**Online Transaction Sentiment and Pain-point Analysis**

Robert J. Reichhard

DSC 550: Data Mining

Dr. Brett Werner

November 20, 2021

**Milestone 5: Case Study**

I work for an online payment processor and have some true-life experience reading customer comments from surveys. I have been asked previously to extract key words, phrases, or themes from the negative customer comments to assist in directionally coaching teammates for customer experience improvement. As a human, reading through these comments was a simple task, albeit time consuming. The difficult part was quantifying why I was surfacing certain phrases or topics. It was also quite a time-consuming process to read through individual comments one by one and attempting to glean insights from them. Depending on the timeframe, there could be a few hundred to thousands of comments to read.

My original intention with this project was to use the Twitter API to scrape thousands of recent tweets regarding online popular transaction processors. I aimed to apply VADER sentiment analysis to the tweets to determine if they were written with a negative or positive tone as well as gather other metadata about the tweets such as source, geographical location, etc. For the actual tweet content, I aimed to also tokenize them to be able to identify n-grams and most frequently used words. It would likely be beneficial to be able to remove stop words from the lists as well as non-English characters like emojis and hashtags.

In the first iteration of this case study I believed this analysis will help identify with statistical accuracy what phrases or words customers are using online to help identify their pain points. Just based off of my own experience, I imagine there will be a great number of tweets related to fraud, limited account usage, and disputed items (item not as described, item not delivered). I think there would also be a business benefit to seeing what reasons customers are making positive tweets.

**Business Problem/Data**

I utilized Twitter’s robust free API to mine tweets and create my own custom dataset. I used the tweepy python library along with my personalized API key and searched for specific hashtags and mentions, specifically eBay, PayPal, Amazon, and Venmo. Before any cleaning, my dataset that was generated from eight different dataframes/hashtags had 7,919 rows and 340 columns. I slimmed down these columns and kept the pertiment fields of created\_at, full\_text, source, and language.

After mining tweets, cleaning them, and applying VADER sentiment analysis, the results were not helpful in a business domain. A majority of the tweets were found to be of a neutral tone as shown below:

Chart, pie chart

Description automatically generated

After isolating only the negative and positive tweets, it was found that the most common words were not something that would help drive a business decision. A majority of the tweets were advertisements or spam in my opinion, being written likely by a bot and aimed at having people mistakenly send them funds.

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Looking at all tweets regardless of sentiment, the most common bigrams were:

Chart, bar chart

Description automatically generated

The business problem then becomes not searching for pain points in the tweets, but rather using the source of the tweet to be able to predict where future spam tweets may be generated from. From a business perspective this could be helpful to identify what platform is generating spam tweets to be able to avoid legitimate users from being misled and clicking on them. Businesses could pre-emptively warn users of potential bad actors and target the direct source such as apple or android for mobile users.

**Graphical Analysis**

Once tweet sources were determined to be useful, the top 5 most common tweet sources were counted and identified.

Chart, bar chart

Description automatically generated

The Twitter web app was the most used by quite a way, followed by twitter for iPhone on apple and twitter for android on non-apple phones. Twitter for iPad sources are like twitter for iPhone as they are both apple products, and tweetdeck was represented in a very small number. I kept the top three sources to move forward with my analysis. This left me with a dataset of 6,601 tweets to test and train the algorithm.

**Feature Engineering**

Since the tweet final text was cleaned and tokenized already, the most important step left for feature engineering was to encode a new categorical feature. I assigned a value of zero to the twitter web app as that is the most common, followed by one for twitter for iPhone and 2 for twitter for android. This creates a numerical value to identify the tweet source to feed into the algorithm. Next, I utilized a term frequency-inverse document frequency for the final text prior to creating a training/testing split of 80/20. This left 5,280 samples for the training dataset. Lastly, the training data is skewed heavily towards twitter web app samples. I utilized the Synthetic Minority Oversampling Technique (SMOTE) to balance the samples. After feeding the training data into the SMOTE() function, the classes were balanced at 33% each:

Chart, bar chart

Description automatically generated

**Model Selection & Evaluation**

Once all three tweet sources were equally represented in the training data, it was time to train some algorithms and evaluate them. First was a Random Forrest Classifier using the “one versus rest” hyperparameter. The ROC AUC score generated was 0.894985, meaning that the classifier can distinguish more true positive & negative predictions than false positive and false negatives. The accuracy score generated was 79.86%. The F1 score for identifying twitter for android tweets was the lowest at 0.47 with a recall of 0.37. I also trained a Naïve Bayes Multinomial classifier using the same dataset which generated an accuracy score of 74.41%.

**Conclusion**

In conclusion, the Random Forest Classifier was the most accurate at 79.86% with a F1 score of 0.80. Looking into each classification in the classification report, twitter for android was the most difficult to identify accurately. The algorithm was too careful of falsely identifying a tweet as being from android with a recall of 0.37.

Reflecting on this project I would recommend designing some method of filtering out spam vs ham tweets. I feel that having too many spam tweets in the dataset all but made the conclusions useless from a business perspective. In the most common words and bigrams, many were asking for retweets (rt) which may not be helpful to the business and indicate it is possibly a spam tweet attempting to get shared more often. If a method of filtering out spam tweets is generated, I think the original business problem of looking for common themes in negative sentiment tweets would be more powerful to a business.

I also would like to mine more android tweets specifically if moving forward with further iterations. While I feel the SMOTE function helped balance the classes, the oversampling of the android class likely did not help with accuracy and precision.