Project Kindle

Devoja Ganguli: M.A. Student, Department of Economics
Nazila Shafiei: Ph.D. Student, Department of Linguistics
Suji Yang: M.A. Student, Department of Linguistics

May 5, 2019

1 Introduction

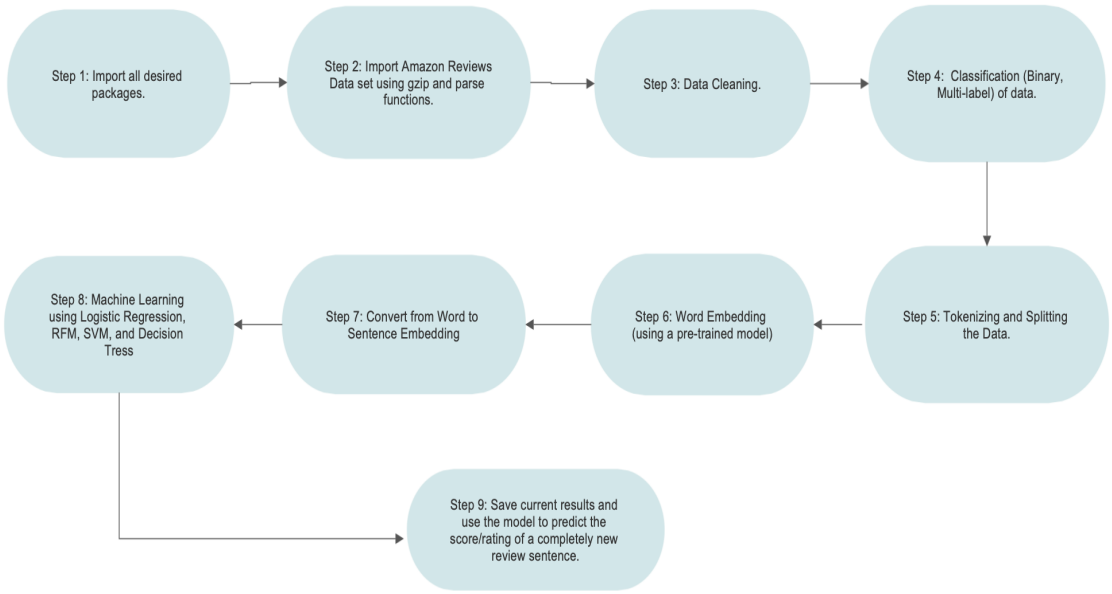
Increasing dependence of mankind on the internet has users relying on product reviews to decide whether or not to purchase a product. While product reviews are important to users who would like to make the decision to purchase, it is also useful to the company selling it. Due to the vast amount of opinion text available on websites, it is impossible for anyone to come to a conclusive opinion about the product by simply reading through the information available. To overcome this hurdle, one can use 'Sentiment Analysis', which is a useful tool for categorizing the data (review text) into positive and negative sentiments. Sentiment analysis or *opinion mining* is currently a popular research topic in the field of Natural Language Processing (NLP) and Linguistics (Liu 2012).

In this project we use Amazon Kindle Reviews (McAuley, UCSD) to perform sentiment analysis and build a model that predicts the reception of the e-book amongst its readers. This kind of a model could in the future be used by Amazon to decide which e-books they should continue to list and which ones they can take off the website. At a broader level, the model could also be used to curate a list of suggestions to a user based on his/her reviews of past preferences and purchases.

2 Overview

The main library used in this project is `scikit-learn` for supervised machine learning. Other libraries used include: `gzip`, `json`, `Numpy`, `Pandas`, `Scipy`, `nltk`, and `gensim`. The original Amazon Reviews data file was available in a zipped file format which required the use of library `gzip` for extraction. Once imported, the file was converted into a data frame to allow for easier analysis of data. The library `json` was also used in the process of importing the data into the python script. `Numpy` and `Pandas` were used throughout the code to perform various operations on the data frame,

e.g. to create empty vectors to store modified variables and to do actual data analysis. The library `nltk` helped with filtering stop-words from the text reviews to achieve a higher accuracy level of the model. `Gensim` library provided a pre-defined `word2vec` model that we implemented in our code to obtain word embeddings for the reviews.



Our entire project can be easily mapped out as a flowchart (included above). The subsequent sections describe in detail each of the steps in the flowchart.

3 Data

We performed our sentiment analysis on Amazon Kindle e-book reviews. The complete dataset is available for free by Dr. McAuley (UCSD), and includes 142.8 million Amazon product reviews spanning between May 1996 and July 2014. The dataset contains not only reviews, but also ratings and helpfulness votes. For the purpose of this study, we use only a small subset of the data, focusing only on those reviews pertaining to the Kindle e-book. We also concentrate only on the product code, the actual review and the 'overall' rating of the e-book. Other product specific characteristics like the 'helpfulness vote', 'reviewer ID' , 'summary' , and timestamps were discarded to create less confusion during the analysis. As already mentioned, the data was available in a zipped format which required the use of the `gzip` library and parse functions, following which we converted the file into a data frame. After the data was converted into a data frame, we proceeded to step 3, Data Cleaning.

Cleaning of data basically implies getting rid of unnecessary data and keeping what matters. The `pandas` library came handy here. After removing all the undesirable columns from our data frame, we stripped the text reviews of any punctuation marks. All uppercase letters were converted to lowercase letters, white spaces and empty lines were removed. A filter for stop words was also applied to remove generic, reoccurring words like 'i', 'me', 'her', 'so', 'do', etc. Removal of stop-words is important because they add little value to the model, and often create a dimensionality problem in the model.

4 Classification

The dataset already had a classification of ratings based on the text review. However, we decided to further classify the reviews in a binary fashion to create our own 'score' column to avoid the dimensionality problem and reduce any confusion in the model. To do so, we used number 3 as a cut-off point. Ratings 3 and below were classified as negative (bad) and ratings 4 and 5 as positive (good). To be able to feed this into the model, we assign number 1 to positive reviews and number 0 to negative values. The classification was done by creating dummy variables for each of the ratings. This gave us 5 new dummies. Next, we added the dummy variables representing ratings 4 and 5. As a result we got a binary score column with ratings 4 and 5 represented by 1 and any other rating represented by 0. The following figure shows the dummy variables created and the 'score'.

asin	reviewText	overall	1.0	2.0	3.0	4.0	5.0	score
B000FA64PK	Another well written eBook by Troy Denning, bu...	3.0	0	0	1	0	0	0
B000FA64PK	This one promises to be another good book. I h...	5.0	0	0	0	0	1	1
B000FA64PK	I have a version of "Star by Star" that does n...	4.0	0	0	0	1	0	1
B000FA64PK	Excellent! Very well written story, very excit...	5.0	0	0	0	0	1	1
B000FA64QO	With Ylesia, a novella originally published in...	2.0	0	1	0	0	0	0

Later on, we re-run our python script without using the binary classification, simply using the original ratings (multi-label classification), and compare the results between the two forms of classification.

5 Tokenizing and Splitting

Step 5 of the process is to tokenize and split the data into lists. Using the function `str.split().tolist()` we are able to convert each individual review into a list of strings. All the lists are appended to

an empty list, and the result is list of lists.

6 Word Embedding and Sentence Embedding

For the data to be accessible by the algorithm, the data needs to be in numeric form (Bengfort, Bilbro & Ojeda 2018).To achieve this we convert our text format to a vectors of numbers. This is known as 'Vectorization', where each word is represented by a one dimensional array of numbers of chosen length 300. Words with similar meaning typically yield numerical word-vectors with a cosine similarity closer to 1 than 0. This is a vital step for creating word embeddings. Word embedding can be done in a number of ways, either by creating one's own model , or with the help of a pre-trained model. Because of our limited time and computational power, we opted to use a pre-trained model. The library `gensim` has an in built `word2vec` model, which as the name suggests converts words to vectors of 300 numbers. We used a pre-trained Google news dataset that uses `word2vec` to create this vectorization. Once our tokenized data was vectorized, we normalized the vectors by their magnitudes. Next we created an empty array in which the word embeddings were added to one another to create sentence embeddings for each of the reviews. This allowed us to have a vector of similar length for all the reviews, making the comparison of different reviews much easier. Creating the sentence embeddings this way also took care of the dimensionality problem that may have arose.

7 Machine Learning And Results

The data is now prepared to be fed into ML models. We split the data frame into training (70%) and testing (30%) and applied Logistic Regression, SVM, RFM and Decision Trees Classifier. The following table gives a comparison of the results of the models on the binary classification of our data. The table reports the weighted average results for precision, recall, accuracy and f1-score.

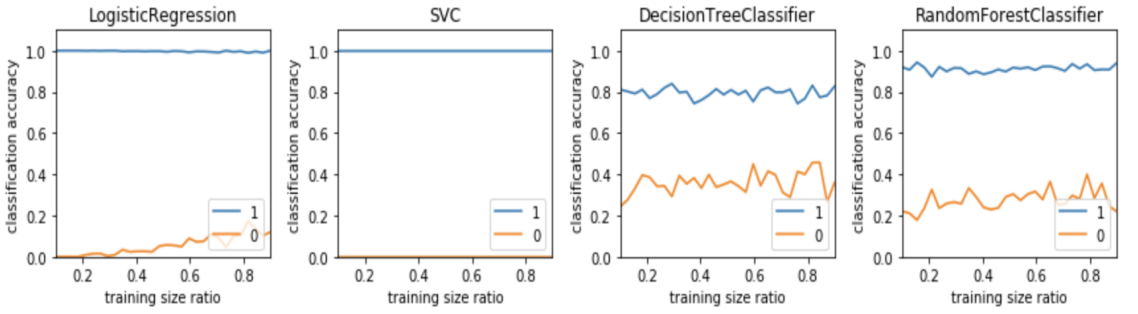
Model	Precision	Recall	f1-score	Accuracy
Logistic Regression	0.79	0.75	0.66	0.75
SVM	0.55	0.74	0.63	0.74
Random Forest	0.76	0.78	0.74	0.78
Decision Trees	0.69	0.71	0.70	0.71

Figure 1: Model Results for Binary Classification

As we can see Logistic Regression offers the highest precision, where 'the precision is intuitively the ability of the classifier not to label as positive a sample that is negative'. For our model, higher precision means that the positive reviews are predicted as positive and not negative, and vise versa.

However, Logistic Regression has the lowest f1-scores. SVM has a consistently low Precision and f1-score. The recall and accuracy score are the same for all four models, although this is not always the case.

The figure below graphs how accuracy of classifiers fluctuates with different training and testing sizes. The plots show extreme contradictions between the accuracy for negative versus positive comments.



The following table shows the performance statistics of the model once multi-label classification is used.

Model	Precision	Recall	f1-score	Accuracy
Logistic Regression	0.47	0.54	0.44	0.54
SVM	0.25	0.50	0.33	0.50
Decision Trees	0.39	0.39	0.39	0.39
Random Forest	0.44	0.48	0.45	0.49

Figure 2: Model Results for Multi-label Classification

The logistic regression model seems to have lost a lot of its precision level when the ratings were classified as multi-label. SVM continues to give lowest precision score. However logistic regression shows better f1-scores in the multi-label classification case than in the binary case. Yet again, the recall and accuracy scores seem to be fairly similar although this is still probably a coincidence.

Moreover, as once can observe, binary classification gives a better prediction in general. However, none of the models stand out in comparison to the others.

8 Conclusion

All the models perform more or less similarly within the chosen classification category. This is probably because some of the information is lost when we combine the data to binary classes. In case of multi-label classification, it could be the large dimension of data that creates some level of confusion.

We finally use our developed model to predict the sentiment of a completely new review. We do this using binary random forest model as it is gives us a better prediction than the others. The `test_new_data` function takes a string/review , tokenizes and vectorizes it followed by which it uses the `saved_model.pkl` to predict the the rating of the new review. The figure below shows a new text with positive strings like 'love' and 'inspiring' classified as 1, implying that it is a positive review.

```
# Using our model to classify a new text

Xnew=test_new_data("I love this book. It is so inspiring! \
I would definately recommend buying it")
```

```
ynew = classifier_lr_load.predict(Xnew)
print(ynew)
```

```
[1]
```

There are however some limitations in the project that are worth mentioning: a) we did not estimate the results multiple times using cross-validation on our data; b)having limited computational power prevented us from running the model on the whole data set; c) Finally, we plotted different training and testing sizes and compared the accuracy of classifiers. However, the plots show extreme contradictions between the accuracy for negative versus positive comments. Moreover, the accuracy of the models do not show positive correlation with the size of training sample, rather they show fluctuation.

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. McCormick, Chris. Google's trained Word2Vec model in Python. http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/

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