Sentiment Classification Use Case

Introduction

As Twitter is one of the most popular platforms on which users can publish their thoughts and opinions, then we can use sentiment analysis to identify user experience or feelings towards a product or idea, which benefits the companies to enhance their performance.
So what is Sentiment Analysis, it is a NLP technique used to determine whether data is positive, negative or neutral. Sentiment analysis in Twitter tracks the problem of analyzing the tweets of the opinions the people express.
The most common type of Sentiment Analysis is “Polarity Detections” which involves classifying statements as positive, negative and neutral.
So with the use of Sentiment Analysis we can know the opinion of the crowd. And this is what I am going to perform today. Using Machine learning and NLP techniques

Data Description

Twitter is a social networking app that lets its users post real time tweets. Tweets have many unique characteristics, which implicates new challenges and shape up the means of carrying sentiment analysis on it as compared to other domains.
Following some key characteristics of tweets:

- Message length: The maximum length of the tweet is 140 characters.
- Writing technique: The occurrence of incorrect spelling and cyber slang in tweets is more often in comparison with other domains, As the messages are quick and short. Also people tend to use acronyms, misspellings, and use emojis and other characters that convey a special meaning.

Our dataset has 74681 rows and 4 columns (“TweetID”, “Entity”, “Sentiment”, “TweetContent”) for training data. And 1000 rows with the same columns for validation data.
Here is a sample from our dataset:

	TweetID	Entity	Sentiment	TweetContent
0	2401	Borderlands	Positive	I am coming to the borders and I will kill you...
1	2401	Borderlands	Positive	im getting on borderlands and i will kill you ...
2	2401	Borderlands	Positive	im coming on borderlands and i will murder you...
3	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder ...
4	2401	Borderlands	Positive	im getting into borderlands and i can murder y...

Baseline Experiments

The goal is to design a model for Sentiment Analysis to classify the tweet whether it is Positive, Negative or Neutral. In order to do that I performed the following this pipeline:

1. Import Necessary Dependencies
2. Read and Load the Dataset
3. Exploratory Data Analysis (EDA)
4. Word2Vec / BOW
5. Preparing data for modeling
6. Modeling
7. Ensemble techniques
8. Model Evaluation

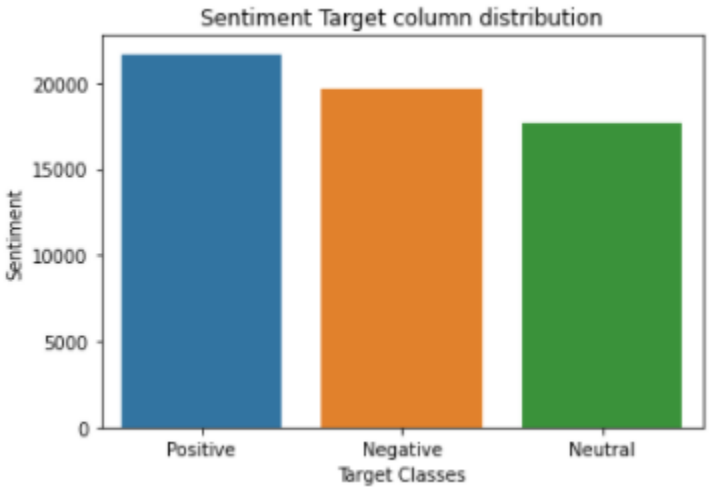
In the phase of reading data I faced the problem of encoding problems, and the encoding method I used to read the data was "`ISO-8859-1`" , but "`utf-8`" didn't work well.

<u>In the phase of EDA we have the following characteristics:</u>
Training data statistics:

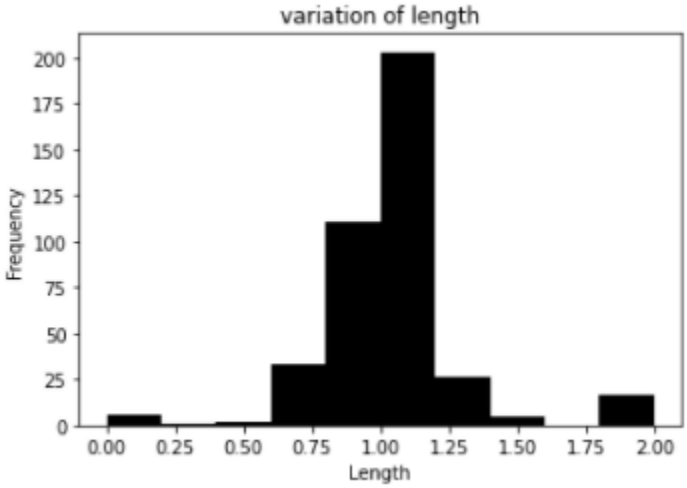
Sentiment	count	mean	std	min	25%	50%	75%	max
Irrelevant	12990.0	5928.771363	3616.510316	5.0	2942.0	5153.0	9369.0	13200.0
Negative	22542.0	6760.267767	3479.188211	1.0	3825.0	7105.0	9384.0	13194.0
Neutral	18318.0	6494.270881	3959.853892	3.0	3173.0	6904.0	10011.0	13197.0
Positive	20831.0	6338.113821	3850.921687	12.0	2755.0	6001.0	9708.5	13198.0

I can notice that we have a good feature distribution, we should be aware of this step as it is an essential step to make sure we don't have biased data as this problem affects the model performance.

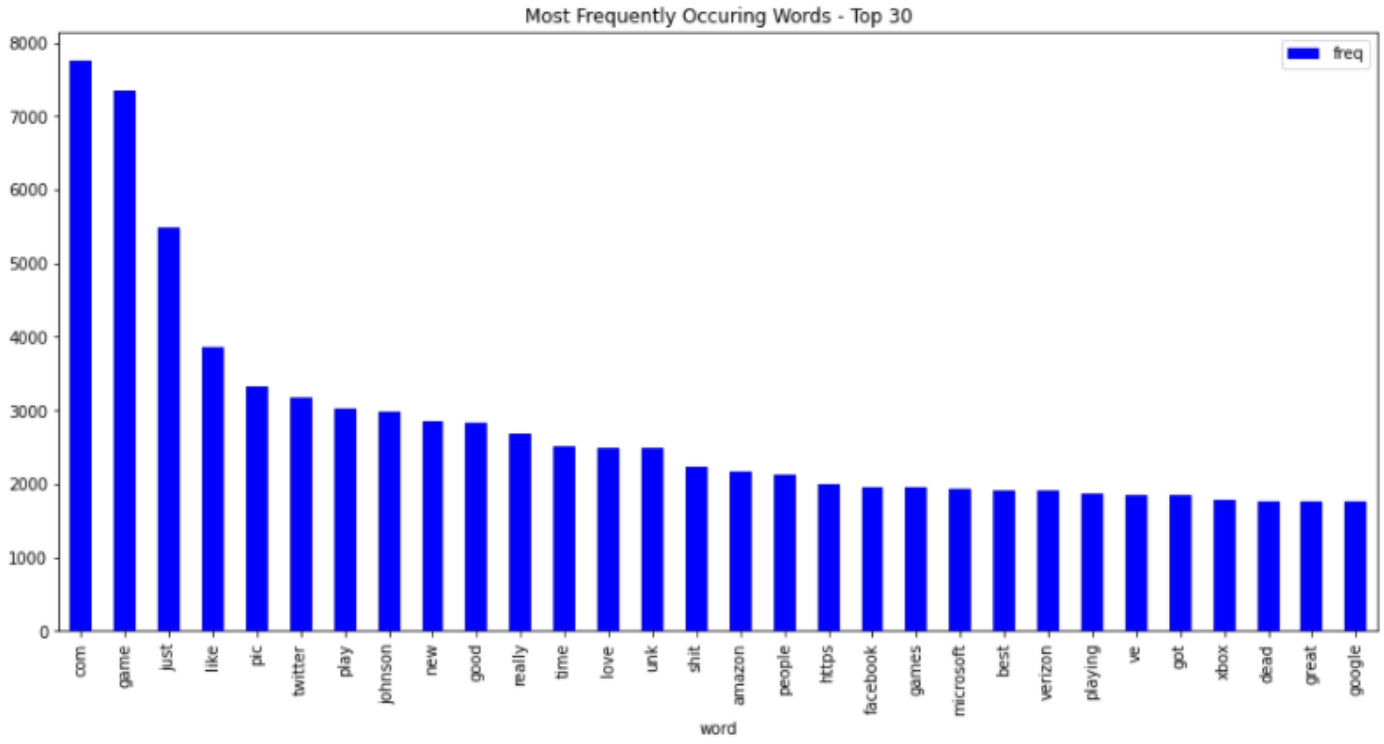
Target Distribution represented in 3 target classes, 0 for "Positive" Tweets, 1 for "Negative" tweets, and 2 for "Neutral" Tweets. I can notice that there is a fair feature distribution in the target classes.



Distribution of tweets lengths: We can notice that most of the length is around (0.75~1.25) and we have an approximate normally distributed features here



Most frequent words repeated in the entire dataset:



Word cloud is an approach to visualize words and counts of words from reviews columns of the dataset, which artistically depict the words at sizes proportional to their counts.

In the phase of applying Word2Vec and BOW:

- *Word2Vec model*:
  The usage of Word2Vec model was for the purpose of detecting synonymous words or suggesting additional words for a word, as the Word2Vec algorithm uses a NN model to learn associations from a large corpus of text. Word2Vec represents each distinct word with a particular list of numbers called a vector. The Vectors are chosen carefully such that a simple mathematical function (the cosine similarity between vectors) indicates the level of semantic similarity between the words, and I can see that here in the given example from the data for texting the word “Twitter” I got the following:

```
[('account.My', 0.5589950084686279),
 ('com?', 0.5582747459411621),
 ('discriminating', 0.5309399962425232),
 ('Whoopsidoodle,', 0.5210988521575928),
 ('irony', 0.5190274715423584),
 ('user:', 0.5103992223739624),
 ('censoring', 0.5045260787010193),
 ('PS5.....', 0.49403929710388184),
 ('coulda', 0.48887813091278076),
 ('facebook,', 0.48229771852493286)]
```

We can see them in descending order, the word “account.My” is the closest word to “Twitter”.

- *Bag of Words (BOW)*:
  I have used the **PorterStemmer** while building the training the testing data. The PorterStemmer removes the suffixes to find the stem, it divides a word in regions and then replaces or removes certain suffixes, if they are completely contained in said region. And I joined back the data with space to build the train/test corpus.
  The BOW model then simplifies the representation of the words as in the model, a text is presented as the bag of its words, and it generates the features which will be fed into the upcoming models.

In the phase of preparing data for modeling:
I used **K-fold Cross Validation** to enhance the model accuracy as it is a resampling procedure used to evaluate the models I am going to use on our data sample. I have used k=10 as mentioned in the project descriptions, K refers to the number of groups that a given data sample is to be split into. As such the procedure is often called K-fold Cross Validation
Then I have applied **StandardScaler** to the data to scale it in the range between 1st Quartile and 3rd Quartile in order to get more robust data and remove any potential outliers as the median and the interquartile range provide better results and outperform the sample mean and variance.

In the phase of Modeling:

Model	Training Score	Validation Score
Random Forest	**0.967**	**0.91**
LGBM	0.716	0.696
Logistic Regression	0.713	0.696
XGBoost	0.615	0.615

After training our models I got these results, as we can see the RandomForest Model here in this problem outperforms the remaining models, with Validation Acc of 0.91.

In the phase of Model Evaluation
For the model evaluation of Voting Classifier I used the **Macro Averaging** method for the evaluations, as it computes the metric independently for each class and then takes the average hence all classes equally. We should only use it when we care about the overall performance of the classes, that's why I needed it here as I care about the 3 classes ("Positive", "Negative", "Neutral") to have predictions as close as possible.

Other experiments

I used Ensembling techniques In order to enhance the model performance, so I applied the Ensembling techniques using **VotingClassifier,** with voting hard and weight of [2, 1] which means more weight for RandomForest, and this method is a voting method assigns various weights to the classifiers based on specific criteria an takes the vote between the LGBM and Random Forest model, when I did that we have *Validation Score* of **0.91465** which is better than the Score of the Random Forest alone.
Also I tried to do some **hyperparameter tuning** to the 4 models we have but I couldn't get train the models to get the results due to some internet problems

Overall Conclusion

Machine learning techniques perform reasonably well for classifying sentiment in tweets. Random Forest outstands the rest of the models but using Ensemble techniques improved the accuracy as the single classifier (Random Forest) may not be the best approach. It would be interesting to see what the results are for combining different classifiers.

Tools

- Google Colab Notebooks
- Python 2.7.12
- Google Drive for storing data

External Resources

https://www.kaggle.com/tanulsingh077/twitter-sentiment-extaction-analysis-eda-and-model
https://towardsdatascience.com/step-by-step-twitter-sentiment-analysis-in-python-d6f650ade58d
https://www.analyticsvidhya.com/blog/2021/06/twitter-sentiment-analysis-a-nlp-use-case-for-beginners/
https://monkeylearn.com/blog/sentiment-analysis-of-twitter/
https://en.wikipedia.org/wiki/Sentiment_analysis
https://aclanthology.org/W11-0705.pdf
https://ieeexplore.ieee.org/abstract/document/6726818
https://ojs.aaai.org/index.php/ICWSM/article/view/14185/14034
https://github.com/sharmaroshan/Twitter-Sentiment-Analysis/blob/master/Twitter_Sentiment.ipynb
https://www-nlp.stanford.edu/courses/cs224n/2009/fp/3.pdf
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.2031&rep=rep1&type=pdf

Questions

What was the biggest challenge you faced when carrying out the project?

Actually I tackled the problem like many other problems I have faced before in competitions on kaggle. The task has some parts new to me in a technical manner, but I know it before theoretically. But I consumed time looking for a dataset with as nice feature distributions as possible. Also I consider the time as a challenge as I wanted to finished the task as fast as possible and I was managed to meet my own deadlines

What do you think you have learned from the project?

I learnt how to apply Word2Vec Algorithm and knew more about how it works internally. Also I knew more about the task of Sentiment analysis as I read papers about this task from various resources.