Natural Language Processing
Fall 2018
Professor James Marting
Assignment 3 (Text classification) Report

Student: **Keval D. Shah**

Task: Implement a text classification system for sentiment analysis, given a data set with segments of positive hotel reviews and a negative hotel reviews. Once the system is implemented, take in reviews as test cases and categorize them as positive or negative.

Approach:-

1. **Dividing main training data:-**

- To keep a track of the progress with accuracy and analyze the effect of an implementation, I have divided the main sets (both positives and negatives) into two sections in the ratio 80:20, calling them training set and dev set respectively. I shall train on the entire main set when building the model for the test set. The review are stored in a list as well as in a dictionary with the review ID as its key.

2. **Parsing the words/sentences into the right format for processing:-**

- The reviews are not tokenized or sentence segmented in any way (the words are space separated). For implementing any system, it is required to separate the words from the sentence and remove the punctuations from the raw text as well. Along-with punctuations, some other regex operators such as /n, /t, /r and /x have to removed as well.
- I have used the *translate()* function in python to clean the words from punctuations.
- After this parsing, we have a list of sentences that is ready to be separated into dictionaries that maintain the count of words.

3. **A system that use the Naive-Bayes approach:-**

- To implement a system that does classification of the input using the Naive-Bayes approach, we have to maintain a model for each class: **Positives** and **Negatives**.
- We are only concerned with building a unigram language model for each class.
- For every input, we calculate the probability of the input belonging to a given class, for each class. Hence, the probabilistic models will look at the count of each word of the test input in its dictionary of training words. It will parse the relative count into the probability of the input belonging to that class. Finally, the label of the class whose model returns the highest probability will be assigned to the input.
- It is likely that the word in the test input is seen for the first time by the system. In this case, we want use add-1 smoothing to consider the contribution of unseen words.

- Coming to the programming flow, I have maintained a dictionary of words corresponding to the frequency of occurrence in the training reviews of that class.
- The accuracy for this approach on the dev set was found to be in the range: **86.48-94.59 %**.
- This accuracy is surprisingly high because of the nature of the dataset and way people wrote reviews. However, there are even better ways to make a more accurate prediction.
- The major drawback in this is the independence assumption. In case of reviews where the emphasis is on the linking/dependency between words/sentences, the Naive-Bayes fails to extract this into a good mapping.