Capstone Project 2 Milestone Report 2:
Understanding Yelp Text and Star Reviews for Business Success
Gurdeep Sullan
5/26/2020

Background:
User reviews can provide useful feedback for business growth and establishment. With the ease of online review apps like Yelp, consumers have added an ever growing amount of text and "star" review data for businesses they visit. Machine learning techniques in NLP can be used to decipher the importance of text features of a business for its success. Specifically, NLP can be used to analyze the actual text of the reviews for the data and perform sentiment analysis. This sentiment analysis can be powerful in gauging public tone. It can be used to understand and respond to customer criticisms of business marketing and products in real time.

Example Yelp star average star rating with business attributes

Problem:
There is a large amount of text review data that is connected to businesses on Yelp. Can we build a model to accurately predict sentiment of reviews (using the star rating as labels)? Can we find meaningful words in reviews that correlate highly with success in star rating?

Client:
The clients for this project would be businesses that are listed on Yelp. An analysis of their text reviews can give businesses a perspective of how they are doing and overall levels of customer satisfaction. On top of the numerical star reviews, the important words from this analysis can point businesses to potential key areas of improvement that will have a greater impact on reviews (and thus performance)

Data:
The data that will be used comes from the Yelp dataset on Kaggle:
https://www.kaggle.com/yelp-dataset/yelp-dataset
The project specifically uses two of the json files in this dataset:

- yelp_academic_dataset_business.json
- Yelp_academic_dataset_review.json

The dataset contains data from businesses in 11 metropolitan areas located in 4 countries. There are 5,200,000 user reviews and information on 174,000 businesses.

Additionally, a recurrent neural network will be trained in tensorflow.keras framework. This model will use byte-pair encoded data that has been generated and is available through tensorflow datasets:
https://www.tensorflow.org/datasets/catalog/yelp_polarity_reviews#yelp_polarity_reviewsplain_text_default_config
This dataset has 560,000 highly polar yelp reviews in its training set, and 38,000 highly polar yelp reviews in its test set. Though this dataset is also from Yelp, it is not the exact same dataset and therefore a 1:1 comparison between the RNN model trained on this dataset and the other ML models trained on the Kaggle dataset cannot be performed. However, it is a useful model to demonstrate and quantify the performance of a RNN model for sentiment analysis.

Approach:

- ➔ The data will be loaded in pandas dataframes using the read_json method. For the reviews dataset, due to the large size of the dataset, the data will be read in as chunks, and a random small sample (1%) of each chunk will be read into a dataframe, which will be concatenated into a larger dataframe after reading through all of the chunks.
- ➔ Data exploration and descriptive statistics will be completed on the data using pandas, scipy, spacy and matplotlib.
- ➔ The spacy NLP package will be used to engineer numerical features for the text data, both in the business dataset and in the review dataset. Due to the large size of the sampled review data, a generator will be used to generate lemmas from the text data.
- ➔ The data will be split into training and test data to be run on a variety of models. The reviews dataset labels will be generated, splitting the data into "positive" and "negative" classes based on the star_rating feature. A positive review is defined as having a star rating that is greater than 3. A negative review is defined as having a star rating that is less than or equal to 3.
- ➔ Models will be trained based on three approaches: (1) a bag of words approach using TfidfVectorizer from sklearn, (2) a BOW approach + SVD using TruncatedSVD from sklearn, and (3) word embeddings using the builtin spacy doc.vector attribute.
- ➔ Additionally, a RNN (recurrent neural network) model using keras in tensorflow will be trained to perform sentiment analysis on the raw text data.
- ➔ Hyperparameters and model performance will be recorded in order to keep track of optimal hyperparameters and models that perform well.

Deliverables:

The output of this project will be a machine learning model on the reviews dataset that predicts sentiment (positive vs negative) based on the text of the review. This model will return the top words that appear in positive reviews as well as negative reviews. There will be an associated hyperparameter table with this model as well.

All of the notebooks for data analysis, statistical analysis, and machine learning model implementation will be put on Github. Additionally, all reports, presentations, and data will be put on Github.

Data Cleaning:

Jupyter Notebooks: Data Sampling & Joining, Data Cleaning & NLP

Because the review data is so large, the data will be randomly sampled (a 1% sample will be taken). This data is joined with the business data on the business_id key, so information about the average rating for a business is included in the data for each review entry.

The labels for this data are created by splitting the reviews into two classes based upon the star_rating feature:

- Class 0 = sentiment is NEGATIVE: Star rating is <= 3
- Class 1 = sentiment is POSITIVE: Star rating is > 3

Thus the label is the new feature called 'sentiment'

The number of characters in each review is calculated by using the len() function, and becomes a new feature in the dataset.

The raw text column will be preprocessed by applying (.apply()) custom functions to strip punctuation, non-English characters, and numbers from the raw text. This creates a new feature, called clean_text. This clean text is input into one of three pipelines depending on the NLP approach:

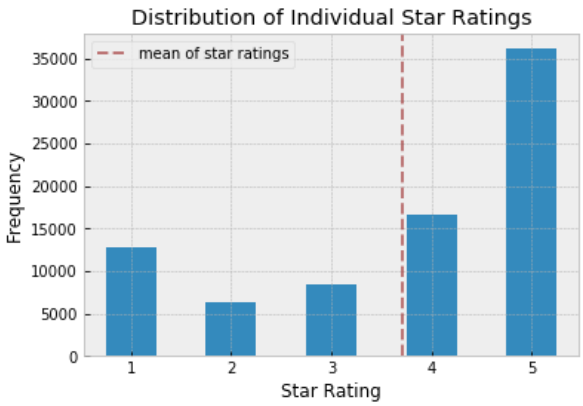
1. Bag of Words (BOW) Approach: TfidfVectorizer
2. BOW + Singular Value Decomposition (SVD) Approach: TfidfVectorizer and TruncatedSVD
3. Word Embedding Approach: Spacy.doc.vector

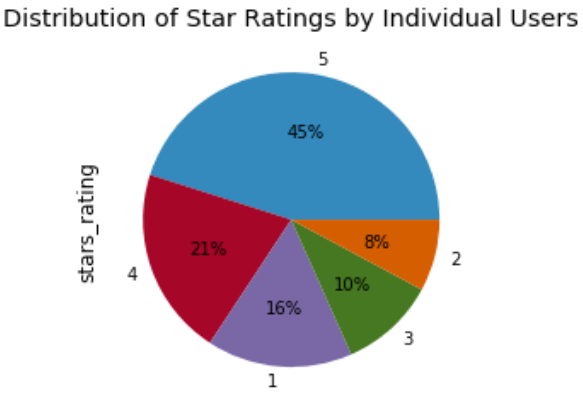
Spacy is the natural language processing library used in this project. The data will be preprocessed using the three above techniques that have been implemented as pipelines. The first method is a bag of words approach, whereby a cleaned text review is split into tokens (words) and each review (document) is represented by a vector with numerical values for each token. The values in the vector are calculated using TfidfVectorizer, with an appropriate *min_df* parameter set to omit words that occur below a minimum number of times (indicating typos, specific names, etc) and *max_df* parameter set to omit words that occur over a certain frequency (indicating stop words like "the"). The second method builds on the first method by applying SVD to reduce the dimensionality to 100 features. The third method does not use any preprocessing to the raw text, but applies the spacy.doc.vector attribute to get a 300 dimension vector representation for each document.

EDA:

Jupyter Notebook: Exploratory Data Analysis

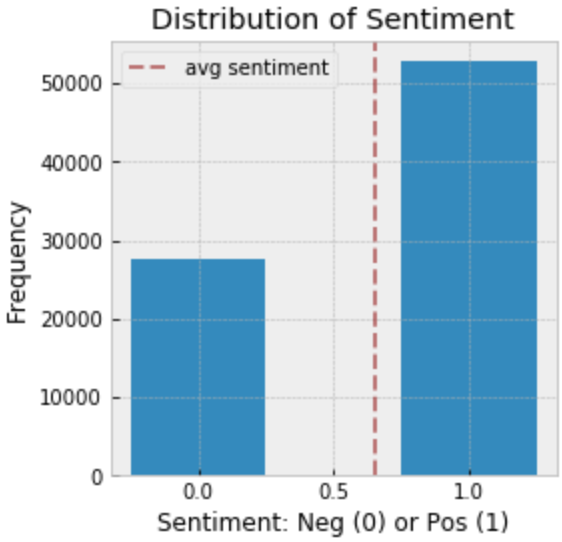
➔ **Rating Distribution:** The first feature explored in this dataset is the stars_rating feature, a column that contains the individual star ratings for each review. What does the distribution of individual user star ratings look like?





➔ It looks like a large number of reviews are very positive, with a star rating of 5. The mean of all user star ratings in this dataset is greater than 3.

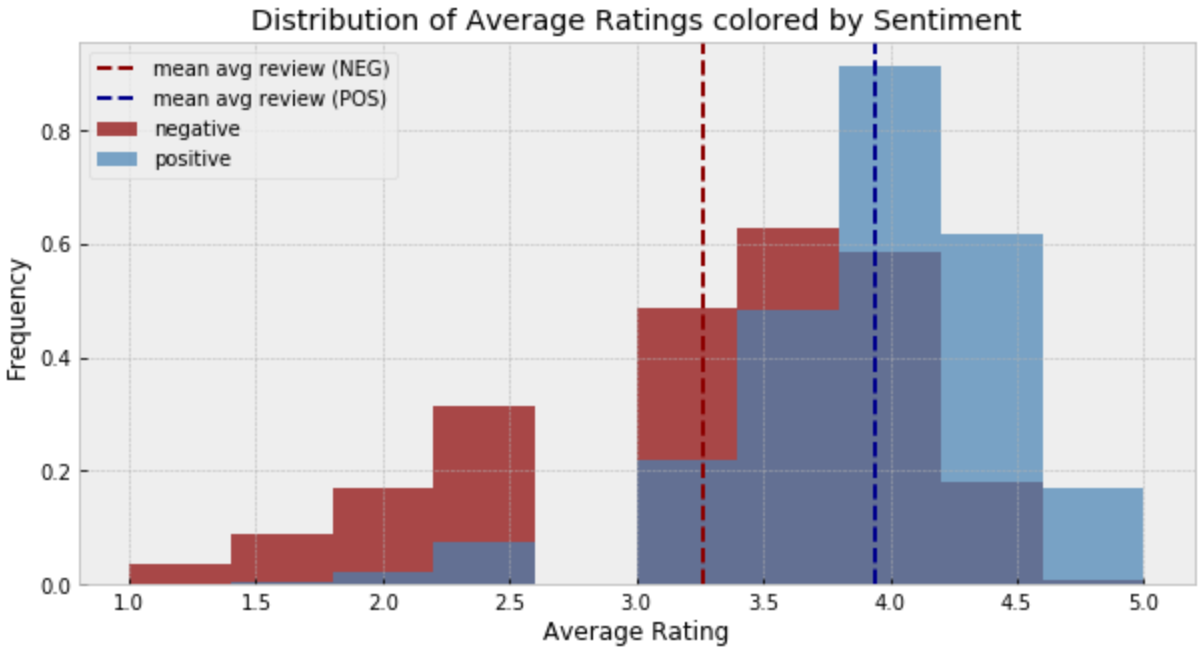
➔ **Sentiment Distribution:** Let's look at what the distribution looks like when we split the data into two classes based on a star rating at 3 or lower for negative and higher than 3 for positive:



➔ **Class Imbalance:** There is a class imbalance here, where the positive class is the majority class. This is an important factor to keep in mind when training and evaluating models.

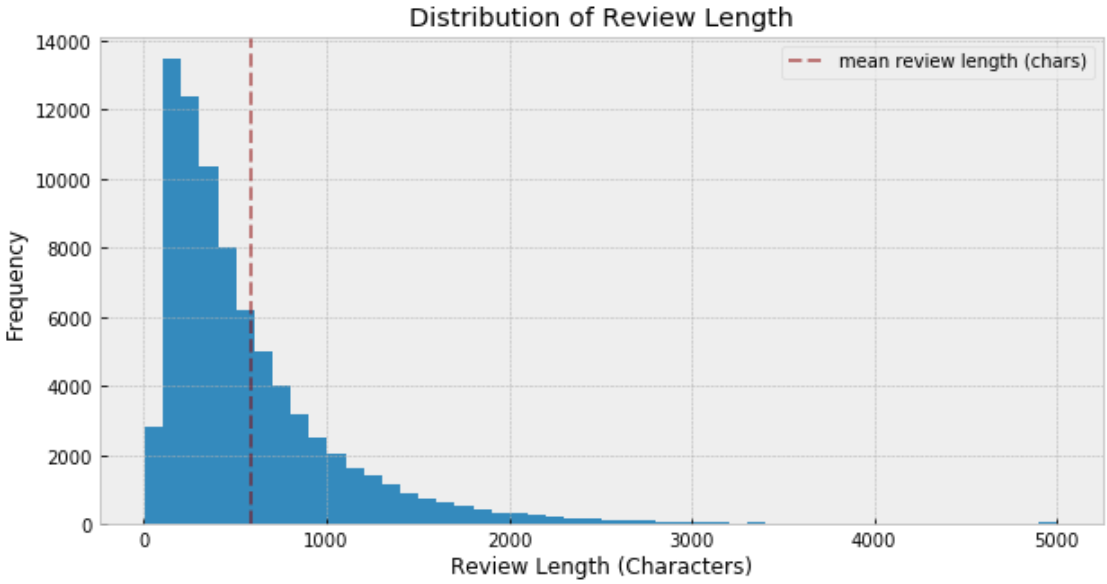
➔ **Avg Star Ratings:** One additional feature we have joined to our dataset is the average star rating for the business that particular review is for. What is the

distribution of the average star ratings for the businesses that have individual reviews?

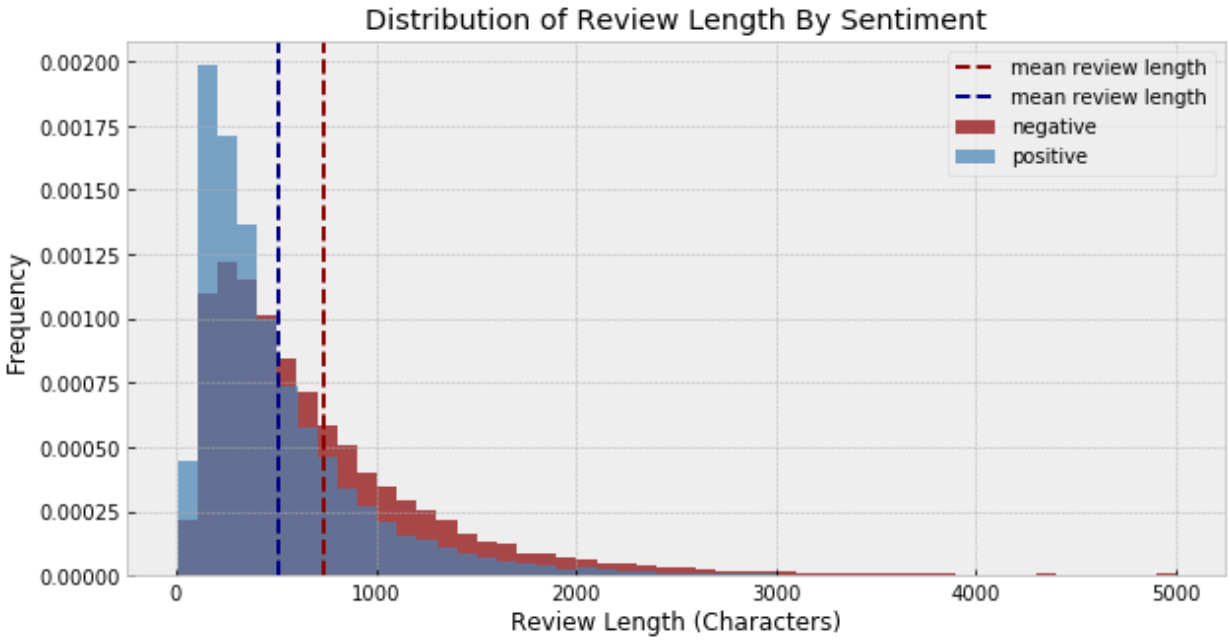


➔ As we would expect, more negative reviews tend to have average business reviews that are lower, and positive reviews tend to have average business reviews that are higher. They are not mutually exclusive, however, reflecting the fact that individual users may have differing opinions than the average rating for a business they visit and post a review for.

➔ **Review Length:** The final feature we will perform EDA on is the review length in characters. What is the distribution of review length in this dataset?



➔ How does this review length change with sentiment?



➔ From a preliminary analysis, it looks like the negative reviews tend to be a bit longer than positive reviews. We will analyze this further via statistical analysis.

Statistical Data Analysis:

Jupyter Notebook: Statistical Analysis

Taking a closer look at the review length feature, there seems to be a difference between the positive and negative class. Is this difference statistically significant?

- Null Hypothesis: There is no difference in mean review length class 0 and class 1
- Alternate Hypothesis: There is a difference in mean review length between the two classes
- Alpha = 0.05

Performing a t-test yields the following results:

- pvalue: 0.00000
- mean of Review Length NEGATIVE: 733.5
- mean of Review Length POSITIVE: 515.9

Because the p-value is lower than the threshold of 0.05, I can reject the null. I will conclude that the mean review length is significantly lower for positive reviews than it is for negative reviews.

Machine Learning Models:

Approach:
Using the three methods, outlined above, a series of machine learning models will be trained on each of the three sets of train/test data (BOW, BOW+SVD, word embeddings). Hyperparameter tuning will be performed using cross validation, and a hyperparameter table will be created and updated with each iteration of each model.

For each of the three NLP approaches, the ML models run were logistic regression, random forest classifier, Multinomial Naive Bayes Classifier, Gaussian Naive Bayes Classifier, and support vector machine classifier.

Additionally, a recurrent neural network will be trained on the tensorflow.keras framework. The data used for this step will come from tensorflow datasets, as outlined under the "Data" section. Due to computing constraints, the data is only trained for 1 epoch.

BOW Method:
Jupyter Notebook: Sentiment Analysis - BOW

In this method, all of the words are included in the dataset used to train the models. This makes models trained on this data easily interpretable. Additionally, the lack of transformation for dimensionality reduction keeps all of the values non-negative, allowing the application of Multinomial Naive Bayes. The other two methods are not trained on MultinomialNB.

Due to the class imbalance observed during EDA, accuracy is not the best metric for evaluation of model performance. Instead, the f1 score, a balance of precision and recall, is a better measure of performance. The best performing model in terms of f1 score was logistic regression for the bag-of-words approach.

BOW + SVD Method:
Jupyter Notebook: Sentiment Analysis - BOW/SVD

Applying SVD to the BOW vectors reduced the dimensionality of the text data to 100. The best performing model for this dataset is again Logistic Regression.

Word Embeddings Method:
Jupyter Notebook: Sentiment Analysis - Word Embeddings

Word embedding uses the spacy.doc.vector feature, which returns a 300-element vector representation for each review. Out of all of the models fit on this dataset, logistic regression performed the best.

RNN:
Jupyter Notebook: Sentiment Analysis - RNN

This model was built using the following tensorflow tutorial:
https://www.tensorflow.org/tutorials/text/text_classification_rnn

The text was byte-pair encoded, meaning that patterns of characters were encoded as bytes. The text vocabulary size was ~8,000. The RNN model has four layers, an embedding layer to encode the data, a Bidirectional layer that propagates the input forward and backward and helps the model learn long range dependencies, a Dense layer with an activation function of ‘relu’, and a final output layer. The model was compiled using the optimizer ‘Adam’, which adjusts the learning rate through gradient descent.