AMS 561 Project Report
How would you rate it: the Kindle project

Team Members

- **Devoja Ganguli** (ganguli.devoja@stonybrook.edu)
  - M.A. Student, Department of Economics
- **Nazila Shafiei** (nazila.shafiei@stonybrook.edu)
  - Ph.D. Student, Department of Linguistics
- **Suji Yang** (suji.yang@stonybrook.edu)
  - M.A. Student, Department of Linguistics

1 Introduction

With the exponential growth of social media, the users tend to rely on the reviews and the content of various website in decision making. However, the amount of opinion text in these websites is too vast that it has made finding and identifying relevant information for an average human reader difficult. Automated sentiment analysis systems are useful tools to help categorize the data into positive and negative sentiments. Sentiment analysis or *opnion mining* is also a popular research topic in the field of NLP and Liguistics (Liu 2012).

The goal of this project is to determine the sentiment of user reviews classifying them as positive or negative. To do so, we use sentence level sentiment analysis on a set of reviews on Kindle books, taken from McAuley.

2 Techniques and Tools

The main tool that was used is the scikit-learn library of Python for supervised machine learning. In addition, we also needed to use other libraries for cleaning, tokenizing, and vectorizing the data, which are explained in the following sections. In the following sections, we explain the steps we took to complete the model.

2.1 Data

To begin with, we needed to have a pre-classified dataset that can be used as an input for machine learning tools. We chose Kindle book reviews on Amazon as our dataset. The data we use are taken from McAuley, with the product reviews spanning May 1996 - July 2014. The dataset contains not only reviews, but also ratings and helpfulness votes. For our purposes, we use the reviews since we want to analyze the sentiment of such sentences and we train our model to determine the rating of reviews based on the sentiment of the reviews. In other words, we look at the correlation between the written text and the rating that a user who has written such a text might provide.

2.2 Cleaning the Data

The data is the json.gz format. Therefore, we used the `gzip` library to read the file in the form of a **DataFrame**. The next step is to clean the data. Cleaning basically means getting rid of unnecessary data and keeping what matters. Dealing with dataframe requires using `pandas` library.

Our dataset includes information about the user ID, reviewer name, helpfullness vote, review time, unix review time, review text, and overall rating. Among these, only the review text and rating is what matter to us. So, we cleaned our dataframe of the unnecessary information and kept the columns needed.

Since we prefer to use binary clustering, to correspond to our positive versus negative comments, we need to assign a threshold to the overall ratings and put them into two classes of positive and negative reviews. To do so, we used number 3 as our cut-off point, meaning that we consider ratings including and below 3 as negative and ratings of 4 and 5 as positive. To be able to feed this into the model, we assign number 1 to positive reviews and number 0 to negative values. In other words, we replace the ratings of 4 and 5 by number 1 and ratings of 1, 2, 3 with number 0. This would allow us to use binary classification of the data.

2.3 Tokenizing and Splitting

Before tokenizing, we need to make sure that the data is clear of punctuation and special characters. This data can now be tokenized. Tokenizing is done on the review text and the end product of it is a list of lists. Each sublist corresponds to one review, and the superlist is a collection of all the reviews.

2.4 Vectorization

For the data to be accessible by the algorithm, the data needs to be in numeric form (Bengfort, Bilbro & Ojeda 2018). This means that we need to convert our text format to numeric format. This is done via vectorization, which allows us to represent words and sentences with an array; and is done in two steps, word embedding and sentence embedding.

To perform vectorization, we use the `gensim` library and also Google's pre-trained Word2Vec model. This model represents words with 300 features, meaning that each word is an array of 300 elements. Using this model, we convert all our tokenized data into vectors. A similar step is required to vectorize sentences and in turn, each review. This would allow us to have a fixed size for all our reviews, i.e. 300, and would make the comparison of different reviews much easier.

3 Training and Testing

4 Conclusion

Selected References

1. Bengfort, Benjamin, Rebecca Bilbro, and Tony Ojeda. 2018. *Applied Text Analysis with Python: Enabling Language-aware Data Products with Machine Learning*. O'Reilly Media, Inc.

2. He, Ruining, and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web, pp. 507-517. International World Wide Web Conferences Steering Committee*

3. Liu, Bing. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies 5 (1)* : 1-167.

4. McAuley, Julian. Amazon product data. http://jmcauley.ucsd.edu/data/amazon/index.html