Sentiment Analysis on Twitter: Affective Keywords Visualization and Like Count Prediction

Tianchen Ye * jackye@umich.edu
Ruipu Li * liruipu@umich.edu
Yuqi Li * liyuqi@umich.edu

1 Introduction: Problem statement

Social media like Twitter, Facebook, and TikTok are playing a more and more important role in modern society. Millions of messages are posted on these platforms every day. People share their feelings, their experiences, and their attitudes in the posts. Thus, these posts contain a huge amount of information on almost every aspect of social life. For instance, when one shares his opinion about some political event, we can look at the upvote ratio and others' comments on this post to decide how people think about this event. In addition, from another perspective, we can extract a large number of posts with some specific tag and analyze the general feelings expressed in these posts. From this, we can potentially learn about how most people feel about the event related to the tag.

Our objective is to analyze the public's attitudes or feelings toward some events based on Twitter posts. Yet human is not capable of handling this task due to the large number of posts uploaded. Therefore, we would like to propose a machine-learning approach to this task. In natural language processing, a popular field called sentiment analysis is an intellectual process of extricating users' feelings and emotions(Devika et al., 2016). Sentiment analysis using machine learning models is an appropriate approach for our objective in that it can detect feelings with high accuracy and is much faster compared with humans. Besides, we will develop some custom metrics, which possibly take the number of likes, retweets, and comments into consideration to measure whether people support the opinion expressed in some posts. Combining the two methods, we should be able to tell how people feel about some event.

In addition, as a classification problem, a number of different methods can be used to process the posts. We will train different models. We are also interested in how these models treat the data or in other words, which words in a post contribute the most to the prediction. Extracting these important words can give us a basic grasp of how the models work. More importantly, these words are highly likely to be relevant to the expression of people's feelings or attitudes. Therefore we can learn more information from the posts.

In summary, we aim to propose a machine learning approach to systemically analyze the public's attitudes towards some events based on Twitter posts and use a model explainer to learn some keywords which express the corresponding attitudes.

2 Related Work

Sentiment analysis has gained widespread acceptance in recent years, not just among researchers but also among businesses, governments, and organizations.

- In 2020, Sujata Rani and Parteek Kumar proposed an aspect-based sentiment analysis using dependency parsing(Rani and Kumar, 2021). They assign a separate sentiment towards the different aspects of a sentence (By generating a dependency graph, they assign the sentiment to an aspect having a minimum distance), as well as evaluate the overall sentiment expressed in a sentence.

- There are also sentiment analysis projects focusing on Twitter, for example, MonkeyLearn is a very famous and effective machine learning platform that is widely used by many companies and organizations.

For the prediction of "the degree of approval", there have been several previous works involving supervised learning tasks on Reddit data, specifically with predicting Reddit post popularity (number of upvotes - downvotes of a post).

*University of Michigan Department of EECS

- (Segall and Zamoshchin, 2012) studied the prediction of net upvote ratio both as a multi-class classification problem where the possible range of upvotes is and a regression problem. The study used a random sample of over 2 million Reddit posts, and used features including title and text embedded using TF-IDF, author, time created, and whether the post is marked as "over 18". The study tried several techniques, including naive Bayes and multi-class SVM for classification, and linear regression for regression.

- (Shuaibi, 2019) also studied the prediction of net upvote ratio, focusing mainly on regression techniques. The study only utilized the title of each post as the only textual feature and utilized other features like the number of comments and whether the post was given any Reddit rewards. The study also used engineered features such as the length of the title and the sentiment of the title. Three regression models were trained on posts across several subreddits from the first 6 months of 2018, including linear regression, KNN regression, and random forest regression.

3 Approach

Our work mainly consisted of two parts: first, we trained our sentiment analysis model to predict the emotion of each tweet; second, we defined a new metric "degree of approval" and trained a model to predict that value for each tweet.

3.1 Sentiment Analysis

Although a lot of previous works did a really good job on sentiment analysis and got very powerful models, most of them just get a high accuracy result and did not explore further why their model makes such predictions.

Therefore, we not only train sentiment analysis models to get the predictions but also visualize the result and learn which words contribute most to the model's result.

We propose a system that first gets a keyword and loads relevant tweets through Twitter API, and then uses a trained sentiment analysis layer to make the classification predictions on those tweets. Besides, it puts the data and the trained model into a model explainer layer, which will give us statistical details about how the model makes such predictions. Through this architecture (shown in Figure 1), we can not only predict the sentiment of each tweet but also see how many contributions each word makes towards that classification.

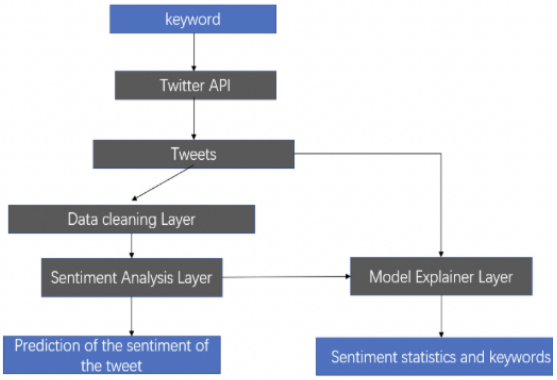
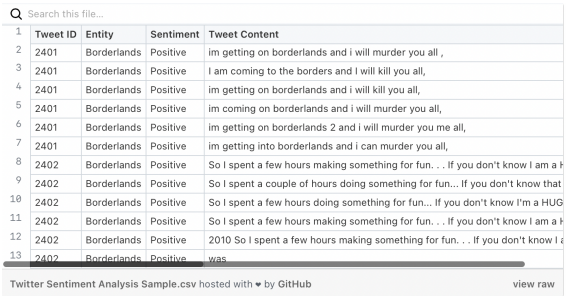


Figure 1: System architecture

3.1.1 Data Collection and Preprossing

The training data we used in the sentiment analysis layer is the "Twitter Sentiment Analysis" dataset from Kaggle. It is a collection of approximately 74000 tweets, and each has a manually assigned sentiment label.


Search this file...

	Tweet ID	Entity	Sentiment	Tweet Content
2	2401	Borderlands	Positive	im getting on borderlands and i will murder you all ,
3	2401	Borderlands	Positive	I am coming to the borders and I will kill you all,
4	2401	Borderlands	Positive	im getting on borderlands and i will kill you all,
5	2401	Borderlands	Positive	im coming on borderlands and i will murder you all,
6	2401	Borderlands	Positive	im getting on borderlands 2 and i will murder you me all,
7	2401	Borderlands	Positive	im getting into borderlands and i can murder you all,
8	2402	Borderlands	Positive	So I spent a few hours making something for fun. . . If you don't know I am a H
9	2402	Borderlands	Positive	So I spent a couple of hours doing something for fun... If you don't know that
10	2402	Borderlands	Positive	So I spent a few hours doing something for fun... If you don't know I'm a HUG
11	2402	Borderlands	Positive	So I spent a few hours making something for fun. . . If you don't know I am a H
12	2402	Borderlands	Positive	2010 So I spent a few hours making something for fun. . . If you don't know I a
13	2402	Borderlands	Positive	was

Twitter Sentiment Analysis Sample.csv hosted with ♥ by GitHub view raw

Figure 2: Kaggle's Twitter Sentiment Analysis dataset

The raw data contains four labels: positive, neutral, negative, and irrelevant; besides, the max length of the tweet content is 957 while most of the tweets have around 150 words. Therefore, we need to do some data cleaning. First, we delete all irrelevant data from the dataset. Then, we remove some too-long tweets to make a relatively uniform distribution. Moreover, since there are many unnecessary or meaningless words in tweets, for example, people's names, URLs, etc., we need to remove them as well. After those preprocessing, the dataset distributions are shown in Figure 2 and Figure 3.

The data we use to evaluate is got through Twitter API. We use Tweepy's method

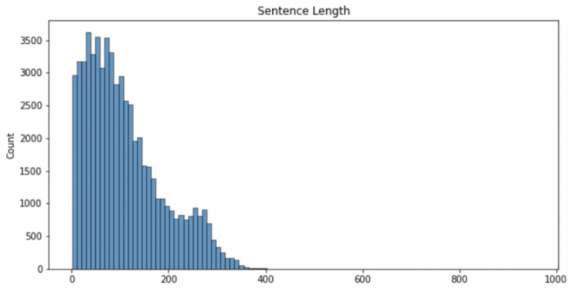


Figure 3: Length distribution

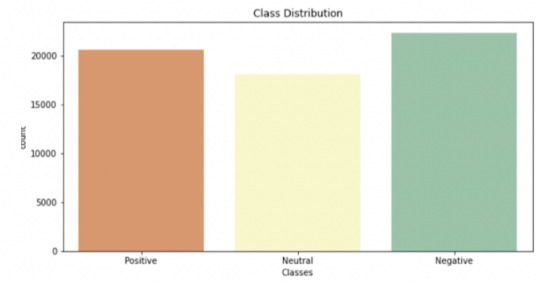


Figure 4: Label distribution

search_recent_tweets(keyword) to get the most recent tweets which contain the keyword specified by us. This method is accessible to all users but the search period is limited to about one week from the current time. Fortunately, for most keywords, we are able to extract enough tweets we need using this method. After we get the tweets we want, we apply the same data cleaning process on those data to make sure they do not contain any meaningless words.

3.1.2 Model

We fine-tuned the pre-trained BERT and RoBERTa models for this task. BERT is a transformer-based machine-learning technique for natural language processing. It contains a variable number of encoder layers and self-attention heads. RoBERTa builds on BERT's language masking strategy, where the system learns to predict intentionally hidden sections of text within otherwise unannotated language examples. It modifies some key hyperparameters in BERT, including removing BERT's next-sentence pretraining objective and training with much larger mini-batches and learning rates. This allows RoBERTa to improve on the masked language modeling objective compared with BERT and leads to better downstream task performance.

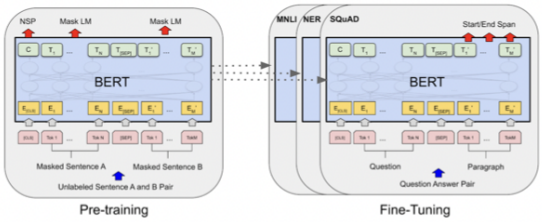


Figure 5: BERT architecture

3.1.3 Explainer Layer

To explain what our model learned from the tweets, we use SHapley Additive exPlanation(SHAP)(Lundberg and Lee, 2017) to quantify the contribution that each word in the tweet brings to the predicted emotion by our model.

By explaining the model's results, we can gain knowledge of how the data is processed by the model and which part of a post contributes the most to the final prediction. Moreover, we will extract some keywords that are most influential to the prediction. We think that these words potentially express the attitudes determined by our models.

For example, given a tweet "I feel so bad today", the SHAP value of the word "bad" is calculated by:

$$SHAP_{bad}(x_0) = [1 \times \binom{5}{1}]^{-1} \times MC_{bad,\{bad\}}(x_0) + \quad (1)$$

$$[2 \times \binom{5}{2}]^{-1} \times MC_{bad,\{so,bad\}}(x_0) + \quad (2)$$

$$[2 \times \binom{5}{2}]^{-1} \times MC_{bad,\{bad,today\}}(x_0) + \quad (3)$$

$$\ldots \quad (4)$$

3.1.4 Word Cloud Extraction

After the explainer layer, we get shap values for each sentence. For a sentence of length l (l words in the sentence), the shap values returned by the explainer layer is a NumPy array of shape (num_labels x l), where in our models, num_labels = 3 (Positive, Neutral, and Negative).

Then we use the shap values to extract the word cloud, which is a collection of words that has a top impact on the model. We think such words summarize the general sentiment about some keywords. First, we extract a number of tweets about a keyword using the method described in section 3.1.1. Then we get shap values for every tweet. From the shap values, we use two methods to extract the word cloud.

Method one is to first take all shap values of a specific label, such as "Positive". Then we take the

average impact of all words and select the words with the highest impact. Basically, the word cloud got from this method contains words with positive and negative impacts on the target class because the impact is measured by the absolute value of the corresponding shap value.

Method two is to first classify each sentence as positive, neutral, or negative using one of our trained models. Then for positive sentences, we take the word with the top positive impact on the positive class. For negative sentences, we take the word with the top negative impact on the negative class, and similarly for neutral sentences. We iterate through all sentences to get a bunch of words associated with each class. We then take the first twenty words with the highest frequency for each class to be our word cloud.

3.2 Like Count Prediction

For the second part, It is hard to get people's attitudes towards a tweet due to the limitation of Twitter API. So, instead of directly predicting how other people think of the tweet, we will focus on how many likes a tweet has. However, the like count cannot be the only metric for the attitude since it is easily influenced by other factors. We will use two approaches to predict the degree of approval towards the tweet:

1. First, we treat it as a classification problem. We will set several thresholds for the like count we collected and divide it into three categories: normal, popular, and very popular. We then fine-tuned a pre-trained Bert model and add a linear output layer to classify the text.

2. Second, we treat the problem as a regression problem. We will train a neural network to predict the degree of approval of the tweet. We first use the "all-mpnet-base-v2" model for sentence vector transformation, then we add several dense layers to learn the information of the sentence vectors. Finally, we add a dense linear layer as our regression layer.

3.2.1 model

Classification Task

The Bert-Base-Case layer is a pre-trained Bert model which has 12 layers of transformer encoder. It will output a [*,768] feature vector. The Relu layer applies the Relu function to the computed results to avoid overfitting. The Linear Layer takes a [*,768] vector as input and outputs a vector of size [*,3]. The classification layer will choose the highest score among the 3 classes and output the predicted popularity classification.

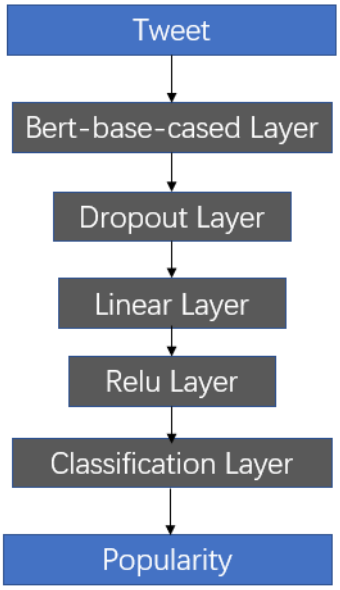


Figure 6: classification model architecture

Regression Task

The first layer is used for sentence embedding. It converts each tweet into a 384-features vector if we use the all-MiniLM-L6-v2 model or generates a 768-features vector if we use the all-mpnet-base-v2 model. Then the vectors will be fed to the dense layer for regression. We added a dropout layer after each dense layer to avoid over-fitting and enhance the generality of our model. After the dropout layer, we also applied a Relu activation function which can avoid linearity. The dense 1 layer outputs a 256 features vector, and the dense 2 layer outputs a 128 features vector. Unlike the classification task, the linear output layer maps to a scalar value representing the like count of the input tweet.

3.2.2 Data Collection and Preprocessing

The dataset we used is the "Elon Musk Tweets Dataset" from Kaggle. This dataset contains all the tweets of Elon Musk and the average like count is relatively large. This enables us to avoid dealing with tweets whose like count is mostly 0 or very small, and can hence facilitate the prediction. What's more, since all the data is from Elon Musk's tweets, we can focus on how his opinions are liked by Twitter users and analyze which kind of sentiment is preferred.

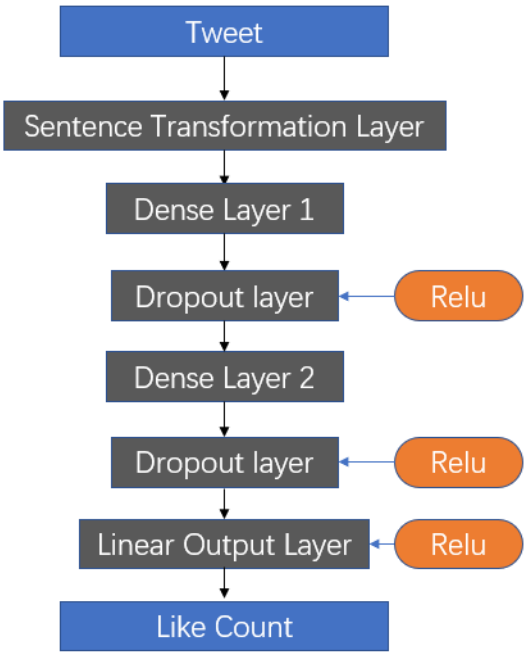


Figure 7: Regression model architecture

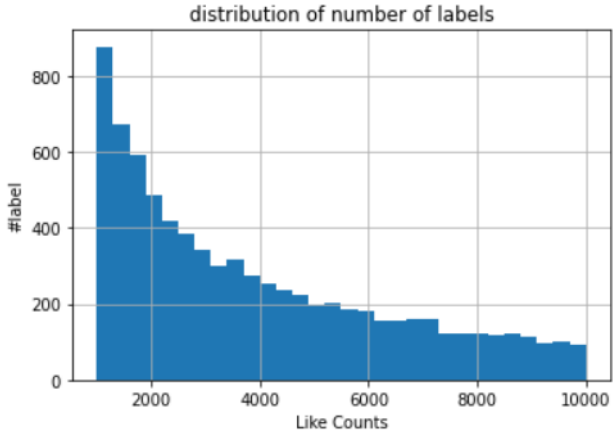


Figure 8: Distribution of Number of Labels

4 Evaluation

4.1 Experiment setup

Our models were trained on Great Lakes. We requested 47GB memory on partition spgpu. The training time was about two hours for our sentiment analysis models. For the data we used for evaluation, we pulled 500 tweets related to the keyword "house of dragons" through Twitter API and apply both our sentiment analysis models and explainers to these data. We extracted several words with strong emotions from the tweets and generated the word cloud graph to analyze Twitter users' attitudes.

4.2 Sentiment Analysis

In this part, we will show the performance of the two sentiment models we trained. We will also pass the sentiment models into our SHAP explainer to show the visualization results. And we will show word clouds we extracted from Twitter using recent tweets. The word clouds will consist of some keywords which were identified as having strong emotional tendencies by our sentiment models.

4.2.1 Model Performance

We evaluated both models on our test set, which consists of 828 tweets unseen by our models. Before evaluation, we extract tweets with "positive", "neutral" and "negative" labels and leave out tweets with other labels like "irrelevant". We also do some data cleaning like lemmatization and removing stopwords, punctuation, URLs, other users' names, and so on. The following table gives the accuracy and the training epochs.

	epochs	accuracy
RoBERTa	15	96.25%
BERT	5	83.17%

Table 1: Model performance

Since our model predicts multiple labels for each example, we also calculated the confusion matrix. The results are also shown as follows.

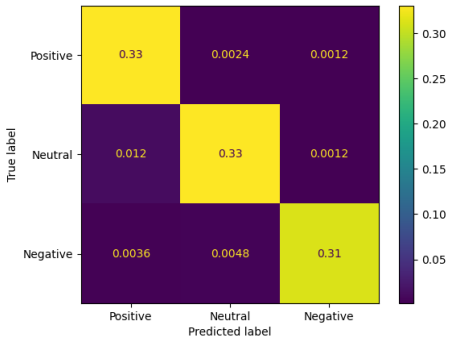


Figure 9: Confusion Matrix of RoBERTa model

From the figure of the confusion matrix of the RoBERTa model, we can see that the predictions on the test set are almost balanced. And the RoBERTa model performs pretty well on it; while the confusion matrix of the BERT model shows that it might

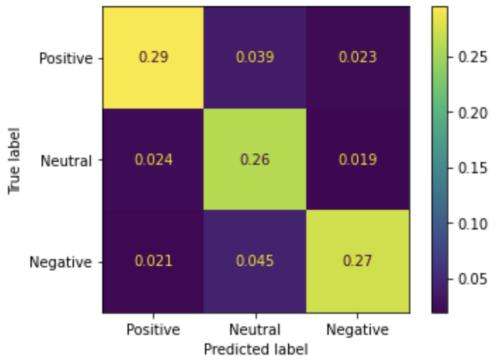


Figure 10: Confusion Matrix of Bert-based-uncased model

have some bias that makes it tend to give "Positive" or "Negative" predictions when the true label is "Neutral".

4.2.2 Explanation on one sentence

To get a better idea of how our models process a sentence, we use SHAP to get a visualization of weights put on each word by our models. Some examples are shown as follows.

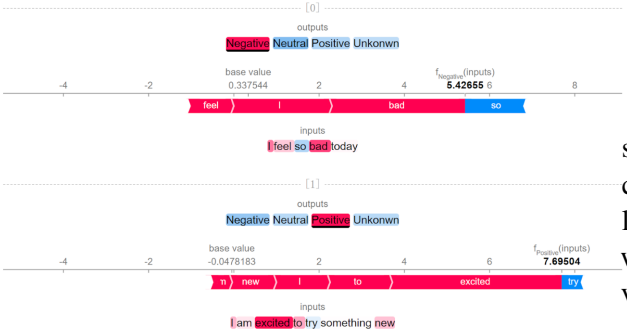


Figure 11: Visualization of shap values on two sentences

From Figure.11, as our expectation, we see that the tweet "I feel so bad today" was identified as negative correctly. In this figure, the words' contributions to the "Positive" label are shown. Red bars are positive contributions and blue bars are negative contributions. And the word "bad" was considered to have the greatest negative contribution. For the tweet "I am excited to try something new", the word "excited" contributes most to the positive sentiment.

4.2.3 Word Cloud Extraction

In this section, we will show 4-word clouds extracted by our models. The first two were extracted by the RoBERTa model using two methods introduced in section 3.1.4. The last two were extracted by the bert-based-uncased model using the same two methods. The keyword is "house of dragons", which is a popular series recently. We pulled 500 tweets from 2022-12-10 to 2022-12-14. All of the words were extracted using the same data.

The following table shows some of the words we extracted. These words have a positive contribution to the "Positive" class so we expect them to be affirmative.

Method	Words
roberta + method 1	good, Looooooooooool, shadwkinq, rewatching, sherlock, but, finally
roberta + method 2	finally, love, month, watching, rewatching, hiphopmusic , good
bert + method 1	greatest, finally, use, considers, sd, good, love
bert + method 2	go, peace, care, thank, t, enjoyed , watched

Table 2: Word clouds extracted by our models

In method 1, to see how important each word selected is, we also visualized the shap values associated with the words as shown in Figure.12. Similarly in method 2, to learn the importance of each word which is reflected by the frequency, we also visualized it in a bar plot as shown in Figure.13.

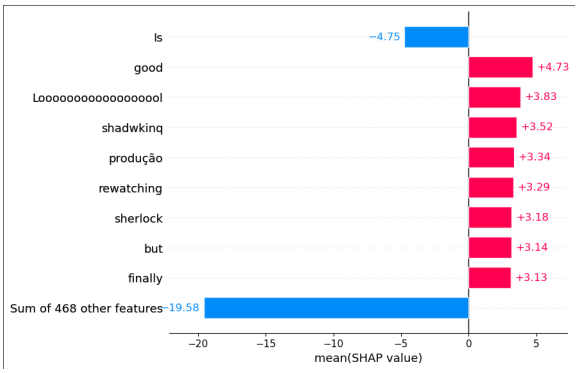


Figure 12: Visualization of shap values on word cloud (roberta + method 1)

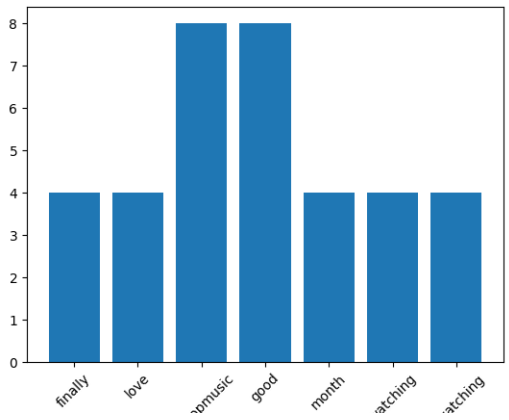


Figure 13: Visualization of frequency of words (roberta + method 2)

4.3 Like Count Prediction

4.3.1 Regression Task

We first define the following loss functions and metric functions:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|.$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{\hat{y}_i - y_i}{y_i}| \times 100\%.$$

We experimented on two different models and tried different loss functions and metrics. First, we set our loss function to be MAE (Mean Absolute Error) and metrics function to be MAPE (Mean Absolute Percentage Error) and the results are shown in Table 3.

model	lr	loss	metric
all-mpnet-base-v2	10^{-5}	1990.38	61.77%
all-MiniLM-L6-v2	10^{-5}	1962.69	59.53%

Table 3: Results for MAE loss and MAPE metrics

We then set our loss function to be MAPE (Mean Absolute Percentage Error) and metrics function to be MAE (Mean Absolute Percentage Error) and the results are shown in Table 4

model	lr	loss	metric
all-mpnet-base-v2	10^{-5}	48.18%	2282.67
all-MiniLM-L6-v2	10^{-5}	47.83%	2275.53

Table 4: Results for MAPE loss and MAE metrics

4.3.2 Classification Task

We first classified the like count into three different degrees of popularity

1. normal (like count < 5000)
2. popular (5000 < like count < 10000)
3. very popular (like count > 10000)

We used the cross-entropy loss as our loss function and got the results shown in Table 5

Task	lr	epoch	BS	Acc
Classif.	10^{-6}	30	256	68.6%
Classif.	10^{-6}	30	128	68.3%
Classif.	10^{-5}	30	256	66.9%

Table 5: Results for classification task

5 Discussion of Results

5.1 Sentiment Analysis

5.1.1 Data Cleaning

For data cleaning, in our system, we simply remove all the URLs and other usernames in tweets. However, in a real scenario, those parts may contain some emotional tendencies as well. Besides, we did not handle some special things like emojis, slang words, and misspellings, which frequently appear on Twitter and could definitely contain some emotional tendencies.

5.1.2 Different Word Cloud Extraction Methods

In this project, we introduce two different methods to generate our word clouds. From section 4.2.3, we observe that the word clouds generated by different methods have a few words overlapped. The reason is that method 1 will focus more on the word's contribution to one tweet; while method 2 considers the word's impact on the whole keyword tweets set.

For example, in our result, RoBERTa + method 1 gives word "Looooooooooool". Obviously, this word shows strong positive emotion and is therefore extracted by method 1. However, this word might only appear once in the "house of dragons" tweets set. Therefore, it will not be shown in method 2's result.

We also observe a difference between word clouds extracted by different models. Comparing RoBERTa + method 1 and bert + method 1,

the word cloud extracted by BERT contains more words that are neutral. For example, "use", "considers" and "finally". We think this can be explained by Figure.10. As the BERT model tends to classify "Neutral" sentences as "Positive" and "Negative", we expect neutral words to have larger contributions to the classification.

5.1.3 Limitation of our method

As is shown in Table.2, some of the words in word clouds are not exactly what we expect. First of all, some of the words cannot give us much information we would like to know. For example, the extracted words of RoBERTa + method 1 contain "but", and "finally". These two words are not very descriptive. In addition, the model often extracts meaningless words when using method 1. An example in Table.2 is "sd" extracted by BERT. Another limitation of our method is in the data collection part. We are not able to collect too many tweets without an academic API. Our word clouds shown in the Section.4 are extracted using 500 tweets, which is not a big number. And many of the tweets are classified as neutral, which means they have no contribution to the final result. We think more data will definitely give us more meaningful words in the word cloud.

5.2 Like Count Prediction

5.2.1 Data Comparison

Our first goal is to treat the task as a regression problem and predict the like count of each tweet. However, the results are far from satisfactory. As you can see in Table 3 and Table 4, we tried two different sentence vector transformation models and trained with two different loss functions: MAE and MAPE. The best MAE value is 1962.69 and the best MAPE value is 47.83%. Although the all-mpnet-base-v2 model is more complicated than the all-MiniLM-L6-v2 model and has more parameters, the all-MiniLM-L6-v2 model performs better. The reason might be that the all-mpnet-base-v2 model is overfitted to the training data and hence perform worse on the testing data.

To improve our prediction model, we then decided to treat the task as a classification problem. We classified the like count into three different degrees of popularity and trained a model that predicts with 68.6% accuracy. The reason that we cannot further improve the accuracy might be that cross-entropy loss is not the most appropriate loss function. In our experiment, a tweet with 4999 likes is considered normal and a tweet with 5001 likes is considered popular. However, they actually have little difference.

5.2.2 Evaluation Metric

Another possible reason that affects our model's performance might be the evaluation metric of degree of likeness. In this project, due to the limitation of Twitter API, which does not allow us to get too much information about a tweet, we only use Likes count as our metric. However, the Likes count does not only depend on the tweet itself but is also influenced by many other factors, like how many users see the tweet and whether the user who posts the tweet is popular.

To further improve the performance of our models, we need to get more data for each tweet, reconsider all those related factors and design a new value metric for evaluation.

6 Conclusion

In this project, we propose a system that could help people to analyze Twitter users' attitudes toward some keywords. We trained two sentiment models and compared their performance. We also applied SHAP explainer on both models and generated word clouds for the "house of dragons" tweets set pulled through Twitter API. By giving the visualized results, we can easily learn why people like "house of dragons" and why they do not like it.

7 Division of Work

Our team consists of three members: Tianchen Ye(@jackye), Ruipu Li(@liruipu), and Yuqi Li(@liyuqi). In this project, we divided the whole work into two parts: sentiment analysis and like count prediction. For the sentiment analysis part, Yuqi and Ruipu trained two different models and put their models to the SHAP explainer to generate their own word clouds. For the second part Like Count Prediction, Tianchen trained the models through two different perspectives: classification and regression. Finally, we evaluate our system together on data pulled from Twitter. We combined all the models and explainers together, compared their performance, and figured out some potential bias in each model.

References

M.D. Devika, C. Sunitha, and Amal Ganesh. 2016. Sentiment analysis: A comparative study on different

approaches. *Procedia Computer Science*, 87:44–49. Fourth International Conference on Recent Trends in Computer Science Engineering (ICRTCSE 2016).

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Sujata Rani and Parteek Kumar. 2021. Aspect-based sentiment analysis using dependency parsing. *Transactions on Asian and Low-Resource Language Information Processing*, 21(3):1–19.

Jordan Segall and Alex Zamoshchin. 2012. Predicting reddit post popularity. *nd): n. pag. Stanford University*.

Ahmed Shuaibi. 2019. Predicting the popularity of reddit posts general machine learning.