Twitter Sentiment Analysis with Emojis

Dane Hankamer
Department of Computer Science
Stanford University
Stanford, California
dhank@stanford.edu

David Liedtka
Department of Computer Science
Stanford University
Stanford, California
dliedtka@stanford.edu

Abstract

The exponential rise in popularity of emoji usage in informal writing mediums, such as Twitter, over the past few years has rendered many previous studies of Twitter sentiment analysis outdated. We hypothesize that deliberate strategies for emoji handling increase sentiment analysis performance. After initially training sentiment classification models on a commonly used but decade-old Twitter dataset, we construct a "modern" Twitter dataset that prominently features emojis and "fine-tune" our models to utilize emoji-based features. We find our hypothesis to be correct, demonstrating that emoji handling clearly and significantly increases sentiment classification power. We conclude that is essential for any modern Twitter or informal writing application to effectively handle emojis, and we identify multiple areas of improvement for even greater potential gains in sentiment analysis performance.

1 Introduction

Sentiment analysis is a seemingly simple problem in Natural Language Understanding (NLU); classify text as positive or negative (or neutral, sometimes), or according to some type of ratings system (one to five stars, scale of one to ten, etc.). However, complexities in human language and the widely varying contexts in which humans receive language increase the depth and importance of sentiment analysis as a sub-problem of Natural Language Understanding. Recent approaches to sentiment analysis look to increase the accuracy of classification into any number of categories, all the while increasing the efficiency of computation. On social media networks, such as Twitter, the general public essentially provides free and unlimited data as any number of worldwide events occur. The ability for an entity to quickly develop an accurate understanding of the sentiment of an interest group, such as consumers of a product or citizens of a municipality, is of immeasurable value in the current day and age.

Specifically, Twitter data presents an interesting challenge for sentiment analysis, as users frequently depart from conventional grammar usage in favor of short, informal language when they express their opinions in a Tweet. Twitter has been a popular subject for sentiment analysis research, and many studies exist using Twitter as the medium for their datasets. However, in recent years, emoji usage as a trend has grown rapidly. Emojis became available on mobile operating systems in 2010 after being encoded into unicode, and they have become so popular that the crying-laughing emoji, 😂, was named the Oxford Dictionaries Word of the Year in 2015. Whereas ten years ago, people used ASCII-composed emoticons, such as *:-D*, to occasionally express basic sentiment, plentiful and colorful emojis seem ubiquitous with Twitter feeds today. Twitter users today often use emojis to convey sentiment in a nuanced way.

In sentiment analysis, and in any machine learning problem, it is important for the distribution of the data that a model is trained on to match the distribution of the data that the model is tested on as closely as possible. Because many past Twitter studies were conducted before widespread emoji usage, we predict that the distribution of those datasets fails to match the distribution of modern Twitter data. We hypothesize that by constructing a "modern" Twitter dataset, one that contains emojis, and by leveraging an effective representation of emojis, we can improve the sentiment classification of modern Tweets.

In this study, we experiment with five different models for sentiment analysis of Tweets: Naive Bayes, Maximum Entropy, Support Vector Machines, a Shallow Neural Network, and a Recur-

rent Neural Network based on an LSTM. We train our models on an existing (but likely outdated) Twitter dataset and then "fine-tune" our models using a representation of emojis on a dataset of recently collected Tweets. We compare the predictions of our models on a test set of modern Tweets to a set of automatically generated predictions we use as ground truth. Finally, we compare and analyze the performance of the different model types after training on just the outdated dataset of Tweets, after being fine-tuned on modern Tweets but without handling emojis, and after being fine-tuned on modern Tweets with emoji handling.

2 Related Work

Pang and Lee serves as a survey of approaches towards sentiment analysis (Pang et al., 2008). Pang and Lee highlights different approaches toward data collection, feature generation, building models, and the utility of sentiment analysis.

Go, Bhayani, and Huang explore Naive Bayes, Maximum Entropy, and Support Vector Machine classifiers on Twitter data, using different combinations of unigrams, bigrams, and part-of-speech tags as feature extraction functions (Go et al., 2009). We utilize each of these models as baseline models to compare with our shallow Neural Network and Recurrent Neural Network models, and we use many of the same features. Go, Bhayani, and Huang also construct the Sentiment140 Dataset using Twitter's API. Tweets in the dataset are labeled positive or negative based on whether they were obtained using a "happy" or "sad" emoticon query respectively. Although the Sentiment140 Dataset does not contain any emojis, as it was constructed in 2009, we find it a useful dataset to initially train our models prior to fine-tuning.

Kouloumpis, Wilson, and Moore explore a variety of features (including slang, hashtags, and emoticons) and models to perform sentiment analysis on Twitter data (Kouloumpis et al., 2011). They find part-of-speech features are not particularly helpful, which we surmise is likely due to the inability to apply a sentiment treebank-like structure efficiently to newly collected data. We seek to expand upon their contributions with emoji handling and utilization of deep learning strategies.

Chen et al. propose a Bi-sense Emoji Embedding and Attention-based LSTM algorithm for Twitter sentiment analysis (Chen et al., 2018). While their paper focuses on their model's performance, we focus on the magnitude of performance gain made possible through emoji handling compared to conventional algorithms (that do not explicitly handle emojis).

Hutto and Gilbert provide an automated, rule-based approach called VADER for classifying social media text (Hutto and Gilbert, 2014). VADER is seemingly among the most reputable sentiment analysis classifiers designed to handle emojis. Using an approach similar to Chen et. al, we use the VADER algorithm to classify Tweets we collect using the Twitter API, and we use the most confident predictions as our ground truth labels for our manually-constructed dataset.

Novak et al. track the usage of emojis through 70,000 tweets in 13 languages, with each Tweet manually labeled as positive, negative, or neutral (Novak et al., 2015). The result is a mapping of 751 emoji characters to sentiment information such as the number of occurrences in each positive, negative, and neutral Tweets. In our experiments, we use these mappings to generate several emoji-based features and encodings.

3 Data

Ideally, to evaluate sentiment classification on "modern" Twitter data, we would make use of a set of fully-labeled Tweets that matches the distribution of modern Tweets. This means the Tweets in our dataset should have an average length similar to the population of Tweets we are interested in, should contain emojis at a similar rate, etc. Unfortunately, many of the most robust and popular Twitter datasets for sentiment analysis are more than five years old, and the explosion in emoji usage is a relatively recent development. In fact, the Sentiment140 Dataset, arguably the most popular dataset used for Twitter sentiment analysis, was released in 2009 and is now 10 years old. To address this, we decide use a mix of the robust, existing Sentiment140 Dataset and our own newly constructed "modern" Emoji Dataset.

3.1 Manually-Generated "Emoji Dataset"

We use Twitter's API to collect Tweets to construct an "Emoji Dataset." Through Twitter's API, a user can submit a query of up to 400 strings at a time, and Twitter will send the user any Tweets that are Tweeted while the query is active that

contain one or more of the strings in the query. Emojis have unicode equivalents, but Python also has an emoji library that generates an emoji when provided the emoji's string alias (e.g.- the traditional smiling face emoji 😀 has the alias *":grinning_face:"*). Based on occurrence information detailed in Section 3.3, we query the Twitter API for the 400 most popular emojis, filtering out any Tweets that are not in English or contain any character that is non-ASCII and non-emoji. Twitter increased the maximum length of a Tweet from 140 characters to 280 characters in late 2017, and knowing that we will be comparing our results to studies conducted before 2017, we additionally filter out any Tweets longer than 140 characters. Over a query running for approximately 12 hours, we collect 194,056 unique Tweets that all contain at least one of the most popular 400 emojis.

We preprocess the collected Tweets, removing any retweet tokens ("RT"), converting the text to lowercase, condensing any letters repeated more than three times to just two repetitions of that letter, replacing any usernames (format "@username") with a "<user>" token, URLs with a "<url>" token, and hashtags with a "<hashtag>" token. We also insert a space before and after each emoji so that each emoji will be interpreted as its own word. We again ensure that no Tweets are repeated in our dataset.

Labeling our collected Tweets is the most significant obstacle we face in constructing our dataset. Hand-labeling thousands upon thousands (or perhaps millions) of Tweets is impractical. Thus, we use the VADER sentiment analysis algorithm, published by Hutto and Gilbert, to automatically label our Tweets (Hutto and Gilbert, 2014). VADER provides a score between -1 and 1 for each text it classifies, with -1 indicating full confidence in negative sentiment and 1 full confidence in positive sentiment. It is not ideal to use a classification algorithm output for our ground truth labels, but it is simply much more efficient to do so. To increase our confidence in these labels, we keep only Tweets that score above 0.7 or below -0.7. Roughly 35,000 Tweets are above the 0.7 threshold, but only slightly more than 6,600 Tweets are classified below the -0.7 threshold. To maintain class balance, we randomly select 6,600 positive-labeled and 6,600 negative-labeled Tweets to keep from our original collection of nearly 200,000 Tweets, and we use the VADER-generated labels as ground truth.

3.2 Sentiment140 Dataset

Containing just 13,200 Tweets, we do not believe that our Emoji Dataset is large enough to produce robust results. Therefore, we choose to use the Emoji Dataset to fine-tune our models after training on a larger, traditional dataset. We choose to train first on the Sentiment140 Dataset, published by Go, Bhayani, and Huang and available on Kaggle, which contains 1.6 million Tweets, 800,000 labeled positive and 800,000 labeled negative (Go et al., 2009). This dataset is the most popular Twitter sentiment analysis dataset on Kaggle, and it interestingly was collected based on querying the Twitter API for emoticons, the ASCII-composed predecessors of emojis. This underscores the need for a modern dataset to conduct effective Twitter sentiment analysis today. The Sentiment140 Dataset does not contain any emojis, but it is large and robust enough to allow for initial training of our models prior to fine-tuning.

3.3 Emoji Sentiment Data

In order to fine-tune our models on our manually-generated Emoji Dataset, we need to employ emoji-based features, discussed further in Section 4.3. Novak et al. provides a mapping to positive, negative, and neutral occurrence information for 751 emojis, also available on Kaggle (Novak et al., 2015). We use this mapping to help generate emoji-based features.

4 Methods

4.1 Experimental Procedure

In order to analyze the impact of emoji handling on the sentiment classification of Tweets, we first divide our datasets as follows:

- Sentiment140 Dataset, 1.6 million Tweets: 99% training set, 1% development set
- Emoji Dataset, 13,200 Tweets: 80% training set, 10% development set, 10% test set

We initially train our models, variants described in Section 4.2, on the Sentiment140 training set. We keep a copy of this *Sent140*-trained model. Next, we fine-tune a copy of the *Sent140*-trained model on the Emoji training set, but we strip out any emojis present in the data, effectively ignoring emojis. This is our *Sent140+Emojiless*-trained

model. Finally, we fine-tune another copy of the *Sent140*-trained model on the Emoji training set, this time using a combination of the emoji-based feature functions described in Section 4.3. This is our *Sent140+Emoji*-trained model.

We evaluate all of our models on the Sentiment140 training and development sets [1], mostly to see if performance on the initial training data is preserved after fine-tuning on the Emoji dataset.

We then evaluate all of our models on the Emoji training and development sets. We compare these results to performance on the Sentiment140 training and testing sets to see how the respective models perform on Tweets without emojis compared to Tweets containing emojis. We also compare results between training and development to check for overfitting. Additionally, we use development set results to influence parameter and feature function selection for our models.

Finally, when it comes time for testing, we find the combination of parameters and features for each model that led to the best performance on the Emoji development set, and we use those parameters and features for testing on the Emoji test set.

4.2 Models

We use several types of machine learning models to evaluate performance. To obtain baseline results, we use the same models that Go, Bhayani, and Huang used 10 years ago (Go et al., 2009). They are as follows:

- Naive Bayes Classifier
- Maximum Entropy Classifier (MaxEnt)
- Support Vector Machine Classifier (SVM)

We use the the *scikit-learn* implementations of these models.

In the past decade, deep learning and neural networks have grown considerably more popular, particularly with applications to language processing and understanding. Hoping to capitalize on these advances, we use the following *PyTorch* implementations:

- Shallow Neural Classifier (Shallow)
- Recurrent Neural Network (RNN)

The RNN is a Long short-term memory (LSTM) based implementation of an RNN. We use these two deep learning models for further performance evaluation.

4.3 Features

For our baseline models, we input into our models a vectorization of traditional unigram and bigram feature extraction functions. We run trials using solely unigrams, solely bigrams, and using both unigrams and bigrams. Because we treat each emoji as its own word, our models learn emoji-based unigram and bigram features in addition to traditional word-based features.

We support both of our deep learning models with pre-trained, 100-dimensional GloVe word representations developed specifically for Twitter data (Pennington et al., 2014). We ensure that we preprocess in a similar (but not exactly the same) way[2], using the same format for token replacement. For the shallow neural classifier, our feature function computes a Tweet's average embedding by summing all of the embeddings of each individual word present in the Tweet (a word that does not have a GloVe embedding corresponds to a vector of zeros) and then dividing by the number of words in the Tweet. This average embedding is our models' input. For the RNN, the RNN processes each Tweet one word at a time, making each word's individual embedding the input into the model. For the RNN, we use a 40,000 word vocabulary (out of approximately 740,000 unique total words that appear in the Sentiment140 and Emoji datasets) and force the vocabulary to contain all emojis present in the Emoji training set. We create a pre-trained word embedding matrix using the GloVe lookup and this vocabulary.

We consider two ways to handle emojis for our deep learning models. In the first, we calculate the average "emoji score" of each Tweet. Based on the occurrence information in Novak et al., for each of the 400 emojis in our dataset, we take the number of positive occurrences of the emoji, subtract the number of negative occurrences, and then

[1] We evaluate all models on the Sentiment140 training and development sets except for the Recurrent Neural Network (RNN), described in Section 4.2. This is because the RNN accepts input one word at a time during training and testing, making evaluation of 1.6 million Tweets impractical given time constraints.

[2] Pennington, Socher, and Manning provide a Ruby script to preprocess Tweets for use with the GloVe Twitter embeddings, but we had already preprocessed our Tweets and obtained baseline results when we found this script. It would have been difficult to backtrack our preprocessing due to the elimination of Tweets that started out different but became identical after tokenization. We address this further in Section 7.1.

divide by the number of total occurrences (including neutral occurrences). For each Tweet, we take the average emoji score of the emojis that occur in the Tweet (weighted by number of appearances in the Tweet), and append that score as the 101^{st} feature of the Tweet.

The second way we handle emojis is through "emoji substitution." To do this, for any emoji that occurs in a Tweet, we consider the emoji's alias, which is a word or multiple words that name the emoji. We average the GloVe embeddings of the words that make up the emoji's alias (again a word without an embedding corresponds to an embedding of zeros), and we use this average embedding as the effective GloVe embedding of the emoji.

For the shallow neural network, we explore using just the emoji score, using just emoji substitution, and using both the emoji score and emoji substitution as features. For the RNN, we explore the same combinations. When we use just the emoji score, we append a 101^{st} feature of 0 to the embedding of each traditional word in the embedding vocabulary and append the respective emoji score to an embedding of 100 zeros for each corresponding emoji. When we use just emoji substitution, we simply substitute the computed 100-dimensional embedding for each emoji's alias as specified above. When we use both the emoji score and emoji substitution, we append a a 101^{st} feature of 0 to the embedding of each traditional word in the embedding vocabulary and append the respective emoji score to the computed 100-dimensional embedding for each corresponding emoji's alias.

4.4 Metrics

The primary metric we use is F-1 score on the positive class. We have an equal number of positive and negative Tweets in both the Sentiment140 and Emoji Datasets, and although the training, development, and test sets do not exactly maintain this class balance (Tweets were randomly distributed into respective sets), there is not much of a disparity. For this reason, we expect the F-1 score on the positive class will be nearly the same as the F-1 score on the negative class, and it would likely be safe to report other metrics such as precision, recall, or even accuracy. We report F-1 score on the positive class for the sake of thoroughness, as it is the most encompassing all-around metric of the choices we have presented, but we also record results for the negative class and the precision, recall, and accuracy metrics, available in Appendix A.

5 Results

Results are presented in Table 1. The rows correspond to results using different models, with *Naive Bayes* referring to the Naive Bayes Classifier, *MaxEnt* referring to the Maximum Entropy Classifier, *SVM* referring to the Support Vector Machine Classifier, *Shallow* referring to the Shallow Neural Classifier, and *RNN* referring to the Recurrent Neural Network. The columns correspond to results after different levels of training, with *Sent140* referring to models after being trained just on the Sentiment140 training set, *Sent140+Emojiless* referring to models after being trained on the Sentiment140 training set and fine-tuned on the Emoji training set without any emoji handling, and *Sent140+Emoji* referring to models after being trained on the Sentiment140 training set and fine-tuned on the Emoji training set with emoji handling.

As explained in Section 4.1, we chose the parameters and feature functions for each function based on which combination led to the best performance on the Emoji Dataset development set. Admittedly, our parameter exploration was minimal. Our deep-learning models were time-intensive, with the RNN in particular taking about 24 hours to run. Additionally, because the Naive Bayes, MaxEnt, and SVM models were baseline models, once we obtained results we decided our effort was likely better spent on implementing and improving our deep learning models rather than attempting to find optimal baseline model parameters. As such, all of our results use default parameters implementation parameters with a few exceptions (random seeding where applicable, "saga" solver where applicable, warm start turned on where applicable, etc.).

However, we did explore different combinations of feature functions, and for each model we obtained the best results on the Emoji development set and then used the same features for testing on the Emoji test set with the following feature combinations:

- Naive Bayes: Unigram features only
- MaxEnt: Unigram features only
- SVM: Unigram features only

Table 1: F-1 scores for the positive class on the Emoji Dataset test set for baseline and deep learning models. Best results are bolded for each level of training.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	0.84362	0.85417	0.97172
MaxEnt	0.83410	0.87735	0.97586
SVM	0.78807	0.88069	**0.98187**
Shallow	**0.87571**	**0.88651**	0.94081
RNN	0.84415	0.86086	0.94413

- Shallow: Emoji score extra dimension only
- RNN: Emoji score extra dimension and emoji substitution for emoji GloVe embedding

We see in Table 1 that for almost all models (with the exception of *Shallow*), exposure to the Emoji training set even without emoji handling led to marginal gains in performance (an average increase of 0.0348 in F-1 score). However, in all cases, emoji handling led to significant performance gains compared to the models trained on just Sentiment140 training data (an average increase of 0.1257 in F-1 score) and compared to the models fine-tuned without emoji handling (an average increase of 0.0910 in F-1 score).

6 Discussion

Based on the results in Table 1, it is clear that our hypothesis was correct, handling emojis in modern informal text, such as Tweets, significantly increases sentiment classification power.

One of the reasons for this performance increase is the expressiveness of emojis. There are examples when an emoji can be substituted with an obvious emotional equiavlent (for "I'm going to my friend's house 😀", the emoji can easily be replaced with a word like "happy"). However, emojis, in one character, often express sentiment that often takes several words to describe. The more convoluted the expression of a sentiment is, if for example it takes a user a whole sentence to describe how they feel, the more difficult it is for the algorithm to determine the appropriate classification. Emojis provide a clear sentiment all at once, making sentiment more easily attainable from clear features.

Another common problem in sentiment analysis that emojis effectively address is negation. This is particularly a problem for collected data, such as Twitter data, where those conducting sentiment analysis do not have the advantage of data that has been fit to the sentiment treebank structure. In a sentence like "Today is not a very good day", it is difficult for low level features, like n-grams, to pick up on the negation "not" (it would require at least 4-grams). However, appending an emoji to the end of the sentence, like ☹, expresses the negation "not good" in one word. We do not need to worry about negations of emojis. For example, when a user wants to express that they are not happy, they simply use a sad emoji, not a "not" emoji eventually followed by a happy emoji. Emojis in many cases reflect an intended negation.

We were surprised that the results from our deep learning *Sent140+Emoji* models failed to improve on our baseline models. There are many nuanced expressions and patterns in Twitter and emoji sentiment that we suspected our simpler baseline models would not be able to learn, but we hoped that the deeper structure of neural networks would lead to performance gains in these areas. Perhaps the absence of these gains is a result of insufficient parameter exploration. However, a more likely culprit is that our Emoji training set was simply not large enough (at $10,560$ Tweets) for our neural networks to learn enough about the nuance of emoji usage. We see that our deep learning *Sent140* models performed better than our *Sent140* baseline models after training on the $1,594,000$ Tweets in the Sentiment140 training set. If we had used an Emoji training set of a similar size as the Sentiment140 training set and of the same distribution as our Emoji test and development sets, perhaps our deep learning models would have been able to better quantify the complex relationships involved in Tweets with emojis and would have performed better than our baseline models.

6.1 Error Analysis

As displayed in the Table 2, we identified four main categories that led to erroneous Tweet classification. First, the automated VADER approach

Table 2: Examples of trends in Tweet misclassification.

Error Type	Example Tweet	True Label	Predicted
Incorrect label by VADER	try to watch this and not smile 😀	negative	positive
Sarcasm	"can you delete this photo you took of me at the club i dont want my girlfriend to see" 😂	negative	positive
Fake/spiteful emoji use	note to self: stop trying so hard for people who don't care. 😀	negative	positive
Neutral/ambiguous Tweets	hahaha. oh yeah, just wait until it hits other regions 😲	positive	negative

occasionally leads to incorrect "true" labels for Tweets. Given that the VADER classification represents our ground truth, these errors reduce our F-1 scores at times when our model's predictions are actually correct. This problem was one of the drawbacks of using the output of a classification algorithm as our ground proof, but it was a cost we were willing to pay in exchange for the benefits of automated labeling. To combat this error, we could increase our threshold for the VADER algorithm and manually correct mislabeled examples (although this would be time-intensive).

Secondly, our models often fail to detect sarcasm, particularly when the sarcasm is represented by an emoji. For the Table 2 sarcasm example, our models view the smiling and laughing emojis in a positive light, while the user is actually reacting negatively to another person's actions. Given the complexity and context needed to detect sarcasm, combined with humanity's imperfect ability to detect sarcasm, this proves to be one of the most difficult NLU tasks. Again, perhaps with a larger dataset our deep learning models would more effectively learn to identify sarcasm.

"Fake" emoji usage also led to misclassification in many cases. In the Table 2 example, the user uses a smile out of spite, which causes our models to incorrectly classify the Tweet as positive. This misleading smile and the use of "note to self" are nuances of the English language that are difficult to take into account, similar to the problems associated with sarcasm. Once again, our model would need to understand the intricacies of the English language or place a smaller weight on the emoji in order to correctly classify Tweets of this nature.

Lastly, many of the Tweets are neutral or ambiguous in nature. Because our models classify every Tweet as positive or negative, there is an increased chance of error for Tweets that lack context or intense emotion. Oftentimes, as in the neutral/ambiguous example in Table 2, we cannot know whether a Tweet is meant to be positive or negative without more context. Given that the shocked emoji could be perceived in a positive or negative light, the neurtral/ambiguous example could be talking about a popular new song or a devastating hurricane spreading to other regions. We could mitigate this error by using a more strict threshold for VADER results, but again this would require gathering a much larger number of Tweets. We additionally could look to change our problem to multiclass classification by beginning to also predict neutral Tweets.

7 Conclusions

We have demonstrated that emoji handling clearly increases one's ability to conduct effective Twitter sentiment analysis.

Today informal mediums such as Twitter are more relevant than ever. Twitter remains a useful communication tool between friends, but it has become a public message board where average users, brands, and even public figures such as politicians and athletes regularly share their opinions or contribute to discussions. Because Twitter is so widely used globally, the ability to perform accurate sentiment analysis, gauging public opinion and perception on any number of topics, is more valuable than ever before. The significant performance increases illustrated in our results show that any modern Twitter sentiment analysis application that is not processing emojis effectively is at a major disadvantage.

Emojis often represent emotions or sentiments that are not easily concisely expressed in words. The number of emojis available and the contexts in which they are used has evolved dramatically in recent years and will likely continue to do so

moving into the future. Emoji handling is an interesting Natural Language Understanding subproblem, and continued developments will only serve to increase the power of sentiment analysis.

Our results are encouraging, and we have identified many areas to improve emoji analysis to further increase Twitter sentiment classification power.

7.1 Future Work

As mentioned in Section 5, time constraints hampered our ability to conduct proper parameter exploration. Testing with different parameters is something we would like to try in the future. In particular, we believe a bidirectional LSTM could greatly increase classification power, as this models offers advantages when it comes to analyzing text streams. We attempted to run a bidirectional version of our implemented RNN, but the model did not finish training in time for the project submission. Related to these time constraints, better utilization of cloud computing services would likely have allowed us to test different parameter and feature combinations more efficiently.

Regarding feature functions, we found that emoji substitution, while an interesting idea, was relatively ineffective in practice. We would like to find a better way to produce GloVe embeddings for emojis. The genesis of our project was that many Twitter sentiment analysis datasets available online are outdated because they do not contain emojis. There may be some aspects of the current GloVe Twitter embeddings that are outdated as well. For example, Pennington, Socher, and Manning provide a Ruby script that handles emoticons when preprocessing datasets to prepare them for usage with the GloVe Twitter word embeddings, but there was no evident emoji handling in the script (Pennington et al., 2014). Inspired by this, it may be beneficial to train the GloVe algorithm on "modern" Twitter data, creating a new set of GloVe embeddings for Twitter data. We would need to use a more representative modern Twitter dataset, as described below (not one where every Tweet must contain an emoji), and to maximize the effectiveness of the GloVe algorithm, we would preprocess our set of Tweets using a combination of the provided Ruby preprocessing script and our emoji separation method. As the Ruby preprocessing script currently is written, their is no emoji-specific handling. Emojis are often included next to a word without a space, which leads them to be interpreted by the GloVe algorithm as part of the word. By separating emojis as their own "words," we may develop a stronger set of GloVe Twitter embeddings.

Related to a new set of GloVe embeddings, while we collected a small dataset of Tweets that contained emojis, our Emoji Dataset was not representative of modern Twitter data. Many Tweets contain emojis, but not all Tweets contain emojis. It would be an interesting study and would likely lead to more reliable results if we were to collect a large dataset without querying specifically for emojis, instead querying for general trending topics or collecting as many live Tweets as possible. This would allow us to train our models on a training set with the same distribution as our development and test sets, and it would remove the need for "fine-tuning" and a dataset like Sentiment140. Additionally, as mentioned in Section 6, a larger training set with the same distribution as our test set would likely increase the performance of our deep learning models in particular.

Finally, other methods of examining our results could give us additional interesting insights into modern Twitter sentiment. One such idea is an analysis of classification accuracy by emoji (for example, perhaps one particular emoji is frequently misinterpreted).

Acknowledgments

We would like to thank our project mentor, Xin Zhou, for her invaluable feedback and support through this process. We would also like to thank Professor Bill MacCartney, Professor Christopher Potts, and the rest of the CS 224U teaching team for putting together an engaging and worthwhile course this quarter.

Authorship Statement

Dane Hankamer provided research on existing datasets, analyzed the results of development and final testing, and conducted error analysis. David Liedtka constructed the manually-generated Emoji Dataset, wrote preprocessing and experiment scripts, and put together the video presentation. Both worked together on different parts of the final paper.

References

Yuxiao Chen, Jianbo Yuan, Quanzeng You, and Jiebo Luo. 2018. Twitter sentiment analysis via bi-sense emoji embedding and attention-based lstm. In *2018 ACM Multimedia Conference on Multimedia Conference*, pages 117–125. ACM.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12):2009.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! In *Fifth International AAAI conference on weblogs and social media*.

Petra Kralj Novak, Jasmina Smailović, Borut Sluban, and Igor Mozetič. 2015. Sentiment of emojis. *PloS one*, 10(12):e0144296.

Bo Pang, Lillian Lee, et al. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

A Appendices

For each model, we chose the parameter and feature combinations that resulted in the best results on the Emoji development set for testing. Table 1, discussed in Section 5, is a presentation of the F-1 score for these models with the chosen parameter and feature combinations on positive labels on the Emoji test set.

The following tables are supplementary results from the same models, displaying performance on the negative class, performance on the development and training sets, and other evaluation metrics. Full results for all possible parameter, feature, dataset, metric, and evaluation label combinations are available in the "results" directory of our source code, which is available at `https://github.com/dliedtka/twitter_emoji_sentiment`.

Table 3: F-1 scores for the negative class on the Emoji Dataset test set for baseline and deep learning models. Best results are bolded for each level of training.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	**0.80711**	0.82500	0.96907
MaxEnt	0.77728	**0.87567**	0.97408
SVM	0.68610	0.86874	**0.98017**
Shallow	0.79623	0.779930	0.94250
RNN	0.79651	0.82811	0.93730

Table 4: F-1 scores for the positive class on the Emoji Dataset development set for baseline and deep learning models. Best results are bolded for each level of training.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	**0.84225**	0.85198	0.95537
MaxEnt	0.83588	0.87067	0.96689
SVM	0.78414	**0.88440**	**0.97432**
Shallow	0.82867	0.83771	0.93673
RNN	0.82117	0.84488	0.93169

Table 5: F-1 scores for the positive class on the Emoji Dataset training set for baseline and deep learning models. Best results are bolded for each level of training.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	**0.83084**	0.85851	0.96150
MaxEnt	0.82746	0.90421	0.98100
SVM	0.77445	**0.93826**	**0.99068**
Shallow	0.81233	0.81955	0.92773
RNN	0.82470	0.84475	0.94509

Table 6: F-1 scores for the positive class on the Sentiment140 Dataset development set for baseline and deep learning models. Best results are bolded for each level of training. We did not evaluate the RNN model on the Sentiment140 development set, because it was impractical given time constraints.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	0.77799	**0.77817**	**0.77807**
MaxEnt	**0.80329**	0.62675	0.63035
SVM	0.76382	0.66756	0.66526
Shallow	0.75384	0.76322	0.75166

Table 7: F-1 scores for the positive class on the Sentiment140 Dataset training set for baseline and deep learning models. Best results are bolded for each level of training. We did not evaluate the RNN model on the Sentiment140 development set, because it was impractical given time constraints (1,584,000 Tweets in the training set).

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	0.80575	**0.80548**	**0.80550**
MaxEnt	**0.81530**	0.62728	0.62979
SVM	0.76495	0.66968	0.66441
Shallow	0.74999	0.76362	0.74824

Table 8: Accuracy scores for the positive class on the Emoji Dataset test set for baseline and deep learning models. Best results are bolded for each level of training.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	**0.82727**	0.84091	0.97045
MaxEnt	0.80985	**0.87652**	0.97500
SVM	0.74697	0.87500	**0.98106**
Shallow	0.81970	0.81061	0.94167
RNN	0.82348	0.84621	0.94091

Table 9: Precision scores for the positive class on the Emoji Dataset test set for baseline and deep learning models. Best results are bolded for each level of training.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	**0.79355**	0.81242	0.96264
MaxEnt	0.76024	**0.90248**	0.97515
SVM	0.69541	0.87000	0.97270
Shallow	0.78200	0.76370	**0.99029**
RNN	0.77709	0.80928	0.92426

Table 10: Recall scores for the positive class on the Emoji Dataset test set for baseline and deep learning models. Best results are bolded for each level of training.

Model	Sent140	Sent140+Emojiless	Sent140+Emoji
Naive Bayes	0.90044	0.90044	0.98097
MaxEnt	**0.92387**	0.85359	0.97657
SVM	0.90922	0.89165	**0.99122**
Shallow	0.90337	0.91801	0.89605
RNN	**0.92387**	**0.91947**	0.96486