Using the Naive Bayes Method to Determine a Phrase Structure Tree for E-Type Anaphora with NP Deletion

Jacob Pawlak
University of Kentucky
845 Red Mile Road Apt#339
Lexington, Kentucky
jacob.pawlak@uky.edu

Abstract

This project's goal was to provide a detailed program that will . Sentences with E-Type Anaphora, or Donkey Sentences, are a linguistic obstacle for semanticists. With NP Deletion, the replaced pronoun becomes ambiguous and makes the sentence difficult to analyze semantically. An example is as follows:

No farmer who buys a donkey can sell that donkey.
*No farmer who buys a donkey can sell it. <- Here the 'it' is an ambiguous pronoun

There are a few movements at the logical level (tree) that make for a correct interpretation, but these really need to be done out on paper after an inspection (before movement).

I wrote a python script that takes in a text file with multiple sentences, and then prints out the correct bracketed expression to be used in tools like Dr. Finkle and Dr. Stump's CATS CLAW.

Keywords:

Natural Language ToolKit (NLTK), Semantics, Natural Language Processing, Python, Syntax Trees

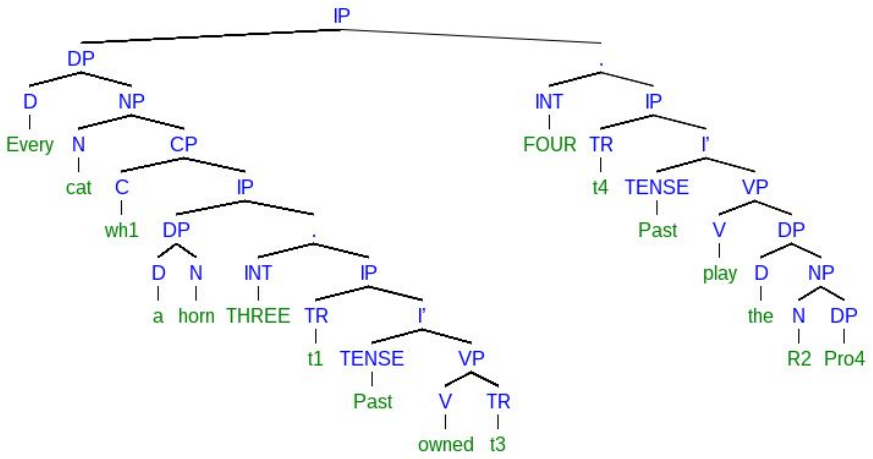
1. INTRODUCTION

Before proceeding, I will outline a few notes about E-Type Anaphora and Naive Bayes Classifiers. E-Type Anaphora is a type of ambiguous pronoun that is very difficult to analyze semantically. On top of that, there are multiple possible ways to represent them syntactically, one of which will be written about in this paper. E-Type Anaphora require lots of movement before they can be drawn in a syntax tree and be successfully evaluated semantically. One such movement is the Cooper Style Analysis, which replaces the ambiguous 'it' pronoun with a DP node - "the R_x pro_y " where the x and y subscripts are integer values representing trace indexes.

2. PROCEDURAL METHODS

The need for this project stems from the need to predict the proper movement needed to produce the correct tree at logical form. This movement also involves a Cooper style analysis of E-Type Anaphora,

changing the logical form tree even further. This project adds value to the world of Computational Linguistics. It will be one of the first online tools for computing the trees for E-Type Anaphora, and it will definitely reduce time spent using drawing tools or pen and paper. E-Type Anaphora prove to be semantically challenging for some computers. Even if the computer has control over lexical semantics, pragmatics of the sentence will often give a different result than the actual outcome. It may also be difficult for a computer to learn the discrete steps of the movement of Determiner Phrases (DP's - the syntactic label for the Cooper Style analysis of E-Type Anaphora). With the Naive Bayes algorithm, we hope to overcome this computational obstacle.



The graphical reference depicts the sentence "Every cat that owned a horn played it". This is an example of a 'donkey sentence' so nicknamed by Peter Geash, they are a product of VP deletion and Cooper's Analysis of E-Type Anaphora. The DP that is headed by 'the' and complement of an NP results from the Cooper Analysis, and the rest of the movement is needed for the Logical Form to be semantically correct for interpretation.

The algorithmic side of this project uses the Naive Bayes Classifier to predict the syntactic label needed for tree diagramming. First, I used an NLTK library called Wordnet to tokenize the sentences and provide their part of speech tag. They were then stored in a list to be used in the Naive Bayes Classifier where they were assigned the correct syntactic label for use in movement. After that I used string manipulation and some code I wrote for processing the movement by splitting the sentence into a subject and a predicate (assuming correct grammar for input sentences), and then determining the necessary movement, and then writing out a string containing the bracketed expression for use in the UKY CATS CLAW - a syntax tree builder. If you copy and paste the output of the program into such a tree builder, you can make a nice image like the one above to model your sentence. The following bracketed expression is the one needed for rendering the above picture:

[IP [DP [D Every] [NP [N cat] [CP [C wh1] [IP [DP [D a] [N horn]] [. [INT THREE] [IP [TR t1] [I' [TENSE Past] [VP [V owned] [TR t3]]]]]]]] [. [INT FOUR] [IP [TR t4] [I' [TENSE Past] [VP [V play] [DP [D the] [NP [N R2] [DP Pro4]]]]]]]]

3. CONCLUSIONS

This program is not yet perfect or done! It

still needs touching up (in the wh movement sector), and could do with some optimization for the Natural Language ToolKit library. In conclusion it was possible to manipulate simple sentences to produce the syntax trees desired for us in CATS CLAW.

4. AKNOWLEDGEMENTS

I would like to acknowledge Dr. Greg Stump of the Linguistics Department at the University of Kentucky for his help with the material and teaching the LIN509 Formal Semantics class where this topic was first introduced to me. I would also like to thank Dr. Jerzy Jaromczyk for his advice on the project, and for being a mentor and great CS315 Algorithms instructor.

5. REFERENCES

1. https://wordnet.princeton.edu/wordnet/documentation/
2. http://www.nltk.org/book/ch05.html
3. http://www.nltk.org/book/ch06.html