Basic Sentiment Analysis using Python NLTK

Shantanu Patil – National Institute of Technology, Delhi FFE SID 15754
Mail: shantanu.patil@nitdelhi.ac.in

Abstract

This project is a sincere effort implement sentiment evaluation of raw text in most basic manner. The project module is in alpha version and requires lot of updates. Currently the module processes a TXT file and generates a composite score of its positivity. Sincere thanks to FFE, Team Xcelerator for giving me this opportunity to work on this project. Special thanks to Mrs. Nirupama Mane for mentoring this project. This project also could not have been possible without the open source support of python.org.

Introduction

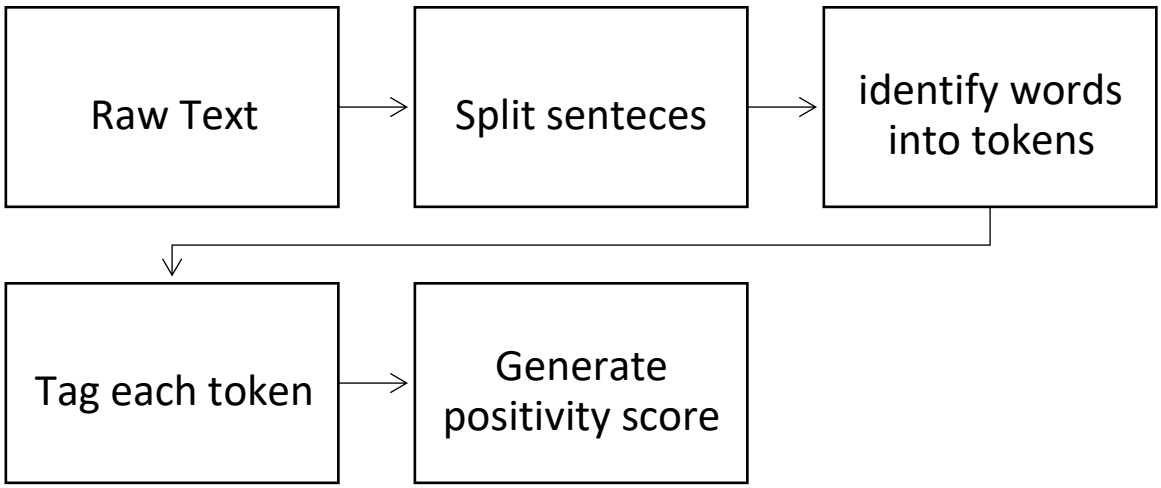
Sentiment analysis (also known as **opinion mining**) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis is widely applied to reviews and social media for a variety of applications, ranging from marketing to customer service.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation affective state (that is to say, the emotional state of the author when writing), or the intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).

Requirements

- Python 2.7.x or python 3.2.x or higher
- NLTK 3.5.x with complete nltk_data
- Python YAML library
- TXT file dataset (sample review data)

Design



Implementation

Software flow:

1. File input read: First the program opens the text file (sample review) and reads it to buffer and awaits processing.
2. Class Splitter: Accepts the text buffer and yield a list of independent strings as output. input format: a paragraph of text. output format: a list of lists of words. e.g.: [['this', 'is', 'a', 'sentence'], ['this', 'is', 'another', 'one']]
3. Class POSTagger: Tags specific parts of speech and identifies them as a specific group of words. input format: list of lists of words e.g.: [['this', 'is', 'a', 'sentence'], ['this', 'is', 'another', 'one']] output format: list of lists of tagged tokens. Each tagged tokens has a form, a lemma, and a list of tags. e.g: [[('this', 'this', ['DT']), ('is', 'be', ['VB']), ('a', 'a', ['DT']), ('sentence', 'sentence', ['NN'])], [('this', 'this', ['DT']), ('is', 'be', ['VB']), ('another', 'another', ['DT']), ('one', 'one', ['CARD'])]]
4. Class DictionaryTagger: The postagger may tag one lexical word with many tags the dictionary tagger solves this anomaly with only one tag for each word. This is determined by two priority rules – longest matches have higher priority and searching is done from left to right.
5. Function sentence_score: Generate a score (points) for each text review on basis of tags. (See Source Code for more info)
6. Function main: Displays the positive points for each review. Positive score signifies positive review. A near zero score is neutral while a negative score asserts a negative review.

Conclusion

The module successfully achieves to determine whether a given paragraph asserts optimistic or pessimistic views about any topic by identifying list of positive and negative words as parts of speech and implement a lemma on them to extract a score.

Challenges

The named entity identification could not be implemented as it turned out to be highly vague and inaccurate in generic reviews. Dictionary tagger sometimes identifies more positive or negative words than observed in the text. (Dangling tags). SentiWord was considered for analysis but later turned down to simplify the code.

Future scope

Text can be directly accessed from html pages through web crawlers/scraping. The lemma can be refined and Dictionary Tagger can be perfected. UI can be implemented for the module. A deep learning algorithm can be defined so that the module is perfected along with processing of each review.

References

- Fang and Zhan *Journal of Big Data* (2015) 2:5
- Chapters from http://nltk.org/book/
- TextBlob documentation