Technical Review - Review of applications for NLTK

Introduction

NLTK is an open source tool for developing Python programs for operating with human language data. It is a suite of open source program modules, tutorials and problem sets for presenting prepared computational linguistics courseware. It incorporates symbolic and statistical Natural Language Processing. The goal of this paper is to give a brief review of three common information extraction processing strategies with this NLTK[1].

Sentiment Analysis

One of the most common uses for NLTK is for text classification to determine sentiment polarity, Sentiment analysis can help us determine the ratio of positive to negative engagements about a specific topic. The analyzing results could be used to obtain insights from the audience in bodies of text, such as comments, posts, and product reviews. NLTK already has a built-in, pretrained sentiment analyzer called VADER (Valence Aware Dictionary and sEntiment Reasoner). Since it is pretrained, we can get results more quickly than with many other analyzers. However, VADER is best suited for social media language, like short sentences with some slang and abbreviations, for example twitter and instagram captions. It’s less accurate when rating longer, structured sentences[2]. Feature selection and feature engineering, can be introduced to figure out which properties of a dataset are useful in classifying each piece of data into a desired category to improve the accuracy of classification[3]. However, this method still has some limitations, which means the final accuracy is still far from perfect at some certain scenarios. For example, if there are any new names or words to VADER or if any uncommon names and words that aren’t necessarily positive or negative[4].

Tokenize Text

Tokenization means to decompose a long sentence into different meaningful little token words, which are very useful for finding patterns and are considered as a base step for stemming and lemmatization. Though we can use Python’s own string manipulation methods, NLKT has two

[1] Reference: https://www.analyticssteps.com/blogs/what-natural-language-toolkitnltk-nlp
[2] Reference: https://realpython.com/python-nltk-sentiment-analysis/#using-nltks-pre-trained-sentiment-analyzer
[3] Reference: https://realpython.com/python-nltk-sentiment-analysis/#using-nltks-pre-trained-sentiment-analyzer
[4] Reference: https://zhuanlan.zhihu.com/p/38231514

important modules for tokenization purpose specifically: word tokenization and sentence tokenization. For word tokenization, *nltk.word_tokenize()* function in the module can split raw text into individual words[5]. The output of word tokenization can be used for better text understanding in machine learning applications. It can also be inputted for further text cleaning steps such as punctuation removal, numeric character removal or stemming[6]. For sentence tokenization, it can be applied to the case when it is needed to count average words per sentence. To accomplish such a task, both NLTK word tokenizer as well as NLTK sentence tokenizer are essential to calculate the ratio, which serves as an important feature for machine training as the answer would be numeric[7].

Stem Text

Stemming is the process of producing morphological variants of a root/base word. For example, it can reduce the words “chocolates”, “chocolatey”, and “choco" to the root word, “chocolate”. Stemming algorithm is beneficial for cleaning textual data to develop a Natural Language Processing Algorithm. NLTK has a couple of stemming algorithms, such as PorterStemmer, which is the standard stemmer; SnowballStemmer, which is an improved version of PorterStemmer; RegexpStemmer, which is rule-based stemmer that focuses on regex rules[8]. Different types of stemmers within the NLTK focus on different languages, rules, or algorithmic rules. Through stemming process, the outputs are helpful to understand the role of a word within the sentence; to understand tense of the sentence and to make irregular words to similar each other for better text understanding[9]. Of course, there are drawbacks of using different stemming in NLTK in different scenarios so it is important to select an appropriate stemming algorithm accordingly. For instance, Snowball Stemmer is more aggressive than Porter Stemmer: words like ‘fairly‘ and ‘sportingly‘ were stemmed to ‘fair’ and ‘sport’ in the snowball stemmer but when you use the porter stemmer they are stemmed to ‘fairli‘ and ‘sportingli‘. Inappropriate stemming algorithms could either over or under stemming, which may lead to not so meaningful or inappropriate stems[10].

---

[5] Reference: https://www.guru99.com/tokenize-words-sentences-nltk.html
[6] Reference: https://www.guru99.com/tokenize-words-sentences-nltk.html
[7] Reference: https://www.guru99.com/tokenize-words-sentences-nltk.html
[8] Reference: https://www.holisticseo.digital/python-seo/nltk/stemming
[9] Reference: https://www.guru99.com/tokenize-words-sentences-nltk.html
[10] Reference: https://www.geeksforgeeks.org/snowball-stemmer-nlp/

Conclusion

In conclusion, features of NLTK that allow us to process text into objects that can be filtered and manipulated. Sentiment Analysis with NLTK makes text data to be analyzable to gain information about its properties. Different classifiers can perform sentiment analysis on data and gain insights about how the audience is responding to content. Tokenization with NLTK helps to divide large quantities of text into smaller parts for word and sentence tokenization purposes, which is an important feature for machine training. NLTK Stemming can be used for understanding how a search engine can use stemmed words to infer a sentence or replace some of the suffixes to extract more information.