Tweeting Emotions: Decoding Sentiment in 140 Characters

Final Report

Edward Wang

Fall 2024

Abstract

Sentimental analysis of social media data, particularly Twitter, has become a popular method for gauging public opinion. This study explores sentiment analysis using the Sentiment140 dataset, focusing on extracting insights from tweets for improving business strategies. We compared traditional machine learning models (Naive Bayes, Logistic Regression) with deep learning models (RNN with LSTM). Data preprocessing involved handling internet slang, text normalization, and lemmatization with part-of-speech tagging to boost accuracy. Logistic Regression reached an F1 score of 0.81, while RNN with LSTM achieved 0.82, showing deep learning's edge in sentiment analysis. However, logistic regression provides fast deployment and computational efficiency, making it ideal for quick to market scenarios. Challenges included handling sarcasm and long training times for some models. Future work aims to implement advanced models like BERT and enhance the preprocessing pipeline.

1 Introduction

Since its inception in 2003, the popularity of social media has surged over 20-times in the past two decades [17]. Currently, more than two-thirds of adults in the US engage with social networking sites, with an astonishing 90% participation rate from young adults aged 18 to 29 [23]. Social media serves as a dynamic platform for users to express their feelings and opinions on a wide array of topic. This ranges from critiques of the latest blockbuster films to in-depth discussions about character development in literature, or even sharing adorable moments of a newborn hippo in Thailand.

Among the various forms of social media, microblogging has gained particular traction. This format "allow users to exchange small elements of content such as short sentences, individual images, or video links" [15]. For example, 𝕏 (formerly Twitter) had more than a quarter of a billion daily active users in 2019, with a notable 10% growth in 2020 [22]. The increasing engagement on these platforms highlights their significance as a medium for both personal expression and community interaction.

As social media continues to grow, companies are increasingly eager to tap into this vast reservoir of user-generated content. The insights from analyzing social media data can provide invaluable information about public perception and brand image. With 500 million tweets sent each day in 2022 [26], 𝕏 (formerly Twitter) offers a rich source of real-time feedback from consumers. Businesses aim to mine this data to gain a deeper understanding of their audience's views and preferences.

Sentiment analysis, also known as opinion mining, plays a key role in processing unstructured data through Natural Language Processing (NLP) to identify emotional tone of the text [31]. By analyzing tweets related to a particular company's product or services, business can uncover trends, gauge customer satisfaction, and identify areas for improvement or expansion. Sentiment analysis enables companies to monitor their public image effectively and engage with their customers. This ultimately helps improve both brand loyalty and marketing strategies.

2 Problem Statement

The objective of this analysis is to identify the most effective models for Natural Language Processing (NLP), with a particular focus on sentiment analysis of online text. This study will utilize various tweets from 𝕏 (formerly Twitter), where the contents of each post will be classified into positive or negative sentiments. By analyzing these tweets, we hope to enhance the understanding of how sentiment analysis can help drive business strategies and improve customer engagement in an increasingly digital world.

3 Data Source

The data source for this analysis is the Stanford Sentiment140 Dataset [9], which contains 1.6 million tweets extracted from 𝕏 (formerly Twitter) API. An example format of a tweet, created using a fake tweet website [30], can be seen in Figure 1. This dataset was collected in 2009 which was an era where posts were limited to 140 characters. This restriction has since been increased to 280 characters. Although the original dataset is no longer hosted by Stanford, it remains accessible through Kaggle [16] and Hugging Face.

Figure 1: Example of a Tweet

The dataset consists of five primary components: the ID of the tweet, the tweet's posted date, the username of the person who sent it, the content of the tweet, and the sentiment of the tweet itself

classified as either positive or negative. Users can tag other users by including their handles (@), and utilize hashtags (#) to highlight keywords or topics relevant to their message [21].

4 Proposed Methodology

4.1 Data Cleaning

In this dataset, tweets often exhibit typical attributes found in online social media. Typical attributes include user mentions, hyperlinks, emojis, abbreviations, and slang [28]. The tweets may also contain informal language, spelling errors, and special characters [5]. While some spelling variations may be errors, others are intentionally stylized for emphasis, which is known as netspeak. For instance, posts might feature elongated words such as "heyyyyy", "i gotta get up eeeeaaarrrrlllyyyyyyyy", or substituting words like "that" with "dat" [29].

The data cleaning process will be broken into three parts. The first part focuses on elements specific to 𝕏 (formerly Twitter). This involves the removal of user mentions and hyperlinks to streamline the data for analysis. The second part addresses standardizing netspeak. In this step, we will standardize abbreviations, correct common spelling variations, and normalize informal language to ensure consistency across the dataset. The third part involves optimizing the text for natural language processing (NLP). This includes tasks such as tokenization, lowercasing, and the removal of stopwords, special characters or other irrelevant elements that may reduce the accuracy of our models.

4.2 Usernames and Hyperlinks

All social media platforms distinguish their users with a username. To prevent specific usernames from influencing sentiment analysis, it is essential to eliminate mentions (e.g., @User123). For instance, a user like @PositiveQuoteDaily, who consistently shares positive content, could inadvertently bias the model towards associating this username with positive sentiment.

Starting with the username, cleaning can be done using regular expressions (Regex) to identify the "@" character that indicates a mention. Usernames on 𝕏 (formerly Twitter) are constrained to no more than 15 characters and must contain only alphanumeric characters, with the exception of underscores [33].

Hyperlinks, on the other hand, are more complex. They generally follow specific patterns, starting with "www", "http://", or "https://". Since 𝕏 (formerly Twitter) is a microblogging platform, hyperlinks are often designed to be clicked immediately. Thus, they tend to follow a standardized format and can be easily identified and removed.

4.3 Standardizing Netspeak

The meaning of words tends to shift both online [19] and in everyday conversations [20]. This can complicate the standardization process. We begin by expanding contracted words back into their full form. This allows our model to focus on each individual word rather than the contraction itself. While this is straightforward for cases like "she'll" into "she will", other cases are more complex. For instance, distinguishing between possession and contraction, as in "it's" versus "its", can be challenging. A more difficult example is differentiating between "the cat's toy" and "that's their toy", using python code, as they both end with "t".

As mentioned above, we need to handle posts that feature netspeak. Regex can help identify repeated letters and remove the excess. The approach is to replace any sequence of three or more repeated letters with just two. This would address cases like "heyyyy" or "hhhhhiiiiii", without impacting naturally occurring double-letter words like "ball" or "butter". While there are rare instances of words with more than two repeated letters, such as "hostessship", these occurrences are infrequent and can be handled separately. Cases like "that" and "dat" are not handled in this cleaning process due to time constraints. Addressing such variations would require an extensive mapping file to account for all possible informal language variations.

4.4 NLP Preprocessing

Preprocessing our text plays a crucial role in building any NLP model [14]. We begin with basic cleaning steps. HTML tags, also known as character entity references, are often used to escape special characters on websites [24]. For example, "&" represents the "&" sign. Using the html package, we can quickly replace these tags with their original ASCII form. Accented characters are also replaced with their standard ASCII equivalents. Finally, the entire text is converted to lowercase to eliminate case-sensitive discrepancies.

To enhance the quality of the data, it is crucial to remove stopwords [8]. These are defined as commonly occurring words in natural language that contribute little to the overall meaning. Stopwords include articles, conjunctions, prepositions, pronouns, and frequently used verbs. For example, some popular ones are: "the", "and", "is", "in", "to" [25].

Since words can take various forms based on tense, techniques such as stemming or lemmati-

zation are necessary to reduce them to their root forms [11]. While both methods aim to reduce words to their base forms, stemming works by simply trimming the start or end of the word, whereas lemmatization considers the word's original meaning. Stemming is computationally simpler than lemmatization, but the latter is generally more accurate [18]. Given the significant advances in computing power over the past decade, the computational cost of lemmatization has become negligible. Therefore, we will be using lemmatization in our process.

5 Modeling

5.1 Model Architectures

The objective of this analysis is to conduct sentiment analysis on online text. A neural network model is well-suited for this task, particularly the Recurrent Neural Network (RNN). RNNs are a type of deep learning architecture designed to handle sequential data, making them especially effective for natural language processing (NLP) and speech recognition tasks [2].

One such model is the Long Short-Term Memory (LSTM) network, which excels at capturing long-term dependencies in text sequences [7]. LSTMs are a specific type of RNN designed to address the limitations of traditional RNNs when learning long-term dependencies. They use a set of gates to regulate the flow of information, allowing the network to selectively retain important information in memory [13]. This makes LSTMs especially well-suited for tasks like sentiment analysis on tweets, where the meaning of a message often depends on the context provided by preceding words or phrases. Furthermore, Bidirectional LSTMs take this a step further by processing input data in both forward and backward directions. This captures context from both past and future states to improve performance.

Beyond RNNs, exploring other types of neural networks, such as transformers, is essential for tackling more complex NLP tasks. For example, state-of-the-art models like BERT [6], developed by Google, have led to significant advancements in NLP and set new benchmarks in the field.

Traditional machine learning models can also be used as our dataset represents a binary classification problem. These models could serve as benchmarks to evaluate the performance of the neural networks. Examples of such models include naive Bayes, K-Nearest Neighbor (KNN), logistic regression, and decision tree. A study that performed binary classified hate speech in tweets, for instance, used similar models [27].

In the example of a decision tree, the text would have to be converted to bag of words or TF-IDF. Bag of words essentially looks at if certain known words are in the text opposed to the location. TF-IDF looks at the term frequency within the text [12].

5.2 Model Tuning

Model tuning plays a critical role in optimizing model performance. Each of the models described earlier has hyperparameters that must be adjusted to find the best performing variation. For neural networks, one of the most important hyperparameters is the learning rate which determines the step size at each iteration during training [32]. Other key hyperparameters include batch size, number of layers, dropout rate, and optimizers. The batch size defines how many examples are processed before updating the model, while the number of layers dictates the complexity of the model. The dropout rate randomly drops certain layers during training, which helps reduce overfitting. The choice of optimizer also influences how quickly the model converges. In models like random forest, hyperparameters such as the number of trees, tree depth, and minimum samples per leaf impact the model's performance by controlling its complexity and how it splits the data.

There are three main approaches to selecting hyperparameters for these models. The first is to use the default parameters or manually chosen values. Often this doesn't provide the best possible model. A more systematic approach is to use grid search, which evaluates a predefined set of hyperparameters in a grid-like fashion. However, a study by Bergstra et al. (2012) argues that random search can be more efficient than grid search in certain cases [3]. While we plan to use grid search initially, we may switch to random search if grid search proves to be too time-consuming.

5.3 Model Metrics

We aim to split our dataset into an 80/20 ratio, with 80% used for training and 20% reserved for testing. Ideally, we would use more than 15% of the data for testing, as suggested in [4]. To determine the best hyperparameters for each model, we will use metrics like accuracy on the training dataset, which will allow us to track the best performance. The testing dataset will then be used to assess the final performance of the model. This is because the testing dataset is new and hasn't been seen before by the model. In addition, we will apply cross-validation to ensure that the model performs optimally and generalizes well across different subsets of the data.

To compare the performance of the various models, we will use the F1 score. The F1 score ranges from 0 to 1, where 0 represents the worst value and 1 represents the best. It combines the precision and recall metrics using the harmonic mean [10]. This balanced metric allows for a fair comparison across models and ensures that both precision and recall perform well overall. However, it is still important to exercise caution and examine the individual precision and recall values. A model with a high F1 score could still have one metric (i.e., precision or recall) that is much lower than the other, which may indicate an imbalance in performance.

6 Analysis and Results

6.1 Exploratory Data Analysis

Before proceeding with the modeling process, we first examined the dataset. The Sentiment140 dataset is evenly split, with 800,000 tweets in each sentiment label. After applying our data cleaning process in python, approximately 3,000 tweets were removed as they consisted solely of direct mentions of other users without providing any meaningful text.

One key decision in our data cleaning process was to avoid removing stopwords. In his article, Aleti argues that indiscriminately removing stopwords can alter the sentiment of the original text [1]. He provides the example of the negative sentiment "The product is not good," which, after stopwords are removed, becomes "product good," changing the sentiment to positive.

When applying lemmatization, some words were not processed correctly. We used part-of-speech tagging to address this issue which improved the lemmatization process. Part-of-speech tags each word as a noun, verb, adjective, or adverb, essentially identifying the grammatical role of each word. With this fix, we were able to ensure that words were lemmatized appropriately based on their context.

The most frequent words in both positive and negative tweets can be observed in Figure 2 and Figure 3 respectively. Some words, such as "go", "now", and "will", appear with similar frequency in both the positive and negative word clouds. The positive sentiment word cloud features more terms related to gratitude and love, while the negative sentiment word cloud highlights themes of work and a sense of loss.

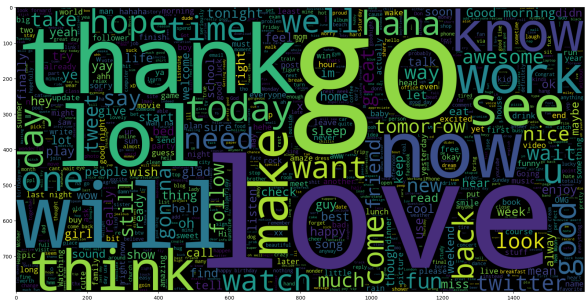

Figure 2: Positive Sentiment Word Cloud

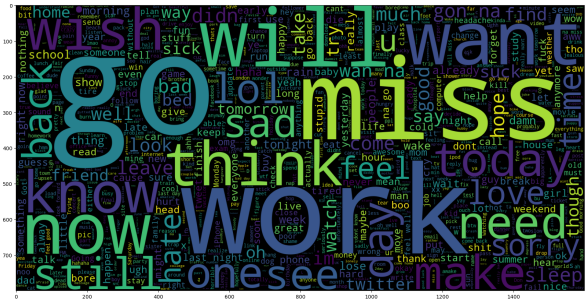

Figure 3: Negative Sentiment Word Cloud

6.2 Modeling Results

The simpler models, naive Bayes and logistic regression, were trained first. The dataset was vectorized using the TfidfVectorizer package in Python, with a maximum of 10,000 features and included both uni-grams and bi-grams. The corresponding hyperparameters are listed in Table 1.

Model	**Parameters**
Bernoulli Naive Bayes	$\alpha : [0.1{-}1.0]$
Logistic Regression	$C : [10^{-3}{-}10^{3}]$
	$penalty : ['l2']$

Table 1: Hyperparameters for Random Search

For naive Bayes, random search identified 0.1 as the optimal alpha value, which improved the F1 score from 0.77 (using default parameters) to 0.78. For logistic regression, random search found that an inverse regularization strength of 1 provided the best performance, which coincidentally matched the default setting. The F1 score for logistic regression was 0.81. Both models were evaluated using 100-fold cross-validation to reduce the risk of overfitting. Finally, the trained models were saved using the joblib package, enabling them to be easily imported for future use.

The Random Forest model took an unexpectedly long time to train. On a Ryzen 3600 CPU, released in 2019, it took approximately 3 minutes per tree. With 100 trees and 5-fold cross-validation, training the model would have taken over a full day.

Model	F1 Score	Accuracy	TP	TN	Confusion Matrix
Bernoulli Naive Bayes	0.77	0.77	126,594	119,824	[[119824, 40263], [32683, 126594]]
Logistic Regression	0.81	0.81	130,303	127,417	[[127417, 32670], [28974, 130303]]
Bernoulli Naive Bayes (RS)	0.78	0.77	126,583	119,853	[[119853, 40234], [32694, 126583]]
Logistic Regression (RS)	0.81	0.81	130,303	127,417	[[127417, 32670], [28974, 130303]]
RNN with LSTM	0.82	0.82	132,752	130,583	[[130583, 26952], [20853, 132752]]

Table 2: Model Performance

This was before any hyperparameter tuning which would have increased the runtime even further. One attempt to reduce training time was to decrease the number of input features from 10,000 to 1,000, and reduce the number of trees from 100 to 20. This cut the training time to about 1 minute per tree. Unfortunately this would have still taken hours to train. Google Colab was also tested, but discrepancies in the results and limited time for root cause analysis led to it being discarded. Given the delays in text processing, the Random Forest model was ultimately removed from the project.

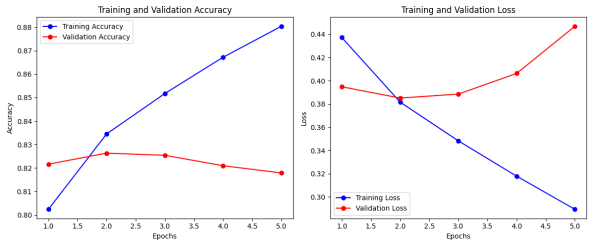

Figure 4: RNN Training and Validation
Note: A larger version can be found in the appendix.

The Recurrent Neural Network (RNN) model was implemented using the TensorFlow package in Python. The architecture consists of six layers: one embedding layer, one LSTM layer, two dense layers, and two dropout layers (See Appendix). Early stopping was used to prevent overfitting. As shown in Figure 8, the model achieved its best performance at epoch 2, where the validation accuracy was highest and the validation loss was lowest. After epoch 2, the validation loss began to rise while the accuracy fell. This suggests the model had reached its optimal performance. The RNN achieved the best overall performance with an F1 score of 0.82 as summarized in Table 2.

7 Conclusion

In this study, we explored sentiment classification using the Sentiment140 dataset. We compared multiple models, including simpler methods like Naive Bayes and Logistic Regression, as well as the more complex Recurrent Neural Network (RNN). After preprocessing the data and addressing issues with lemmatization, we trained the models while leveraging hyperparameter tuning and cross-validation. The models were evaluated using F1 scores, and overfitting was prevented through techniques such as early stopping and random search for optimal parameters.

Our results demonstrate that the simpler models, Naive Bayes and Logistic Regression, performed adequately with F1 scores of 0.78 and 0.81 respectively. These were fast and easy to implement, but failed to fully capture the sequential nature of text. The RNN model achieved the best overall performance, with an F1 score of 0.82. This highlights the effectiveness of using deep learning techniques for sentiment analysis, particularly in capturing the nuances of language.

Given time constraints in a competitive business environment, logistic regression offers a practical solution. It provides solid performance with faster training times compared to complex models like neural networks, allowing companies to deploy models quickly. This speed enables businesses to stay ahead of competitors by iterating faster and responding more swiftly to market demands, without sacrificing too much accuracy.

For future work, several areas can be explored. One limitation in this study was the handling of sarcastic comments, which were not addressed in the modeling process. Another direction is revisiting the Random Forest model to overcome its limitations and improve its performance. It would also be valuable to experiment with the Bag of Words method as an alternative feature extraction technique. We woudl want to compare its results with those obtained using TF-IDF. Although BERT was considered for this study, there was not enough time to fully implement and compare its performance. Exploring BERT and other advanced models in future work could offer significant improvements in sentiment classification.

7.1 Lessons we have learned

Several important lessons were learned during this process. The most significant one was the failure to train a random forest model within the allotted time. In hindsight, this issue likely stemmed from the decision to retain stop words, which in-

creased the number of input features and ultimately slowed the model's training down to a crawl.

Another key lesson was that lemmatization alone was not sufficient. Words can have multiple meanings depending on context which cans issues further down the line. To make the lemmatization process more effective, it became clear that incorporating parts of speech was essential for achieving the desired results. Additionally, text preprocessing turned out to be more complex than initally anticipated.

Finally, narrowing the project scope or adding more team members would have greatly helped in meeting all the original objectives within the planned timeline.

References

[1] Sunil Aleti. Don't blindly remove stopwords in sentiment analysis. *DEV Community*, 2020. Accessed: 2024-11-09. URL: https://dev.to/sunilaleti/dont-blindly-remove-stopwords-in-sentiment-analysis-3nok.

[2] Shervine Amidi. Recurrent neural networks cheatsheet, 2021. Accessed: 2024-10-20. URL: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] Google Developers. Dividing datasets, n.d. Accessed: 2024-11-07. URL: https://developers.google.com/machine-learning/crash-course/overfitting/dividing-datasets.

[5] Bhuwan Dhingra, Zhong Zhou, Dylan J. Fitzpatrick, Michael Muehl, and William W. Cohen. Tweet2vec: Character-based distributed representations for social media. *CoRR*, abs/1605.03481, 2016. URL: http://arxiv.org/abs/1605.03481, arXiv:1605.03481.

[6] Hugging Face. Bert model documentation, 2024. Accessed: 2024-10-20. URL: https://huggingface.co/docs/transformers/en/model_doc/bert.

[7] Usha G, Priyan M K, Gokulnath Chandra Babu, and Gayathri Karthick. Sentiment analysis on twitter data by using convolutional neural network (cnn) and long short term memory (lstm). *arXiv*, February 2021. Accessed: 2024-11-03. doi:10.21203/rs.3.rs-247154/v1.

[8] Kranti Ghag and Ketan Shah. Comparative analysis of effect of stopwords removal on sentiment classification. In *Proceedings of the 2015 International Conference on Communication, Control, and Computing Technologies (IC4)*, pages 1–6, September 2015. Accessed: 2024-11-03. doi:10.1109/IC4.2015.7375527.

[9] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. In *CS224N Project Report*, volume 1, page 12. Stanford University, 2009.

[10] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview. *arXiv preprint arXiv:2008.05756*, 2020.

[11] Stanford NLP Group. Stemming and lemmatization, n.d. Accessed: 2024-10-20. URL: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

[12] Yaakov HaCohen-Kerner, Daniel Miller, and Yair Yigal. The influence of preprocessing on text classification using a bag-of-words representation. *PloS one*, 15(5):e0232525, 2020.

[13] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[14] Subbu Kannan, Vairaprakash Gurusamy, S Vijayarani, J Ilamathi, Ms Nithya, S Kannan, and V Gurusamy. Preprocessing techniques for text mining. *International Journal of Computer Science & Communication Networks*, 5(1):7–16, 2014.

[15] Andreas Kaplan and Michael Haenlein. The early bird catches the news: Nine things you should know about micro-blogging. *ResearchGate*, 2010. Accessed: 2024-11-03. URL: https://www.researchgate.net/publication/223080854_The_early_bird_catches_the_news_Nine_things_you_should_know_about_micro-blogging.

[16] Kazanova. Sentiment140 dataset, 2017. Accessed: 2024-10-20. URL: https://www.kaggle.com/datasets/kazanova/sentiment140/data.

[17] Simon Kemp. Digital 2024: Global overview report, January 2024. Accessed: 2024-11-03. URL: https://datareportal.com/reports/digital-2024-global-overview-report.

[18] Divya Khyani, BS Siddhartha, NM Niveditha, and BM Divya. An interpretation of lemmatization and stemming in natural language processing. *Journal of University of Shanghai for Science and Technology*, 22(10):350–357, 2021.

[19] S. Lerique and C. Roth. The semantic drift of quotations in blogspace: A case study in short-term cultural evolution. *Cognitive Science*, 42:188–219, 2018. doi:10.1111/cogs.12494.

[20] Kenneth Liberman. Semantic drift in conversations. *Human Studies*, 35, 05 2012. doi:10.1007/s10746-012-9225-1.

[21] Metricool. How to tag on twitter, 2024. Accessed: 2024-11-03. URL: https://metricool.com/how-to-tag-on-twitter/.

[22] Oberlo. How many people use twitter?, 2024. Accessed: 2024-11-03. URL: https://www.oberlo.com/statistics/how-many-people-use-twitter.

[23] Andrew Perrin. Social networking usage: 2005-2015, October 2015. Accessed: 2024-11-03. URL: https://www.secretintelligenceservice.org/wp-content/uploads/2016/02/PI_2015-10-08_Social-Networking-Usage-2005-2015_FINAL.pdf.

[24] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. Html 4.01 specification. *IETF HTML WG*, 1997.

[25] Serkan Sarica and Jun Luo. Stopwords in technical language processing. *PLoS ONE*, 16(8):e0254937, 2021. Accessed: 2024-11-03. doi:10.1371/journal.pone.0254937.

[26] David Sayce. Number of tweets per day 2019, 2019. Accessed: 2024-11-03. URL: https://www.dsayce.com/social-media/tweets-day/.

[27] Akuma Stephen, Tyosar Lubem, and Isaac Adom. Comparing bag of words and tf-idf with different models for hate speech detection from live tweets. *International Journal of Information Technology*, 14(9), 2022. Accessed: 2024-11-03. doi:10.1007/s41870-022-01096-4.

[28] Irina Temnikova, Carlos Castillo, and Org. The case for readability of crisis communications in social media. In *Proceedings of the 2015 ACM Conference on Human Factors in Computing Systems*, 05 2015. doi:10.1145/2740908.2741718.

[29] Shalini Raj Thangaraj and Mahendran Maniam. The influence of netspeak on students' writing. *Journal of Education and Learning (EduLearn)*, 9(1):45–52, 2015.

[30] Typefully. Fake tweet generator, 2024. Accessed: 2024-11-03. URL: https://typefully.com/tools/fake-tweet-generator.

[31] Mayur Wankhade, Chandra Sekhara Rao Annavarapu, and Chaitanya Kulkarni. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(6):2389–2412, February 2022. Accessed: 2024-11-03. URL: https://link.springer.com/article/10.1007/S10462-022-10144-1, doi:10.1007/S10462-022-10144-1.

[32] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimizationb. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019. URL: https://www.sciencedirect.com/science/article/pii/S1674862X19300047, doi:10.11989/JEST.1674-862X.80904120.

[33] X. X username rules, 2024. Accessed: 2024-11-05. URL: https://help.x.com/en/managing-your-account/x-username-rules.

A Larger Versions of Figures

Figure 5: Tweet format

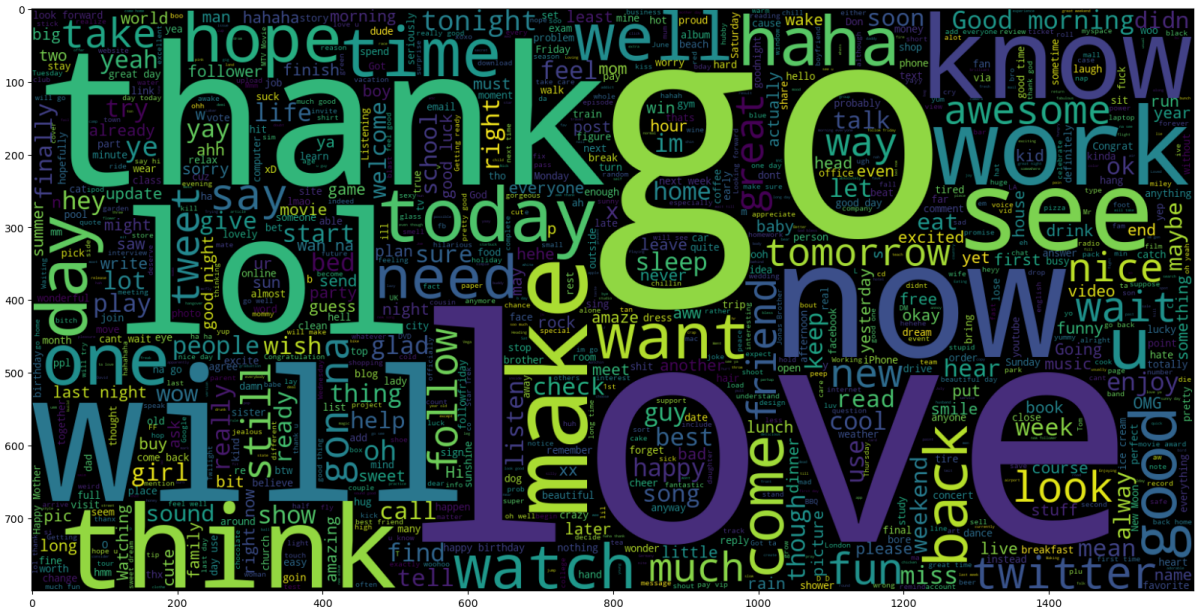

Figure 6: Positive Sentiment Word Cloud

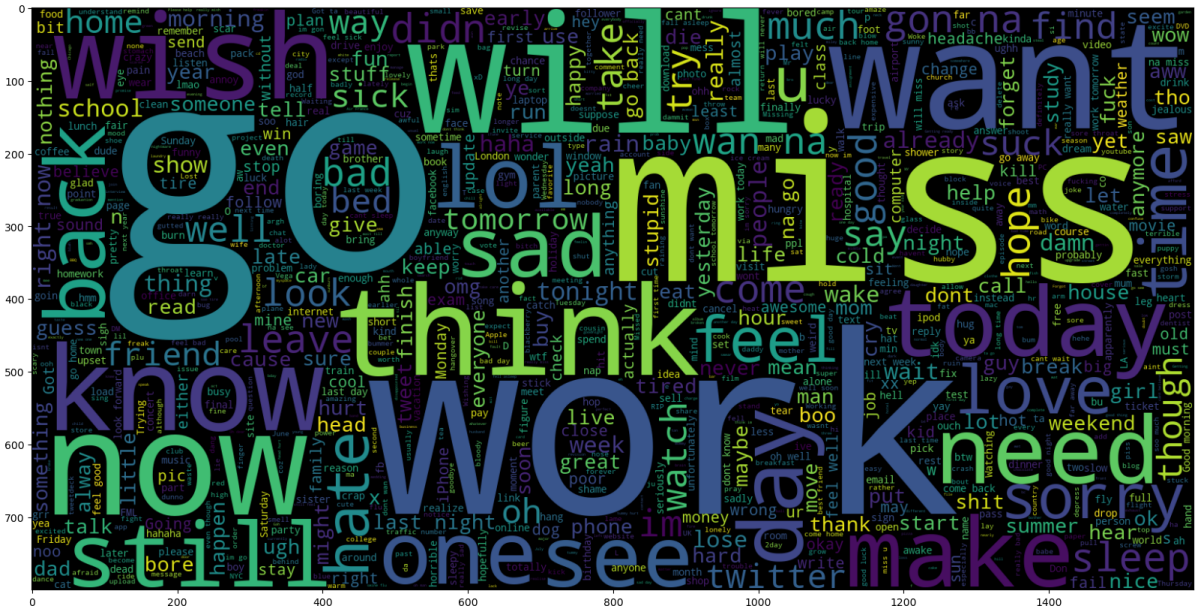

Figure 7: Negative Sentiment Word Cloud

Figure 8: RNN Training and Validation

B Additional Figures

id	sentiment			text
1827461707	positive	@nat	http://twitpic.com/	_y - Bang! Bang! Bang! Giddy-Giddy!
1980916125	negative			eww. jus had the worse dream ever
1693405239	positive	http://tinyurl.com,		5q < THE MOST AMAZING collab that I happen to be in.
2324905275	negative	i cannot wait for a @The_		)0 show! i can only make one date on the july tour
1881293346	positive			got back from rehearsals... recording in two weeks can't wait ^^

Figure 9: Sample Dataset

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 30, 100)           10000000

 lstm (LSTM)                 (None, 64)                42240

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 32)                2080

 dropout_1 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 10,044,353
Trainable params: 10,044,353
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 10: RNN Layers

```
Tweet: I love the product, it's fantastic!
Predicted Sentiment: Positive

Tweet: Terrible experience, will not buy again.
Predicted Sentiment: Negative

Tweet: Service was average, not great.
Predicted Sentiment: Negative

Tweet: Worst purchase ever.
Predicted Sentiment: Negative

Tweet: Very satisfied with the quality.
Predicted Sentiment: Positive

Tweet: I just LOVE waiting on hold for 10 HOURS!
Predicted Sentiment: Positive
```

Figure 11: Model Predictions on Sample Tweets