Restaurants: Who Wants Some Eats?

Eriq Augustine
Department of Computer Science
Cal Poly, San Luis Obispo
eaugusti@calpoly.edu

I. Introduction

The methods discussed in this paper will only be methods that made it to the final implementation. There were many methods that did not make it and they may or may not be mentioned in this paper. If you are curious about them, I would love to talk about them and they should still be lurking in my code.

Along the way, I used both "Natural Language Processing with Python"[**?**] and "Speech and Language Processing"[4] for inspiration.

On of my themes on this project was not using external dictionaries. They are not as fun. Therfore, I avoided pre-tagged word corpera such as Sentiwordnew[**?**] and ANEW[3].

II. The Team

It is just me :.([1]

III. Predicting Paragraphs

For paragraph prediction, I ended up using the mean of two orthogonal methods.

A. Naive Bayes Classifier

The first method was a Naive Bayes classifier from the NLTK package[2]. Most of the work with the classifier went into feature selection. However, I did do something a little different with the classifier. I choose to use a Laplace Probability Distribution instead of the default ELE Probability Distribution.

B. Feature Selection

My feature selection can be divided into two parts: grams and features of the text.

1) Grams: I found that trigrams were not very informative, so I stuck with bigrams and unigrams. I choose to stem unigrams, but leave bigrams unstemmed. Experiment has shown that this works a little better than stemming or not stemming both. I used a Porter's stemmer, in the past I have found that it works well.

The gram selection takes two stages. The first stage is feeding all of the training data into my feature set generator. I then find the frequencies for all of the unigrams and bigrams. I then discard all of the grams that do not occur in at least two different paragraphs and less than 80% of the paragraphs. This allows me to get rid of local stopwords without having to rely on an external list, and git rid of somewhat unique words.

When finding the gram features for a specific document, the present features were given an "True", and the absent features were turned off and given a "False". I have found that the NLTK Native Bayes classifier does not do well when the absent features are not denoted with a false.

2) Text Features: I also tried to capture some information about the text and make them into features. I have found that the most informative text features that I found were:

- Number of Unique Words
- Mean Length of Words
- Length of Entire Document
- Number of Total Words

What all of these features are pretty obvious, but there is more to be done before putting them into the classifier. The NLTK Naive Bayes classifier does not handle numeric properties well. So in order to combat this, I rangified the featured. Therefore instead of saying that a document has 55 unique words, I say that this document has 20, 25, 30, 35, 40, 45, 50, and 55 unique words. This way, the classifier can more easily make splits.

I tried using POS tagging, and then grouping together similar tags eg. NN, NNS, NNP, and NNPS all go into the "noun" group. And then using the counts as features. However, I did not find these informative and POS tagging takes too long.

C. Word Weight

The second paragraph predicting method I used is a word sentiment based method that I call "Word Weight".

When training my Word Weight classifier, I first find them mean and median of all the scores in the training set. I then assign a score to every word that is the median of all the scores for paragraphs that the word occurs in. I ignore words that occur in less than six documents and that occur in more than 40% of documents. These values were found experimentally.

When classifying a document, I first lookup all the scores from the training for each word in the document. I then keep two running totals. One total for words with a score that is less than the mean score of the training set, and one for all words with a score greater than or equal to the mean from the training set. I go through each word and add the absolute difference between the training mean and the words score to its respective sum. If there are no words that fall into one of the categories, then I give it a value of 0.0005. Difference of the two scores over the sum of the two scores and multiply

that by 0.5 and add it to the mean from the training set. I then round that number, and that is my score for the document.

The idea behind this is that the ratio of positive to negative words will drag the prediction away from the global mean. If there is a tie, then the mean is guessed. I have found that rounding usually decreases my RMSE, so I do that.

D. Combination

I have found that my two orthogonal methods tend to cover for each other deficiencies. So to combine the results from each, I just take the mean of the two.

IV. Predicting Overall

My overall prediction scheme is highly dependent on my work for paragraph prediction. I simply take the mean of my guesses for the "food", "venue", and "service" paragraphs. I do not use my prediction for the "overall" paragraph because I feel that it would provide an unnecessary skew. Also, I feel that using a value already provided to predict the same value would most likely contribute to a negative feedback loop.

V. Predicting Author

My author prediction in very simple. I just use a Naive Bayes classifier trained with the features that I discussed earlier. I did not try to do any thing with the scores, because I fell that on this particular dataset they are fairly arbitrary and do not provide useful insight into the author's identity.

VI. Author Similarity

I did the author similarity exercise. As my similarity metric I used Jaccardian distance, which is actually dissimilarity, but I just subtracted it from 1. I concatenated the paragraphs for each review and then broke them up into the same features that I used from the author prediction. I then just took the Jaccard distance between each feature set.

Because I did the similarity comparison bassed off of review and not author, the matrix is too large to be placed into this paper. It is placed inside the included "matrix.txt" file.

The closest similarity I have is 0.278 and that is with Ryan Verdon and test08. After investigating, I found that test08 is Ryan reviewing Denny's while the one it is similar to is Ryan reviewing IHOP. It makes sense that Ryan reviewing two similar places yields a similar score. He even mentions the other place in each respective review. Ryan also had a 0.146 similarity to another one of his reviews.

Jacob Muir also had a 0.142 similarity with himself.

However, for the most part there is not a large correlation between the Jaccardian distance and the author.

VII. Conclusions

In the end, I don't think that this is a very informative dataset. It was small and the documents were short, a deadly combination. I think what could have helped most was going over some reviews more as a class. Making sure that everybody conforms to the format and scales. For example, I know some people who thought that it was a four point scale; and there we some people who placed the paragraphs out of order. In a small dataset such as this, that could have a large impact.

References

[1] Augustine, E., Cushing, C., Dekhtyar, A., McEntee, K., Paterson, K., and Tognetti, M. Outage detection via real-time social stream analysis: leveraging the power of online complaints. In *Proceedings of the 21st international conference companion on World Wide Web* (New York, NY, USA, 2012), WWW '12 Companion, ACM, pp. 13–22.

[2] Bird, S. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions* (Stroudsburg, PA, USA, 2006), COLING-ACL '06, Association for Computational Linguistics, pp. 69–72.

[3] Bradley, M. M., and Lang, P. J. Affective norms for English words (ANEW): Stimuli, instruction manual, and affective ratings. Tech. rep., Center for Research in Psychophysiology, University of Florida, Gainesville, Florida, 1999.

[4] Jurafsky, D., and Martin, J. H. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*, 2 ed. Prentice Hall, 2008.