Sentiment Analysis of Cancer Immunotherapy-Relevant Tweets

Darren Wright
Indiana University
School of Informatics, Computing, and Engineering
Bloomington, IN 47408, USA

ABSTRACT

Research and funding for precision medicine continues to increase as big data technologies increase the potential for addressing a patient's individual, unique medical needs. Immunotherapy is a form of precision medicine that leverages a patient's immune system to fight cancer. It is a promising alternative for treating cancers not responsive to traditional treatments, though it also poses the danger of harming non-cancerous cells.

This paper discusses the development and results of Python code written to assess the sentiment of streaming tweets relevant to this topic to better understand the state of this field in real-time. The analytic pipeline resulting from this effort targets advancing improving real-time, topic-based social media analysis to establish a solid foundation for future research.

1 INTRODUCTION

Research and funding for precision medicine-based cancer treatments using the patient's own immune system to improve treatment results continues to grow. In particular, the field referred to commonly as cancer immunotherapy attracts medical professionals who treat cancers that do not respond well to traditional treatments like chemotherapy, radiation, and gamma knife. While immunotherapy poses significant risks, it provides hope to cancer patients with few or no treatment alternatives [1].

Analysis of social media may benefit the advancement of this field by leveraging real-time data to gain a more holistic look at the state of this field beyond clinical trials and journal articles [11]. In an approach akin to crowd-sourcing, this project is an initial foray into analyzing streaming Twitter data to establish a method for later work. This project provides insight into the following: (1) geographical locations with the highest frequency of immunotherapy-related tweets, (2) overall perception (*sentiment*) of cancer immunotherapy, and (3) suggestions for future research and analysis. The resulting code, written in Python 2.7.14, provides an analytic pipeline as depicted in Figure 1 to execute sentiment analysis of real-time Twitter data.

[Figure 1 about here.]

2 DATA

To develop training and testing data sets for use with a machine learning algorithm to predict sentiment, I developed two sets of data as follows:

- Train Data: Natural Language Toolkit's (*NLTK*) Twitter sentiment corpus, which provides two files: one with positive tweets and one with negative tweets formatted as shown in Figure 1. These two files were used to create the training set for the machine learning (ML) algorithm selected to predict the sentiment for collected tweets [19].
- Test Data: Approximately 1,300 tweets collected via a live-streaming Twitter feed. These tweets were then partially preprocessed while being saved as in CSV format for later usage.

[Figure 2 about here.]

For sentiment analysis, the code integrates NLTK's Twitter sentiment corpora. This corpora provides two files labeled `negative_tweets.json` and `positive_tweets.json`. As the file names suggest, one file contains positive tweets and the other negative tweets, but sentiment scores are not provided. NLTK only categorizes the tweets as positive or negatives based on the file they belong to,sentiment and polarity scores were not provided. Thus, each of the three files were processed and formatted to contain three columns – sentence (*tweet text*), polarity, and sentiment [12].

[Figure 3 about here.]

3 METHODS

Linking various packages to create a pipeline is difficult for streaming a Twitter feed directly into a sentiment analysis pipeline. This challenge is compounded by the numerous possible approaches to choose from related to Python's main three packages for Twitter streaming: Twitter, Tweepy, and Twython. After multiple variations were attempted to develop a code based on these three packages, Twython was selected due to its flexibility and compatibility with other packages like Pandas and CSV [7] [5] [20] [13]. This package is the only one that provided solutions to the following critical issues:

- automatically breaking with the Twitter feed,
- excluding all re-tweets,
- allowing for multiple search terms
- compatibility with the CSV package,
- limiting the content saved from each tweet,
- option to specify number of tweets and files.

Initial processing and visualization of the tweets was executed with Python packages Re and NLTK. NLTK's frequency distribution tool was used to create the line chart in Figure 4 [2]. This chart provides an initial insight into potential themes and qualitative sentiment associated with the topic of immunotherapy. Interestingly, there are no negative individual terms within the 40 most common words depicted in the chart.

[Figure 4 about here.]

Seaborn and Matplotlib.pyplot were used to visualize locations with the most tweets as shown in Figure 5. Although this data sample is small, previously collected at least 5000 tweets that also showed an overwhelming majority of immunotherapy-related tweets originate from the United States. Europe and Australia also had a significant amount of tweets, though all other countries fell far

short of these three locations. However, search terms in other languages like Chinese, Japanese, and Arabic were not included, which may significantly skew the data.

[Figure 5 about here.]

Figure6 also demonstrates the rather positive sentiment toward immunotherapy. As this figure shows, the majority of tweets are of neutral sentiment, though there are several small peaks on the positive side that outweigh the single peak on the negative side of the chart.

[Figure 6 about here.]

Each file was then analyzed for sentiment using TextBlob, which provided both sentiment and polarity scores (Figure 6). Since NLTK already provided a positive and negative file where sentiment had already been assessed, TextBlob processing simply provided more accurate scoring [10]. Sentiment analysis performed on the collected tweets can be used later to compare this method with the machine learning approach described later in this report.

Before settling on an algorithm to predict tweet sentiment, several sub-modules from Sklearn were used cross-validation and accuracy scoring for three different machine learning algorithms: naive Bayes, support vector machine, and Sklearn's multi-layer perceptron (MLP) neural network [3] [22] [17].

[Figure 7 about here.]

[Figure 8 about here.]

4 RESULTS

The neural network algorithm performed the best out of the three original algorithms considered for this project, which also included Naive Bayes and Support Vector Machine. Sklearn's neural network algorithm resulted accuracy scores above 60 percent. However, since the desired score was 80% or more, additional research posited the Extreme Gradient Boosting (*XGBoost*) algorithm as a likely better candidate [4] [14] [16]. After testing this algorithm with the rest of the code, it provided significantly higher results with a top accuracy score of 77%.

Since 2016, multiple Kaggle competition winners successfully applied XGBoost along with Pythonfis Scikit-learn (*Sklearn*) to improve machine learning algorithms for processing disparate, sparse data. Academic research further supports the utility of XGBoost as an optimal approach to increasing the precision (the precision (efficiency + accuracy) through more fully leveraging data and handling imbalanced data. The website Analytics Vidhya sums up the advantages of using XGBoost as follows [9]:

- reduces overfitting,
- parallel processing through building parallel decision trees,
- highly flexible, providing adjustable parameters for optimization and evaluation criteria,
- handles missing values,
- removes splits in the tree when they are no longer beneficial.

Figure 9 shows one set of parameters used for XGBoost as an example of just a few of the aspects of the algorithm the user can adjust.

[Figure 9 about here.]

[Figure 10 about here.]

Despite the relatively computationally intensive parameters selected after several trials (Figure 5), XGBoost functioned without a hitch. The program ran through 24 iterations of the code in as little as 20 minutes on a notebook computer (although the highest score was achieved with a runtime of 52 minutes). The program ran on a Dell Inspiron 7559 with the following specs:

- Intel Core 5th-generation i5-6300HQ (2.30GHz, quad-core)
- Ubuntu 16.14 run on Oracle Virtual Box with Windows 10 as host system
- 500GB Samsung SSD 850 EVO
- NVidia GeForce 960M (4GB)

Access to multiple adjustable parameters proved beneficial. One parameter that made a significant difference is the *nthread*, which defines the number of CPU cores used, significantly improved accuracy and speed. By increasing nthread (number of CPU cores used) from 1 to 4, the difference in computing time for each of the folds went from around 5 to 8 minutes to between 30 and 45 sections each. Despite using all 4 cores on the laptop, other programs not memory-intensive ran without any major lag. However, using 4 cores and increasing max_depth had a large impact on computer performance.

5 DISCUSSION

This project aimed to create an easily executable program providing insight for topic-based research through leveraging real-time, streaming social media. The resulting code allows for future development of additional training and testing data sets as new data becomes available. The code also contains the following unique characteristics:

- creates integrated pipeline for streaming, storing, processing, visualizing, and saving analysis in a single program;
- scalable as multiple search terms can be queried on separate virtual instances running the programming;
- decreases data bias by excluding re-tweets;
- allows for flexible searches by allowing user to use more than one search term;
- executable in either Python 2.7 or 3+.

Overall, the analytic pipeline discussed in this report was successful in achieving project goals. Multiple issues related to streaming, collecting, and storing tweets plaguing the first half of this project were resolved. Moreover, multiple packages and methods for analyzing sentiment were integrated to make the pipeline more robust. The issues discussed later in this section outline some of the unique challenges posed by open source programming, especially to newer Python users.

Before discussing the issues, several data scientists who are generous with sharing their work online along with open source Python publications were critical in the success of this report [21] [15]. Most significantly, Dr. Jason Brownlee, the author of the website *Machine Learning Mastery*; Brett Romero (www.brettromero.com; and Tianqi Chen, the author of the XGB script benefited this project from the micro level and on the macro level continue to contribute to the international data science community [3] [18] [4].

Challenges: Significant issues, both expected and unanticipated, appeared consistently throughout the course of developing the code. The lack of sufficient and useful approaches to using Python

for analyzing streaming Twitter or other social media data was surprising, requiring me to spend a lot of time either manipulating existing code or developing my own to accomplish were seemingly simple tasks, including:

- real-time processing and saving tweets to CSV-formatted files;
- decreasing data bias by excluding re-tweets;
- analyzing sentiment for newly collected data;
- converting tweet text to a binary format;
- having the program break when a specified number of tweets have been saved.

Python's rapid evolution: Python and its packages evolve and/or become deprecated at a rapid pace. This element of using Python as the programming language posed significant challenges throughout the course of completing this project. Changes in code compatibility created difficulties. Since the original code often worked when tested, it became very difficult to resolve problems that appeared as authors of the respective python package either changed or abandoned the package while other packages needed to complete the code became incompatible.

Incompatibility of new Python packages: Python 2.7+ is increasingly difficult to use to integrate updated or new packages. One frustrating instance arises from Python 2.7's handling of text encoding. When using Textblob and Pandas to process tweets for sentiment, Python processed the text as ascii-encoded, but Textblob requires UTF-8. Using Jupyter Notebook further compounded the issue because the simplest solution fi?? using *reload(sys)* and then setting the default encoding to *utf-8*fi?!caused the results of the code to print in the terminal window and not the notebook itself. There is no resolution for this small, albeit very inconvenient, issue.

Useful test cases are scarce : Many aspiring data scientists post or publish their knowledge of Python to collect and store, process, and visualize real-time streaming social media. However, only after attempting to fit their techniques to real-life application for topic-based analysis did I discover that virtually none use data sets not already processed and structured for immediate use with machine learning algorithms.

As an example, one tutorial involves a drawn-out, but seemingly promising, approach to conduct detailed sentiment analysis. After an extended period was spent figuring out why the model failed to work after modifying the code repeatedly, it was discovered that the pipeline used pre-processed data sets. The author never used the pipeline for processing original data to create training or testing data sets [6].

Another example regards examples of using Twitter's geolocation data to create maps either through Matplotlib's Basemap or another package named Folio. If a sufficient number of tweets containing geo data can be collected, both packages enable the user to create impressive graphics. However, immunotherapy-related tweets with geolocation data are very rare; out of 4,000 tweets collected at different times, only 10 included coordinates. The author used the query term Halloween, a generic search term, to mine tweets [8]. Thus, the results of this example are misleading because it there are many more tweets concerning Halloween than immunotherapy. In most cases, the *ReadtheDocs* content written by the program developers is the most useful information to use while selecting an approach and relevant packages to execute the code.

These challenges underscore the significant amount of work to do to provide sufficient programming to assist scientists and experts leverage big data and social media for topic-specific research. This project provides a solid foundation for future efforts to leverage streaming Twitter data and perhaps data from other platforms like Instagram, LinkedIn, and Facebook. However, the one-sided nature of the barchart (5) in this report depicting the tweet source locations is likely significantly skewed from not including foreign-language data. As such, future efforts will target integrating data from social media platforms from other countries.

Subsequent work for improving upon this project will look to analyze immunotherapy-relevant tweets over specified time intervals to track events or new approaches leading to protracted negative or positive perceptions. Network analysis is another target for later work. With ongoing collection of data and the addition of other sources of social media, the potential for more precise research for analyzing individual immunotherapy treatments will increase.

REFERENCES

[1] 2017. Immunotherapy: Precision Medicine in Action. Web Site. (Nov 2017). https://www.hopkinsmedicine.org/inhealth/policy-briefs/immunotherapy-precision-medicine-action-policy-brief.html

[2] Steven Bird. 2016. *Natural language processing with python.* OReilly Media.

[3] Jason Brownlee. 2016. Machine Learning Algorithm Recipes in scikit-learn. Web Site. (Sep 2016). https://machinelearningmastery.com/get-your-hands-dirty-with-scikit-learn-now/

[4] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16.* ACM Press. https://doi.org/10.1145/2939672.2939785

[5] Clay. 2013. Data mining Twitter for sentiments about the new consoles. Web Site. (Nov 2013). https://forum.quartertothree.com/t/data-mining-twitter-for-sentiments-about-the-new-consoles/73751/2

[6] Ryan Cranfill. 2017. Building a Sentiment Analysis Pipeline in scikit-learn Part 1: Introduction and Requirements. Web Site. (2017). https://ryan-cranfill.github.io/sentiment-pipeline-sklearn-1/

[7] Python Code Search Engine. [n. d.]. Twitter User Data. Web Site. ([n. d.]). http://pythonexample.com/code/twitter%20user%20data/

[8] ianbroad. 2014. Converting Twitter tweets into points? Web Site. (Oct 2014). https://gis.stackexchange.com/questions/119923/converting-twitter-tweets-into-points/119939

[9] Aarshay Jain, Pranav Dar, and Pranjal Srivastava. 2016. Complete Guide to Parameter Tuning in XGBoost (with codes in Python). Web Site. (Aug 2016). https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[10] Shubham Jain, Tavish Srivastava, Pranav Dar, Aishwarya Singh, and Faizan Shaikh. 2018. Natural Language Processing for Beginners: Using TextBlob. Web Site. (Feb 2018). https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/

[11] Buket Kaya and Mustafa Poyraz. 2017. Extracting Relations Between Symptoms by Age-Frame Based Link Prediction. *Lecture Notes in Social Networks Prediction and Inference from Social Networks and Social Media* (2017), 85fi??95. https://doi.org/10.1007/978-3-319-51049-1_4

[12] Nikhil Kumar. [n. d.]. Twitter Sentiment Analysis using Python - GeeksforGeeks. Web Site. ([n. d.]). http://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python

[13] mirosval. 2015. csv: writer.writerows() splitting my string inputs. Web Site. (Dec 2015). https://stackoverflow.com/questions/34285154/csv-writer-writerows-splitting-my-string-inputs

[14] Ismail Babajide Mustapha and Faisal Saeed. 2016. Bioactive Molecule Prediction Using Extreme Gradient Boosting. *Molecules* 21, 12 (jul 2016), 983. https://doi.org/10.3390/molecules21080983

[15] Tony Ojeda. 2014. *Practical data science cookbook.* Packt Publishing.

[16] phunter. 2016. xgboost with GridSearchCV. Web Site. (2016). https://www.kaggle.com/phunter/xgboost-with-gridsearchcv

[17] Sebastian Raschka and Randal S. Olson. 2016. *Python machine learning: unlock deeper insights into machine learning with this vital guide to cutting-edge predictive analytics*. Packt Publishing.

[18] Brett Romero. 2016. Data Science: A Kaggle Walkthrough - Cleaning Data. Web Site. (Mar 2016). http://brettromero.com/data-science-kaggle-walkthrough-cleaning-data/

[19] Edward Loper Steven Bird, Ewan Klein. [n. d.]. NLTK 3.2.5 documentation: Twitter HOWTO. Web Site. ([n. d.]). http://www.nltk.org/howto/twitter.html Last accessed on 11/05/2017.

[20] 200 success. 2014. python - Printing out JSON data from Twitter as a CSV. Web Site. (March 2014). https://codereview.stackexchange.com/questions/44349/printing-out-json-data-from-twitter-as-a-csv

[21] Al Sweigart. 2015. *Automate the boring stuff with Python: practical programming for total beginners*. No Starch Press.

[22] JT Wolohan. 2017. Module 9 - Algorithms. (2017). This is an mp4 file from the School of Informatics and Computing, Indiana University at Bloomington Fall 2017 Applied Data Science course.

List of Figures

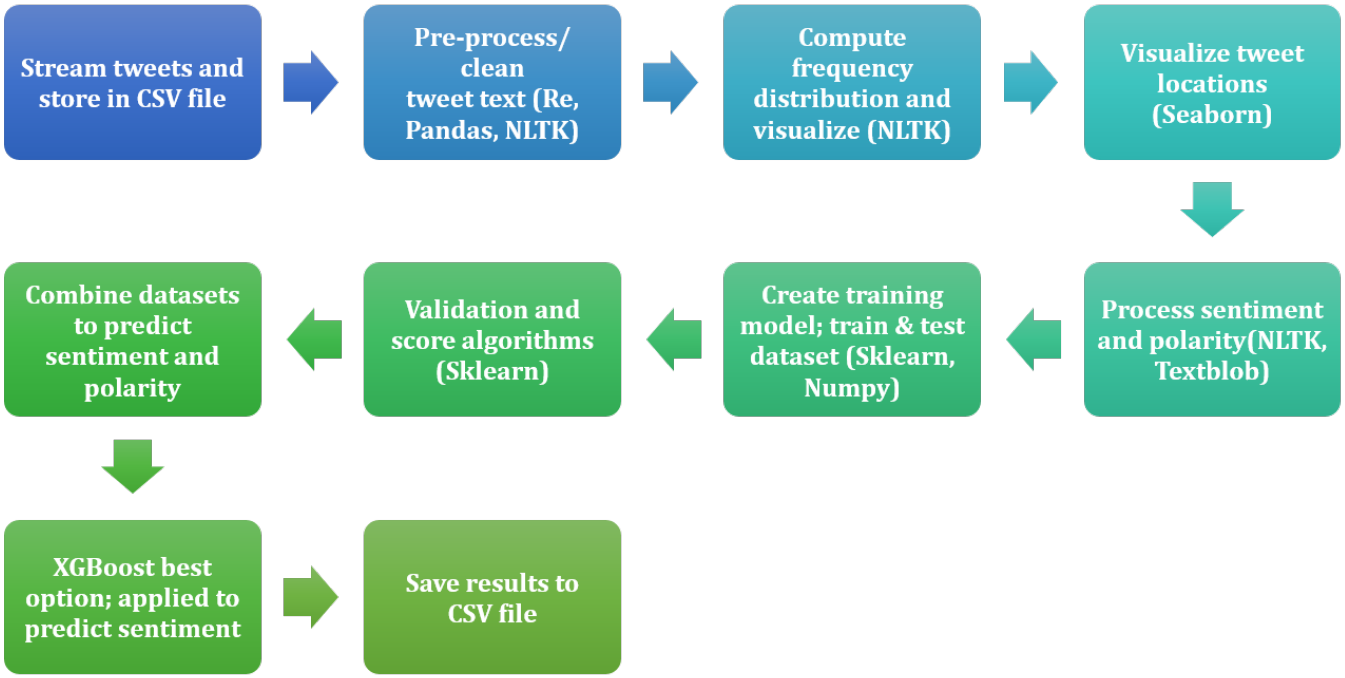

Figure 1: Overview of Sentiment Analysis Pipeline

```
In [4]:
1 df = pd.read_csv("/home/dj/Documents/positive_tweets.csv")
2 df.head()
3
```

Out[4]:

	Unnamed: 0	contributors	coordinates	created_at	entities	favorite_count	favorited	geo	id	id_str	...	possibly_sensitive	quoted
0	0	NaN	NaN	2015-07-24 08:23:36	{u'user_mentions': [{u'indices': [14, 26], u'i...	0.0	False	NaN	6.244951e+17	6.244951e+17	...	NaN	
1	1	NaN	NaN	2015-07-24 08:23:35	{u'user_mentions': [{u'indices': [0, 8], u'id'...	0.0	False	NaN	6.244951e+17	6.244951e+17	...	NaN	
2	2	NaN	NaN	2015-07-24 08:23:35	{u'user_mentions': [{u'indices': [0, 16], u'id...	0.0	False	NaN	6.244951e+17	6.244951e+17	...	NaN	
3	3	NaN	NaN	2015-07-24 08:23:35	{u'user_mentions': [{u'indices': [0, 8], u'id'...	0.0	False	NaN	6.244951e+17	6.244951e+17	...	NaN	

Figure 2: Positive Tweets file from NLTK

Out[5]:

	sentence	polarity	sentiment
0	sentence	0.0	0.00
1	Tweet	0.0	0.00
2	['#AmyloidosisSupportGroups thanks @ldanderson...	0.1	0.25
3	['Cancer drugs shed light on rheumatism\n\nhtt...	0.4	0.70
4	["22 (3-3): Molecular 'magnets' could improve ...	0.0	0.00

Figure 3: Collected Tweets Processed Using TextBlob

]images/linechart.png.pdf]images/linechart.png.png]images/linechart.png.jpg]images/linechart.png.mps]images/linechart.png.jpeg]images/linechart.png.jbig2]images/linechart.png.jb2]images/linechart.png.PDF]images/linechart.png.PNG]images/linechart.png.JPG]images/linechart.png.JPEG]images/linechart.png.JBIG2]images/linechart.png.JB2]images/linechart.png.eps

Figure 4: Tweet terms with highest frequencies

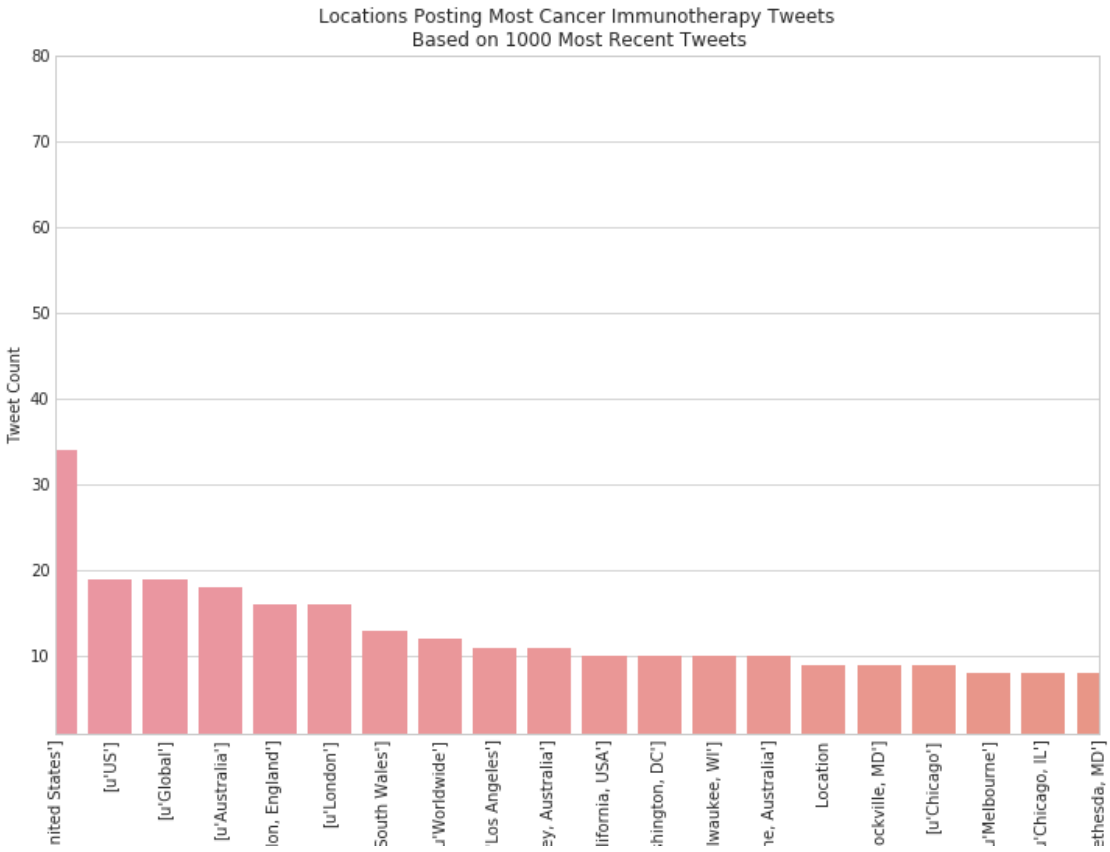

Figure 5: Locations with most immunotherapy-related tweets

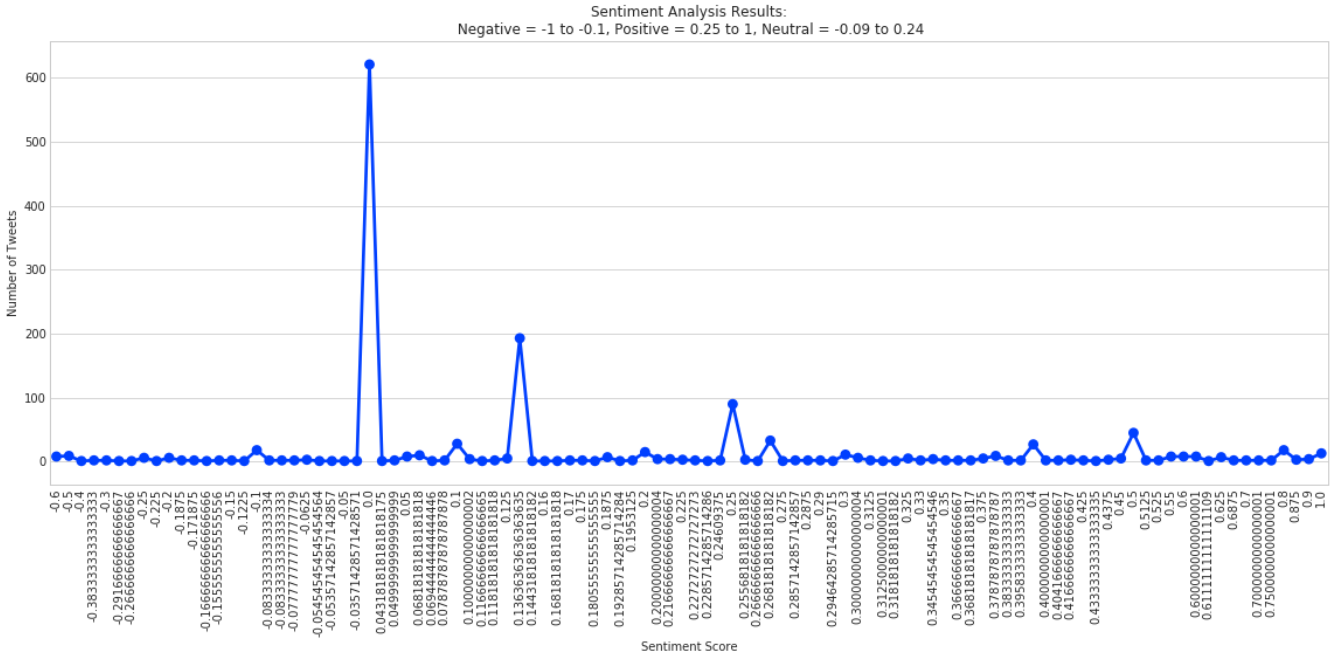

Figure 6: Locations with most immunotherapy-related tweets

Out[5]:

	sentence	polarity	sentiment
0	sentence	0.0	0.00
1	Tweet	0.0	0.00
2	['#AmyloidosisSupportGroups thanks @ldanderson...	0.1	0.25
3	['Cancer drugs shed light on rheumatism\n\nhtt...	0.4	0.70
4	["22 (3-3): Molecular 'magnets' could improve ...	0.0	0.00

Figure 7: Sentiment and Polarity Scores after TextBlob Processing

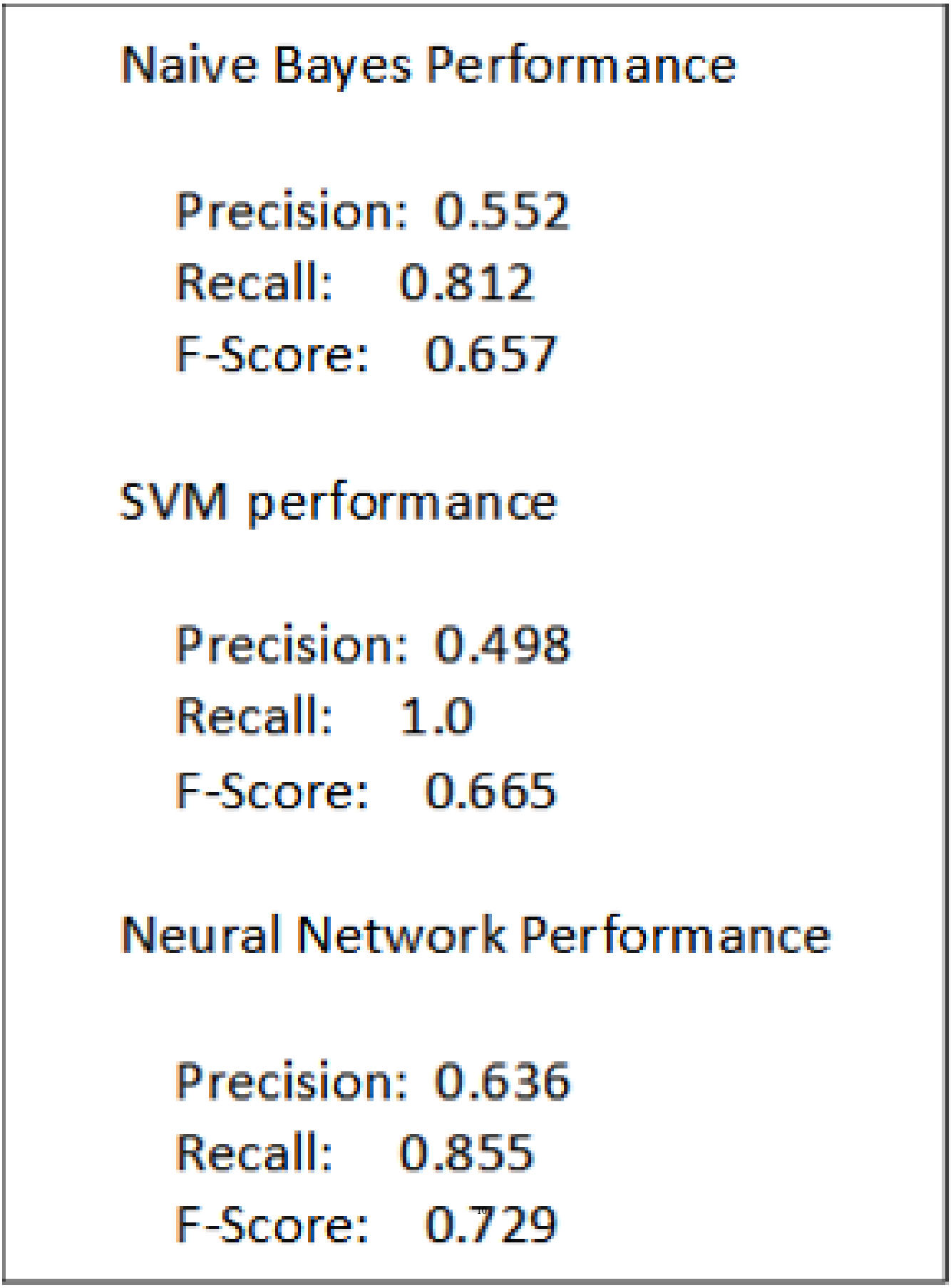

Figure 8: Performance of Naive Bayes, Support Vector Machine, and Multi-Perceptron Neural Network

```
  if diff:
[CV] ... learning_rate=0.3, max_depth=3,
         score=0.751200 -  39.6s
[CV] learning_rate=0.3, max_depth=3 ...
```

Figure 9: Top Accuracy Score Predicting Sentiment with XGBoost

```
[I 01:38:34.287 NotebookApp] Starting buffering for 4a645546-1510-4e05-ae52-
a0b405d4e67d:8d2f093ad74b4c15872c7c8de04643cf
GridSearchCV(cv=4, error_score='raise',
      estimator=XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
      colsample_bytree=0.5, gamma=0, learning_rate=0.1, max_delta_step=0,
      max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
      n_jobs=3, nthread=2, objective='binary:logistic', random_state=0,
      reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
      silent=True, subsample=1),
      fit_params={}, iid=True, n_jobs=1,
      param_grid={'learning_rate': [0.2, 0.3], 'max_depth': [3, 5]},
      pre_dispatch='2*n_jobs', refit=False, scoring='accuracy',
      verbose=10)
Fitting 4 folds for each of 4 candidates, totalling 16 fits
[CV] learning_rate=0.2, max_depth=3 ..................................
```

Figure 10: Example of XGBoost Parameters