byrdwcrawford / NLP-Sentiment-Classification

Code Issues Pull requests Actions Projects Wiki Security Insights Setting

0 stars 0 forks 1 watching 1 Branch 0 Tags Activity

Public repository

1 Branch 0 Tags Go to file t Go to file + Add file Code ...

byrdwcrawford finished 35a2740 · 6 minutes ago

Images	almost done	yesterday
Notebooks	finished	6 minutes ago
PDF's	finished	6 minutes ago
.gitignore	Initial commit	2 weeks ago
NLP Notebook-ColumnTrans...	finished	6 minutes ago
README.md	finished	6 minutes ago
tweets.csv	starting	2 weeks ago

README

NLP-Sentiment-Classification

Will Byrd

Flatiron School 2024

Introduction

This is a multi-classification NLP project. Tweets from 2011 regarding the South by Southwest (SXSW) tech and arts conference are preprocessed and and analyzed to create ML models that can predict sentiment. I will create a variety of models and iteratively improve upon them-finally using ensemble methods to optimize the predictions.

Business Understanding

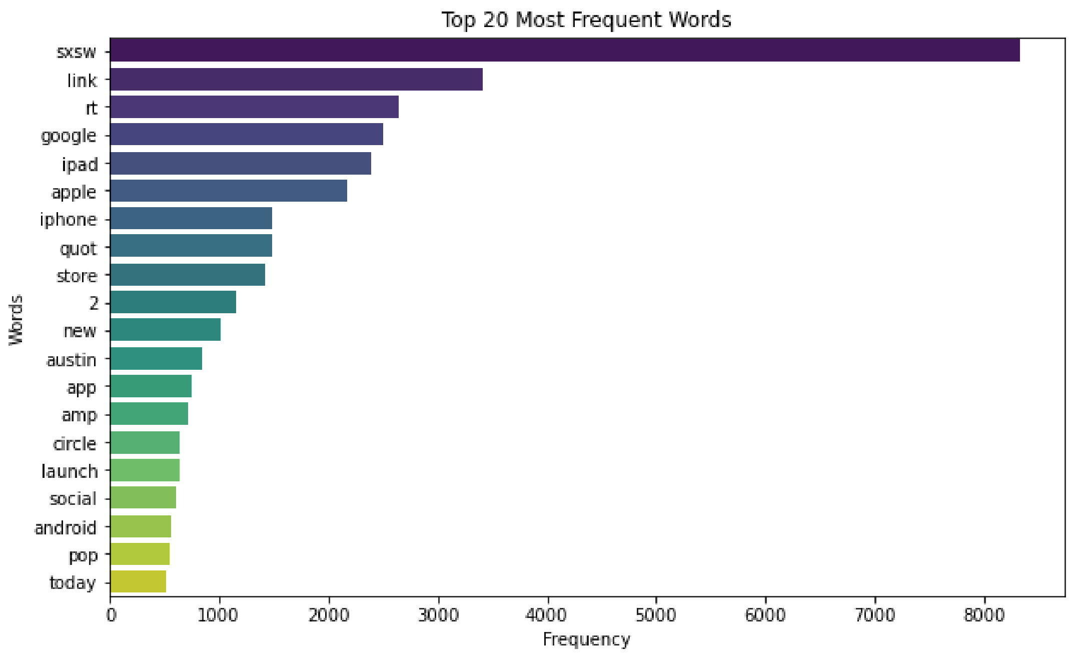
We have downloaded a dataset that contains over 8000 tweets. Tweets have been reviewed by humans to determine if the sentiment is negative, indifferent, positive, or if the human judge could not tell. Some of the tweets have also been attributed to be about specific brands like Apple, Google, or Android. Our dataset is from CrowdFlower via data.world.

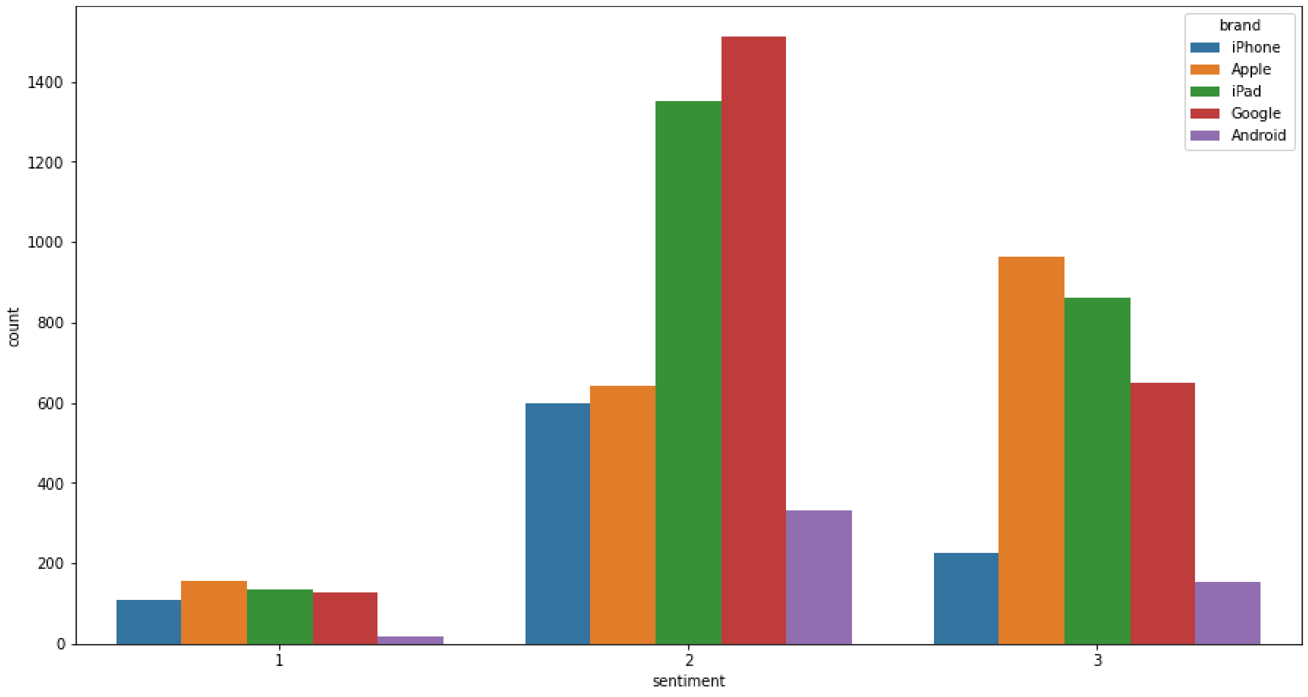
We have 2 main goals here. The first is to **understand overall public opinion of specific brands and products at the SXSW conference** and the second is to **demonstrate proper NLP preprocessing techniques.**

EDA

First thing we notice is an overwhelming amount of NaN values. We use imputation and mapping to double the amount of tweets attributed to specific brands and then consolidate the amount of brands to :

- iPhone
- iPad
- Apple
- Google
- Android



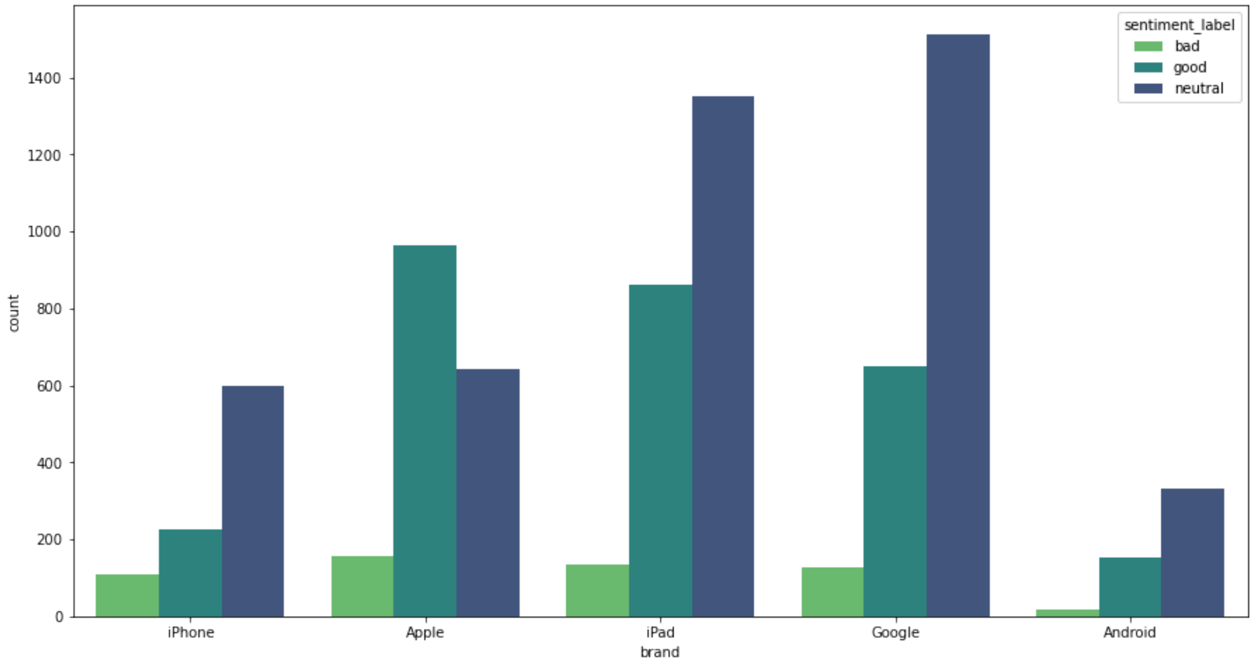


Before we can really do any analysis or modelling, the data must be preprocessed. We use the following steps in this notebook:

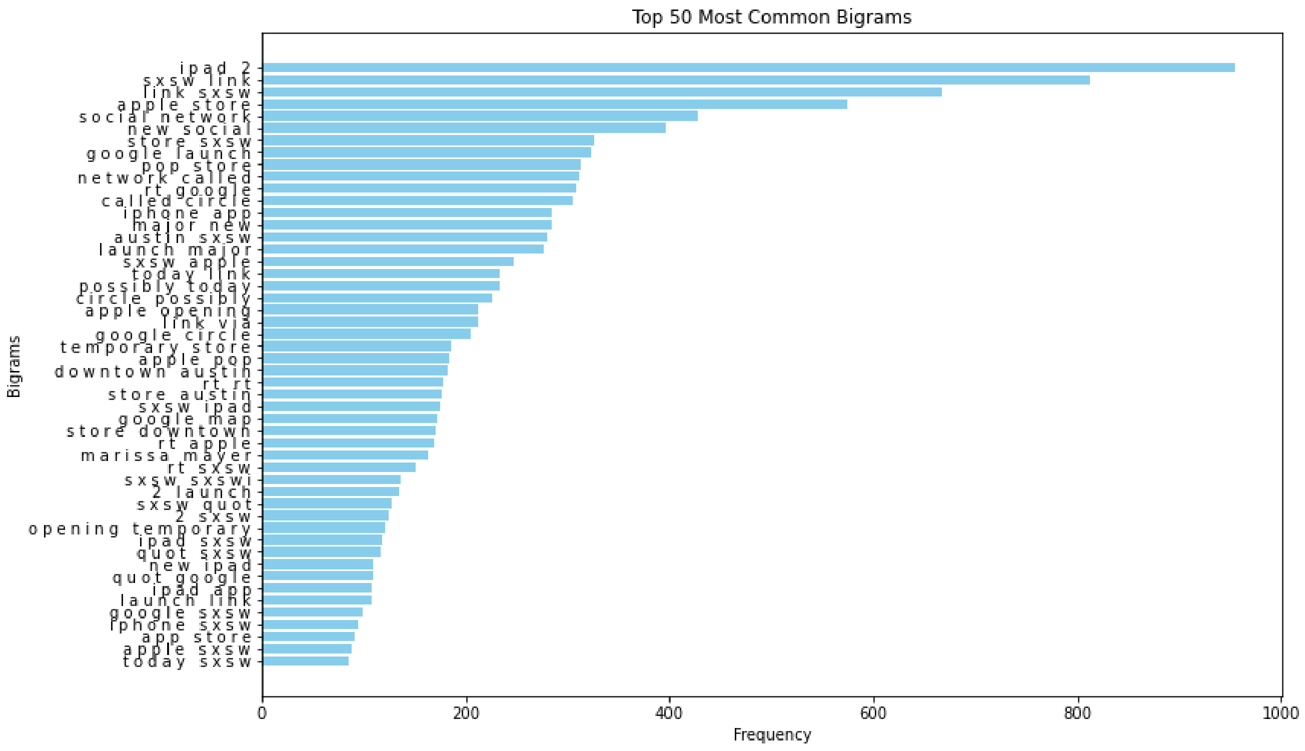
- Regex
- Lemmatization
- Stemming
- Removing Stopwords
- Removing nonword characters

Next step is to use LabelEncoder for our categorical variable **'sentiment'**. The rows where sentiment was unknown have been dropped since all NaN values will need to be removed prior to modelling. We can take advantage of the ordinal nature of this column making:

- Negative = 1
- Neutral = 2
- Positive = 3



Then **OneHotEncoding** is used for all of the brands mentioned above. Now we can focus on the tweets again. We will tokenize the tweets so that we can then vectorize them, creating a bag of words for our corpus. These 2 steps will allow us to view frequency of words in our corpus, create TF-IDF Histograms, and view bigrams and trigrams.



Another important step is to perform Part of Speech tagging. POS tagging is important to understand grammar and context of words and also helps improve modelling.

Modelling

Now that our data is cleaned and preprocessed for modelling, we begin to build the following models:

- Logistic Regression
- Decision Tree
- K-Nearest Neighbors
- Gradient Boosting

Each model will be improved upon using GridSearchCV which will find the best parameters for us. We will then take the best models and use **Stacking and Voting Ensemble methods** to create even better models.

Ensemble methods in ML is the process of combining multiple models to create one model that should outperform all others. This works, because the Ensemble method is able to utilize the strengths of the models and is able to sort out the weaknesses of each individual model.

In this notebook, we utilized 2 Ensemble Methods-Stacking and Voting. Here's a brief synopsis of how these work:

Stacking-Multiple models called base learners are trained on the dataset and then the predictions of these models are the input features of a secondary model referred to as a meta-learner.

Voting-the models are all trained independently and then combined via voting. There are two types of voting-hard voting and soft voting. In hard voting, the final prediction is determined bu majority vote. In soft voting, each model provides the probability estimate for each class and the final prediction is made by averaging these probabilities.

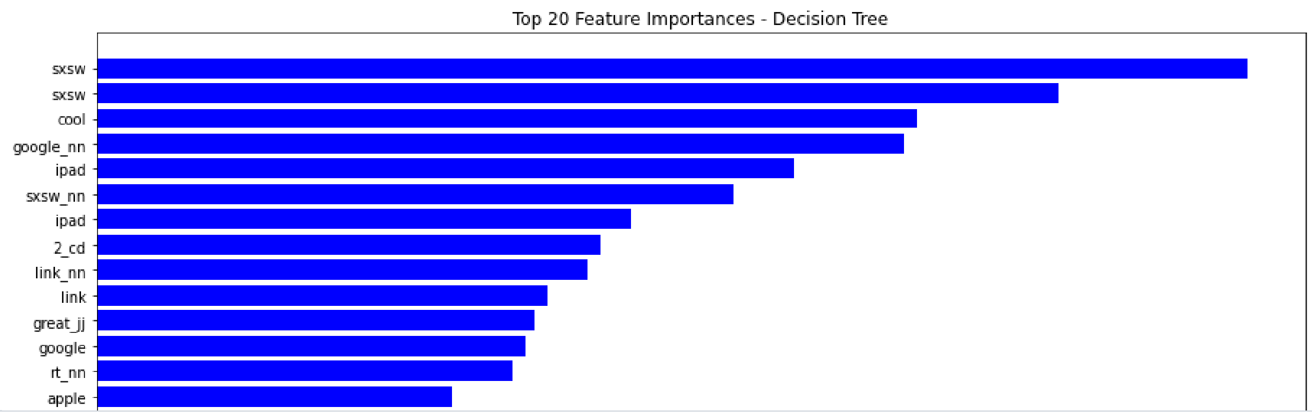
Conclusion

Here is what we can determine from our results:

- Our model can predict the correct class of tweet(negative, nuetral, positive) with 69% accuracy. This is actually fairly accurate as there are 3 classes, so a model that was purely guessing would be accurate roughly 33% of the time.
- Class 1
  - **Precision**- Our model predicts tweets fall into class 1 61% of the time.
  - **Recall**- The instances where the actual value of a tweet is class 1 were correctly identified 20% of the time
  - **F1-score**- Low F1-score is due to the sample size
- Class 2
  - **Precision**- Our model predicts tweets fall into class 2 69% of the time.
  - **Recall**- The instances where the actual value of a tweet is class 2 were correctly identified 85% of the time
  - **F1-score**- High F1-score is due to the sample size
- Class 3
  - **Precision**- Our model predicts tweets fall into class 3 68% of the time.

- **Recall**- The instances where the actual value of a tweet is class 3 were correctly identified 53% of the time
- **F1-score**- Moderate F1-score is due to the sample size

Recomendation


Top 20 Feature Importances - Decision Tree
sxsw
sxsw
cool
google_nn
ipad
sxsw_nn
ipad
2_cd
link_nn
link
great_jj
google
rt_nn
apple