Final Report: Twitter Sentiment Analysis

Krit Verma

Abstract

The task I wanted to complete was exploring Twitter sentiment analysis through multiple machine-learning models and exploring how well these models generalize and fine-tune. I explore three methods: lexicon-based approach, Naive Bayes, and Transformers. Transformers were overall the best model, followed closely by Naive Bayes and Reccurent Neaural Networks. The lexicon-based approach lacked significantly in predictions and generalization. The transformer models were not only the best general predictor, but when fine-tuned, they were significantly better than the others.

1. Introduction

My project will be around the sentiment analysis of tweets and focusing on how different models do when exposed to different more specific datasets. The models will take in a string of comments or tweets as data and tell the user whether it is positive or negative. Twitter users can use this to filter out negative comments or tweets. Brands could also use it to limit or remove negative comments from their sponsored posts or view the sentiment of the engagement from the tweet. Since tweets and comments are similar, The same model, with different training data, could be used to understand the sentiment for comments for articles, restaurants, etc.

I have focused the final project on how different models succeed in generic Twitter sentiment analysis versus how they are fair for more specific tweets. I compared three different approaches. The only nonmachine learning approach is a Lexicon-based approach using pre-learned dictionaries of sentiment for words. The model does not change whether the data is generic or specific. Overall, the models did not perform very well. While they were accurate in the tweets, they predicted positive and negative; they mostly predicted neutral sentiment when the data had no neutral examples.

The Naive Bayes model was decent when predicting generic tweets. For specificity training, the dictionary and counts of words were changed with the training examples of the additional dataset and tested; when blindly shown new specific data, they performed better than both models and also improved when the training data for the new data was well.

The transformer model was also fairly accurate for generic tweets. When retraining a pre-learned classifier like Bert for more generic examples, the models revealed the best predictor for the sentiment140 dataset. When blindly shown new data, it struggled for the movie data set but did well on the company dataset. The model really excelled when fine-tuned to the new datasets.

The Recurrent Neural Network model had similar results to the transformer in generic tweets. I wanted to make it a lot different than the Transformer, so I decided to train my own word embeddings using kernas. It lacked compared to Transformers when generalizing but still was decent.

2. Related Work

Sentiment Analysis has been widely explored, and many machine-learning and nonmachine-learning approaches have been attempted. There are two main approaches.

2.1. Machine Learning Approaches

In 2018, Ankit and Saleena used Naive Bayes, Random Forest, Support Vector Machines, and Logistic Regression. Naive Bayes had an average acc. of 75.71. Random Forest had an average acc. of 74.185. Support Vector Machine had an average acc. of 75.927. Logistic Regression had an average acc. of 74.717. Dataset: Stanford - Sentiment140 corpus, Health Care Reform (HCR), First GOP debate Twitter sentiment dataset, Twitter sentiment analysis dataset

In 2019, Sailunaz and Alhajj detected sentiment and emotion from tweets and their replies and measured the influence scores of users based on various user-based and tweet-based parameters. The experiments were conducted in multiple steps. They applied the Naive Bayes classifier to classify the text according to corresponding sentiment and emotion classes using 3-fold, 5-fold, and 10-fold cross-validation. Finally, they used the classifier on cleaned text and NAVA words to measure accuracy. Their accuracy was 66.86. Datasets: EmoBank, The Valence and Arousal Facebook Posts, The Emotion in Text data set, EmoLex, ISEAR

In 2021, Machuca used COVID-19 pandemic tweets to

analyze sentiment using a logistic regression algorithm on positive and negative labels. Their reported accuracy was 78.50 Dataset - Self Parsed and Labelled Dataset.

These are a few machine learning research on Twitter sentiment analysis. Challenges arise when dealing with sarcasm, irony, new vocabulary, and contextual understanding. My models also struggled with these common issues during text and sentiment analysis.

2.2. Lexicon-Based Approaches

Researchers have used lexicons or sentiment dictionaries to assign sentiment scores to words in tweets and then aggregate these scores to determine the overall sentiment of the tweet.

2011 Maite Taboada, Julian Brooke, Milan Tofiloski, and Kimberly D. Voll published their research using a lexicon-based approach. They assume each word has a prior polarity and use dictionaries to store these polarities. They used dictionaries for adjectives, nouns, verbs, and adverbs. They also had intensifiers and negating words in their dictionaries. They also had other features, but the basic idea was to create a dictionary and values and use a formula to calculate the tweet. They reported an overall accuracy of 71.1

2.3. Differences

While widely explored, I did not find many studies on general model training vs specific model training, My inspiration behind this choice were some articles I read on how data is the most important part of any machine learning model and how some researcher at OpenAI was saying that many models were converging to the same point given the same data. I am intrigued by the importance and differences data makes on a model's abilities and want to explore it further.

3. Dataset and Evaluation

I had mentioned using multiple datasets to see the performance over different categories. I still plan on doing this in the final report and after I have trained more expressive and complex models. However, I will use the most extensive general dataset, Sentiment 140. The dataset is 1.6 million tweets labeled negative and positive. The dataset has 800,000 positive labeled tweets and 799,998 negative labeled tweets. The large general dataset is a good baseline for developing my machine-learning pipeline.

For the lexicon-based approach, there is no need for any split of data. The lexicon dictionary does not need any data to learn, so I will use all 1.6 million tweets to test its performance.

I have fixed the error in my midterm report of not testing on the same data. Now I have the files unprocessedData.csv that are used in all of the models. I am using an 80/10/10 Train, Dev, and Test split. I am making sure to always use the last 10 percent of the data for testing across all the models in order to standardize results.

Since the dataset is essentially perfectly balanced, accuracy will be an essential metric in the performance of models. I will also use the f1 score to provide an additional metric to consider precision and recall with one score.

3.1. Specific Datasets

On top of the huge sentiment140 dataset, the following datasets test more niche tweets to test generalization.

For specific datasets, I used the following:

Sample	Tweets	Num of Data
Stanford dataset	Movie Reviews	50000
Github Corpus	Company Tweets	1092

links: Movie Dataset, Company Dataset

4. Methods

4.1. Lexicon-Based Approach

The lexicon-based approach I tried uses the AFINN dictionary. The AFINN lexicon assigns words with a score that runs between -5 and 5, with negative scores indicating negative sentiment and positive scores indicating positive sentiment. AFINN has 2,477 words with 878 positive and 1598 negative words. This is the baseline to test my other models against.

4.1.1. Preprocessing

To effectively use the AFINN, it is important to preprocess the data. This is also crucial in the machine learning models, so I will discuss how and why I used the methods I did.

Tweets are very casual and have many characters and weird spellings. For example, "@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D". Ats (@switchfoot) and links are often found in tweets and do not offer much, if any, towards any sentiment analysis, so it is important to remove them to have less noisy data and faster running times. Additionally, words are often elongated and misspelled. Unfortunately, I did not find a great way to fix incorrect words, but I did apply stemming. Stemming is a natural language processing (NLP) technique that reduces words to their base form. The goal of stemming is to simplify and standardize words, which helps improve the

performance of information retrieval, text classification, and other NLP tasks.

In addition to stemming, since the two methods I tried today do not learn relationships of words, I implemented the removal of stopping words. Stop words are a set of commonly used words in a language. Examples of stop words in English are “a,” “the,” “is,” “are,” etc. These words do not carry sentiment and are unimportant to Lexicon or Naive Bayes methods.

After preprocessing, tweets are shortened to their most essential words in lowercase. Example:

Original Tweet: @JimMacMillan I love New Zealand, I go there every two years, terrific place. Have fun.

Proccessed: love new zealand go every years terrific place fun.

4.2. Machine Learning Based Approach

4.2.1. Naive Bayes

For the first machine learning-based approach, I used Naive Bayes. I assumed that, conditioned on the label y, each word of x is sampled independently. Additionally, I will also invoke a second assumption that is specific to text documents: I will also assume that for every j, the distribution $P(x_j|y)$ is identical, similar to how we did in class.

While both of these assumptions are not true, it is practical. I expect that the set of words in a positive tweet will be similar to other positive tweets. In a situation where a tweet uses the same words but in a different order to change the sentiment, it will perform poorly, but I assume situations like these will be rare.

When I test Naive Bayes's flexibility and generalization over other topics. I will store the vocabulary from the train set of the original and add the vocabulary of the training set of the other datasets.

4.2.2. Transformers

Transformers are very powerful text analysis tools and have been the cause behind the AI craze recently. I was really interested in their abilities to do sentiment analysis and their ability to generalize. The preprocessing for transformers is not as extreme as the Naive Bayes or Lexicon-based approach. Since it wants to learn relationships between words, I still want the tweets to be complete sentences and do not care if there are words like ”of” or ”the.”

I will start by using the Bert embedding and tokenizers and the train on top of this. The pre-learned transformer is an encoder only, which is the only thing I care about with sentiment analysis.

The tweets become tokenized and then inputted into the model. The Bert tokenizer breaks down each tweet into a sequence of subword tokens, capturing both individual words and subword units to preserve the contextual information inherent in the language. Given the informal nature of tweets, this tokenization process is beneficial for handling abbreviations, slang, and emoticons commonly used on Twitter. The resulting tokenized tweets serve as the input to the model for training and testing.

This will overcome the lack of attention in the two other models. Both other models only look at singular words and make incorrect assumptions. With transformers and more specifically, attention, this will overcome those weaknesses and be able to learn more complex relationships and sentiment in tweets.

When tasked with generalization, I will test it before tuning it to the dataset and compare the results to the predictions on sentiment140 after fine-tuning.

4.2.3. Reccurent Neural Networks

RNN is a good choice for sentiment analysis as, like transformers, they improve upon Naive Bayes and Lexicon-based approaches through learning relationships. Since tweets and language are sequential, the RNN will allow the words to be able to build off of each other.

I played around with different layer sizes, lengths, epochs, and activation functions but felt three layers of sizes 24, 8, and 1 using real and 2 epochs gave me the best result. I choose teh same max length of 64 for the same reason as I did for the transformers.

5. Experiments

5.1. Lexicon-Based Approach Results

After using the lexicon-based approach for the 1.6 million tweet dataset, the confusion matrix is below.

Table 1. Confusion Matrix for Processed Data using AFINN Dictionary

Actual/ Predicted	Positive	Negative
Positive	454790	74499
Negative	236192	304752

The resulting accuracy of the approach was 82.54% for the ones it predicted, positive or negative. However, the major problem is that over half of the data results in a neutral score. 529765 tweets AFINN says are neutral, which makes the accuracy only 52%

For example, the processed tweet is Tweet: rain rain go

away, Score: Negative. But Afinn does not have any of those words in the dictionary, resulting in a score of 0.

To try to fix this, I used a more extensive dictionary. The Bing dictionary had 6789 words compared to Afinn's 2477 words. The confusion matrix is below.

Table 2. Confusion Matrix for Processed Data using Bing Dictionary

ACTUAL/ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	443988	74748
NEGATIVE	251978	241114

The accuracy is 84.68%, but it still suffers from a neutral sentiment for 588170 examples making the true accuracy around 50% as well/

To ensure it was not due to over-processing the data and possibly removing important words, I ran the examples with AFINN and BIng without preprocessing. While improved, there were still 506318 tweets with neutral sentiment.

There were still important aspects learned from the lexicon-based approaches. The most common positive and negative words are below:

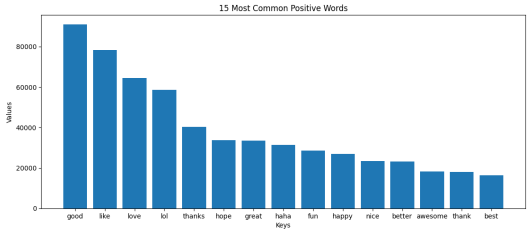

Figure 1.

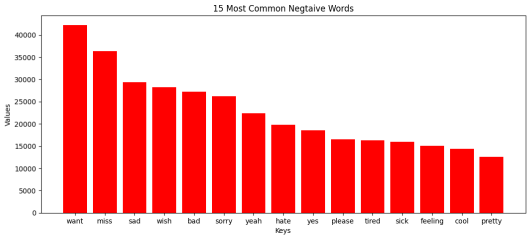

Figure 2.

These will be words we expect to be influential in our multinomial Bayes approach and future Machine Learning models.

5.1.1. LEXICON GENERILIZATION

While the method is not very accurate, it generalizes pretty well. It produces around 50% for both specific datasets. Intuitively, it makes sense as the dictionaries learn sentiments of common words you will probably see regardless of the topic.

5.2. Naive Bayes Approach Results

Using the 80/10/10 Training Split, I ran the model with unprocessed and processed data.

For the unprocessed data, the confusion matrix is below:

Table 3. Confusion Matrix for Unprocessed Data using Naive Bayes

ACTUAL/ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	58552	21316
NEGATIVE	13609	66524

The test accuracy was 78.172%. The accuracy is much better than the lexicon-based approaches while being faster to compute.

To analyze what the model has actually learned, I printed out the 20 most influential words for each category. To do this, since we are feeding our model one word at a time, we can use the equation:

$$P(y = \tilde{y}|w) = \frac{P(\tilde{y}) * P(w|\tilde{y})}{\sum_{\tilde{y} \in L} P(\tilde{y}) * P(w|\tilde{y})}$$

Where L is the label, the higher values signify a strong association with the label.

For the unprocessed data, the tables below show the strongest associative words.

While the performance is not bad, the model seems to be learning abstract data. The most influential values are links, @, and other words, you would expect to not be as influential and common. You can assume these values would not generalize well.

For the processed data, the confusion matrix is below:

The training accuracy was 79.931%, and the test accuracy was 76.873%. The f1 score for the test set was .77. This is a decent f1 score, and the accuracy is less than the unprocessed data.

While the accuracy is lower, whenever looking at the most influential words, the model seems to learn a more "correct" set of influential words. Instead of their being links, @'s, and other not sentimental words, the most influential negative words are oucchhh and hurtsss, which are more applicable to the actual English language. The same is evident

Table 4. Most Influential Words for Unprocessed Data

LABEL	WORD	VALUE
NEGATIVE	FAWCETT	0.994
NEGATIVE	ISPLAYER	0.993
NEGATIVE	#INAPERFECTWORLD	0.992
NEGATIVE	BUMMED.	0.991
NEGATIVE	DIED!	0.987
NEGATIVE	#DONTYOUHATE	0.986
NEGATIVE	#TAG	0.985
NEGATIVE	HTTP://TR.IM/LVBU	0.982
POSITIVE	WWW.TWEETERADDER.COM	0.998
POSITIVE	WWW.TWEETERFOLLOW.COM	0.998
POSITIVE	WWW.IAMSOANNOYED.COM	0.995
POSITIVE	TIME..FOLLOW	0.992
POSITIVE	@BANKSYART	0.992
POSITIVE	LAUGH..HAVE	0.987
POSITIVE	HTTP://WWW.4OFFICEAUTOMATION	0.987

Table 5. Confusion Matrix for Processed Data using Naive Bayes

ACTUAL/ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	60670	19198
NEGATIVE	17806	62327

in the positive words as we see hearty and appreciative.

Table 6. Most Influential Words for Processed Data

LABEL	WORD	VALUE
NEGATIVE	FAWCETT	0.996
NEGATIVE	OUCHHH	0.993
NEGATIVE	SADDD	0.980
NEGATIVE	HURTSSS.	0.978
NEGATIVE	SADFACE!	0.978
NEGATIVE	OUCHIE	0.975
NEGATIVE	HEARTBROKEN	0.974
NEGATIVE	OWIE	0.968
POSITIVE	DIVIDENDS	0.984
POSITIVE	HEARTY	0.967
POSITIVE	RECOMMENDS	0.977
POSITIVE	SUBSCIRBEEE	0.961
POSITIVE	WAHAHAHA	0.957
POSITIVE	APPRECAITE	0.950

5.2.1. NAIVE BAYES GENERALIZATION

For the unprocessed data, I said how the abstract words it had learned would be bad at generalizing. When exposed to the other two datasets, the performance suffered greatly. When completely blind, it performed better than I initially expected:

The accuracies, respectively, are 71.23% and 69.739%. While not great, it is still much greater than the lexicon-based approach when it comes to generalizations as well.

Table 7. Confusion Matrix for Unprocessed Data using Naive Bayes Blind to New Company Tweets

ACTUAL/ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	78	34
NEGATIVE	29	78

Table 8. Confusion Matrix for Unprocessed Data using Naive Bayes Blind to New Movie Tweets

ACTUAL/ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	4743	887
NEGATIVE	2308	2620

Additionally, when exposed to the train set of the datasets, the accuracies rose to 77.626% and 82.137%, much more respectable results.

Surprisingly, the processed tweets performed worse when generalized. For the company tweets, when blind, the model scored 70.776%, and when exposed to the training set, it scored 73.059%. For the movie tweets, it scored 67.295% and 82.118% when exposed to its training data.

5.3. Transformers

Since transformers rely on relationships, the only preprocessing I did was remove obscure characters and links. I used the Bert Transformer Encoder as I wanted a good starting point for my transformers. Since I am using transformers for the first time, it provides a baseline accuracy for me to build upon. The model has 12 layers, as I did not add any to the Bert model.

For my transformer, I chose a max length of 64. Since tweets are only 280 characters, most will have less than 64 words. Additionally, I believe that the sentiment will have already been revealed if a tweet is longer than 64 words.

After being trained on the sentiment140 dataset, the transformer had an accuracy of 81.4 percent.

5.3.1. TRANSFORMERS GENERALIZATION

When the transformer model was blind to the movie data set, the confusion matrix and classification report are below.

Table 9. Confusion Matrix for Transformer Model Blind to New Movie Tweets

ACTUAL/ PREDICTED	POSITIVE	NEGATIVE
POSITIVE	1834	656
NEGATIVE	1018	1771

Classification Report:	precision	recall	f1-score	support
0	0.64	0.74	0.69	2490
1	0.73	0.63	0.68	2789
accuracy			0.68	5279
macro avg	0.69	0.69	0.68	5279
weighted avg	0.69	0.68	0.68	5279

Figure 3. Classification Report for Transformer Model Blind to New Movie Tweets

I was fairly disappointed with this result as it was underperforming compared to the much simpler Naive Bayes. However, when exposed to the dataset and allowed to train and fine-tune, it became much better.

Classification Report:	precision	recall	f1-score	support
0	0.87	0.82	0.84	2490
1	0.85	0.89	0.87	2789
accuracy			0.86	5279
macro avg	0.86	0.85	0.86	5279
weighted avg	0.86	0.86	0.86	5279

Figure 4. Classification Report for Transformer Model Tuned to New Movie Tweets

This was by far the best result I had reached. The same trend was seen in the company tweet dataset as well.

These results, when transformers were fine-tuned to niche datasets, were the best models that were created.

5.4. RNN

The RNN had very similar results to the Transformers.

However, it performed worse on the other datasets

6. Discussion

What I think I could have improved on my models, especially transformers and RNN was to add more layers. I was constrained by time as well as resources on Google collab. I believe that if the models had more layers and were able to learn for longer, they would have been able to capture more complex relationships and produce better results across all

Classification Report:	precision	recall	f1-score	support
0	0.88	0.89	0.88	115
1	0.87	0.86	0.87	103
accuracy			0.88	218
macro avg	0.88	0.88	0.88	218
weighted avg	0.88	0.88	0.88	218

Figure 5. Classification Report for Transformer Model Tuned to New Company Tweets

Figure 6. Classification Report Sentiment140 using RNN

Classification Report:	precision	recall	f1-score	support
0	0.81	0.83	0.82	80132
1	0.83	0.81	0.82	79868
accuracy			0.82	160000
macro avg	0.82	0.82	0.82	160000
weighted avg	0.82	0.82	0.82	160000

datasets.

My transformer and RNN models both got one of the examples I highlighted in the midterm report wrong still.

Label: Negative, Tweet: 'happy birthday babe even tho think, hes mad'

The goal of using RNN and Transformers was to learn that "even tho hes mad" is the part that creates a negative connotation regardless of the earlier words. I believe this is a sign of the transformers and RNN models not having enough depth and parameters to learn the complex relationships that occur in human text.

Additionally, tweets with sarcasm seem to be very hard to find a model to decode

Label: Negative. Tweet: Love getting my teeth cleaned

I am not sure exactly how to tackle this problem. Tweets with this kind of sarcasm may not be able to be decoded by transformers or RNN, This would require prior knowledge fo whether or not the person likes the dentist or not.

Finally, emojis offered problems. I think I should have created a dictionary to map emojis to words that may have helped the models as many tweets had emojis that were the main contributor to positive or negative, leading to the model learning and predicting on some ambiguous tweets

7. Conclusion

Overall, the new transformer and RNN model do well in generic classification. I am happy with how the Transformer

Figure 7. Classification Report Movie Dataset using RNN

```
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.59      0.61      2490
           1       0.66      0.71      0.68      2789

    accuracy                           0.65      5279
   macro avg       0.65      0.65      0.65      5279
weighted avg       0.65      0.65      0.65      5279
```

Figure 8. Classification Report Company Dataset using RNN

```
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.66      0.72       115
           1       0.68      0.81      0.74       103

    accuracy                           0.73       218
   macro avg       0.74      0.73      0.73       218
weighted avg       0.74      0.73      0.73       218
```

was able to fine-tune and predict the niche datasets very accurately. The transformer seems to be a more flexible model than the Lexicon and Naive Bayes-based approaches, as it can keep getting fine-tuned. It seems the most practical model for buisness to understand responses to their tweets and perceptions about their company.

Link to Github - https://github.com/kverma2002/467FinalProjectCode