REPORT

PROJECT 3

PREPARED FOR

Dr. Sabine Bergler

PREPARED BY

Karandeep

40104845

TABLE OF CONTENTS

Introduction:

About NLTK - NLTK aka the Natural Language Toolkit, is a suite of open source Python modules, data sets, and tutorials supporting research and development in Natural Language Processing. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

Purpose:

Hidden information often lies deep within the boundaries of what we can perceive with our eyes and our ears. Some look to data for that purpose, and most of the time, data can tell us more than we thought was imaginable. But sometimes data might not be clear cut enough to perform any sort of analytics. Language, tone, and sentence structure can explain a lot about how people are feeling, and can even be used to predict how people might feel about similar topics using a combination of the Natural Language Toolkit, a Python library used for analyzing text, and machine learning.

For this project I was to develop a Feature grammar to parse given sentences and judge the sentiment behind the sentence. The console has options to run **Sentiment Analyzer** either on a good inbuilt data or bad inbuilt data sentences to parse. Analysis of data has been done in a way that is easily comprehensible. Overall goal of the project was to create a parser and a FCFG which filters data by using NLTK access commands and creates sentence trees and visualize them in a more informative way. NLTK is not perfect and lacks in some areas which is being explored below.

Getting Started:

1. Open CMD (for Windows), Terminal (for MacOS).
2. Goto directory of the script.
3. Type in command “python3 sentiment.py”

Design Justification:

Why did I make the grammar do what it does ?

What ?

1. The grammar defined is a feature grammar built on NLTK feature grammar.fcfg.
    - **Grammar.fcfg** is a FCFG file where grammar is defined. Sentiment.py reads grammar.fcfg and parses sentences from it.
2. Grammar has been modified, 5 new Non-terminals are added:

SPR	DT	PRP	ADJ[SNT= POS]	ADJ[SNT= NEG]

Why ?

To add a feature to the adjective, all the adjectives both **Positive** and **Negative** from the **opinion lexicon** and **Sentence Polarity** have been added to judge the **sentiment** of the sentence.

DATA SET:

Opinion Lexicon

Positive opinion lexicon words are used to populate the ADJ with feature **'SNT=POS'**

Negative opinion lexicon words are used to populate the ADJ with feature **'SNT=NEG**'

Sentence Polarity

- Two sentences are picked from Sentence polarity dataset to add rules to feature grammar.
- To accommodate these adjectives additional rules are added so to parse the sentence successfully.

Conjunction

AND, OR, BUT

- The grammar takes into account the **conjunctions** , if the conjunction is 'and' then the sentiment is calculated on the basis of total pos or neg tags.
- If the conjunction is 'but' then more consideration is given to the tags after 'but'.

Why did you make it not do what it doesn't?

Negations: The grammar does not consider negation sentences. If a sentence uses negation it would be incorrectly tagged.

Why - Adding logic to identify negations is a vast process and goes beyond the scope of the project. Although this can be used as an enhancement in Project 4.

Oxymorons: The grammar does not consider sentences that contain oxymorons like '**pretty reckless'.**

Why - This is scheduled to be an iteration for the 4th project to enhance the grammar's coverage.

Design Critique:

The grammar does not include the following points that should be considered as important:

Negations: Sentences with negations are not analysed correctly by the grammar and hence they are not annotated in a correct manner.

Eg: It's not that bad.

Oxymorons: The grammar does not identify oxymorons and instead gives the words one positive and one negative tag.

Eg: It's pretty disgusting.

Conjunctions: Some sentences containing conjunctions like 'but' might not always be correctly identified.

Eg: He was supposed to arrive at 11:am but he came at 10:30am.

Sentences above all represent a sentiment to them, whether it's positive, negative or neutral. But This sentiment analyzer tags them as the wrong category.

Semantics Handled

The grammar can include sentences in the following formats:

Ideas:

1. Coverage of more simple declarative sentences.

Work:

Additional words from more range will be added in the grammar file, the logic in sentiment.py to parse it remains the same.

2. Sentences with conjunctions like 'that', 'but', 'and', 'or'.

Work:

Current logic identifies the position of a conjunction and takes only the sentiment after the conjunction, further it can be improved by taking in consideration all the adjectives in the sentence and adding more conditions into consideration.

3. Sentences with negation are also correctly identified but they are not annotated correctly.

Work:

Conditions to detect the presence of negations along with the sentence structure can be added to detect the actual sentiment of the sentence.

4. To improve the grammar furthermore words need to be added in nouns, verbs and adjectives.

Work:

Good nouns and bad nouns like, “Hazard” can be added to Grammar to take into consideration the real sentiment of the sentence.

References:

- https://github.com/nltk/nltk_teach/blob/master/examples/grammars/book_grammars/feat1.fcfg2.
- https://www.nltk.org/book/ch09.html