JSC270 Winter 2022 Assignment 4

Authors: Steven Liu, William Zhang

** All work are done together **

Google Colab Link:

https://colab.research.google.com/drive/1uTHEc-ZqtLL8JpjVor_pMPg1wjy_PvzR?usp=sharing

Part I: (codes are in Colab)

1. **Consider the training data. What is the balance between the three classes? In other words, what proportion of the observations (in the training set) belong to each class?**

   For the training data, the proportions of observations belonging to the three classes are 0.374, 0.187, 0.438, which means the majority observations have either sentiment of “negative” or “positive”.

2. **Remove all punctuation (,.?!;:’") and special characters(@, #, +, &, =, $, etc). Also, convert all tokens to lowercase only. Can you think of a scenario** when you might want to keep some forms of punctuation?

   Yes certainly under some scenarios we would want to keep some punctuation. For example, a word like “well-known”. It is better to be treated as one phrase instead of “well” and “known”, splitting it into two words might result in twisting the meaning of that word. Another scenario is that puncatations like the exclamation mark or question mark might indicate some kind of sentiments, such as

excitement or confusion, thus under those circumstances we should keep the relevant punctuations.

3. **Now stem your tokens. This will have the effect of converting similar word forms into identical tokens (e.g. run, runs, running → run). Please specify which stemmer you use.**

   We used the Porter stemmer.

4. **Now convert your lists of words into vectors of word counts. You may find Scikit-learn's CountVectorizer useful here. What is the length of your vocabulary?**

   The length of our vocabulary is 52572. However, we're only taking the top 1000 features otherwise the runtime would be too long each time.

5. **Fit a Naive Bayes model to your data. Report the training and test error of the model. Use accuracy as the error metric. Also, report the 5 most probable words.**

   The training accuracy is 0.680 (68%). The test accuracy is 0.668(67%).

   The 5 most probable words for "negative" are 'coronaviru', 'covid19', 'price', 'food', 'thi' with counts (6703, 4862, 4332, 3622, 3206)

   For "neutral" the words are 'coronaviru', 'covid19', 'store', 'supermarket', 'price' with counts (3792, 2751, 1581, 1435, 1361)

For “positive” the words are 'coronaviru', 'covid19', 'store', 'thi', 'price' with counts (7466, 6000, 3895, 3770, 3322)

6. **Would it be appropriate to fit an ROC curve in this scenario? If yes, explain why. If no, explain why not.**

   It would not be ideal to fit an ROC curve in this scenario because there are three classes that we care about, rather than the normal binomial class setting which we can easily define positive and negative. Although we could use the “one” vs “rest” strategy, it is not the most ideal way since we care equally alike for all three classes.

7. **Redo parts G-H using TF-IDF vectors instead of count vectors. You might find Scikitlearn’s TfidfVectorizer() transformer useful. Report the training and test accuracy. How does this compare to the accuracy using count vectors?**

   The training accuracy for TF-IDF is 0.664 (66%), while the test accuracy is 0.654 (65%). Compared with using count vectors, the performance for both training and test decreased slightly, which is quite surprising.

8. **Redo parts E-H using TF-IDF vectors instead of count vectors. This time use lemmatization instead of stemming. Report train and test accuracy. How does the accuracy with lemmatization compare to the accuracy with stemming?**

When using lemmatization instead of stemming, the train accuracy is 0.657 (66%), the test accuracy is 0.637(64%). It seems that compared to stemming, the accuracy with lemmatization decreased furthermore.

9. **(Bonus) Is the Naive Bayes model generative or discriminative? Explain your response.**

   The Naive Bayes model is a generative model. The main reason is that we're utilizing the joint probability of labels and tweets to construct our model, rather than directly using the conditional probability to predict. Another reason is that it assumes all the features (the words in our case) are conditionally independent, which is only assumed for a generative model.

Part 2 on next page

Part II: Having fun with NLP using the Twitter API

(I) **Problem description and motivation**.

The question we want to answer is “Can we use models to predict whether a tweet about Clash Royale will either be approval, disapproval, or neutral towards the recent game updates and balance?” Knowing how the player base reacts to each game update is often a crucial demand for every game developer. However, traditional methods of survey usually take a period of time and do not provide instant feedback which are considered more valuable often, and survey is also very much a passive way of collecting data. Other methods like analysis of revenue gain, or player count changes are even more on the inefficient side, and would require a long wait period before these data can even be collected. Thus, analysis of Twitter data would provide a quick and easy access of the needed data that offers some insights into the immediate reaction of the players as soon as an update is released. Furthermore, building a machine learning model classifier could very well take into account the fact that they’re a mixture of opinions through designing labels and categories to represent different attitudes. Although none analysis have been found directly related to data that of the game “Clash Royale”, similar analysis on game reviews have appeared at least a few times over the course of investigation, for example, “Twitter sentiment analysis of game reviews using machine learning techniques”[1] is a fairly similar one which we will use as comparison throughout this analysis.

[1] Kiran, T. D. V. et al. “Twitter sentiment analysis of game reviews using machine learning techniques.” (2016).

(II) **Describe the data**.

Using the Python library ‘Tweepy’, we extracted 397 relevant tweets with the search queries “@ClashRoayle Update” and “@ClashRoyale Balance”, these two are the main phrases that players typically include as they tweet about the updates. We’ve restricted the search results to tweets only for easier analysis purpose and have used the parameter “tweet_mode = extended” to disengage the 140 characters limit of the Twitter API. Then, we manually labeled the sentiment of the tweets as either ‘0’, for being neutral about the update or informative matters like reporting a bug, ‘1’ for being in favor/approval of the update, and lastly ‘2’ for being disapproving/hating the update. Then we’ve made the tweets into a 70-30 split for a training and testing set.

Some strengths about our data are that they’re quite relevant and recent (tweets within 7 days) to the topic; they’re all manually labeled, which allows us to use supervised learning methods; and lastly, most tweets have “strong opinions”, especially for the negative ones where even vulgar languages are used, which in turns is actually advantageous for NLP. The limitations are in a sense related to the strengths. Manual labeling are affected by human inconsistency (which is reduced through having both individuals labeling the data together), recent data means that the data are collected within a short period of time which make them more prone to being biased, lastly, since tweets contains many ambiguous wordings involving internet slang, abbreviations, and sometime sarcasms, these would’ve likely reduced the performance of our analysis in some degree. In comparison, the game review analysis had 21,000 tweets as their data, yet they still reported very similar concerns about the limitation of their data and the difficulties with tackling these problems.

(III) **Exploratory data analysis**.

After manually labeling all the tweets, we have firstly created a bar plot demonstrating the distribution of people's attitude toward the game to have a better understanding with our data. It turns out to be around 40% tweets are either irrelevant or neutral, and the ratio for both positive tweets and negative tweets are 30%. Then, we have performed a series of preprocessing steps to transform our data into a computer-understandable form while extracting key information from the raw dataset. This includes: Tokenization, which we splitted the tweets into individual words; URL, special character and stop word removal[2], which helps us exclude unnecessary (meaningless) words that have no real effect on the result; Lemmatization and Lowercase the words, which convert all vocabulary to their simplest form to avoid the duplication in meaning. Lastly, we have converted these lists of words into a word count matrix by using Count Vectorization and a word ratio matrix by using TF-IDF so that we have numerical representations of our tweets that can be easily fitted into a model.

(IV) **Describe your machine learning model**.

As a main goal for our investigation, we want to explore the best model that can categorize the tweets most accurately. Thus, we have constructed four different models in which we can make comparisons among them to choose the "best". The first two models are built based on the Naive Bayes model. One of them is using the count vector as input and the other one is using the TF-IDF as input. The Naive Bayes algorithm has constructed a posterior distribution which allows us to predict the most

[2] We have removed the Top 100 English stop words by applying Python package nltk.stopwords

probable class with the given dataset (count vector). As a condition, it is a supervised model which requires the training dataset to be labeled. Since our data is properly pre-labeled, the Naive Bayes model is an appropriate choice for our research question. As strengths, Naive Bayes is highly related to every word feature. However, if we have too many unnecessary words, this can also be a weakness which decreases the significance of key words. To solve this weakness, we have also constructed a Classification Decision Tree with a depth of five. With a Classification Tree, we can visualize how each class is being categorized under some key words. By using a depth of five, it is still related to some of the word features while not getting overfitting. As a biggest weakness, the Classification Tree only makes predictions based on a small part of word features. But it is still appropriate especially for us to visualize our model and it works better when we have limited word features. Lastly, we have fitted in a support vector machine model in order to compare with our existing models. Similar to the Naive Bayes algorithm, SVM also minimizes the error while splitting classes with given input. It is another great classification model which has been applied in many other NLP researches. For each model, we compute both its test accuracy and train accuracy, as well as its confusion matrix to test their overall behavior and the evidence of overfitting. In addition, we have introduced another variable named “accuracy for class 1(positive) & 2(negative)” as an indicator to models’ behavior, which tests the probability of correctly identifying the positive and negative tweets.

(V) **Results and Conclusions**.

It is evident that our models were, although not in the best forms possible, but still to some degree assisted with predicting the corresponding attitude of a tweet about the Clash Royale update. Out of all the methods, the Naive Bayes classifier with the countvectorizer performed the best (Highest overall accuracy and accuracy for class 1 & 2). It has a train accuracy of 92%, a test accuracy of 67% and a test accuraacy on Class 1 & 2 of 61%. Yet all models demonstrate signs of overfitting, seen by a large difference between the train and test accuracy. This is most likely due to the small dataset that we've fed into our models and would be better addressed with more data. In regards to the initial question, we can claim to have at least found a more efficient way to determine the general reactions and feedback of the players after an update of Clash Royale, where there are certainly still rooms for improvements. Perhaps a smarter EDA process such as removing repeated tweets and filtering spam would be a start to improving the accuracy.