (S, SBAR, NP). It also contains specific dictionaries about what parts of speech are considered plural (NNS and them/their). It also encodes gender parts of speech, although in this case

For gender, I didn't see any male or female specific words in the sentences or grammar file. While I do instantiate "female" and "male" dictionaries, my rules only check the male one because it is a clone of the female one. For gender, I'm only check if it is "u" (unknown) or "m" (male).

Plurality, however, I feel works fine. I do give the plurality dictionary the correct part of speech (NNS), but it also has specific pronouns that are plural as well (they, them).

I was going to implement this with just the grammar file, but I like the approach of having it in the Rules.py class because there are certain rules that cannot be coded in the grammar file that must be coded in Rules.py - such as the correct part of speech of antecedents or the funtion to determine if a given antecedent actually comes before the pronoun to be compared with.

My file is printing out similarly to how the simple_example.txt is printing out - however the instructions on the website indicate that if a programmatic approach is taken then to just print out all the antecedents. I did go a bit further and specified agreement as well. I have an extra space to indicate when the file is finished processing.

Starting on line #50 in Hobbs.py, I manually list out the steps taken in code as specified on the web site.

This was a fun assignment, thank you for the great class!