Ahmad Elshenawy
Ling 571 – Wi2014
March 18, 2014

Hw7 write-up

**results-unannotated is a file with the raw results, and is the output of hw7.cmd
results is the file with annotations added

I decided to try my hand at the coding version of this assignment, changing the Hobbs algorithm a little bit to work with how I understood to use NLTK. There is an explanation for how my code works at the end of this write-up. The results are in the following format:

Sentence_1
Sentence_2
Pronoun_1 sentence_1_parse sentence_2_parse
Antecedent_1_flat_tree
Antecedent_1_plain_text
System_judgment
Hand Annotation

Antecedent_2_flat_tree
Antecedent_2_plain_text
System_judgment
Hand Annotation

Pronoun_N sentence_1_parse sentence_2_parse
Etc.
Etc.
Etc.

I have this output, which technically prints more than the example given on patas, because of how NLTK printed my trees. I slightly modified the grammar to attach AGR feature structures to the relevant nouns in this grammar, but when printing, NLTK prints **every** feature, making the trees harder to read, so I added those extra lines to make things more readable. There was also a problem that I couldn't figure out where sometimes a flattened parse tree didn't really print out flattened. I couldn't figure out why, and since it technically just throws off the formatting, and no results, I didn't get to worried about it.

I also want to mention that my code used only the number one best parse as determined by NLTK's nbest_parse() function. Some of the sentences have parses that aren't exactly the best parse though (e.g. 'restored immunity in mice with a weak immune system' ➔ (restored immunity (in mice) (with a weak immune

system)) and not (restored immunity (in (mice with a weak immune system)))), but I think my code should work fine with these sentences for any different parse.

My program has three possible judgments for any given proposed antecedent. One, it could be judged 'Reject' by the system, meaning there was some conflict making this antecedent unacceptable. Two, it could be judged 'Acceptable', meaning it is technically an acceptable pronoun (didn't really work **all** the time). Three, it could be judged 'Acceptable – preferred', meaning that it is correct in the ways that matter, and it matches up with the case of the pronoun it's being bound to, which is one of the concerns discussed in class for determining antecedence (although this bites me for the final test sentence).

As far as results, I took any labels of "Acceptable – preferred" to be the system's chosen antecedent, and for the most part, all of the "Acceptable – preferred" selections were correct, except for in the final pair of sentences, which is honestly rather ambiguous. Since my code gives preference to an antecedent that is the same grammatical case as the pronoun, the program selected what is a somewhat strange seeming antecedent. My code decided that the one's with sepsis were the scientists, and not the mice with a weak immune system. Technically, both are acceptable, and while 'mice' is more natural, 'scientists' would in fact have been the better choice.

Another strange sounding labeling came for the sentences:

```
Scientists restored immunity in mice with a weak immune system.
They injected them with a live vaccine.
```

Since my method of determining antecedence was so simplistic, 'Scientists' and 'mice' were each considered acceptable for both pronouns 'They' and 'Them'. The "Acceptable – preferred" judgments were correct in this case.

I also want to note that S nodes kind of get the shaft in my code. They are always included for consideration, but since our examples are so few and specific('Them' and 'They' don't match with S, and we have no examples with 'That'), it doesn't really get considered for antecedence practically.

All in all, the code I have now is **highly** tailored to NLTK and for this specific grammar we have for this assignment. For more complicated sentences like "Bill's father's picture of him", my code would select both "Bill" and "Bill's Father" as acceptable antecedents, when only "Bill" would be correct, and "Bill's father" would require a reflexive for such a binding. Also, "it' would not work well with the system I have at present, and there is much to be improved with how I try to match for case in bindings, and the preferences weighted with case matchings.

==========

==========Explanation of Code==========
==========

The algorithm starts (once a pair of sentences has been given) by building a parse tree for each sentence using NLTK's nbest_parse(). Then, my code runs through the parse tree, subtree by subtree, for sentence_1, grabbing any S and NP node that it encounters and adding that to a list of candidate antecedents. My code implements a very basic case system, were case is a variable initialized at None, and is changed to 'ACC' when a VP has been encountered. Each candidate antecedent is stored as a tuple (subtree, case).

Next, the program runs through all of the subtrees in sentence_2. Once again, any S or NP that is encountered is added to the list of candidate antecedents. However, if it reaches a node that is a pronoun (POS tag = PRP or Poss), then this subtree is appended to a list of pronouns in a tuple (pronoun, feature structure, case, list of candidate antecedents thus far). This way, any pronoun in sentence_2 is able to look at any NPs preceding it in its own local S tree to find an antecedent.

Then the code iterates through all the pronouns it has found. Since the list of candidate antecedents was compiled from the top-down, and Hobb's Algorithm requires you to look up from the starting node, the program iterates through a reversed list of candidate antecedents for this pronoun, checking for Feature Structure agreement and case agreement.