Predicting Reddit Forums: Multinomial and Bernoulli Naive Bayes Models

Lilly Sharples lillysharples@uga.edu
Department of Computer Science, University of Georgia
Kimberly Nguyen Kimberly.Nguyen@uga.edu
Department of Computer Science, University of Georgia

Abstract

The objective of this research is to accurately classify what forum a Reddit post came from, given the text of the post. Reddit is an online network of communities, where users can post, vote, and comment in forums of their interest. These targeted forums are called subreddits, designated to users talking about a specific topic. Our focus was on the subreddits of Cryptocurrency, WallStreetBets, VaccineMyths, Coronavirus, worldnews, and AndroidDev. The aforementioned six subreddit datasets were appended into one large dataset, giving each entry a label of the associated forum name. Combining the posts of each of these subreddits brought a total of 1,368,659 entries, 1,360,455 of those being non-null. This dataset was then split 75%-25% into training and test sets.

To predict which forum a post came from, we trained two variations of Naive Bayes classifiers- Multinomial and Bernoulli. Using the test data on these models, the Multinomial Naive Bayes had an accuracy of 96.54%, and the Bernoulli Naive Bayes had an accuracy of 93.6%.

1. Introduction

Reddit is a centralized online platform, founded in 2015, attracting more than 430 million monthly users with varying interests, making it the seventh most popular social network app in the US [**dean˙2021**]. A Reddit subforum contains many posts of a relevant topic. As of January 2021, there are over 100,000 communities, and more than 2,620,000 subreddits [**dean˙2021**]. A post can be text, website links, images, and photos. When a user posts, other users can comment on that post. Both comments and posts can be upvoted or downvoted; content with more upvotes will generally be seen by more users. The focus of our research was on Reddit posts/comments in text form.

The objectives of our research project were to build a model that could accurately classify a post or comment into it's given subreddit. For this project, we chose six various subreddits, whose descriptions are given in the table below:

r/CryptoCurrencies	The official source for cryptocurrency news, discussion, and analysis, with around 2 million users.
r/worldnews	A place to discuss major events around the world, with over 26 million members since its creation in 2008.
r/WallStreetBets	Participants of this subreddit discuss stock and option trading. Since being founded in 2012, there are now almost 10 million active users in this forum.
r/AndroidDev	News for Android app developers, where the 172,000 members are able to ask questions and share projects.
r/VaccineMyths	A smaller subreddit where users discuss various vaccine myths.
r/Coronavirus	This subreddit has over 2 million members, monitoring the spread of the pandemic with high-quality posts.

Subreddit forum descriptions

With 1,368,659 entries of Reddit posts from six subreddit forums, it was first necessary to get the data ready to be processed. After combining all six files of data into one dataset (see the Data-Preprocessing section for more details), we checked if there were any null entries. To do this, we called the *.isnull().sum()* [**reback2020pandas**] methods on our dataframe, and found there were 8,204 null entries coming from three of the subforums. To deal with this problem, we removed all null entries using the *dropna()* method [**reback2020pandas**].

Next, we added a column of text with the stopwords removed, then split our data into training and test sets(see the Data-Preprocessing section for more details). It was then necessary to convert our collection of text documents to a matrix of token counts, using the *count_vectorizer* method from sklearn [**scikit-learn**] (see the Data-Preprocessing section for more details).

After preprocessing our data, we created three classifiers to train: Bernoulli Naive Bayes, Multinomial Naive Bayes, and TFIDF Multinomial Naive Bayes. The TFIDF classifiers convert a collection of raw documents to a matrix of TF-IDF(term frequency–inverse document frequency) features, which help show how important a word is within a text document or corpus [**scikit-learn**].

We chose to use Naive Bayes classifiers as they are both fast and proven to be good for text classification. We decided to create a Multinomial classification, as this is usually the best suited for word counts by using the frequencies. The Bernoulli classifier works slightly differently, working with boolean inputs on whether a word occurs or not. [**horbonos˙2020**].

The accuracy results of our four classifiers are as follows:

Bernoulli Naive Bayes	86%
Multinomial Naive Bayes	96.50%
TFIDF Multinomial Naive Bayes	86%

before removing stopwords

Bernoulli Naive Bayes	93.6%
Multinomial Naive Bayes	96.54%
TFIDF Multinomial Naive Bayes	95.1%

after removing stopwords

As Multinomial Naive Bayes classifiers are commonly the best suited for text classification, having the highest accuracy in our dataset makes sense. Overall, we received higher accuracy among all of our classifiers after removing the stopwords from the text.

The Data-Preprocessing section of our paper will describe in further detail the steps taken to prepare our data for the classifiers.

2. Data – Preprocessing

All 6 datasets are from Kaggle. r/AndroidDev posts from 2018-2020, generated by Viktor Arsovski, contains the subreddit posts from the most popular android developer community, and the r/Coronavirus dataset was created by Manish Sharma. The r/CryptoCurrency 2018 subreddit dataset was generated by Kaggle user 'nkrnrnk', and the r/worldnews subreddit comes from Kaggle user 'Chris'. The r/WallStreetBets and r/VaccineMyths subreddit datasets were generated by Data Scientist Gabriel Preda, using the Python Reddit API Wrapper (PRAW) to collect reddit posts in text form. PRAW is a python package that allows easy access to Reddit's API, so that posts can be read [**boe˙2012**].

Across our six datasets, we had a total of 1,368,659 entries. Of those entries, 8,204 were null, giving us 1,360,455 non-null entries. In order to use the data from six different datasets, it was necessary to combine these into one dataframe. Each dataset was imported using Pandas read_csv [**reback2020pandas**] method, reading in the github raw data link to the dataset as well as the *usecols* [**reback2020pandas**] parameter. The *usecols* parameter allowed us to save space and time, since the only feature we needed to download from each dataset was the text, ignoring fields such as usernames and time/date/location posted. Among the six datasets, different column names were used to refer to the body text of the reddit post- some used 'text','title', or 'body'. Therefore, after reading in the specific column, it was necessary to rename the column so everything stayed uniform. For example, these were the steps taken to read, use the correct column, rename the column, and give the data a forum name for the r/WallStreetBets data.

```
wsb = pd.read_csv(’Data/WallStreetBets.csv’, usecols=[’title’])
wsb.rename(columns=’title’:’body’, inplace=True)
wsb[’forum’] = ”r/WallStreetBets”
```

The r/Android dataset required an extra step. Each year of data was in its own csv file, so we used Pandas *concat* [**reback2020pandas**] method to combine two csv files so the data from 2018-2019 and 2019-2020 into one dataset.

```
android = pd.concat([android, android2])
```

Each dataset was also given a column forum, where we named them according to the subreddit forum name. After the six datasets were ready, we used the Pandas *DataFrame().append()* [**reback2020pandas**] to create one large dataset, containing text entries and their respective forum name.

```
data = pd.DataFrame().append([crypto,wsb,vm,cv,android,news])
```

Among our six datasets, half had missing values. The r/Crypto dataset had 3 null entries, r/VaccineMyths had 365 null entries, and r/AndroidDev had 7836 null entries. In order to address the missing values, we used the *dropna()* method, dropping all null entries. As we were working with large text sets, it made more sense to get rid of the null entries instead of trying to replace them.

```
df = data.dropna()
```

The dataset is imbalanced, with forums ranging from 973 entries (r/Coronavirus) to 790958 entries (r/Crypto). The tables below show each forum and their respective entry counts.

r/CryptoCurrencies	790958
r/worldnews	509236
r/WallStreetBets	44234
r/AndroidDev	21780
r/VaccineMyths	1478
r/Coronavirus	973

Table 1: Total entries

r/CryptoCurrencies	790955
r/worldnews	509236
r/WallStreetBets	44234
r/AndroidDev	13944
r/VaccineMyths	1113
r/Coronavirus	973

Table 2: Non-null entries

After creating one large dataframe of non-null entries, we added another column of entries after removing the english language stopwords. The purpose of this is to remove frequently used words that add little meaning to the text. For example, ”the”, ”a”, and ”in” have little importance. We used the Natural Language Toolkit [**bird2009natural**] in order to do this. Now, our dataframe has a column of the original body text, the body text with stopwords removed, and the forum name.

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
df['without_stopwords'] = df['body'].apply(lambda x: ' '.join([word for word in x.split() if
word not in (stop)]))
```

The train_test_split method from the model_selection package [**scikit-learn**] was used to split our original dataset into training and testing sets. Our overall dataset was passed as a parameter, a random_state value of 5, the default train_size of .25, and the default test_size value of .75. The random_state value was chosen by manually testing different values and comparing the accuracy.

```
train, test = train_test_split(df, random_state=5, shuffle=True)
```

Creating a count vectorizer and transforming text data is a necessary step before training our classifiers. We created two count vectorizers; *sklearn.feature_extraction.text.CountVectorizer* and *sklearn.feature_extraction.text.TfidfVectorizer*.
Next, we fit the train and test data to these count vectorizers.

```
count_vect = CountVectorizer()
X_train = count_vect.fit_transform(train.remove_stopwords)
X_test = count_vect.transform(test.remove_stopwords)
tfidf_count_vect = TfidfVectorizer()
tfidf_X_train = tfidf_count_vect.fit_transform(train.remove_stopwords)
tfidf_X_test = tfidf_count_vect.transform(test.remove_stopwords)
```

3. Experiments

The data was preprocessed to exclude stopwords, which greatly improved the accuracies. More specifics of the preprocessing can be found in the **Preprocessing** section.

Naive Bayes classifiers were then performed to classify texts into their associated subreddits, because naive bayes is commonly used for text classification. There are many other advantages of using Naive Bayes models. One of them includes the fast runtime in comparison to other classification models due the independent estimation of each distribution, which refers to the “naive” assumption of conditional independence between each event. They also “require a small amount of training data to estimate the necessary parameters” [**naive˙bayes**].

Specifically, the ML frameworks used to classify texts based on their subreddit were Multinomial Naive Bayes, Bernoulli Naive Bayes, and TF-IDF Multinomial Naive Bayes. Both the Multinomial and Bernoulli Naive Bayes are suitable for discrete data; however, the difference between them is that the Multinomial Naive Bayes counts every instance of a word occurrence while the Bernoulli Naive Bayes can only detect whether a word occurs or not (boolean).

Multinomial Naive Bayes Formula:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Bernoulli Naive Bayes Formula:

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$ [**naive˙bayes**]

In addition to using the Multinomial and Bernoulli Naive Bayes model, we decided to utilize TF-IDF to determine the importance of words in a document by outputting fractional counts, which provides information on how often the word is seen in the document and if the word fits into the message or theme of the document.

TF-IDF Formula:

tf-idf(t,d) = tf(t,d)*Log(N/(df+1))

[**tfidf**]

The following steps outline the process of the model creation:

The training and testing datasets (corpus) are transformed into document-term matrices, which would later be used to fit the models. CountVectorizer() was used for the Multinomial and Bernoulli Naive Bayes models while TfidfVectorizer() was used for the TF-IDF model.

```
count_vect = CountVectorizer()
X_train = count_vect.fit_transform(train.without_stopwords)
X_test = count_vect.transform(test.without_stopwords)
tfidf_count_vect = TfidfVectorizer()
tfidf_X_train = tfidf_count_vect.fit_transform(train.without_stopwords)
tfidf_X_test = tfidf_count_vect.transform(test.without_stopwords)
```

The target training and testing sets are formed.

```
Y_train = train.forum
Y_test = test.forum
```

The Multinomial Naive Bayes, Bernoulli Naive Bayes, and TF-IDF models were then fitted given the training data.

$$\begin{gathered}\text{clfM} = \text{MultinomialNB().fit(X\_train, Y\_train)}\\ \text{clfB} = \text{BernoulliNB().fit(X\_train, Y\_train)}\\ \text{clfTM} = \text{MultinomialNB().fit(tfidf\_X\_train, Y\_train)}\end{gathered}$$

Predicted targets were made by the models given the actual targets.

$$\begin{gathered}\text{Y\_predM} = \text{clfM.predict(X\_test)}\\ \text{Y\_predB} = \text{clfB.predict(X\_test)}\\ \text{Y\_predTM} = \text{clfTM.predict(tfidf\_X\_test)}\end{gathered}$$

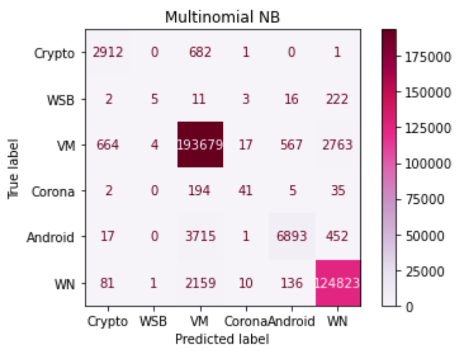
All the Naive Bayes and TF-IDF classifiers were tested using a 2015 MacBook Pro with 2.9 GHz Dual-Core Intel Core i5 processor in version 11.2.2 of macOS Big Sur and a 2018 MacBook Air with 1.6 GHz Dual-Core Intel Core i5 processor in version 10.15.4 of macOS Catalina. They were coded in the The Scientific Python Development Environment, more commonly referred to as Spyder. Git version control software, namely Github, was also used to remotely track changes. None of the ML frameworks require significantly more resources or memory than the others. No noticeable problems were encountered when running the experiments, as well.

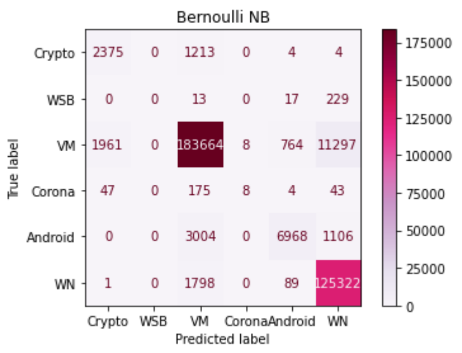
4. Analysis

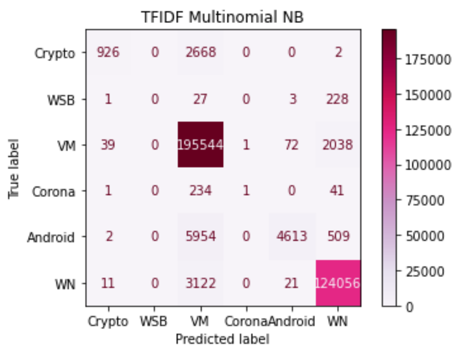
Legend:

crypto → r/Crypto
wsb → r/WallStreetBets
vm → r/VaccineMyths
corona → r/Coronavirus
android → r/AndroidDev
wn → r/worldnews

Confusion Matrices:



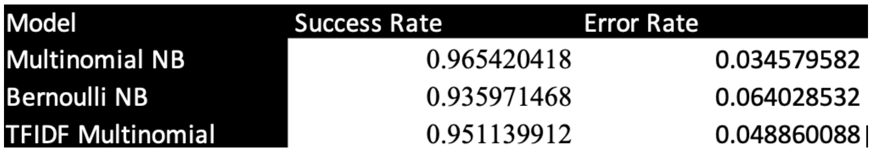




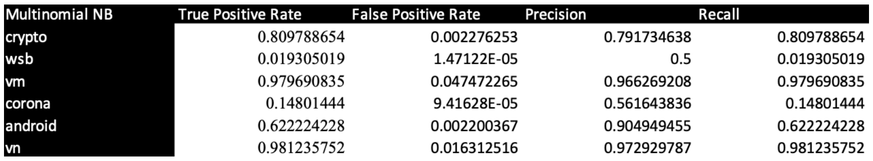
The following metrics were calculated: success rate, error rate, true positive rate, false positive rate, precision, and recall. Here is an example of how the metrics are interpreted for the Multinomial Naive Bayes model, specifically for the class r/CryptoCurrency:

96.54% of the predictions made by the Multinomial Naive Bayes model were accurate (success rate), which means 3.46% of the predictions were incorrect (error rate). In the specific class of r/CryptoCurrency, 80.98% were true positives (true positive rate, recall). 0.23% were false positives (false positive rate). 79.17% were not false positives (precision).

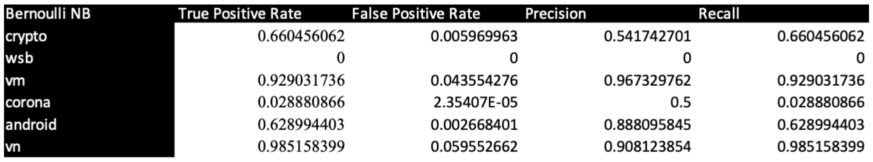
Results:



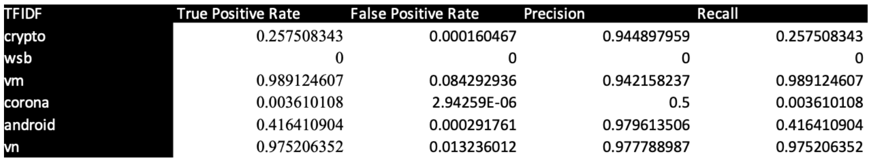
Model	Success Rate	Error Rate
Multinomial NB	0.965420418	0.034579582
Bernoulli NB	0.935971468	0.064028532
TFIDF Multinomial	0.951139912	0.048860088



Multinomial NB	True Positive Rate	False Positive Rate	Precision	Recall
crypto	0.809788654	0.002276253	0.791734638	0.809788654
wsb	0.019305019	1.47122E-05	0.5	0.019305019
vm	0.979690835	0.047472265	0.966269208	0.979690835
corona	0.14801444	9.41628E-05	0.561643836	0.14801444
android	0.622224228	0.002200367	0.904949455	0.622224228
vn	0.981235752	0.016312516	0.972929787	0.981235752



Bernoulli NB	True Positive Rate	False Positive Rate	Precision	Recall
crypto	0.660456062	0.005969963	0.541742701	0.660456062
wsb	0	0	0	0
vm	0.929031736	0.043554276	0.967329762	0.929031736
corona	0.028880866	2.35407E-05	0.5	0.028880866
android	0.628994403	0.002668401	0.888095845	0.628994403
vn	0.985158399	0.059552662	0.908123854	0.985158399



TFIDF	True Positive Rate	False Positive Rate	Precision	Recall
crypto	0.257508343	0.000160467	0.944897959	0.257508343
wsb	0	0	0	0
vm	0.989124607	0.084292936	0.942158237	0.989124607
corona	0.003610108	2.94259E-06	0.5	0.003610108
android	0.416410904	0.000291761	0.979613506	0.416410904
vn	0.975206352	0.013236012	0.977788987	0.975206352

The results are significant, because the preprocessing, which involves the removal of stopwords, was consistently applied for all the models before classifying text. The results of the various ML frameworks were not significantly different with a difference of only 0.3 between the best and the worst recorded accuracy. The order of the models with the best accuracy to the worst accuracy is as follows: Multinomial Naive Bayes (0.9654), TF-IDF using Multinomial Naive Bayes (0.9511), and Bernoulli Naive Bayes (0.9360).
However, it is possible that the results may be slightly misleading due to the nature of the ML frameworks and the imbalance between datasets. A possibility for its lower accuracy compared to the Multinomial Naive Bayes model, the Bernoulli Naive Bayes model only accounts for word occurrences that are present so it may have underutilized the data compared to the Multinomial Naive Bayes. The same may also explain why the TF-IDF model had a slightly lower accuracy than the Multinomial Naive Bayes model since it was more selective when classifying the text; the TF-IDF model accounts for the importance of the words

rather than solely counting the occurrence of the words. Also, the subreddits' sample sizes varied greatly, in which the smallest sample size was 973 entries belonging to r/Coronavirus and the largest sample size was 790958 entries gathered from r/CryptoCurrencies. Therefore, there was a great imbalance between the different datasets that were used to train and test the model. More information on the subreddits' sample sizes can be found in the **Preprocessing** section.

5. Conclusion

Three models were created to classify texts based on their respective subreddit: Multinomial Naive Bayes, Bernoulli Naive Bayes, and TF-IDF. Overall, the model with the best performance was the Multinomial Naive Bayes model with an accuracy of 0.9654, followed by the TF-IDF model with an accuracy of 0.9511, and the Bernoulli Naive Bayes model with an accuracy of 0.9360. Although the results seem significant due to the standard procedure of preprocessing (ex. removal of stopwords) for text classification, there may be some flaws in the design, which can interfere with the results' accuracy. Some of which are the dataset imbalance and the nature of the machine learning frameworks that were used. The current work can be extended to generate a list of the most commonly used words found in each subreddit and compare the list across multiple subreddits. Another possible future work would be creating models that can generate a summary of the subreddit given a set of posts (corpus) found from the subreddit.