**Sentiment Analysis: How is Your Brand Viewed on Twitter?**

Introduction

One of the more interesting areas of data science is the ability to use machine learning to detect sentiment in a particular string of text. The processing and model aspects of text is interesting and sometimes conceived as difficult. This project will highlight how it is easy and straightforward to process text and create a sentiment model around it. From there a model will be deployed on a web-based dashboard to show sentiment of tweets about a particular brand over a period of time. By doing this a brand could identify any key events that lead to a positive or negative impact on the brand.

Abstract

With the advent of Social Media, all brands need to be aware how they are being portrayed online. It been common these days to see a brand make a poor choice or ignore customer feedback which results in the deterioration of the brand. Having the ability to view recent social media posts, in this project tweets from Twitter, can give businesses an easy way to identify any business opportunities or area of improvement. For this project, tweets about shopDisney.com (Disney’s eCommerce business) will be analyzed over a period of time from June 14th until June 21st. During this range of dates there was a planned product release that has caused some strong feeling. However, this release was cancelled (during the same time range). By looking at this range of dates, it will be determined if a single event can lead to a deterioration of the brand.

Research Questions

The two main research questions for this project are:

1. Is it feasible to use a dataset containing already containing classified tweets to create a sentiment model?
2. Can a dashboard be created in Python using Flask that would rival popular dashboard programs such as Tableau and Power BI?

Methods

In order to accomplish the goal, perform a sentiment analysis on tweets, a dataset must be located that a model can be trained on. The dataset that will be used is called Sentiment140. This dataset contains over 1 million tweets that have already been classified as one of three different polarities: 0, 2, and 4. (Sentiment140, n.d.) Tweets that are classified as 0 are considered negative, 2 is considered neutral, and 4 is considered positive. For purposes of modeling, the neutral tweets are going to be filtered out so the problem is just binary classification.

When it comes to modeling, the first methodology that was investigated was Apache Spark. Since there is a large amount of data, Spark might provide a quick and easy way to model. What was discovered is that while Spark could handle the data without a problem, when it comes to modeling there was issues. After attempting multiple times to create a logistic regression model, the resulting accuracy was around 30% (this on top of the fact that it took about an hour to create the model). This was not a desirable outcome as most of the time it would be incorrect. Because of these issues the methodology had to change. The alternative way to model is by using scikit-learn. The code required to create the model is very similar to Spark, making the adjustments easy. Ultimately this was the better way to model for this project because a more accurate model was created in a much shorter period of time. Also, it is easier to deploy the model from scikit-learn for the dashboard section of this project.

Diving more into the methodology of the modeling, the main libraries needed are nltk and scikit-learn. The nltk library is used in the first step of text processing to help tokenize the string. What’s great about this is that there is a tokenizer function just for processing tweets. Once the tokenizer is defined, it is integrated with the CountVectorizer function to return a matrix of token counts. The next definition is the TfidfTransformer which takes the matrix of tokens and normalizes it. Lastly, the LogistRegression function is used to create the model. Since this is a binary classification problem the liblinear solver is used. With these functions defined a Pipeline is created to make training as simple as one line of code.

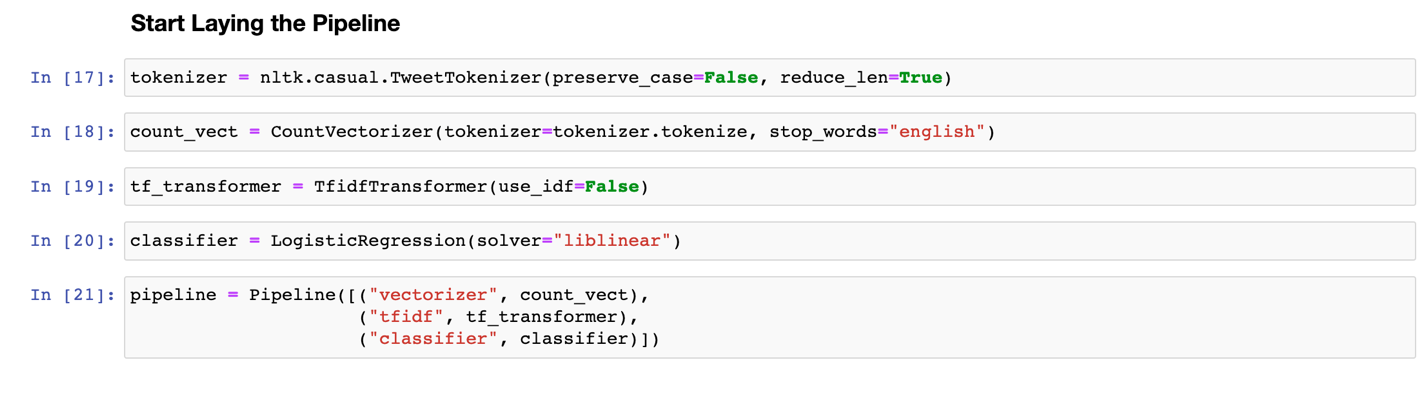


Figure 1 - Code sample showing all of the components of the Pipeline used to create the Sentiment Model.

In order to create a dashboard to view tweet sentiment, there must be a way to save/load the model as well as pull the data from Twitter. Saving and loading the model is easiest when you use the pickle library. Pickle makes it easy to save the model with a few lines of code. However, there is one caveat to this method. If this model is used, it should be used with the same version of scikit-learn it was created, otherwise there will be a few warnings when you run the dashboard. The tweepy library is one of the easiest ways to interact with the Twitter API. A single Python3 program was created to pull the data from Twitter and save it in a CSV file for consumption in the dashboard.

When it comes to the dashboard, there are multiple options to run the backend as well as the frontend. Since all of the code creating the model is in Python, it makes the most sense to use a Python backend. The simplest and easiest one out there is Flask. Flask is considered a microframework, in just a few lines of code you will have a web server up and running. (Hunt-Walker, 2018) When it comes to frontend, this project will go with the basics. Web pages written in HTML using Bootstrap 4 as its CSS/JS framework. In order to show plots on the dashboard, the JS library Charts.js will be used. This package makes it easy to create a multitude of graphs and supports variables being passed through Flask pretty easily. (Bjerrome, 2017) Since knowledge of Flask is basic, this dashboard will not be in real-time per say. When the server starts up a Python program will run that will fetch data from the Twitter API, process it, then save it to a CSV file so the server can process and show the information to the user.

Results

With all of the methodology outlined, it’s time to look at how the model performed! After training the pipeline and predicting the test labels, the resulting accuracy is around 81%. This accuracy is much better than the 30% that was achieved when created a model in Spark. In Figure 2 (below), a normalized confusion matrix is shown. From this figure it can be seen how each class performs separately. In this instance it performs (slightly) better at identifying positive tweets than negative tweets. While it might not be 100% accurate, for the purposes of this project it is accurate enough to give the user a good idea on the brands sentiment.

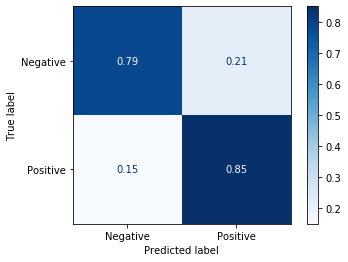


Figure 2 - Normalized confusion matrix showing the performance of the model.

Before moving onto the dashboard, a test case was used to determine if everything was working properly. This case looked at tweets from a different time range (June 7th – June 12th) and calculated the sentiment based on the trained model. Figure 3 (below) tabulates the results found for this time period.

|  |  |
| --- | --- |
| **Date** | **Sentiment Score (Positive)** |
| June 7th, 2020 | 0.58 |
| June 8th, 2020 | 0.50 |
| June 9th, 2020 | 0.65 |
| June 10th, 2020 | 0.64 |
| June 11th, 2020 | 0.63 |
| June 12th, 2020 | 0.59 |

Figure 3 - Table showing sentiment score for Tweets containing "shopDisney" from June 7th through June 12th.

This test showed it was feasible and easy to calculate the sentiment score. Now time to move onto the dashboard.

The goal of the dashboard is to show quick and memorable metrics about how the brand is viewed on Twitter. The first section displays four cards containing different metrics: total number of tweets, number of positive and negative tweets, and a rolling average (of positive tweets) over the last seven days. To view more detailed data, a trends section is featured below showing a line graph of the observed sentiment values. This plot features both positive and negative values. The purpose of this is to be able to identify dates where something went wrong. Figure 4 (below) shows the range of dates outlined in the abstract. During this time range there is a clear shift from June 17th to June 18th. Between these two dates sentiment shifted from around 70% positive to around 45% positive. What happened during this time range to cause a huge change? During this time shopDisney made an announcement that a popular product release had been cancelled due to unknown issues. While necessary, this action shows how event a necessary event can reflect badly on the brand. The good news is that days later sentiment seems to be returning to “normal” levels.

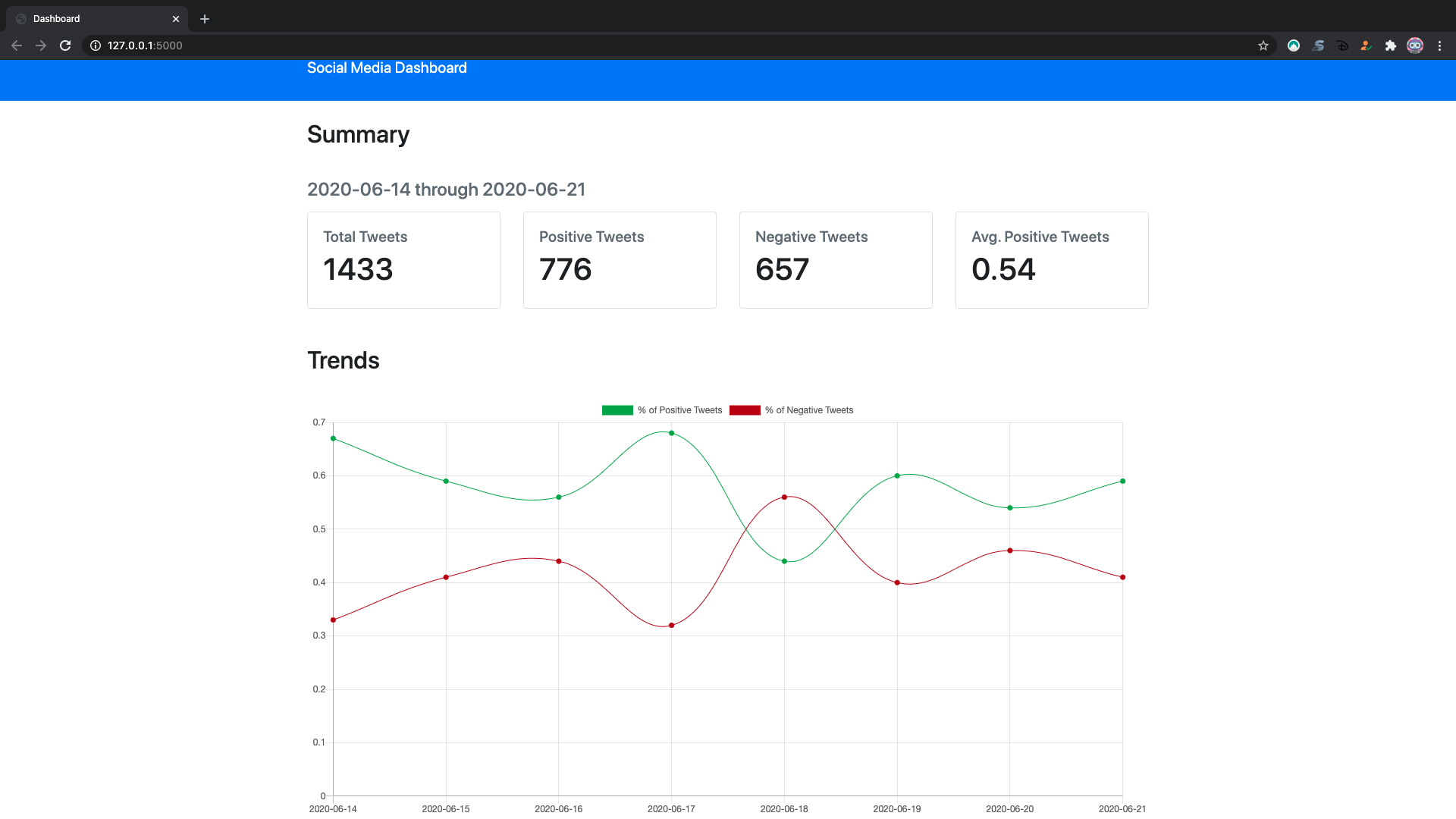


Figure 4 - Sentiment trends for tweets containing “shopDisney” from June 14th through June 21st

Conclusion

The goal of this project was to create a program that would help create a model to predict sentiment, then take this model and produce a web-based dashboard to view the results. What was discovered is that creating a model in scikit-learn was quite easy. By using a pipeline to tokenize, vectorize, transform, and classifier, a good performing model was created in under 30 minutes. Once this model was complete, the web-based dashboard was created. By using the pickle library, the model was able to be saved and imported easily. Ultimately the dashboard came together by using Python’s Flask and standard HTML/CSS/JS. In the end, this project shows that it is easy to take the power of social media posts and dissect them to get a better insight to your customers. Looking at the range of dates from June 14th through June 21st showed that between June 17th and 18th something happened to anger consumers. Having this information can help decision makers analyze what went wrong, how can it be prevented, and if it can’t what measures can be in place to soften the blow. Ultimately, if they are happy, keep doing what’s right, if they are not happy, time to make some changes.

# Bibliography

Bjerrome, T. A. (2017, March 5). *Data visualization with Chart.js: An introduction*. Retrieved from TobiasAhlin.com : https://tobiasahlin.com/blog/introduction-to-chartjs/

Hunt-Walker, N. (2018, April 2). *An introduction to the Flask Python web app framework*. Retrieved from OpenSource.com: https://opensource.com/article/18/4/flask

Sentiment140. (n.d.). *For Academics*. Retrieved from Sentiment140: http://help.sentiment140.com/for-students

Figures

[Figure 1 - Code sample showing all of the components of the Pipeline used to create the Sentiment Model. 3](#_Toc43051867)

[Figure 2 - Normalized confusion matrix showing the performance of the model. 4](#_Toc43051868)

[Figure 3 - Table showing sentiment score for Tweets containing "shopDisney" from June 7th through June 12th. 4](#_Toc43051869)