Data Mining

Using NLP to Predict Authenticity of Tweets about Disasters

Fall 2022

Jake Brehm
32381023
jbrehm@ur.rochester.edu

Josh Wang
32376968
jwang247@ur.rochester.edu

Contents

1 Abstract

Our project uses data provided by Kaggle that contains information about tweets which use disaster-related language. Using this data, we aimed to predict whether or not a tweet was announcing a real disaster. To do this, we used multiple techniques, including topic modeling, naive Bayes classification, and long-short term memory neural networks, and evaluated them using k-fold cross-validation. Other exploratory analysis techniques were used, such as sentiment analysis and word clouds, in order to determine if there was any correlation between tweet sentiment and authenticity of a disaster. Our project serves as a baseline for real time disaster monitoring tools.

2 Introduction

2.1 Background and Motivation

"Wisdom of the crowd" is the idea that collective human knowledge can be more accurate than that of a single expert, due to the fact that a large number of responses can cancel out the noise associated with each individual judgement.[1] The advantage of wisdom of the crowd extends to crowd-sourcing in general: it is believed that groups of people are collectively better at problem-solving and innovating than an individual.

In recent years, Twitter has become an important communication channel in times of emergency. By taking advantage of the concept of crowd-sourcing, it is postulated that tweets can be analyzed to provide accurate and real-time reports of natural disasters. Because of this, many agencies such as news stations and disaster relief organizations are actively monitoring Twitter for natural disaster reports.

However, human speech has an overwhelming amount of complexities that can make it unclear whether a tweet is announcing a real natural disaster or not. These complexities include aspects of speech such as sarcasm, figures of speech/idioms, usage of emojis, etc.[2]

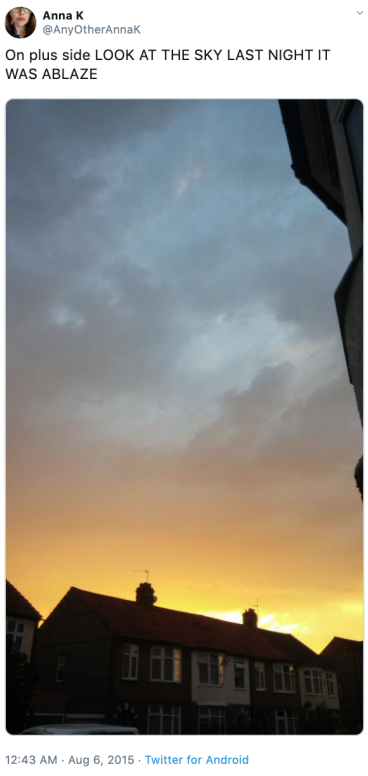


Figure 1: Tweet using figure of speech

For example, a twitter user tweeted that the sky was "ablaze", as shown in Figure 1.[3] While it is clear to humans that this is a figure of speech–the Twitter user is not seriously trying to report that the sky was on fire–it can be a challenge for a machine to discern the meaning of the tweet.

In this project, our aim is to build a predictive classification model to discern which tweets are about real natural disasters and which tweets are not.

Using natural language processing (NLP) techniques, we will process and analyze unstructured text data from Twitter.[4] The final product of our project could serve as an impactful utility for disaster-monitoring agencies to more accurately monitor the occurrence of natural disasters.

2.2 Problem Statement

> *"How accurately can we predict whether or not a Tweet is announcing the occurrence of a natural disaster using NLP?"*

3 Dataset

3.1 Overview

The dataset that was used originates from this Kaggle competition, titled *Natural Language Processing with Disaster Tweets*, and contains tweets that were hand classified: "1" for disaster or "0" for not a disaster. This dataset was created by the company *Appen* (formerly *figure-eight*) and originally shared on their *Data For Everyone* website here.

Since the dataset comes from a Kaggle competition, it has already been cleaned, at least to a reasonable extent. However, that does not mean that it is immediately ready for analysis or training purposes; some additional data preprocessing needed to be performed before we could train any model or perform any analysis.

3.2 Data Format

The data was provided in two parts: `train.csv` and `test.csv`. As denoted by their appropriately chosen names, `train.csv` is intended to be the data set used for training any models, while `test.csv` was intended to be used to test those models.

We decided to combine these two datasets into one for exploratory visualizations such as word cloud and sentiment analysis. This allowed us to get a closer look at a larger data set instead of just being limited to training data, and potentially catch any extra data cleaning that we must perform. Once combined, the 7613 observations in the training data set and the 3263 rows in the testing data set became 10876 tweet observations in total.

For supervised learning models, we split the `train.csv` into 5 folds using *StratifiedK-Fold* for cross-validation. The `test.csv` was not used for supervised learning since no class label was provided.

The attributes of these data sets are as follows:

- `id`, which is a unique identifier for each observation/tweet. This attribute is not particularly useful when training models or performing any kind of analysis, as any particular id value has no relationship to any other value in its respective observation.
- `keyword`, which is a particular keyword that is found in the text of the tweet. A value for this attribute is optional, and thus may be blank.

- `location`, which is the location that the tweet is sent from. Further analysis could determine that the geographical location that a tweet is sent from could be a significant factor in determining if a tweet is announcing a real disaster. Since not every tweet will have such metadata, a value for this attribute is optional, and thus may be blank. We decided against using this feature since it is highly inconsistent and mostly missing.
- `text`, which is the text of a tweet. This is the main attribute that exploratory analysis–i.e. word cloud and sentiment analysis–and models will use.
- `target`, which denotes whether a tweet is about a real disaster or not. It is a binary value, so if a tweet is about a real disaster, its value will be `1`, and if a tweet is not about a real disaster, its value will be `0`. This attribute is located *only* in `train.csv`.

3.3 Data Preprocessing

Once `train.csv` and `test.csv` have been combined into a single dataset, which was done by reading both csv files into a single *pandas* dataframe, we were able to begin exploring the data and seeing what kinds of data cleaning needed to be performed.

One of the first things we noticed about the data is that many observations in the text column included URLs. These URLs could have been included in the tweet in many ways–most commonly, the user could have added the link to the tweet themselves, or they could have posted a photo which would have been replaced with a link by the Twitter API or manually when the data was being acquired. Having URLs in text is undesirable because of the natural language processing and other exploratory techniques that we wanted to perform. Therefore, it was determined that it would be best to remove URLs from each text value via regular expression replacement.

Similarly, we found that many text values were actually mentions–a type of tweet that contains an "@" symbol, followed by another user's Twitter username.[5] Because usernames are not typically relevant to the subject of a tweet, a username might, for example, skew the results of sentiment analysis. For this reason, anything immediately preceded by an "@" symbol was removed.

There were also a few hundred observations that included HTML entities in the text. An HTML entity is a string that begins with an ampersand (&) and ends with a semicolon (;). These entities are typically intended to display reserved characters, as well as invisible characters.[6] One of the most common HTML entities found in the dataset was the entity corresponding to the ampersand, `&`. In order to avoid having strings such as *amp* be interpreted as a word and subsequently included in analysis and visualizations such as the word cloud, these entities had to be removed. Doing this was quite simple, as it only involved using the `unescape` function from Python's `html` library.

Next, we wanted to clean each text value in such a way that it no longer had any special characters, with a few exceptions. By utilizing regular expression substitution, we were able to remove all non-alphanumeric characters from the text column of the dataframe, with the exception of spaces (to allow for the tokenization of individual words)

and quotation marks (to preserve certain grammatical features such as contractions). It was important that this was done *after* removing the HTML entities, as if it hadn't, the entities would never have been found and would be included in the later analysis, just without the non-alphanumeric characters.

Then, we noticed that some observations contained one of a few canned phrases such as

- "I liked a YouTube video"
- "I added a video to a YouTube playlist"

that were not actually written by the author of the tweet. Thus, we removed these strings in order to uphold the integrity of our analyses.

Finally, we converted all text to lower case and dropped all duplicates. The case conversion was done mostly in order to ensure that each word, which can be written in multiple ways, was treated by our analyses as a single word. This also benefits us by improving the aesthetics of our word cloud since it makes sure all words are in the same case, although this is quite subjective and overall a minor side effect.

4 Exploratory Analysis and Modeling

One of the first things we tried to do was determine if there was some kind of association between word count and whether or not a tweet was announcing a real disaster.

A visualization of this analysis can be found in Figure 2.

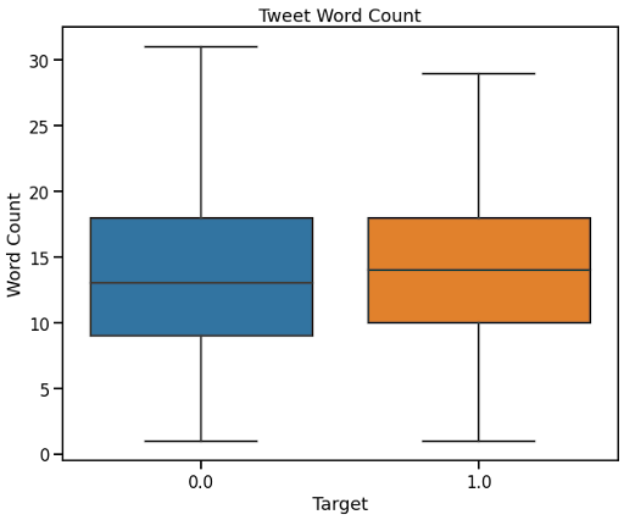


Figure 2: Word count analysis between types of tweets.

The boxplot clearly shows that there was no association between the number of words in a tweet and whether or not the tweet was about a real disaster.

4.1 Word Cloud

Word clouds are graphical representations of how frequently individual words appear in text; words that appear more frequently appear to be larger (and sometimes bolder) in the visual. While word clouds aren't normally useful for quantifiable analysis due to the lack of a size scale and the arbitrary nature of the word placement, it is a great way to illustrate the general sentiment or theme of a text.

In the context of this problem, generating a word cloud from the entire data set might not be very meaningful, as it would *only* depict the most frequent words–a good word cloud should also convey some theme or tell some kind of story through the most frequent words as well.

That said, as alluded to previously, word clouds can be useful tools to summarize the *views* of an audience or *theme* of a text. Thus, we split the training data set (i.e. data that has already has a `target` column) into two dataframes: tweets concerning real disaster, and tweets that are not about real disasters. Two separate word clouds were then generated, one for each data frame, and these visuals can be seen in Figure 3.

(a) Tweets *not* about real disasters.

(b) Tweets about real disasters.

Figure 3: Word clouds of tweets.

As demonstrated by the two word clouds, the frequent words in real disaster tweets are generally more negative, intense, and/or vulgar; comparatively, tweets that aren't about real disasters seem to have words that generally have more of a positive connotation. It is, however, clear as to how those tweets could be misinterpreted as announcements of real disasters. Many of the words in Figure 3(a) are words that have multiple meanings, or are used as slang terminology. For example, "fire" is a word that, in today's slang, could be a synonym for "cool"–however, this is a word that also obviously could be describing a real disaster. The tweet

> *this is about to be a bomb ass firework picture http://t.co/lr4BTvuEoM*

is a clear example of this phenomenon, except using the word "bomb".

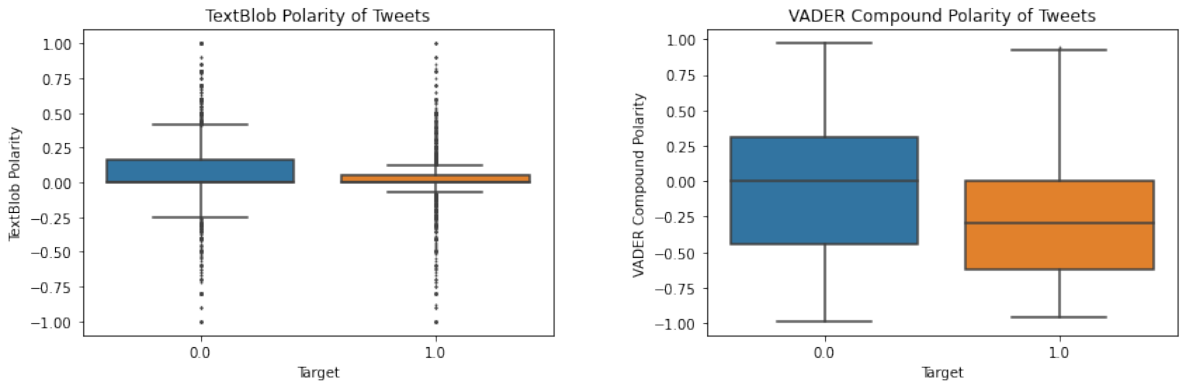
4.2 Sentiment Analysis

Two Python libraries were used to perform sentiment analysis: `textblob` (TextBlob) and `vaderSentiment` (VADER).

TextBlob is a lexicon-based sentiment analyzer that uses the NLTK (Natural Language Toolkit) and, by feeding it just a single string, it gives us a polarity score that ranges from -1 to 1. A score of -1 means that mostly words with negative connotations were used in the text, e.g. ”awful” or ”disgusting”, whereas a score of 1 means that the string was mostly positive.

VADER is another lexicon-based sentiment analyzer that gives a polarity score over four separate scores: positive, negative, neutral, and compound. Essentially, the input string is scored based on how positive, negative, and neutral its sentiment is, and these scores are returned separately; then, those three scores are normalized and returned to us as a compound score which ranges between -1 and 1. When comparing to results from TextBlob, we want to use VADER's compound score.

We used these two libraries to analyze the sentiment of both disaster and non-disaster tweets. The results are shown in Figure 4.



(a) TextBlob polarity.

(b) VADER compound polarity.

Figure 4: Polarity of tweets.

The polarity boxplots illustrate that there is quite a significant difference in polarity and overall sentiment between tweets that refer to a real disaster and tweets that do not refer to a real disaster.

Additionally, we performed two-sample t-tests for the results of both the TextBlob and VADER analyses, shown in Table 1.

Method	Test Statistic	p-value
TextBlob	8.6716	5.2427e-18
VADER	19.326	4.2857e-81

Table 1: t-test results for different sentiment analysis methods.

These results confirmed what was shown in Figure 4–the sentiments of disaster tweets are significantly different from the sentiments of non-disaster tweets.

4.3 Topic Modeling

In NLP, topic modeling is used for discovering the abstract ”topics” or hidden semantic structures that appear in a text or a collection of texts. In the context of this research, we can use a topic model to determine which topics were most frequently discussed in non-disaster tweets and compare those results to the topics that were found in disaster tweets.

To perform this analysis in Python, stop words were removed from the text using `spacy`, and the text was then lemmatized using the same library. We then performed tokenization using term frequency-inverse document frequency (TF-IDF), which is a numerical statistic that is meant to illustrate the importance of an individual word in a text. Then, we used three different dimensional reduction approaches to obtain topics from the tweets:

1. Non-negative matrix factorization (NMF) from the `sklearn` library
2. Latent dirichlet allocation (LDA) from the `gensim` library
3. Latent semantic analysis (LSA) from the `sklearn` library

After some testing, we found that NMF generated the most interpretable results. Figure 5 compares the results of the topic modeling for both non-disaster and disaster tweets.

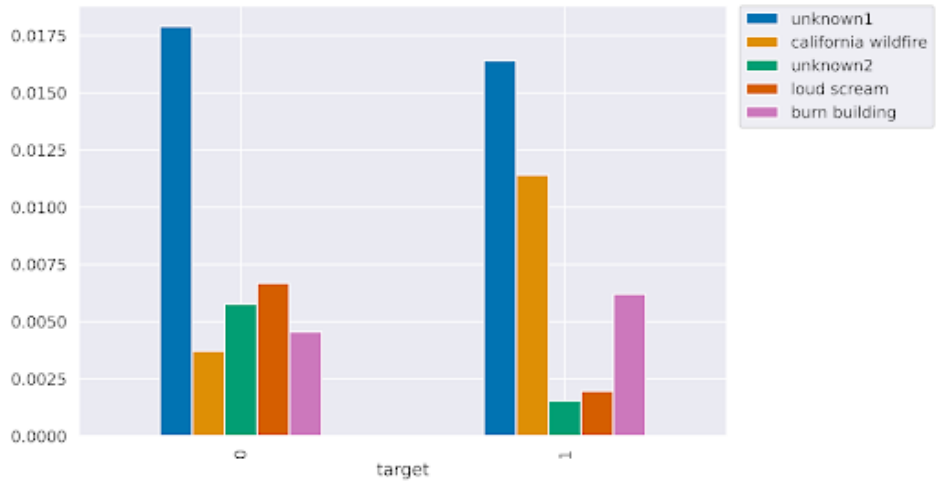


Figure 5: Results of topic modeling.

We can see that disaster-related topics: “california wildfire” and “burn building”, are more highly represented in disaster tweets, whereas random topics without discernible meanings: “unknown1”, “unknown2” and “loud scream” are more highly represented in non-disaster tweets.

The top 5 topics and the top 5 keywords of each topic are tabulated in Table 2 as well.

Topic	Keywords (most to least significant)
Unknown1	go, get, know, bomb, think
California Wildfire	fire, forest, wild, forest fire, truck
Unknown2	body, bag, body bag, cross, cross body
Loud Scream	scream, love, loud, hear, ass
Burn Building	burn, building, burn building, fire burn, crash

Table 2: Top 5 keywords for each of the top 5 topics.

4.4 Naives Bayes Classification

Naive Bayes classifiers are a family of multiple individual classification algorithms based on Bayes' Theorem. These algorithms share a common principle: "every pair of features being classified is independent of each other".[7]

Out of the collection of naive Bayes classifier algorithms, we decided to use Multinomial Naive Bayes (abbreviated MNB, called `MultinomialNB` in the `sklearn` Python library). In this implementation, the probabilities $P(feature|class)$ follow a multinomial distribution (word counts, probabilities, etc.).

First, we checked for class imbalance (Figure 6) in our data.

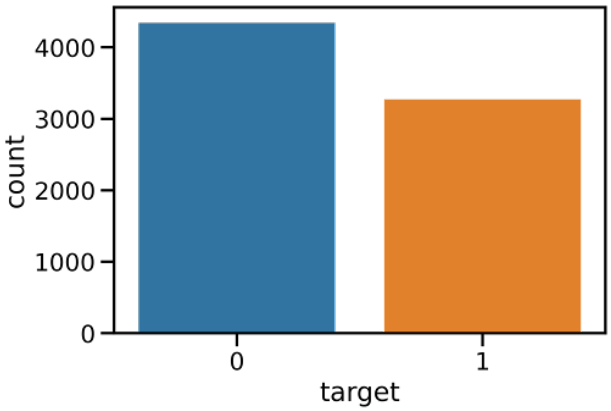


Figure 6: Results of naive Bayes classification.

We determined that there is no severe class imbalance, so there was no need to use methods such as undersampling or oversampling.

To begin our analysis in Python, we removed stop words from and lemmatized the text using the `spacy` library. To prevent any data leakage, we incorporated TF-IDF tokenization into an `sklearn` 5-fold cross-validation pipeline. Hyperparameters were tuned using GridSearchCV, which is part of the `sklearn` library.

Because there is no class imbalance, we decided to look closely at accuracy. However, since the use case of this research is to closely monitor natural disaster, and false negatives may lead to undesirable consequences, we thought it may be more important to focus on our model's ability to identify the positive class (i.e. the disaster tweets). Thus, we also looked at the F1-score, which is the harmonic mean of the precision and recall scores.

The baseline results for our classification were:

```
parameters:
    TFIDF_ngrams = (1,1),
    TFIDF_max_features = all,
    MultinomialNB_alpha = 1.0,
    fit_prior = True
accuracy: 0.7845
f1: 0.6949
```

The results when we tuned towards high accuracy were:

```
parameters:
    TFIDF_ngrams = (1,2),
    TFIDF_max_features = 5200,
    MultinomialNB_alpha = 0.40,
    fit_prior = True
accuracy: 0.7933
```

The results when we tuned towards a high F1-score:

```
parameters:
    TFIDF_ngrams = (1,2),
    TFIDF_max_features = 5200,
    MultinomialNB_alpha = 0.44,
    fit_prior = False
f1: 0.7293
```

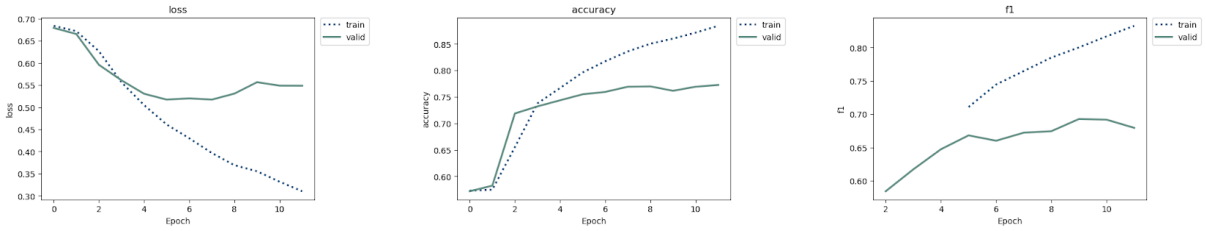
4.5 Long-Short Term Memory

Traditional neural networks (NNs) suffer from short-term memory and the vanishing gradient problem, which is especially apparent with longer data sequences. Long-Short Term Memory (LSTM) neural networks efficiently improve performance in this scenario by memorizing relevant information and then finding patterns. This makes LSTM great for NLP tasks, as it can memorize important information in sequential data.

We used Python's `torchtext` and `spacy` libraries to load the tweets and build vocabulary. We then used the `BucketIterator` from the `torchtext` library to group the tweets of similar lengths into batches. Having batches with similar length examples provides a lot of gain for recurrent NNs and transformers where padding will be minimal. We then used

3-fold cross validation and trained the LSTM in 12 epochs for each fold. Both accuracy and F1-score were considered.

Our results are shown in Figure 7.



(a) Loss of validation set. (b) Accuracy of validation set. (c) F1-score of validation set.

Figure 7: Results of LSTM model analysis.

As illustrated by the three plots, the loss, accuracy, and F1 in the validation set level off at around epoch 8. Note that missing data in the train F1 line in the F1 plot is due to division by zero in the calculation of F1, which was caused by misclassifications early in the epochs.

```
Mean Validation Loss: 0.5765
Mean Validation Accuracy: 0.7638
Mean Validation F1: 0.6692
EMBEDDING_DIM: 300
NUM_HIDDEN_NODES: 64
NUM_LAYERS: 3
BIDIRECTION: True
DROPOUT: 0.3
NUM_HIDDEN_NODES: 64
```

Removing URLs, mentions, HTML entities, and most non-alphanumeric characters from the tweets did not improve the performance of the model.

Overall, naive Bayes classification performed better than LSTM. This is possibly due to the small size of the dataset; deep learning models usually benefit from a large dataset, while machine learning techniques such as naive Bayes classification are more suitable for small data sets such as this one. In addition, naive Bayes models have fewer hyperparameters compared to LSTM, and our naive Bayes model has the benefit of Grid Search tuning.

5 Future Work

In the future, we could try using other classification models such as random forest (after vectorization of the text data), deep learning models such as GRU, and transformers such as BERT.

Furthermore, we could improve our models using prior or additional knowledge– assuming it is possible to collect extra data about the tweets; for example, we could segment our dataset by geographical region. Tweets coming out of California with keywords such as ”fire” or ”smoke” could be given more weight to help more accurately detect forest fires. Similarly, tweets coming out of the central United States (termed Tornado Alley) with keywords such as ”storm”, ”tornado”, or ”cloud” could be given more weight when patrolling for tornadoes or other natural disasters.

Once an acceptable performance is achieved, we could deploy our model to monitor tweets in real time. This would, of course, benefit society in general, but more specifically the authorities that are trying to prevent and/or aid when disasters occur.

6 Conclusion

In most aspects, the results are not exactly surprising. The number of words contained in a tweet does not correlate with whether or not a tweet is about a real disaster; however, the polarity that was calculated using sentiment analysis *is* correlated. Further exploratory analysis using word clouds verified that there is a very real and distinguishable theme to disaster tweets when compared to non-disaster tweets, and topic modeling confirmed that certain topics were more frequently discussed in disaster tweets. Naive Bayes classification performed better than LSTM, most likely due to the relatively small size of the dataset.

Overall, this project taught us a lot about how to apply the data mining knowledge we’ve acquired on a more real-life scenario. This is the type of problem that data mining and its associated analysis and machine learning techniques excel at, and it is also the type of problem that could benefit society as a whole.

References

[1] Sheng Kung Yi, Mark Steyvers, Michael D. Lee, and Matthew J. Dry. The wisdom of the crowd in combinatorial problems. *Cognitive Science*, 36(3):452–470, 2012. doi:10.1111/j.1551-6709.2011.01223.x.

[2] Raymond W. Gibbs. On the psycholinguistics of sarcasm. *Journal of Experimental Psychology: General*, 115(1):3–15, 1986. doi:10.1037/0096-3445.115.1.3.

[3] Anna K. On plus side look at the sky last night it was ablaze pic.twitter.com/qqsmshaj3n, Aug 2015. URL: https://twitter.com/anyotherannak/status/629195955506708480.

[4] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016. doi:10.1613/jair.4992.

[5] About different types of tweets. URL: https://help.twitter.com/en/using-twitter/types-of-tweets.

[6] Entity. URL: https://developer.mozilla.org/en-US/docs/Glossary/Entity.

[7] Mar 2017. URL: https://www.geeksforgeeks.org/naive-bayes-classifiers/.