Sentiment Analysis Project

Problem: User feedback is arguably the cornerstone of any product or service. Feedback is what allows the developers and designers to effectively evaluate whether their product or service is resonating with the audience. However, with so many channels to evaluate products and so many users to pull opinions from, its difficult to understand whether your product is sticking from an aggregate level. This project specifically focuses on reviews about Yelp.

Goal and purpose of the project: The purpose of this project is to introduce a pipeline where reviews can be analyzed quickly, and their sentiment correctly classified based on the language used. This project explores several approaches to build a natural language processing classification model and explores several levels of granularity. This model’s intent is to be used in user-evaluation contexts, so that developers can quickly understand how users are reacting to their product on a broad level.

Project Process and Notebooks

1. App Store Scrape Script
   1. This script uses an Apple App Store API to scrape raw reviews on Yelp from the Apple App Store. It then saves them to a CSV file called yelp\_reviews.csv
2. Yelp Reviews Preprocessing
   1. This notebook contains preprocessing code to prepare the text for conversion to matrix data. Several methods are used including converting text to lowercase, handling common chat words (abbreviations), handling negations and contractions, removing punctuation, and labeling parts of speech with the spaCy package
   2. The notebook then saves the prepared data to a CSV file called reviews\_prep.csv
3. Yelp Reviews Sentiment Analysis EDA
   1. This notebook analyzes the text data to see what language is being used in each category
      1. The updated version of this notebook pulls data from review\_prep\_v2.csv, an updated version of reviews\_prep.csv that contains 20,000 reviews instead of the initial 3,000 reviews. All the initial reviews are included.
   2. The notebook creates word clouds for each category with common stop words in general English language and stop words specific to this use case removed
   3. The notebook also separates and counts words unique to each category to examine whether certain language is being used in each review label, finding several select words that are used repeatedly in the extreme contexts, but not in the neutral context
   4. The notebook also discovers that reviews sometimes talk about Yelp, and other times simply talk about their experience with a business
4. Initial Modeling
   1. This notebook built an initial random forest model using a couple different matrices, with their metrics posted for comparison. The following matrices were used:
      1. Bag of words with CountVectorizer
      2. Tf-idf with TfidfVectorizer
      3. N-grams with CountVectorizer
   2. Before modeling the data was downsampled to ~140 cases each. Five classes were used (1-star to 5-star ratings) to train each model
   3. A random forest model was also built using just the titles of each review to see if that would create better accuracy
   4. The classification reports were the following:
      1. Random forest model with CountVectorizer: 41%
      2. Random forest model with just title tokens and CountVectorizer: 23%
      3. Random forest model with TfidfVectorizer: 43%
      4. Highest performing Random forest model with n\_grams: 41% (with uni-grams)
   5. The following next steps were then taken to increase the accuracy, to which each of these approaches would be compared:
      1. Utilize the gensim package to use word embeddings, which take context into account as opposed to the word frequency
      2. Collect more data
      3. Group reviews with similar sentiment together
5. Word Embeddings and Neural Network Training
   1. More data was collected to train the neural network effectively (20,000 reviews as opposed to ~3,000 reviews)
      1. “Word Embeddings with Gensim” was used to understand how the gensim package works
      2. The App Scrape Script notebook was used again and the result was saved in a new dataset called yelp\_reviews\_v3.csv
      3. To understand how many weights the neural network had to solve for the number of words in the new corpus was calculated in “Total Number of Words in Yelp Reviews Datasets”
   2. The gensim package was used to train a neural network that predicted the next word based on the context surrounding the word. The weights solved for in this task are the word embeddings. The resulting word vectors then had 300 components (default is 200)
      1. The words were preprocessed using simple\_preprocess from gensim utils
   3. The model was then saved to disk to load in future notebooks
6. Modeling and Performance Evaluation with Gensim
   1. This notebook created random forest and logistic regression models with both binary and multiclass tasks and the performance was compared between each to find which was ideal for accuracy
      1. The binary classes were derived in the following way:
         1. 1 and 2 stars were classified as negative and given a 0
         2. 4 and 5 stars were classified as positive and given a 1
         3. 3-star classes were dropped
   2. Initial binary class (positive and negative sentiment) random forest model reported an accuracy of 92%. This performance was encouraging so a multiclass task was tested next
   3. An initial multiclass task with a random forest model yielded an accuracy of 52% on test data, better than previous models. However, when evaluated on the training data, it was clear the model was overfitting
   4. A gridsearch was then performed to see if model performance could be improved, and the same was done with logistic regression models. The optimal performance came from a logistic regression model with a C penalty of 0.1 and an l2 regularization scheme. This model had an accuracy of 48% on training and test data. Other models displayed more overfitting
      1. A confusion matrix found that this model was good at predicting extremes (1 and 5 star reviews) but struggled to predict neutral reviews (f1 scores of 0.61 and 0.66 for 1 and 5 star reviews and 0.34 for neutral reviews
7. Binary Modeling Comparison
   1. To compare the binary performance on matrices built from CountVectorizer and TfidfVectorizer, random forest models were trained and each returned an accuracy of 91% and 92% respectively

Results

* Multiclass problems were more challenging for the model, the best model returning an accuracy of 48% on training and test data, built with logistic regression with a C penalty of 0.1 and an l2 regularization scheme. This was an improvement from the initial random forest models that were built on 3,000 reviews (best accuracy was 43%)
* Models built off word embedding matrices performed slightly better than matrices built from CountVectorizer or TfidfVectorizer
* Binary classification models performed very well yielding an accuracy of 92% for each
* EDA supports this as more signal came from the extremes than the middle

Discussion and Limitations

* This shows an effective sentiment analysis where positive and negative predictions can be made with confidence
* The neutral data appears to be noisier which translated into decreased model performance
* This model can’t process words that don’t appear in the corpus and would need a broader gensim model loaded to create the correct word embeddings

Conclusion and Next Steps

* This shows promise for deployment in the future to help developers understand user sentiment around their products more effectively, particularly around positive and negative sentiment
* More data will be needed to create a more robust word embedding scheme
* Other methods for feature extraction might be worth exploring