Name: Brenda Boudaie, Jose Quillada, Francisco Vilaboa

Intro

Our group decided to test our knowledge on NLP by implementing a program using Sentiment Analysis. Sentiment Analysis is used in Natural Language Processing to systematically extract and identify subjective information. By assigning polarity to text, applications of Sentiment Analysis quantify patterns and gain a better meaning to the word or sequence. Some real world examples include: examining a person's psychological state based on their verbal behavior, examining emotional scales, or detecting polarity on reviews.

Classifying the data can be done with many methods. In our textbook, we learned different regression methods with Maximum Entropy Models. These classifiers extract a set of features from a dataset and use that information to classify some other text. There is Linear Regression, and Logistic Regression. While both examine values from features and their weights, there are slight differences between the two.

Linear Regression predicts real valued outcomes given some observation, where the features and observations follow a linear relationship. For example, a realtor will notice that the relationship between vague words in a house listing has a linear relationship with the pricing of the house — the more vague words, the lower the sale goes for. These values are computed by the dot product of all weights and features in the feature vector.

Logistic Regression predicts values based on the ratio of probabilities. Specifically, it calculates the probability of an event occurring, over the probability of it not occurring. This method uses conditional maximum likelihood to pick the best weight for the features. A good example of Logistic Regression is tagging a new word in a dictionary with a part of speech.

Our Project

For our actual program, we used Sentiment Analysis to classify binary values: positive or negative. We initially tried to extract polarity using Naive Baye's Rule: P(polarity|words) = P(words|polarity)P(polarity). We did so with the use of importing sentiment within our python script. This simple algorithm used built in functions to train a dataset and calculate probabilities with Naive Baye's. We quickly experienced a lot of issues with this method. We got completely incorrect values, and experienced the issue of scarcity very quickly.

To fix inaccurate results, we implemented another program that uses ideas of Maximum Entropy Models. The basis of our code is as follows:

- We train the data from Amazon reviews, downloaded from kaggle.com. The link included two separate sets: trained-positive text file (2.1MG) and trained-negative text file (3.5MG).
- First, we put the two training sets in our cleanUpWords function. This function uses regular expressions that take out unnecessary characters (like punctuation), and stores the split words in a list.

Name: Brenda Boudaie, Jose Quillada, Francisco Vilaboa

- Then, these lists are called in our getProbability function. This function calculates the occurrence of each word within the positive set, and the negative set. It does so by dividing the number of times a word occurs divided by the total count of words.
    - To further optimize this function, we eliminate words that are in our dictionary of neutral words (7KB). This action increases efficiency when calculating the occurrence of words, as well as total words.
- Now that all of the training data is set up, we use this information gathered to classify the user's input review. This is done by cleaning up the words in the review, and passing that list into our printAnalysis function. This function simply prints if the review was classified as positive or negative, based on the boolean function isPositive (explained in next bullet)
- isPositive compares the words in the user review to those in the positive and negative cleaned datasets. There are four if-statements assigning weights to these features:
    - if word is in positive set: probability of input being positive increments by log weight
    - if word is in negative set: probability of input being negative increments by log weight
    - if word is in positive, but not negative: the probability of input being negative is incremented by a very small log value — 0.000000000000000000001.
    - if word is in negative, but not positive: the probability of input being positive is incremented by a very small log value — 0.000000000000000000001.
- After the analysis is completed, the data is input into our countAllReviews function. A fairly simple function, calculating and outputting all of the negative and positive reviews of all previously user-inputted reviews. All reviews are stored in its own txt file, where reviews are being appended every new entry.

Other Remarks

We are really excited and happy with our project. We were intrigued to implement concepts from the class in a real world application. One aspect to highlight, is our program's inability to recognize certain complex reviews. Reviews that contain double negatives or have objective meanings tend to get classified with the wrong polarity. As a future solution, we have discussed adding bigrams and maybe even larger n-gram models to take this issue into consideration. This method would help because of the n-gram model's ability to check probabilities of multiple words (ie: 'the movie was not greatly terrible' — funky wording but you never know what type of review will be an input).

Name: Brenda Boudaie, Jose Quillada, Francisco Vilaboa

Sample Outputs:

```
...LOADING SENTIMENT ANALYSIS...

(Enter 'q' to exit)
Write your review: I loved this movie! The acting was amazing, and the plot really kept
 me excited.
review is POSITIVE
1 positive reviews out of: 2 total reviews
1 negative reviews out of: 2 total reviews

Write your review: Omg, terrible movie. Waste of my time and money!!!
review is NEGATIVE
1 positive reviews out of: 3 total reviews
2 negative reviews out of: 3 total reviews

Write your review: Wow wow wow ... Must see!
review is POSITIVE
2 positive reviews out of: 4 total reviews
2 negative reviews out of: 4 total reviews
```

storing all reviews in txt file: