COMP 6751
Assignment 1A

Introduction:

For this assignment, we were to work with Cornell movie review dataset. This dataset contains 2000 files each containing a movie review. 1000 files contain positive and the other 1000 contain negative movie reviews. Each review contains at least 20 sentences. Display and analysis of data had to be done in a way that was easily comprehensible. The pipeline also had to handle the data that was not already in the dataset.

In order to do this, the code I made handles all data per sentence. When a text is entered the sentence splitter divides the data into numbered sentences and prompts the user to enter a sentence number. Once the user enters the sentence number they want to analyse, that particular sentence is then analysed.

For the purpose of this assignment Stanford's parser available at http://corenlp.run has been used as the Gold standard. The errors listed are in comparison to that parser.

Pipeline:

1. Sentence splitting:
    a. Module used- sent_tokenize()
    b. This step was pretty straightforward, the Cornell movie review dataset already came with line breaks ("\n").
    c. Made a text parser that splits strings as line breaks are encountered.
    d. Also implemented Punkt sentence tokenizer. It uses an unsupervised learning algorithm to identify where text should be split. *[1]*
    e. Finally used sent_tokenize() which itself is a pre-trained version of Punkt tokenizer. *[2]*

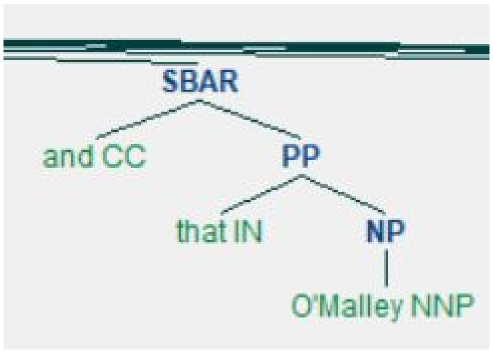
       <u>Issues:</u>
        - Splitting by looking for line breaks will only work on text that already has line breaks. We have to build a code that can parse other text as well. To counter this, I included **sent_tokenize()** method.
        - We have a large corpus available for movie review dataset so training data can be obtained from there. However, no such training data will be provided for custom text so **sent_tokenize()** would be a better alternative to Punkt tokenizer in this case. sent_tokenize() is the recommended sentence splitter included in nltk.

       <u>Errors:</u>
        - sent_tokenize() cannot detect " . . . ". In movie review dataset " . . . " was used quite often and sent_tokenize() would return each full stop as separate sentences. Tried Punkt by training with movie reviews, didn't solve the error.

2. Word tokenize:
    a. Module used- word_tokenize()
    b. Tried making a word tokenizer for this step using split() function. Because of the below-mentioned issue changed split() to split(“ ‘ “) but the results obtained were still weren’t satisfactory.
    c. Then used word_tokenizer() which is the default tokenizer present in nltk. It assumes that the text has already been segmented into sentences, e.g. using sent_tokenize(). *[3]*
    d. This did not separate by punctuation in numbers, e.g. $5.88 was not separated to [“ $ ” , “ 5 ” , “ . ” , “ 88 ”] instead it was separated as [“ $ ” , “ 5.88 ”]
    <u>Issues:</u>
        - By default split function separates using whitespace in sentences, so it cannot separate, for example, David’s car to [‘David’ , ’’s’ , ’car’] instead it separates it as [‘David’s‘ , ‘car’].
        - It separated abbreviated words like “can’t” as [“ ca ”, “ n’t ”]. Here POS tagger will not be able to identify “ ca ”. This form of tokenization works in words where individual tokens are machine recognisable e.g they’re would be separated as [“ they ” , “ ‘re ”], here POS tagger can mark they as PRP and a better sense of the sentence could be obtained later on.
    <u>Errors:</u>
        - Tried customising split by using split(“ ‘ ”), this, however, gives [‘David’ , ‘s’ , ‘car’] as output. We can observe that the output is missing an apostrophe with s.

3. Part of speech tagging
    a. Module used- nltk.pos_tag()
    b. This is the tagger recommended by NLTK to tag a given list of tokens. It uses the Penn treebank tagset by default which is used by corenlp as well *[5].* It contains several tags to cover all encountered variety of tokens.
    <u>Issues:</u>
        - When input proper nouns with lowercase initials the tagger fails. All text in movie review dataset is given in lowercase, that includes proper nouns as well so pos_tag() failed in this instance. E.g when entered a name “paxton” the result obtained would be NN which denotes singular noun where NNP was expected since the token is a proper noun.
        - Similar to the previous error pos_tag() gives different tags to i and I. It tags i as NN and I as PRP. Stanford’s corenlp also fails in with this issue. It marks i as LS and I as PRP.
    <u>Errors:</u>
        - It does not consider the context of token usage. E.g. in a movie review the line says “ joe is too big for that cage ”, here the tagger correctly tags cage as NN (noun). However, in line “ cage that beast ! ” cage acts as a verb, here the tagger incorrectly tags cage as NN (noun).

4. Parsing
    a. Module used- chunk = nltk.RegexParser(); chunk.parse()
    b. Used chunking to parse sentences and make a comprehensible tree-like structure. Defined rules based on observations and resulting parser gives results very close to the ones obtained with corenlp.
    Issues:
        - Same rules don't work for different sentences. Since a sentence structure is not fixed the rules do not hold for many sentences and an erroneous tree structure is obtained.
        - Dividing sentences because of connecting conjunctions was an issue. This was corrected by taking inspiration from corenlp constituency parse and defining a new rule for a sentence within the bigger sentence structure. E.g. in one of the test cases the sentence was joined by " and " conjunction. This was separated as-

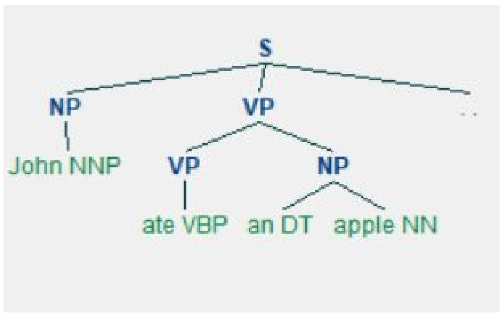


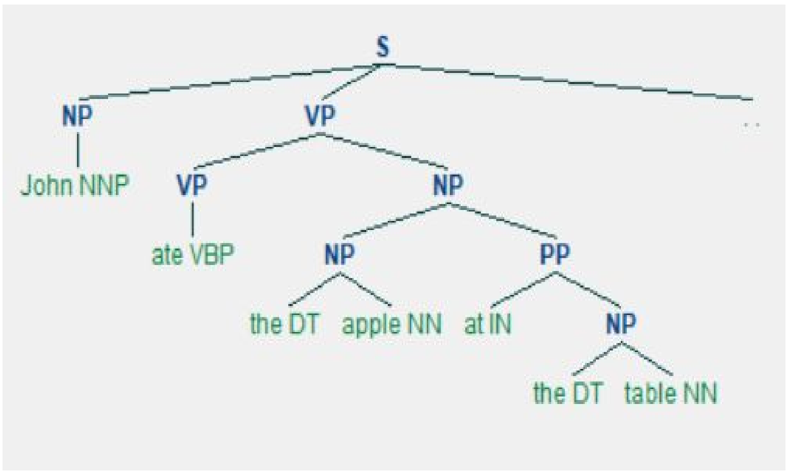
    Errors:
        - Python for some reason could not detect the PRP$ tag. Because of this, the chunking for NP (noun phrase) was not successful if a token with the PRP$ tag was encountered. It can identify PRP (Personal pronoun), just not PRP$ (possessive pronoun)

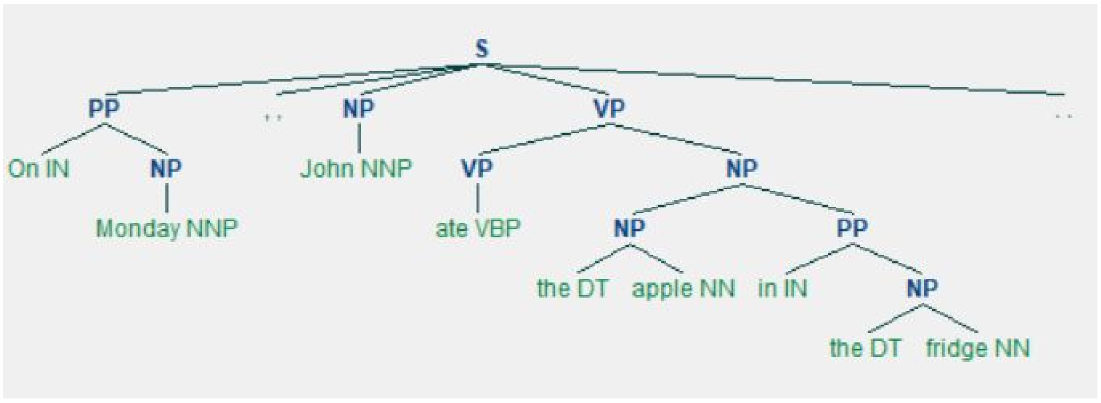
5. Name entity recognition
    a. Module used- nltk.ne_chunk()
    b. This function requires a list of tagged tokens.
    c. Here the attributes associated with particular tokens were analysed. It was observed that tokens with POS tag NNP were considered.
    Issues:
        - Since this is dependent on POS tagger to tag tokens with NNP, when the POS tagger failed (because of the issue mentioned) ne_chunk() also failed.
    Errors:
        - In some instances, it was not able to identify PERSON attribute when just the first name was mentioned. However when the text was changed to the full name the function was able to tag tokens present in the name.
        - It could not chunk together a first name and last name

Validation text-

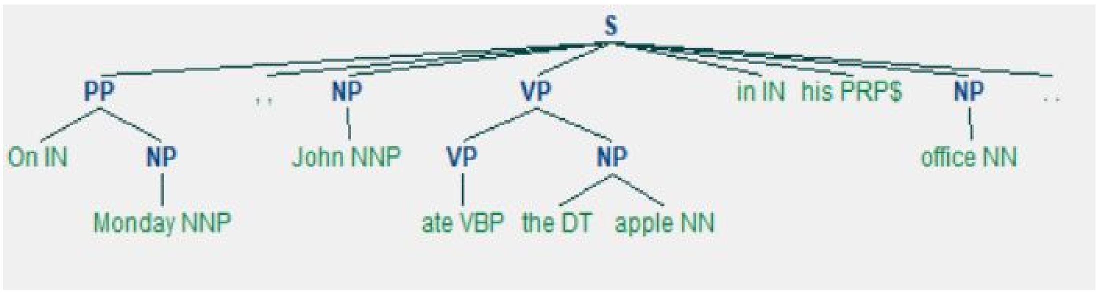
Following are the observations for various validation text provided in the assignment:

1. 

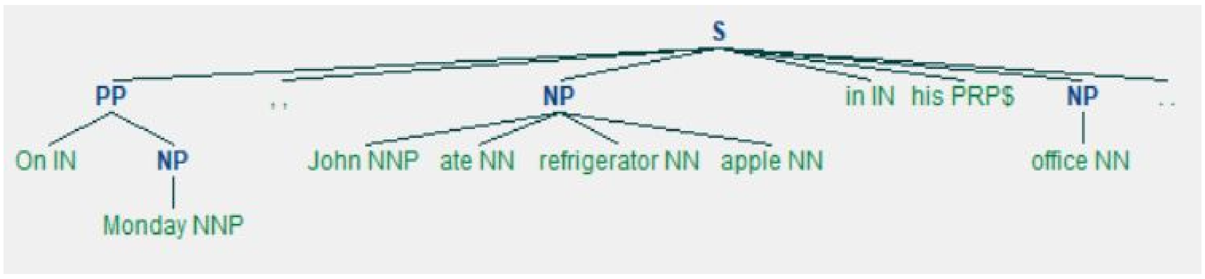
2. 

3. 

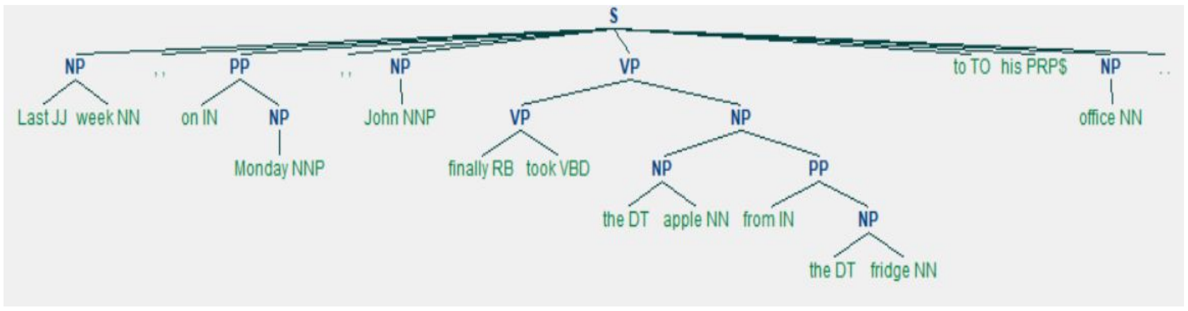
- The first 3 sentences had a simple structure and were easily parsed. The pipeline was able to identify entities present in the sentence, i.e John.
- Monday was not annotated in the third sentence as an entity.

4. 

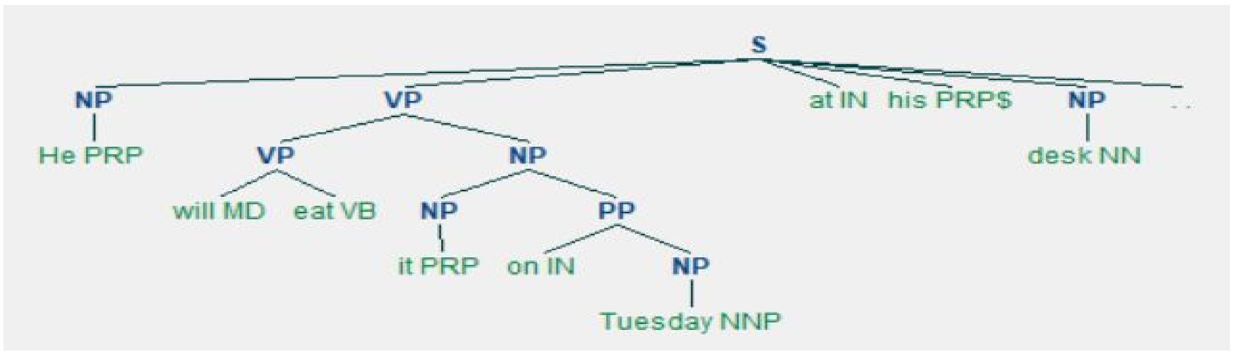
- This is where the first error was encountered. This sentence contained a token with PRP$ tag. As mentioned in Parser section above, this code was not identifying PRP$.

5. 

- Here, the pos tagger failed and incorrectly annotated “ate” as a noun. Because of this, the tree does not have VP (verb phrase)
- If the tagger correctly tagged ate as a verb code would’ve chunked a VP

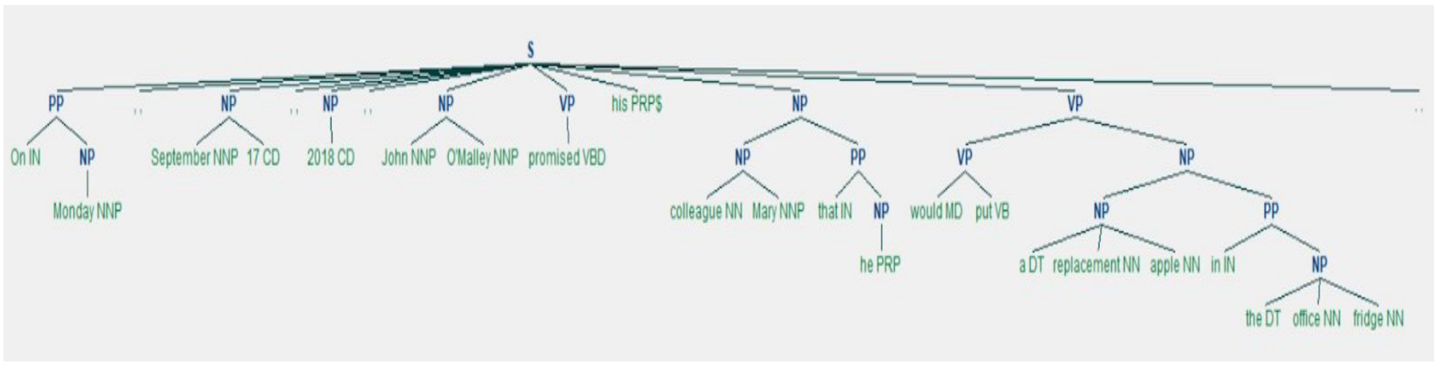
6. 

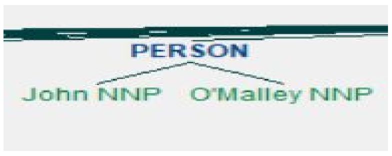
- Again in this example, PRP$ was not recognised which is why the tree is skewed towards the end.
- Unlike corenlp, name entity recogniser in this pipeline did not annotate Monday as DATE.

7. 

- This text had multiple sentences. The code split the text into multiple sentences and here’s the parsed tree for the second sentence.
- In this example the code was able to chunk PRP (personal pronoun) and not PRP$ (possessive pronoun). This is inline with our previous observations and is a consistent error.

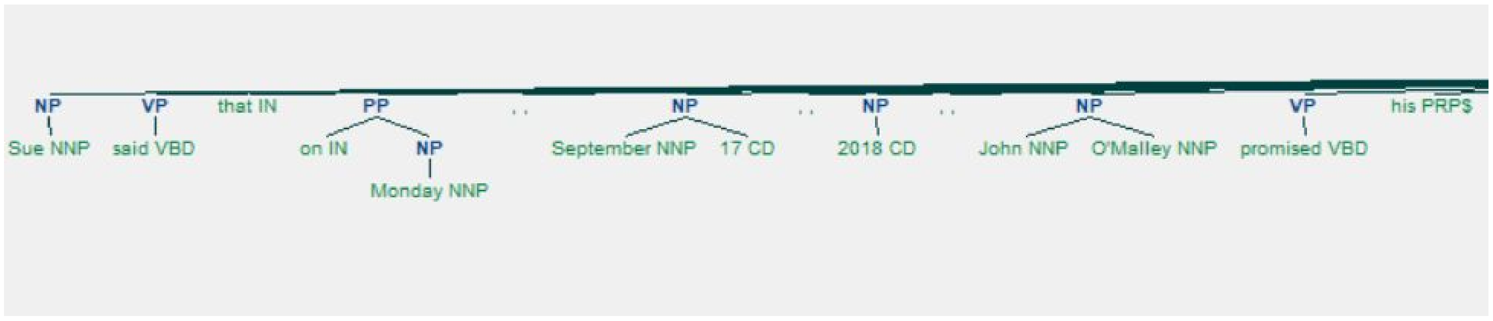
8.

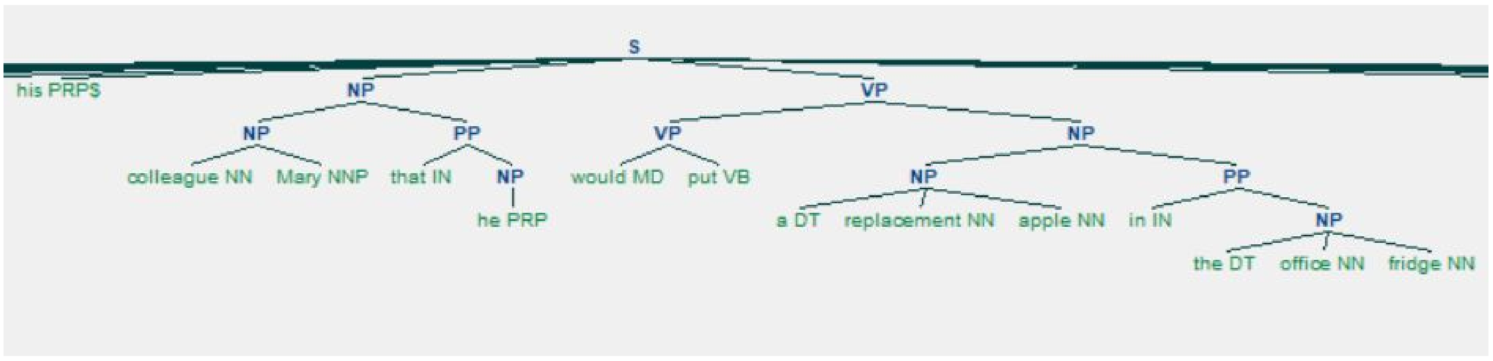


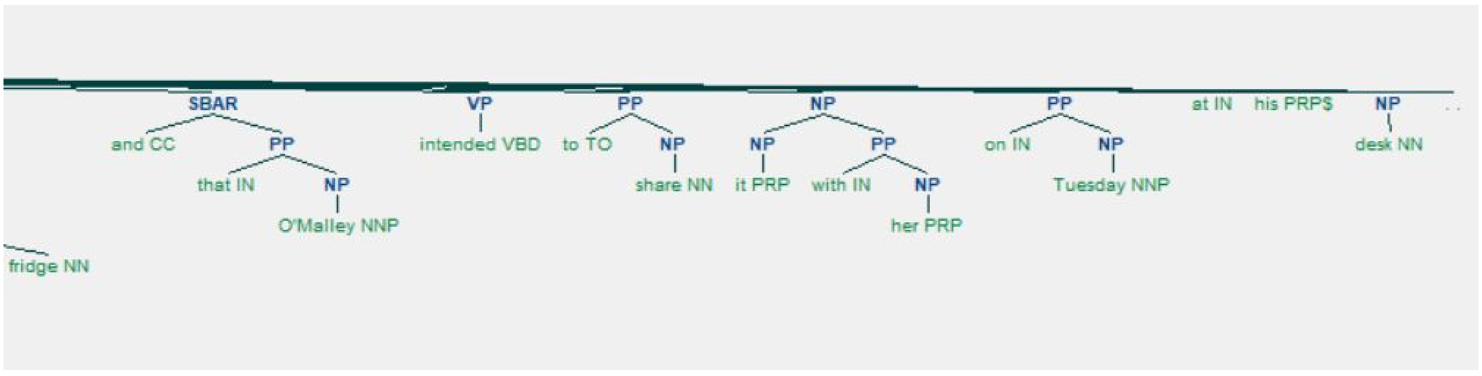


- NER fails to recognise dates (September 17, 2018) in this sentence.
- NER did chunk John and O'Malley which are first and last names of a one person under one PERSON annotation.

9.







- Here we can observe the code was able to break the sentence in two parts when the conjunction was encountered.

- This was a very long sentence with 44 tokens which is why the tree obtained is wide. It was expected that the structure contains deeper layers but the code formed a tree just 5 layers deep.
- The POS tagger incorrectly annotates “ share ” as NN.
- In the context of this sentence, the word “replacement” is used as an adjective for the noun “apple”, however, the pos tagger ignores the context and incorrectly annotates “replacement” as NN

Use-case for this sentence structure:

- This data model is at a primitive stage compared to some of the deep learning models available today. The chunking rules are not as flexible as there are exceptions in sentence structure in the English language.
- NER can be used to extract specific information from a sentence, e.g. in line 3 of the above text if we were given this sentence and asked: “Who ate the apple?”, the answer could be generated by the pipeline since John is the only one token with PERSON annotation mentioned, we can say that John ate the apple.
- The resulting tree structure we obtain from this pipeline is based on constituency-based. After some preprocessing we can obtain a dependency chart that can be further expanded to form dependency treebanks as done in *[6],* the treebank thus obtained can be used dependency-syntactic relationship. The algorithms used for the above-stated conversion are available at *[7]*
- We can create an advanced sentence splitter that separates sentences at conjunction points. This will enable us to extract information from broken down smaller sentences which in turn will take less semantic analysis. The code separates, as shown in the third screenshot of sentence 9, the sentence is split using SBAR tag.

The semantics of a sentence :

- Semantic role labelling has obvious applications for template-filling tasks such as information extraction and has also been applied in question answering, entailment recognition, summarization and text categorization. *[8]*
- It has been shown that dependency representations can serve as the input for semantic role labellers and achieve good results. This is important theoretically since it makes the syntactic-semantic interface conceptually simpler and more intuitive but also has practical significance since there are languages for which constituent annotation is infeasible.*[9]*
- Context clues are important to get the semantics of a sentence e.g.-
    1) Mary went to a fair.
    2) Lady justice is fair.

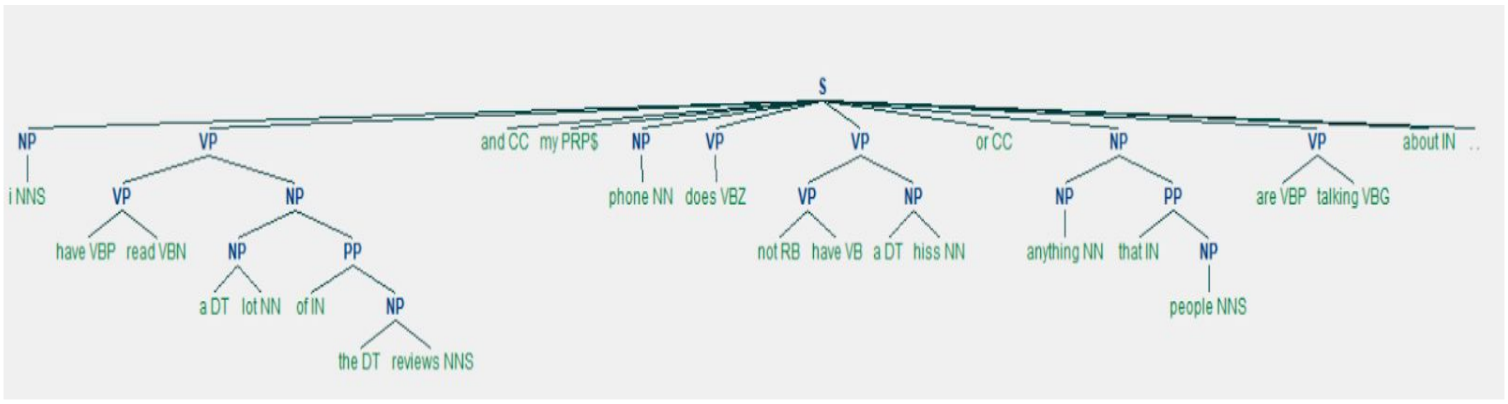
  In both sentences meaning of the word fair is different. In the first sentence, fair refers to a place whereas in the second sentence fair refers to a quality. This ambiguity in language can be removed by context clues which can help determine the semantics of the sentence. *[10]*
- Punctuations play an important part determining semantics of a sentence. For e.g. “the panda eats, shoots and leaves”, the sentence means that panda ate shoots (noun) not that he shoots (verb) and leaves but the misplacement of the comma makes the sentence semantically incorrect.

Challenge data-

Findings:

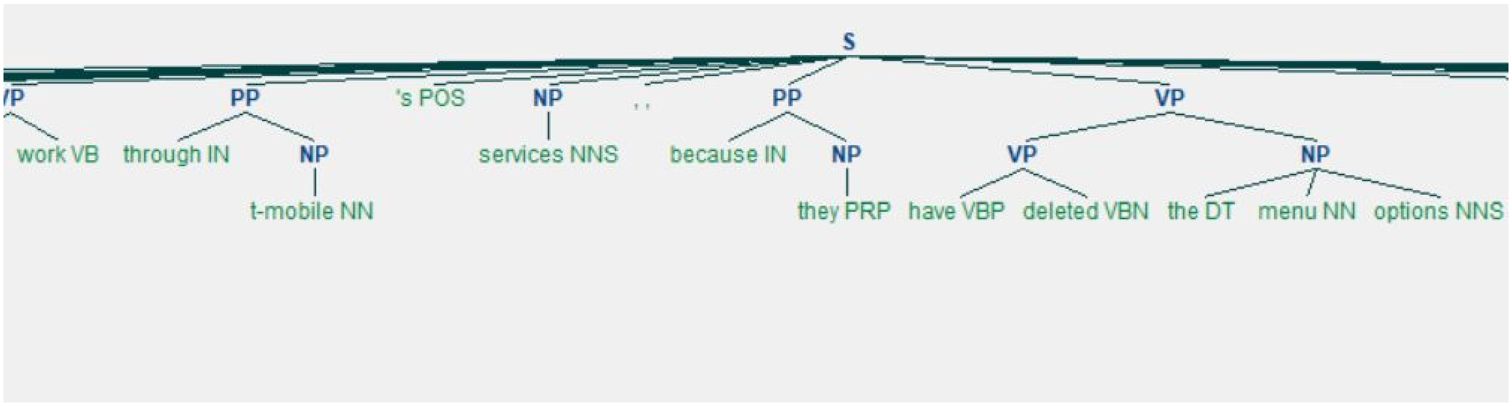
Review 1-

- The first review worked as expected. PRP$ was not recognised because of which it wasn't included in NP.
- "i" in this sentence is tagged as NNS which represents Noun, plural. We know that "i" is not plural so the POS tagger failed in this instance.
- Similar to both errors reported in Pipeline as well.

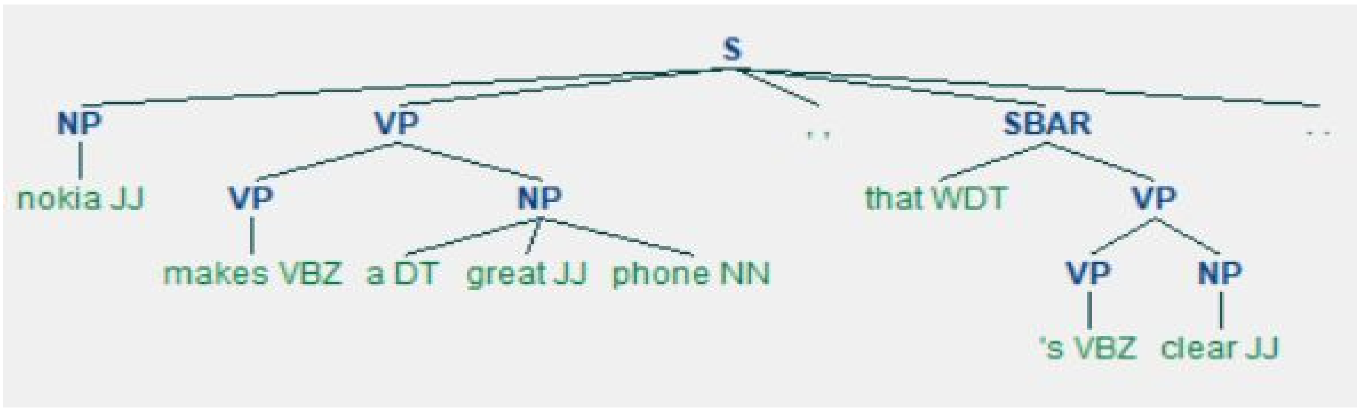


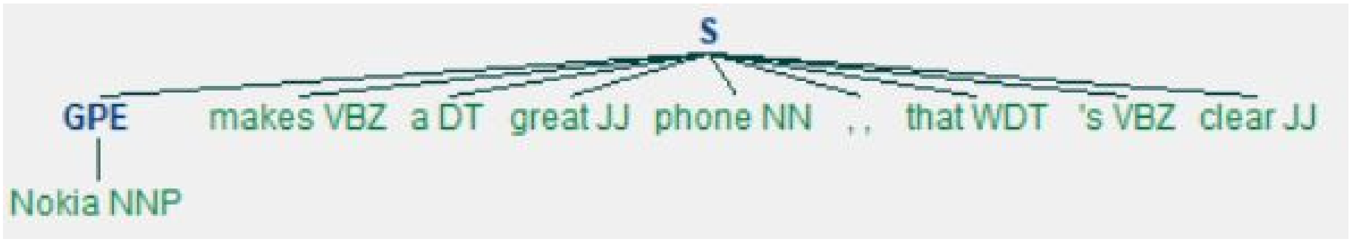
Review 2-

- Sentences in this review had generally complex language structure because of which the tree is not as layered as it should be.
- In this case, PRP is recognised which is why it is chunked as NP, we could look for a way to rename PRP$ so that it is chunked with NP as well.
- Here the code fails to recognise t - mobile as a named entity. This could be because the initials of t-mobile were not in uppercase or nltk does not attribute t-mobile as an organisation. To clarify I entered the text by changing the case of t-mobile but the code still didn't work. That would prove that nltk does not have ORGANISATION attribute associated with T-Mobile

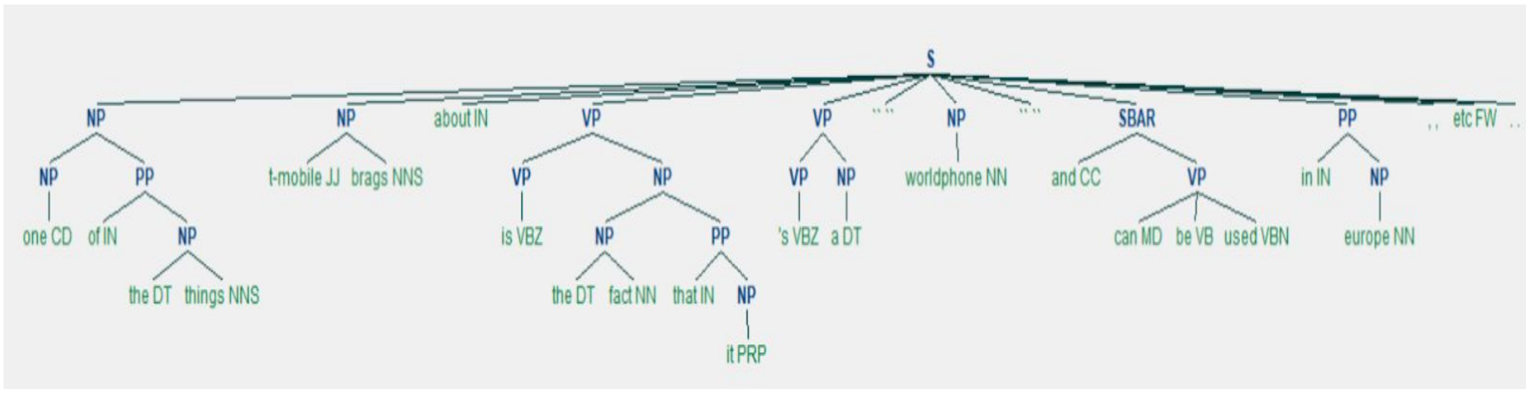


- In 2nd sentence of review 2 “nokia” is annotated as an adjective which is wrong. This is because the initial is in lowercase that the POS tagger fails to mark it as NNP. When the same sentence was entered with uppercase “N” the POS tagger correctly annotate as NNP, however, now ne_chunk() recognised as “Nokia” as GPE (Geopolitical Entity) and not ORGANIZATION.

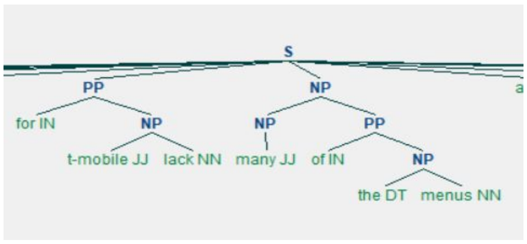




- In the following example, the code fails to chunk “ ‘ ” as part of NP. In future iterations, NP can include a rule to include quotation marks as well so they don’t form a separate branch in this tree.



- Here the POS tagger incorrectly marks menus as NN (Noun, singular) where in fact it is NNS (Noun, plural)



Comparison to validation text:

- Most of the faults observed in the challenge data were already observed in validation text or movie review dataset.
- Not including PRP$ is an error which was seen in all tested datasets.
- Failure to recognize some named entities.
- Failure to annotate proper nouns as NNP when their initial character is in lowercase was also consistent throughout.
- Some places where context changes the part of speech of a particular token pos tagger fails and annotates incorrectly.

References:

1. dl.acm.org/citation.cfm?id=1245122
2. https://stackoverflow.com/questions/35275001/use-of-punktsentencetokenizer-in-nltk
3. https://www.nltk.org/_modules/nltk/tokenize/treebank.html#TreebankWordTokenizer
4. https://www.nltk.org/_modules/nltk/tag.html#pos_tag
5. https://nlp.stanford.edu/software/pos-tagger-faq.html#tagset
6. Chapter 4 of the following dissertation https://pdfs.semanticscholar.org/8998/02b7a5795f1696c67d7c653b8b8ce0e6974c.pdf
7. Algorithm 4.1 - 4.7 https://pdfs.semanticscholar.org/8998/02b7a5795f1696c67d7c653b8b8ce0e6974c.pdf
8. Introduction of the following dissertation https://pdfs.semanticscholar.org/8998/02b7a5795f1696c67d7c653b8b8ce0e6974c.pdf
9. Abstract of the following dissertation https://pdfs.semanticscholar.org/8998/02b7a5795f1696c67d7c653b8b8ce0e6974c.pdf
10. https://study.com/academy/lesson/using-syntactic-semantic-context-clues-to-determine-meaning.html
11. http://www.nltk.org/book/

Code: https://github.com/a-sid1996/COMP-6751