Tweet Text Extraction Based on Sentiment

CHENG, Lin
HKUST
Big Data Technology
20711693
lchengaq@connect.ust.hk

LAU, Cho Han
HKUST
Big Data Technology
20729585
chlauay@connect.ust.hk

QIAO, Shuyu
HKUST
Big Data Technology
20747563
sqiaoac@connect.ust.hk

***Abstract*—Sentiment analysis is a common task in the field of Natural language processing (NLP). It is performed to identify and extract the sentiment or opinion of a sentence. Knowing the sentiment of a sentence is important for companies or brands to evaluate the impact of the sentences. However, we do not know which word is used for showing the sentiment. By extracting the keywords or keyphrases from a sentence, we are able to identify the sentiment of the sentence. In our project, we are extracting the words or phrases that were used for sentiment description from Tweets. We are trying to construct a model that can look at the labeled sentiment for a given tweet and figure out what word or phrase best supports it. We use nltk, SpaCy and RoBERTa models for training the data and find that RoBERTa produce the best result.**

***Index Terms*—Sentiment Analysis; Extract keywords; nltk; NER; SpaCy; RoBERTa; Text Mining.**

I. Introduction

Twitter is one of the most popular Social Networking Sites that people use nowadays. People share their thoughts and feelings on Twitter. Twitter is also a platform used by many companies, influencers, and celebrities to announce and share important information. With millions of tweets posted, liked, and retweeted around, it is difficult to evaluate whether the sentiment of a sentence will have an impact on a company, a person, or a brand. Evaluating the sentiment of sentences is thus essential when decisions and reactions are made and updated in seconds. However, we do not know which word is used for sentiment description. In general, the sentiment of a sentence can be classified as positive, negative, and neutral. Some words or phrases are extracted from a sentence to reflect its sentiment. For example, given a sentence: ”My ridiculous dog is amazing.” [sentiment: positive]. The sentiment of this sentence is positive, then we need to extract the words that can fully express this positive emotional information. For instance, the word ”amazing” in the sentence can express positive emotions. In this project, we extracted part of the tweets, usually some words or phrases, that are used for evaluating the sentiment. We used several models to train our data, including Natural Language ToolKit (nltk), SpaCy, and A Robustly Optimized BERT Pretraining Approach (RoBERTa). We found that RoBERTa produced the best result among all the others.

For the contribution of this project, Shuyu is responsible for doing the data preprocessing, computing exploratory data analysis (EDA). Cho Han is responsible for building the nltk model. Lin is responsible for building one of the SpaCy models, which uses all three sentiments (positive, negative, and neutral) for training the model and building the second SpaCy model, which uses the three sentiments separately for model training. Finally, the three of us build the RoBERTa model together.

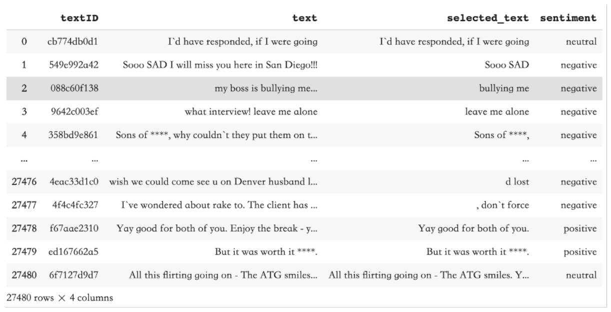
II. Related Works

Several previous studies have addressed the techniques of text extraction task. In the study of Greenberg (1998), he pointed out the powerfulness of NLP. NLP was useful in accessing and handling a large number of electronic archives. It could be used as an information seeker, which enabled searches of a collection of text to take place. An NLP search was conducted by matching a document’s free texts with its controlled vocabulary simultaneously, also known as keyword searching. Yet, the NLP mentioned by Greenberg was unable to understand the content of the archival records. Thus, other techniques, like sentiment analysis, were crucial in understanding the content of a text. In our project, we used a few other techniques, such as building the Named-Entity Recognition (NER) model and RoBERTa model, to perform the data extraction task. The data extraction task we do in our project was in line with the study of Kim, Medelyan, Kan, and Baldwin’s study (2012). They analyzed the result of a keyphrase extraction task competition. Participants were required to extract keyphrases from a given document. Keyphrases extracted automatically by the system were compared with those extracted by the participants. A traditional means of matching was used to evaluate the original keyphrases with the keyphrases generated by the participants. In specific, Kim et al. computed TF x IDF n-gram based baseline, using both supervised and unsupervised approaches. They first identified keyphrases candidates by generating 1-, 2-, and 3-grams for both testing and training data. For the supervised learning system, a maximum entropy (ME) learner was used to learn a supervised baseline model by looking at the keyphrase candidates and TF x IDF scores of the original keyphrase generated. For the unsupervised learning system, ranking the keyphrase candidate by TF x IDF scores was used as the basis. Finally, F-scores were computed for each team of participants. It was found that the results of the participants were not very good. The F-scores were found to be relatively low since keyphrase extraction was a subjective task. Also, a strict evaluation system was used and thus many semantically-equivalent keyphrases were found to be false. Therefore, keyphrase extraction is found to be a common task in data mining, which is similar to what we are doing for our project. In another study, Al-Jumaily, Martinez, Martinez-Fernandez and Van der Goot (2011) suggested using Named-Entity recognition (NER) to perform text mining tasks for the Arabic language. They stated that NER is a good source for capturing the semantic content of a written text. In their study, they built a system to detect the existence of some named entities and a set of common words written in Arabic. They divided named entities into 3 categories, including person, location, and organization. A recognition task was applied to detect nouns and verbs for common words. For easy understanding of Al-Jumaily et al.'s system, they first introduced Arabic and the system architecture built for Arabic text mining and analysis. A system dictionary called PMG, including Prefixes P, Morphological Resources M, and Gazetteer G, was created to save the patterns retrieved from several information resources. It was found that NER was useful in the verification process of the recognition results. Thus, we also use the NER model in the SpaCy model of our project. In a study by Liu, Ott, Goyal, Du, Joshi, Chen, Levy, Lewis, Zettlemoyer and Stoyanov (2019), they first performed a replication study of BERT pretraining (Sevlin et al., 2019) and then proposed an improved model of BERT named RoBERTa. They stated that the comparison between different language models was challenging and showed that different hyperparameter choices would lead to a significant impact on the final result. They later found that RoBERTa could compute the same result and even exceed the performance of every other model published after it. In specific, RoBERTa was first modified in the following ways: models were trained with bigger batches and more data, the next sentence prediction objective was removed, longer sentences were used for training, the masking pattern used in the training data was changed. Other improvements to the BERT model were implemented to form the final RoBERTa model. The modifications included training with dynamic masking, full-sentences without Next Sentence Prediction (NSP) loss, large mini-batches, and a larger byte-level Byte-Pair Encoding (BPE). As for our project, we also use the RoBERTa model to perform the keywords/sentences extraction task.

III. Dataset

This is a dataset of tweets which consists of 27,481 rows and 4 columns(partial shown in Fig. 1).

- textID - unique ID for each piece of text
- text - the content of the tweet
- sentiment - the general sentiment of the tweet, including 3 types: neutral, negative and positive
- selected_text - (train only) a subtext of text that supports the tweet's sentiment



	textID	text	selected_text	sentiment
0	cb774db0d1	I`d have responded, if I were going	I`d have responded, if I were going	neutral
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative
2	088c60f138	my boss is bullying me...	bullying me	negative
3	9642c003ef	what interview! leave me alone	leave me alone	negative
4	358bd9e861	Sons of ****, why couldn`t they put them on t...	Sons of ****,	negative
...	...	...	...	...
27476	4eac33d1c0	wish we could come see u on Denver husband l...	d lost	negative
27477	4f4c4fc327	I`ve wondered about rake to. The client has ...	, don`t force	negative
27478	f67aae2310	Yay good for both of you. Enjoy the break - y...	Yay good for both of you.	positive
27479	ed167662a5	But it was worth it ****.	But it was worth it ****.	positive
27480	6f7127d9d7	All this flirting going on - The ATG smiles...	All this flirting going on - The ATG smiles. Y...	neutral

27480 rows × 4 columns

Fig. 1: Dataset

IV. Methods and Technology

Firstly, we partition the raw dataset according to the ratio of 8:2, of which 80% is the training set and 20% is the test set. Then we do lots of exploratory data analysis (EDA) on the training set, so as to have a general understanding of the data. Through the analysis and understanding of the training set, we use several methods to do text extraction based on sentiment. In the beginning, we use the more familiar nltk library for simple text processing and analysis, and the accuracy was not high. Then we use SpaCy to build our NER model. When we build the same model on three sentiments, the effect is not as accurate as building different models on each model separately. Finally, we construct RoBERTa model on the training set, after five-fold cross-validation, we get better results on the test set than the previous methods, and the Jaccard score reach 70.9%.

A. Data Pre-processing

In this project, we remove the missing values, and divide the original training data (27,481 observations) into two datasets, in the ratio of 8:2 as the new training data and new test data separately. The training set we handle contains 21,984 observations, and the testing set contains 5,496 observations. In the subsequent model training, we only train the training set, and test the trained model on the test set to obtain the Jaccard scores between prediction and ground truth.

B. Exploratory Data Analysis

Since the training dataset includes the key emotional words and classifies them in different kinds of sentiment, we split them to count the number of emotional words selected and the number of words in the original text, also the differences between them. We apply Jaccard index to calculate the similarity between selected emotional words and the original text, which conducts a new dataframe with the Jaccard scores from 0 to 1 representing similarity from weak to strong. After checking the missing values in this dataframe, we count the numbers of tweets posted which imply different sentiments in descending order, 8,924 for 'neutral', 6,841 for 'positive' and 6,219 for 'negative'. The funnel-chart(Fig.2) below shows the distribution of sentiments with their proportions. Neutral tweets take up the most position as 40.6% while only 28.3% for negative tweets.

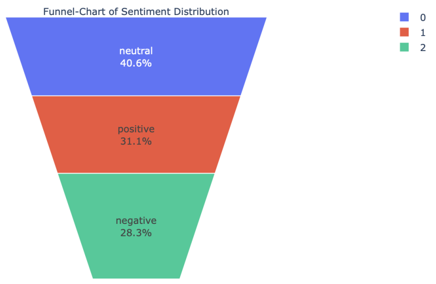


Fig. 2: Funnel Chart of Sentiment Distribution

We explore the distribution of meta-features(Fig.3). The distribution of words' numbers combines both words count from selected and original text. The histogram is right-skewed, which displays that most tweets have less than 25 words.

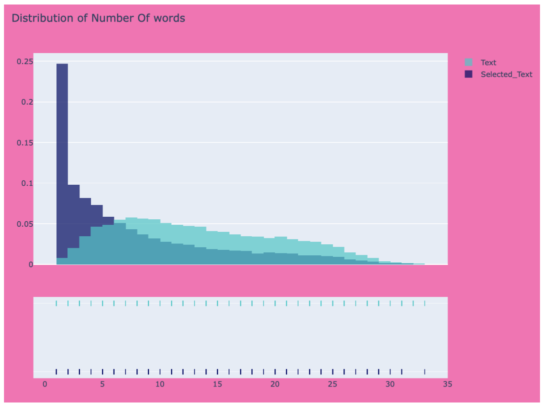


Fig. 3: Distribution of Number of Words Between Text and Selected Text

The plot of difference in number of words under the neutral tweets(Fig.4) shows that there is almost no difference between the number of words selected and original text as the most differences are zero. The plot of Jaccard scores((Fig.5) for neutral sentiments verifies that the Jaccard scores for most neutral tweets are 1.0s. It can be observed that the selected text and the original text are mostly the same for neutral sentiment.

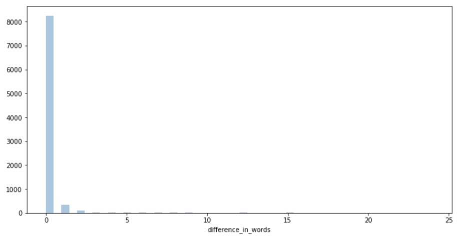


Fig. 4: Plot of Difference in Words' Number for Neutral Sentiment

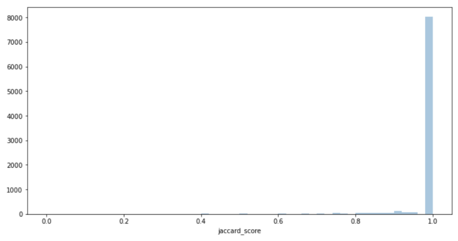


Fig. 5: Plot of Jaccard Scores for Neutral Sentiment

To compare the Jaccard scores for these three sentiments together, we contribute the kernel distribution of Jaccard scores(Fig.6) across all three sentiments. This plot presents two obvious trends. Neutral tweets concentrate in one region around the peak around a score of 1.0. Positive tweets and negative tweets have similar distributions, their patternings overlap a lot. Both of them concentrate in two regions with high densities around two peaks, with scores around 0.1 and 1.0 separately. In this case, we conclude that neutral tweets have low kurtosis, positive and negative tweets have high kurtosis where kurtosis is one measure for the peak of one distribution and the spread around this peak.

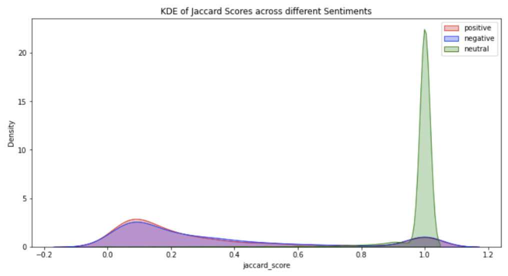
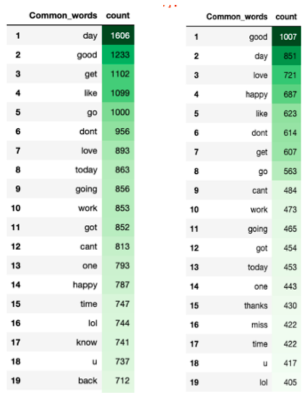


Fig. 6: KDE of Jaccard Scores Across Different Sentiments

There is one common observation for these three sentiments that all have the peak around Jaccard score of 1.0. This implies a high similarity existing in a cluster of tweets between the selected and original text. Then we can predict text for selected text independent of the segment of text through finding these clusters for different sentiments. Filtering the text with no more than two words to get the selected text with high probability used completely from the original text. We calculate the average Jaccard scores for neutral, negative, and positive sentiments, as 0.97, 0.79, and 0.76 individually. These three average scores are all close to 1.0 which shows the similarities between selected and original texts. We closely check each sentiment's information, the most texts used as the selected text. Therefore, we improve this observation by preprocessing the word count of texts within two. In addition, exploring the most common words that expressed sentiments is meaningful for this project. Before this exploratory, we clean the corpus. Making all text lowercase, removing text in square brackets, links, punctuations, and words containing numbers to get the clean dataset with only words. The most three common words in the selected text are ’i’, ’to’, and ’the’; the least is ’good’. In this case, it is hard to find the representative words for sentiments. Then we remove all stop words and count again. From the tables((Fig.7) shown below of counting the top twenties common words in both texts and selected texts, the most common words appear in texts also appear in selected texts, such as ’day’, ’good’ and ’like’ etc.



	Common_words	count
1	day	1606
2	good	1233
3	get	1102
4	like	1099
5	go	1000
6	dont	956
7	love	893
8	today	863
9	going	856
10	work	853
11	got	852
12	cant	813
13	one	793
14	happy	787
15	time	747
16	lol	744
17	know	741
18	u	737
19	back	712

	Common_words	count
1	good	1007
2	day	851
3	love	721
4	happy	687
5	like	623
6	dont	614
7	get	607
8	go	563
9	cant	484
10	work	473
11	going	465
12	got	454
13	today	453
14	one	443
15	thanks	430
16	miss	422
17	time	422
18	u	417
19	lol	405

Fig. 7: Common Words Count in Text(left) and Selected Text(right)

These three tree graphs(Fig.8) below displayed the most common words in positive, negative, and neutral sentiments. Positive words like ’good’, ’happy’, ’love’, ’thanks’, and ’great’, appearing frequently in positive tweets. Negative words such as ’miss’, ’sorry’, ’bad’ and ’hate’, strongly expressed unsatisfied negative sentiment in tweets posted. Other words in neutral sentiment actually do not express any distinct sentiments, like ’work’, ’going’ and ’get’ etc.

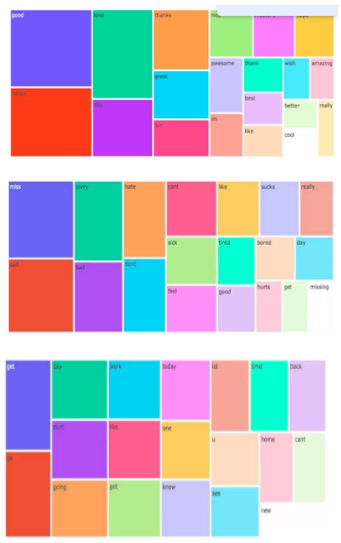


Fig. 8: Tree Graphs in Positive(top), Negative(middle) and Neutral (below)

Because of the unofficial posts on Twitter, the expression of tweets is random. Most people write ’thank’ for short as ’thnx’, ’good’ as ’goood’ also a way to emphasize their satisfaction for something. As these words are presented, we consider them as unique words and take a look at them in each segment. These three DoNut plots(Fig.9) of unique words in order of positive, negative, and neutral sentiments give a much clearer version of our dataset, which indicates that unique words play a significant role in determining the sentiments.

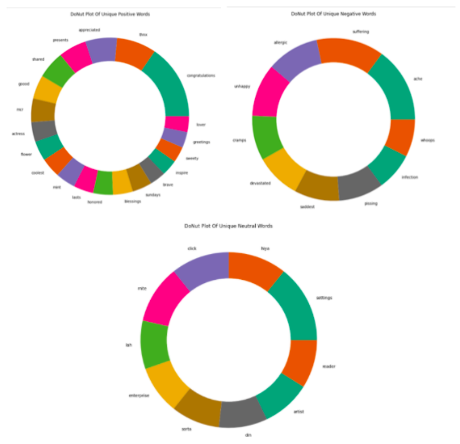
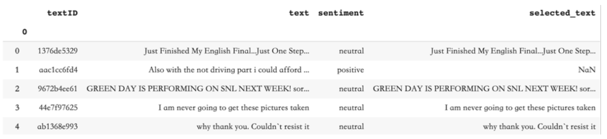


Fig. 9: DoNut Plots for Unique Words

C. *nltk*

After EDA, we have noticed that the average Jaccard score between text and selected text on the training set is around 97.6% when sentiment is neutral. That means the selected text and original text are almost the same. So we consider keeping all the words as the selected text we extracted in neutral tweets(Fig.10).



	textID	text	sentiment	selected_text
0				
0	1376de5329	Just Finished My English Final...Just One Step...	neutral	Just Finished My English Final...Just One Step...
1	aac1cc6fd4	Also with the not driving part i could afford ...	positive	NaN
2	9672b4ee61	GREEN DAY IS PERFORMING ON SNL NEXT WEEK! sor...	neutral	GREEN DAY IS PERFORMING ON SNL NEXT WEEK! sor...
3	44e7f97625	I am never going to get these pictures taken	neutral	I am never going to get these pictures taken
4	ab1368e993	why thank you. Couldn`t resist it	neutral	why thank you. Couldn`t resist it

Fig. 10: Result of Keeping Neutral Tweets

For positive and negative tweets, We used Python’s nltk to do NLP(Fig.11).

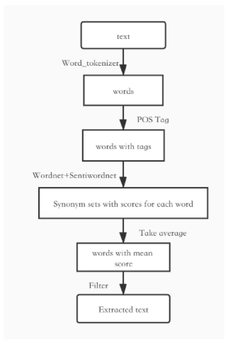


Fig. 11: Nltk Flow Chart

Firstly, we tokenize the text and extract the stem of every word. Then, we attach a part-of-speech tag to each word, converting the P.O.S tags to simple Wordnet tags. According to the tag, we find the set of synonyms for each word in the nltk corpus, and calculate the sentiment scores for each word in all synonym sets. Finally, the average sentiment score of each synonym set is used to represent the sentiment score of the corresponding word, including positive score and negative score. If there is no stem or no synonyms found for the work, then an empty set is returned. After we get the sentiment scores of all words in each text, we extract the corresponding sentiment scores (pos or neg) against the sentiment of the text. At this point, we set a threshold to filter out words with scores less than this threshold, and the final words are the predicted selected-words.

We use different thresholds for text extraction on the training set and analyze their Jaccard scores(Fig.12).

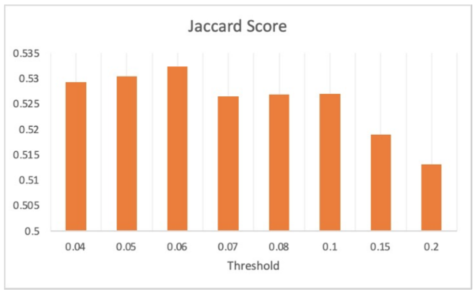


Fig. 12: Accuracy of Different Threshold

We choose 0.06, the threshold with the highest score on the training set to predict on the test set, and the final Jaccard score is 52.6%. The result of this prediction was not very good, and the words filtered by the threshold are not continuous, which would cause the extracted text to be inconsistent.

D. SpaCy

After consulting relevant information, we decide to use NER to solve this problem. First, we convert the training data into the format required by SpaCy and train a SpaCy NER model by adding our custom entities present in the training dataset. The training process consisted of the following five parts(Fig.13):

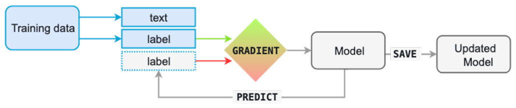


Fig. 13: SpaCy Flow Chart

1) Create an empty model using spacy.blank ("en") and add the entity recognizer to the pipeline. If there is already a model, just load the existing model and disable all other pipeline components during training.
2) Add the new entity label to the entity recognizer.
3) Loop over the examples and call nlp.update, which steps through the words of the input. At each word, it makes a prediction. It then consults the annotations, to see whether it was right. If it was wrong, it adjusts its weights so that the correct action will score higher next time. We can set different parameters, such as the value of batch_size, n_iter and drop, which determine how big the batch is for each training , how many the number of iterations are, and how much the data is memorized. We batch up the examples using SpaCy's minibatch and use spacy.util.compouding to iteratively generate a series of sizes.
4) Save the trained model.
5) Test the model on a validation set. Adjust the important parameters in the fourth parts by the Jaccard scores on the validation set.

Then, we select the model that performs best on the validation set to test on the testing set. The Jaccard score is 57.8%, which is higher than nltk. Next, we consider creating models separately on tweet text of each sentiment(Fig.14), because the NER model for different sentiments should be different. The experimental result shows that this has a higher accuracy rate with a Jaccard score of 63.2%.

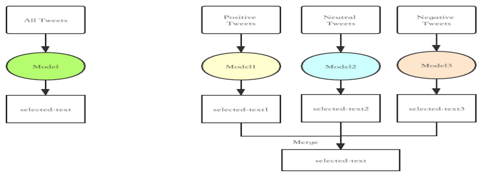


Fig. 14: SpaCy Model vs SpaCy Models

E. RoBERTa

In order to achieve a higher Jaccard score, we decide to try to build a RoBERTa model. So we use the TFRobertaModel in HuggingFace transformers. Like SpaCy, the word tokenization is performed using roBERTa vocabulary first, it converts text to character level tokens, and then text and sentiment are spliced to construct a data type that RoBERTa understands(Fig.15). Here we will set a MAX_LEN(the length of the maximum sequence length specified by BERT).Sequences with a length less than the value will be padding processed, tokens greater than the value will be cut.

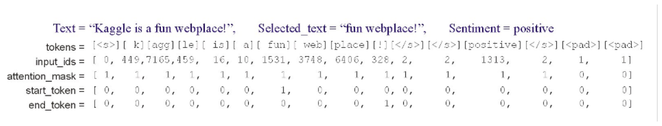

```
Text = "Kaggle is a fun webplace!",      Selected_text = "fun webplace!",      Sentiment = positive
      tokens = [<s>][ k][agg][le][ is][ a][ fun][ web][place][!][</s>][</s>][positive][</s>][<pad>][<pad>]
   input_ids = [ 0, 449,7165,459,  16, 10, 1531, 3748, 6406, 328, 2,     2,    1313,     2,    1,     1]
attention_mask = [ 1,   1,   1,  1,   1,  1,    1,    1,    1,  1, 1,     1,       1,     1,    0,     0]
 start_token = [ 0,   0,   0,  0,   0,  0,    1,    0,    0,  0, 0,     0,       0,     0,    0,     0]
   end_token = [ 0,   0,   0,  0,   0,  0,    0,    0,    0,  1, 0,     0,       0,     0,    0,     0]
```

Fig. 15: Input Embeddings

As shown above, we need five embeddings called tokens, input_ids, attention_mask, start_token and end_token. After that we download the pretrained RoBERTa model and its configuration files in Tensorflow and import them. The model building process includes the following steps(Fig.16):

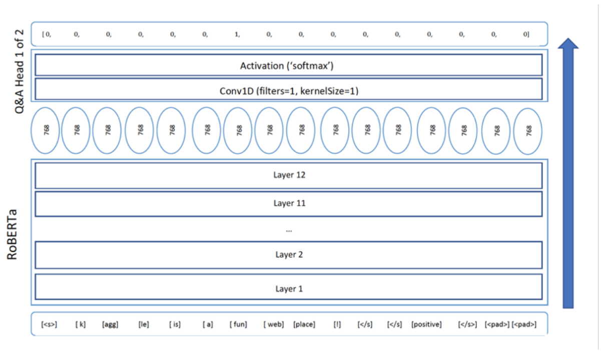

Q&A Head 1 of 2
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Activation ('softmax')
Conv1D (filters=1, kernelSize=1)
768
RoBERTa
Layer 12
Layer 11
...
Layer 2
Layer 1
[<s>] [k] [agg] [le] [is] [a] [fun] [web] [place] [!] [</s] [</s] [positive] [</s>] [<pad>] [<pad>]

Fig. 16: RoBERTa Structure

1) We input the data into the RoBERTa model and get the output of RoBERTa, which is the encoding vector corresponding to each token.
2) Because the answer is composed of consecutive tokens in the text, the process of predicting the answer is essentially the process of determining the location of the token at the beginning and end of the answer. We apply a drop layer to forget some message, apply a Conv1D layer and a Flatten layer to get the logit value at the beginning and end of the answer, finally, we use a softmax layer to get the corresponding probability. After data post-processing, the predicted answer can be obtained.

We train with 5 StratifiedKFolds, and do the five-fold cross-validation. Each fold, we set checkpoints to find the best model weights and save it.The final five-fold cross-validation average Jaccard score is 70.5%. The distribution of Jaccard scores is as follow(Fig.17):

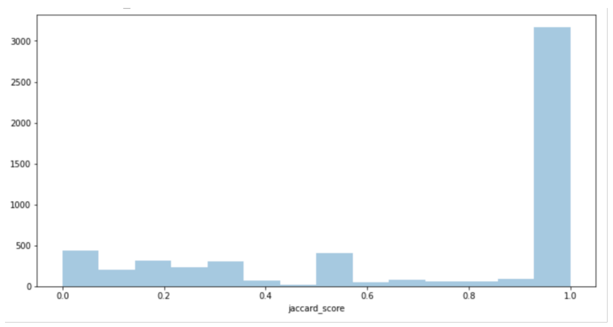


Fig. 17: Distribution of Jaccard Scores

Then we load the best model to predict the testing set and the Jaccard score on the testing set is 70.9%. As shown in the figure above, the horizontal axis is the Jaccard score and the vertical axis is the number of texts. Compared with the previous method, the RoBERTa model performs quite well.

V. Evaluation Index

The Jaccard index, also known as Intersection over Union and the Jaccard similarity coefficient (originally given the French name coefficient de communauté by Paul Jaccard), is a statistic used for gauging the similarity and diversity of sample sets. Its formula represented by sets is defined as follows:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$

It is basically a formula for measuring how much overlap there is between A and B. It can be easily visualized using venn diagrams(Fig.18)

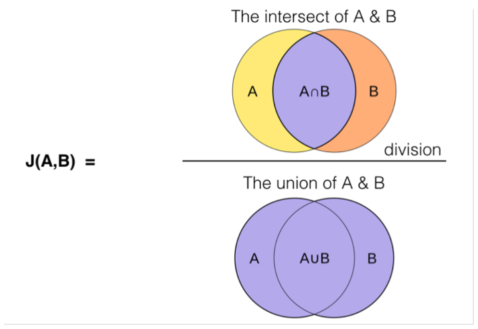


Fig. 18: Jaccard Venn Diagram

In text, Jaccard similarity refers to the size of the intersection of two sentence words divided by the size of the union of two sentence words. In this report, we call it Jaccard score. In our model evaluation, we calculate the Jaccard similarity score between prediction and ground truth. The higher the score, the higher the similarity, that is, the more accurate our model predicts.

VI. Conclusion

A. Summarization of work

The results of several models are compared as follows:

TABLE I: Testing Accuracy

Model	Default
nltk	52.6%
SpaCy1	57.8%
SpaCy3	63.2%
RoBERTa	70.9%

It is crucial to evaluate the sentiment of a tweet. However, we do not know which word is used for emotional description. Extracting some words or phrases from sentences to reflect their emotions (neutral, positive, or negative) is needed. In this project, we use a variety of models to train our data and predict the word or phrase from the tweet that exemplifies the provided sentiment, including nltk, SpaCy and RoBERTa. We find that RoBERTa achieves the best results than the others.

B. Limitation and Future Work

The result of prediction based on nltk is not very good, and the words filtered by the threshold are not continuous, which would cause the extracted text to be inconsistent. We need to adjust the algorithm to make the sentence continuous. There are many parameters in the pre-trained RoBERTa model and they are complex, and we are not very good at tuning parameters. We still need to continue to learn and be familiar with BERT model.

References

[1] ”Tweet Sentiment Data” https://www.kaggle.com/c/tweet-sentiment-extraction/data.

[2] Al-Jumaily, H., Martinez, P., Martinez-Fernandez, J. L., Van der Goot, E. (2011). A real time Named-entity Recognition system for Arabic text mining. Springer Science+Business Media B.V., May 2011.

[3] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language under- standing. *North American Association for Com-putational Linguistics (NAACL).*

[4] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer L. Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. July, 2019.

[5] Greenberg, J. (1988). *The American Archivist*, 61(2), pp. 400-425, 1998.

[6] Kim, S. N., Medelyan, O., Kan, M. Y., Baldwin, T. (2013). *Language Resources and Evaluation*, 47(3), pp. 723-742, September 2013.