The University of Texas at Dallas
CS 6320 Natural Language Processing
Spring 2021
Class Project Report

Project Title: Affect Processing from Social Media

Team 8

Students: Yibo Cheng, Chaoran Li, Mo Han, Yinglu Huang

Problem Statement

The purpose of this report is to present the process and result the team obtain from developing affect processing system operating on Twitter postings. Over past one year, Covid-19 has become the main-stream emphasis on both social media and political campaign. Information regarding to the Covid-19 has emerged and spread rapidly, especially in its mysterious origin, its side effect, mortality ratio and etc. In the meanwhile, misinformation on Covid is also obtained attention and sent out wrong and unproven messages. In this project, based on 10 misinformation, the team will decide the certain tweet's stance against misinformation and explore the improvement on accuracy with incorporation of sentiment, emotion and topic analysis.

In the course of developing the model, the team not only learn using open resource on pre-trained model, but also practiced incorporating one neural architecture into the other one. However, it is a rough start in the beginning as the team searched for available model and planning seemed to be a chaos. Once the baseline model is selected, each team member has its own tasks and communication in the team is seamless and on-time. This is the main reason the team is able to deliver the result on time and performs well in the accuracy ranking.

Member: Chaoran Li(cxl190012)

Stance Detection Model

a) A figure of the neural architecture:

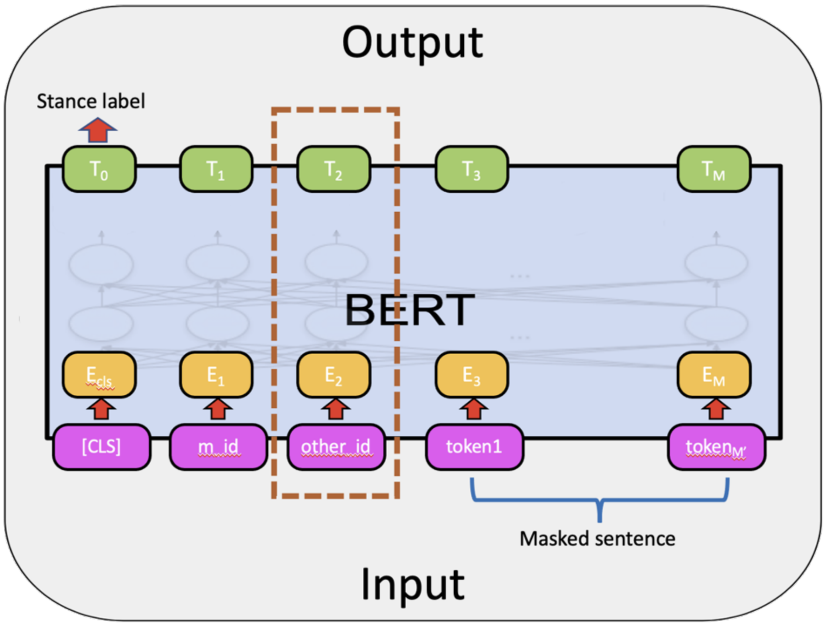


Figure 1: Stance Neural Architecture

For simple stance detection for our case, we need to input m_id(misinformation id) and text together at the same time. If we need to consider other information like sentiment, emotion, topic, etc, we also need to input extra embedding along with the two we said above. The dotted line indicates that it is only needed when other information is included.
For masked attention, we need to go through Wq, Wk, and Wv. After that, we would use softmax to get the final weight as output. Compared with self attention, we need to include matrix M in softmax.
Equations:

$$Q = XW_Q$$

$$K = XW_K$$

$$V = XW_V$$

$$Attention(Q,K,V) = Softmax\left(\frac{QK^T \odot M}{\sqrt{d_k}}\right)V$$

For output, the first digit represents the predicted embedding. Use np.argmax() function to transform an embedding into an id which represents a label of stance detection.
b) How the architecture was trained and what results it obtained on the development set:

Member: Chaoran Li(cxl190012)

After that, we can train the data with the pretrained ‘bert-base-uncased’ model. During the training, we can adjust batch size, learning rate and epochs to get better results. For the result, I firstly tried possible values for batch size and learning rate. I choose batch size to be 3 and learning rate to be 1E-5 with epsilon which should be extremely small to prevent the denominator from being zero. Then, I saved the model after each epoch. I will then choose the best model from saved models.
Below is the result during training in first 15 epochs:

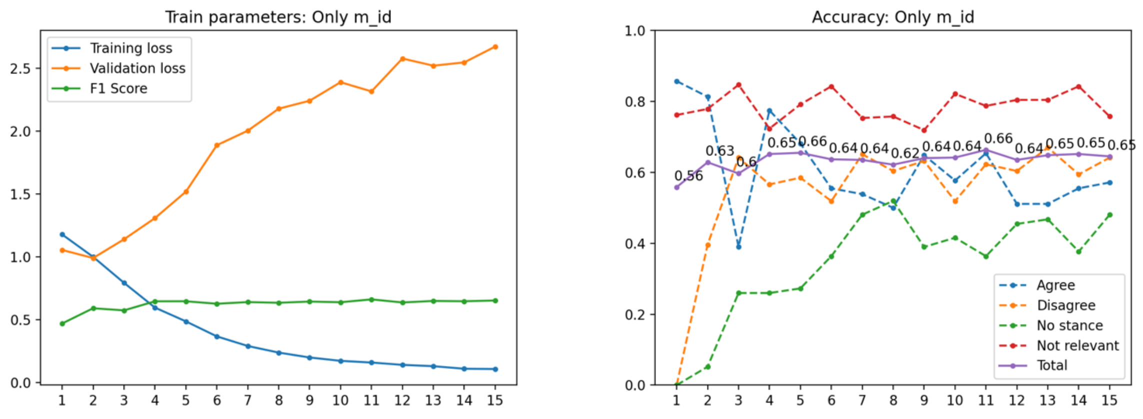
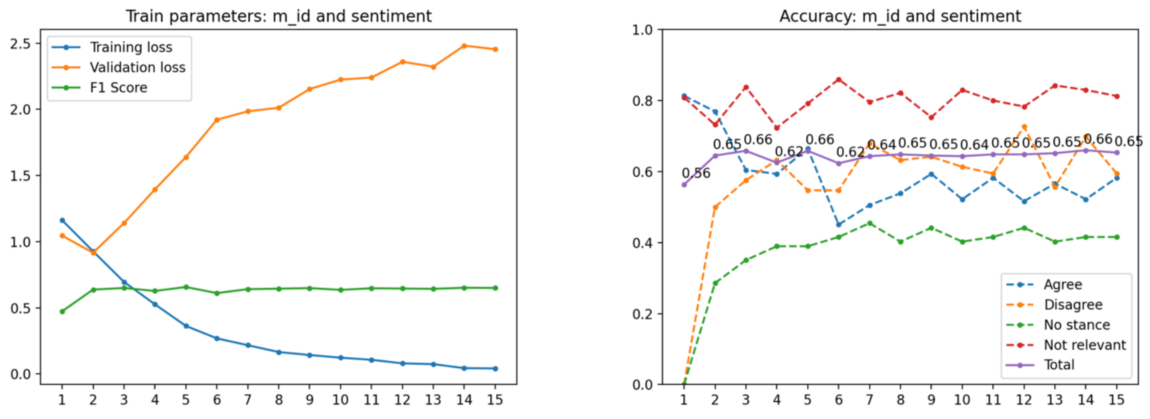


Figure 2: Results from 15 epochs

When the train is processing, the training loss decrease and the validation loss increase. After about 6 epochs, the F1 score and the accuracy nearly unchanged.
Besides, I also count the result collaborate with other information:



Member: Chaoran Li(cxl190012)

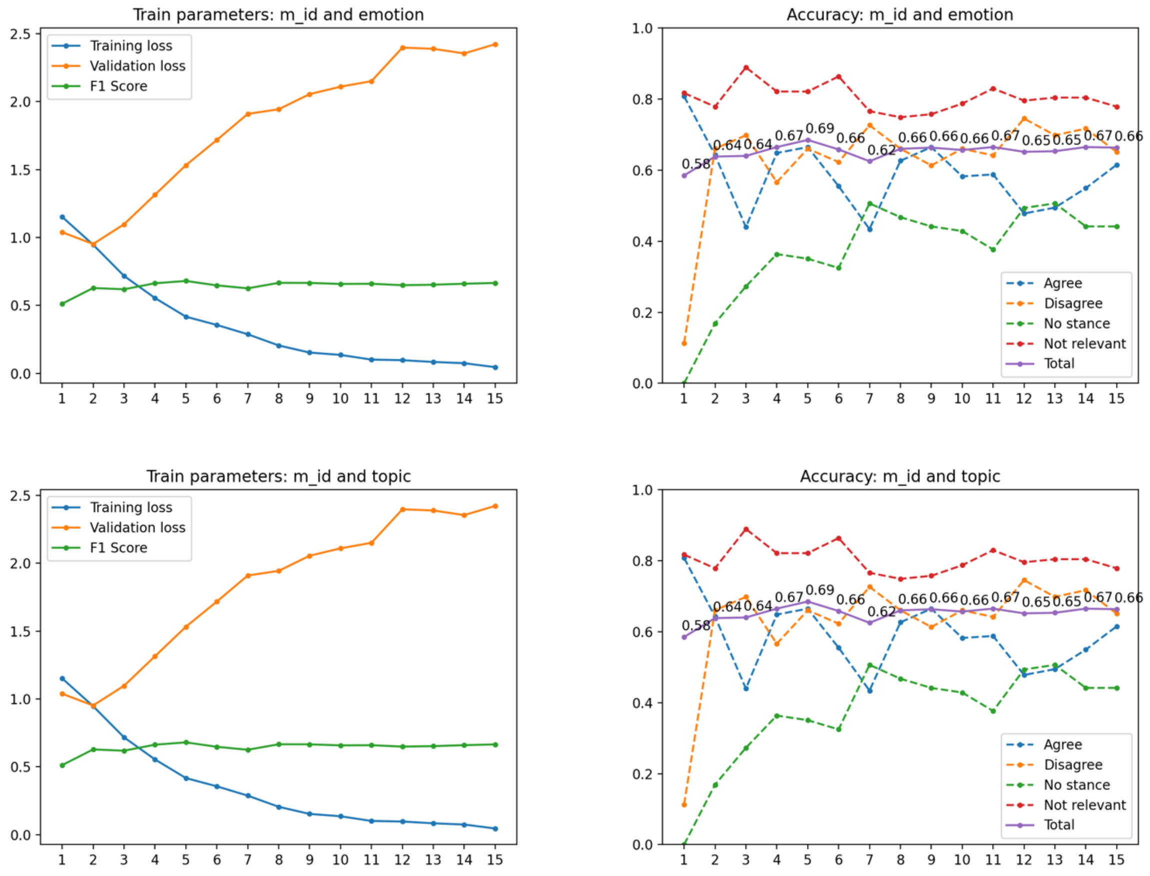


Figure 3: Accuracy Trend in Model

c) Baseline architecture using BERT:
For BERT, all the input should be encoded before entering the model. We use the BERT tokenizer for pretrained ‘bert-base-uncased’ model to encode input.
For validation, we have labeled tweet which we can directly set the input. But for prediction, we could not get labels before entering the model. However, we still need to use the forward method. Hence, we should assign the label to be 0 before prediction.
Below is a part of output from the model:

```
[[ 2.0438879   1.3755342   0.39373523 -2.1641233 ]] -> 0: Agree
[[ 2.9964478 -3.548699  -0.6125754  1.3325485]] -> 0: Agree
[[ 2.1602232 -0.6231929  1.7521578 -2.0171134]] -> 0: Agree
[[ 4.66987    -0.77939284 -0.34308136 -2.1094923 ]] -> 0: Agree
[[-1.760359   -0.83027035  2.34561     0.04697713]] -> 2: No stance
[[ 4.552128    0.08585838 -1.3349531  -2.1103904 ]] -> 0: Agree
[[ 5.0619965 -1.9012371 -1.0287616 -0.8097608]] -> 0: Agree
[[-1.1732289   3.7125492  -0.71068025 -1.2867408 ]] -> 1: Disagree
[[-1.5200349  -2.687817    0.98776805  2.9413579 ]] -> 3: Not relevant
```

Member: Chaoran Li(cxl190012)

```
[[-0.6751543  3.9518397 -0.363368  -2.1552498]] -> 1: Disagree
[[ 5.3149695 -1.3324903 -1.4584866 -1.1959326]] -> 0: Agree
...
```

Use np.argmax() function to transform an embedding into an id which represents a label of stance detection.

d) How to make available for the other affect processing tasks?

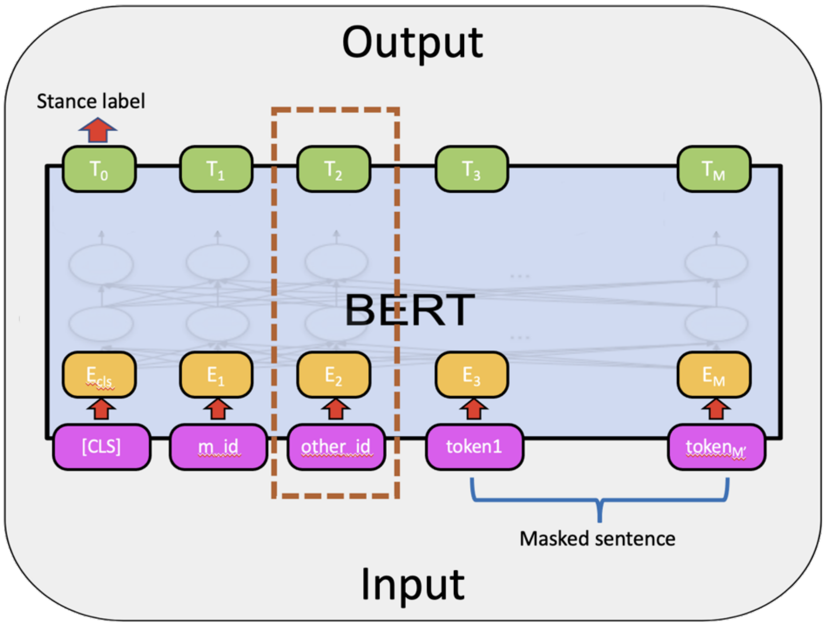


Figure 4: Bert Baseline Model

Since our results are represented in json format, we decide to make our program communicate by json files. Like for stance detection, I first read text, m_id from json files as well as other information if necessary. Then, I train the model with these labeled data. After that, I read the test text and m_id from json files. I will also label with other information from collaborating model.
After that, I will generate the result from trained model.
For the situation when we have to consider other information, this time, I directly join the m_id, other_id and text together as input. For example if m_id = 2, emotion_id = 0 and text = “aaa bbbb.”, I will use “2 0 aaa bbbb.” as input.
However, I think a Esep might be better than this way. Hence, I would try to add Esep between different elements if I have extra time to try.

What I have learned:
I met a bug caused by default format transformation when reading dataframe from json file. It results that we read an error tweet_id from json file. And the most impressive thing is that,

Member: Chaoran Li(cxl190012)

when write back to json file, it went back to original value. That is why I did not find this bug at the beginning. I read several articles about the fundamental theory behind this and it reminded me of some knowledge I have learned in other class.

Member: Mo Han(mxh190001), Chaoran Li(cxl190012)

Stance Detection with Sentiment Detection

This part mainly focuses on exploring effects brought by incorporating sentiment recognition in to stance detection system. A BERT model is introduced to make sentiment recognition over the training tweets. First, dataset for sentiment recognition from Internet is used for training and test. After the BERT model is fine-tuned, we use it to predict sentiment over the tweets. The sentiment label is merged into the stance prediction system. After training and validation process, we can get a fine-tuned stance detection system with sentiment information. Then we can use this system to predict the stance of tweets.

(1) Architectures of Sentiment Detection System and Stance Detection System

Figure 5 shows the neural architecture for sentiment recognition. The solution is using a stack of BERT and LSTM layers.

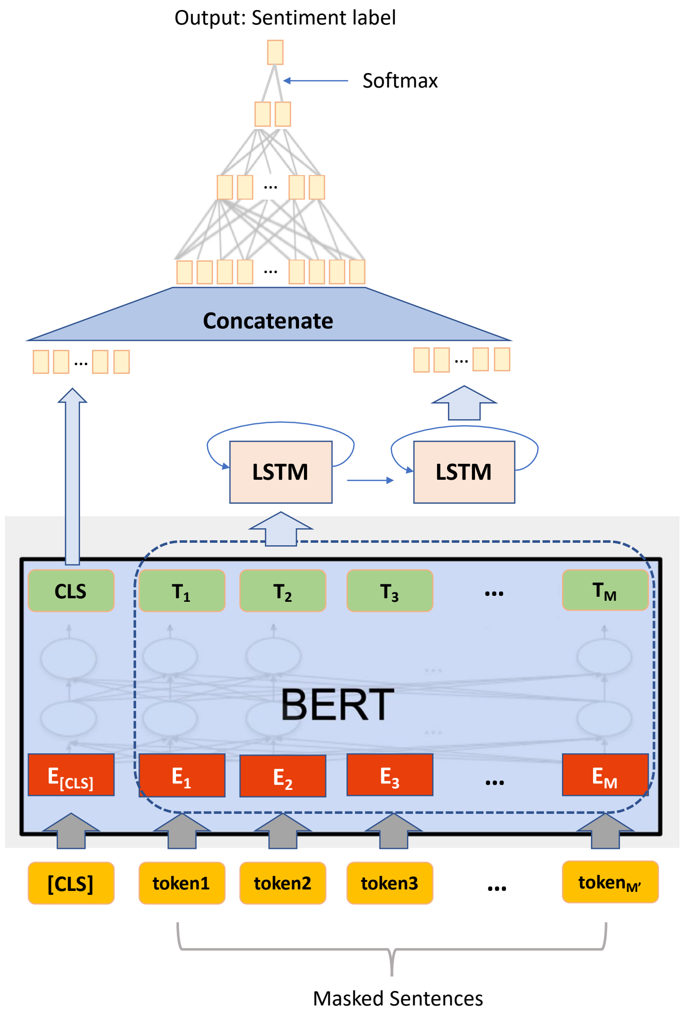


Figure 5. Neural architecture for sentiment recognition.

Member: Mo Han(mxh190001), Chaoran Li(cxl190012)

Figure 6 shows the input embeddings of BERT. There are three parts in the input embeddings, the token embeddings, the segmentation embeddings and the position embeddings. The sum of these three embeddings is the whole input.

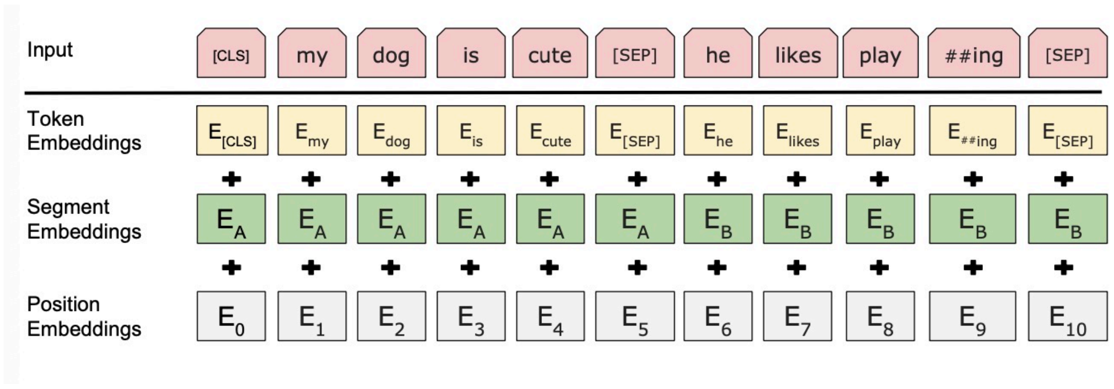

Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 6. BERT input representation.

The input of Sentiment Detection System is the text and tweet_id of tweets. Firstly, we take the output vectors of all the word tokens of tweet. Then the tokens are sent as input through LSTM layers. The output from LSTM layers are concatenated with the [CLS] token from Bert. It is finally passed to a fully connected neural network to perform the final classification.
In this sentiment detection system, Softmax activation function is used to map sentiment label to tweet. With the equation of np.argmax(), probability can be transformed into a sentiment id which represents a label of sentiment extraction. Label 1 means positive and Label 0 means negative.
Figure 7 shows the neural architecture for stance recognition. The architecture of stance recognition system is also BERT. In this system, we first get the sentiment label from the sentiment detection system. Then the sentiment label is used as one part of the token embeddings in the input of stance detection system.
BERT is formed by 12 transformers. The core parts of transformer are self- attention and feed forward networks. Equations are as below:

$$K = W^K X$$
$$Q = W^Q X$$
$$V = W^V X$$
$$SelfAttention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where K, Q and V are key, query and value vectors and X is the input embedding vector, d_k is the dimensionality of Q or K vectors.

$$h = WX + b$$
$$a = \sigma(h)$$
$$y_i = \sum_{j \le i} \alpha_{ij} v_j$$

where W is the weight matrix in the feedforward layer and b is bias. X is the output from previous layer, y_i is used for input of output layer, α_{ij} is the attention weights learned by head and v_j is the value vector.
Output:

$$Y = Uy$$
$$\hat{y} = softmax(Y)$$

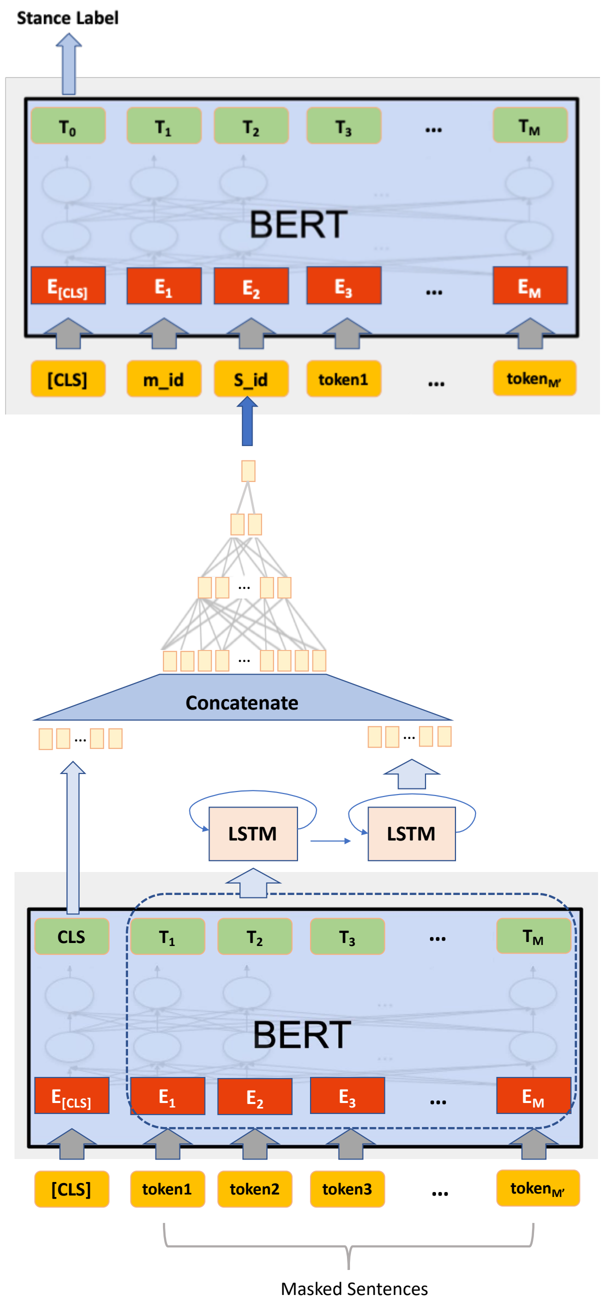


Figure 7. Neural architecture for stance sentiment recognition.

Member: Mo Han(mxh190001), Chaoran Li(cxl190012)

(2) Training process of Sentiment Detection System and Stance Detection System

Firstly, we choose a BERT base model to pre-train. In this model, there are 24 layers, 1024 hidden, and 16-heads. A batch size of 32 is used for model.

I chose to tune BERT layers with a lower learning rate of 3e-5, and a fixed weight decay of 0.001, and to optimize all other custom layers on top with a conventional learning rate of 1e-3 without no decay. A single layer, bidirectional configuration for LSTM layers is also applied with a hidden size of 128.

We use dataset from Internet to train the BERT system for sentiment detection.

Figure 8 shows the train loss and test loss during 10 epochs. Figure 9 shows the train accuracy and test accuracy.

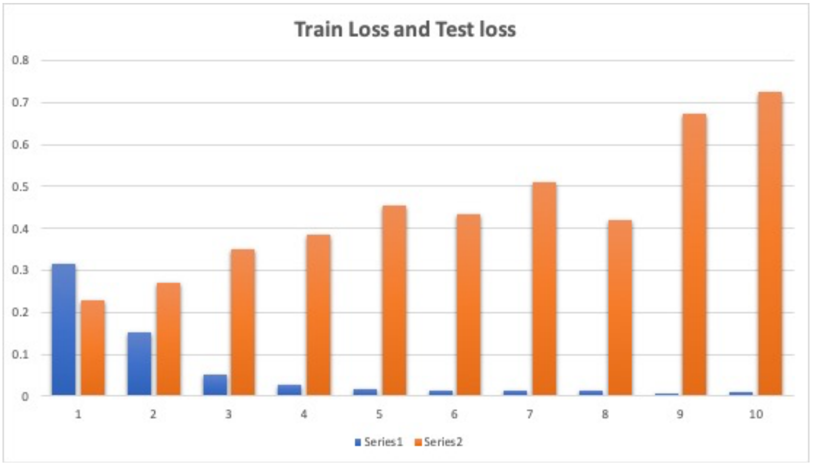


Figure 8. Train loss and test loss of the sentiment detection in different epoch.

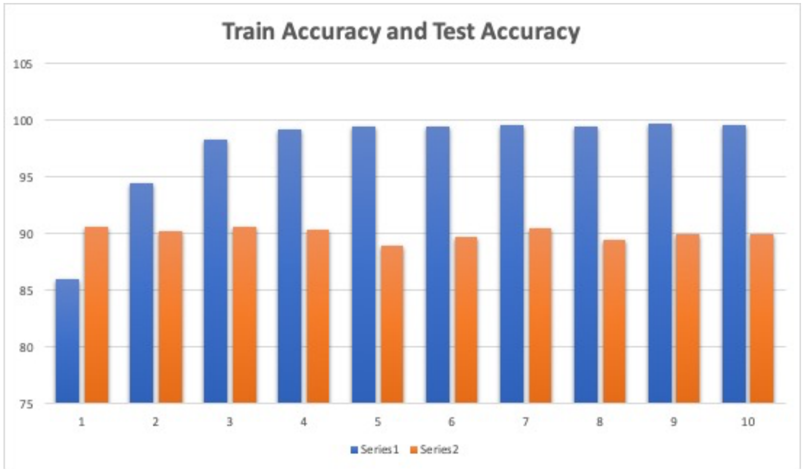


Figure 9. Train accuracy and test accuracy of the sentiment detection in different epoch.

Member: Mo Han(mxh190001), Chaoran Li(cxl190012)

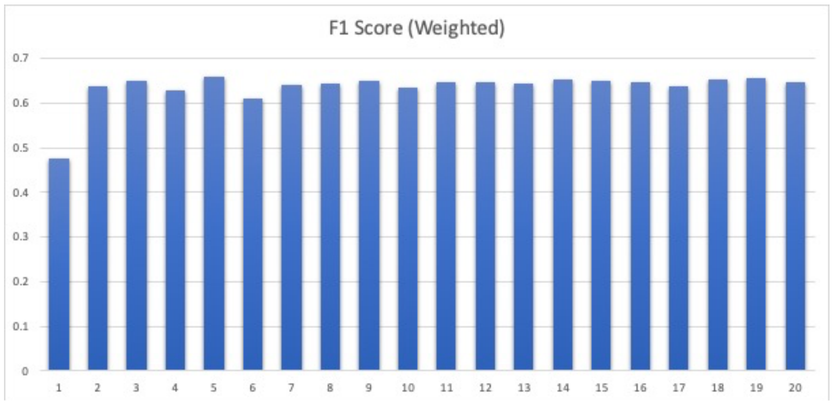


Figure 10. F1 score in the training process in different epoch.

Then we use the fine-tuned sentiment neural network to work with the stance detection system. When we get the input file of tweets with information of tweet_id and text. We can use the sentiment detection system to get the sentiment label of the tweets, that is positive(1) or negative(0) according to the text of tweets. Then we combine the sentiment id with the text of tweets as the input for the stance detection system. After training the system, we can use it to predict the stance of new tweets. Figure 10 shows the F1 score in different epoch. Figure 11 shows the details of accuracy of validation. We find that the accuracy is a little higher compared with the stance system without sentiment label.

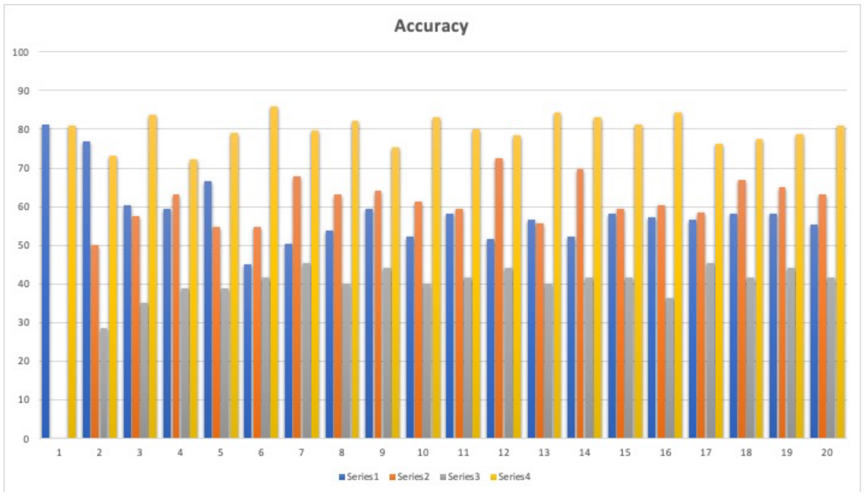


Figure 11. Accuracy of different stances in validation process in different epoch (Series1 is agree, Series2 is disagree, Series3 is no_stance, and Series4 is not_relevant).

Member: Mo Han(mxh190001), Chaoran Li(cxl190012)

(3) Collaboration with the student X1.

Chaoran is responsible for the stance detection. We have about 1600 pairs of tweets and misinformation after Phase A. Chaoran and I discussed a lot about how to combine the sentiment part with stance. Finally, we decided to add the sentiment label as one of the tokens of the input for stance detection system. Then I tried to find a fine-tuned sentiment detection system and modified the codes to get output files with suitable format. After that, Chaoran tried to design the neural network and training it with the added sentiment label. Then I tried to transfer the sentiment label to the fine-tuned new model and predict the stance of tweet. Finally, we find there is some improvement after adding the sentiment label in the stance detection system.

Member: Yinglu Huang(yxh180064), Chaoran Li(cxl190012)

Stance Detection with Emotion Detection

Generally speaking, emotion detection is the process of identifying human emotion. It aims to detect and recognize feelings such as anger, happiness, sadness, disgust, fear and so on. In this project, we need to generate an effect processing system operating on Twitter postings, which can be used to detect misinformation targets related to COVID-19 vaccines. And in this part, the emotion detection can be concatenated with stance detection to get a possible improvement of accuracy.

1) A Figure of the neural architecture for emotion recognition and equations, input/output explanation. How emotion recognition was tuned on the training set.

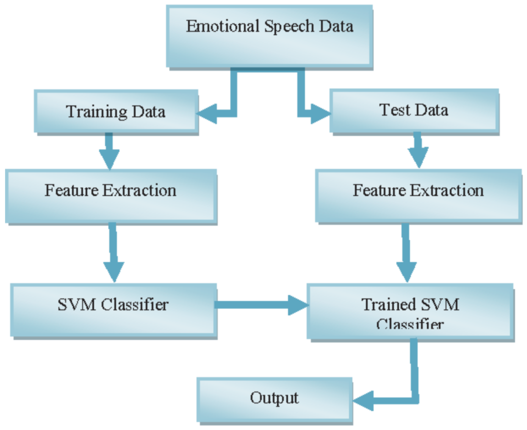


Figure 12: Overview of Emotion-Stance Architecture

- Equations

```
lsvm = SGDClassifier(alpha=0.001, random_state=5, max_iter=15, tol=None)
lsvm.fit(X_train_count, y_train)
y_pred = lsvm.predict(X_val_count)
```

- input/output

input: tweets.jsonl file which include “tweet_id” and “tweet_text”.

output: emotion_output.jsonl file which includes “tweet_id”, “emotion_result” and “emotion number”.

- How emotion recognition was tuned on the training set

I use a dataset which has 40,000 tweets, labeled into different human emotions. And my emotion detection system aims at accurately identifying new tweets’ emotions.

Member: Yinglu Huang(yxh180064), Chaoran Li(cxl190012)

First I convert all these textual data into mathematical data by making words lowercase, removing punctuation and stop words. And I also remove all rarely appearing words.

```
# Making all letters lowercase
data['content'] = data['content'].apply(lambda x: " ".join(x.lower() for x in x.split()))
# Removing Punctuation, Symbols
data['content'] = data['content'].str.replace('[^\w\s]',' ')
# Removing Stop Words using NLTK
stop = stopwords.words('english')
data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split() if x not
in stop))
freq = list(freq.index)
data['content'] = data['content'].apply(lambda x: " ".join(x for x in x.split() if x not
in freq))
```

After that, we need to perform feature extraction, and we consider two features, TF-IDF and count vectors. But before that, we need to split the data into training and testing parts before feature extraction.
Then we can train our models and test it after that.

2) A Figure for its incorporation in the stance detection system and equations, input/output explanation. How emotion recognition was tuned on the training set.

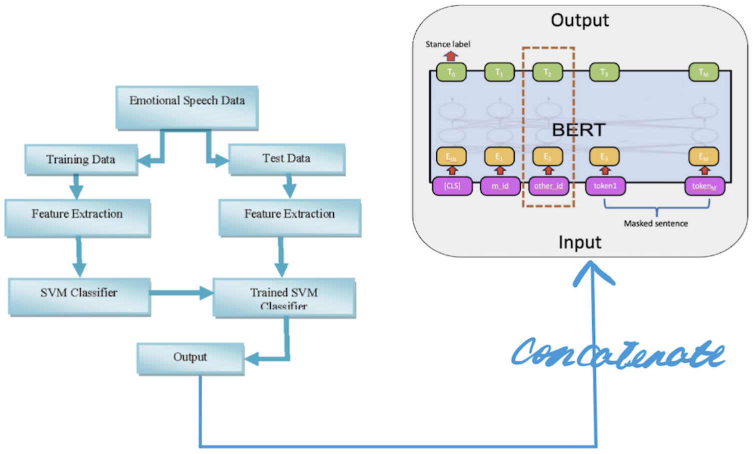


Figure 13: Combined Architecture

- Equations

```
predictions = np.concatenate(predictions, axis=0)
return np.argmax(predictions, axis=1).flatten()
```

- input/output

input: tweets.jsonl file which include “tweet_id”, “tweet_text” and “misinformation_id”.
output: emo_stance_output.jsonl file which include “tweet_id”, “misinformation_id”, “emotion_result” and “stance_label”

- How emotion recognition was tuned on the training set

Member: Yinglu Huang(yxh180064), Chaoran Li(cxl190012)

As we concatenate the emotion detection system and the stance detection system, the data treatment is similar.

3) How the stance detection using emotion recognition was trained and what results it obtained on the development set (10 points).

We got an embedding result from the stance system, then our emotion detection also gave an embedding result. And they can be concatenated together to convert into four class classification results, which could improve recognition ability further.
The result should be the combination of stance label and emotion recognition.

4) Elaborate on how you have collaborated with the student responsible for the stance detection for incorporating emotion detection (5 points).

In our team, Chaoran is responsible for the stance detection. After completing phase A, our team had 1000 train tweets. At the beginning, Chaoran and I were struggling for our own. After we built our own systems, and got the emotion_result and stance_result respectively, we started to cooperate with each other. Firstly, Chaoran used all of our other analysis results to make predictions of our 1000 train tweets, and get different accuracies. We chose the highest one and improved it, which is prepared to be used for the final test. And then, we nearly had meetings everyday. Chaoran explained his system to us, and for the emotion detection part, Chaoran and I discussed the possible ways to concatenate stance and emotion. Finally we reached an agreement, Chaoran gave me an embedding result, then my emotion detection also gave an embedding result. And I can concatenate them to convert into four class classification results, which could improve recognition ability further.

Member: Yibo Cheng(yxc190039), Chaoran Li(cxl190012)

Stance Detection with Topic Model

This part mainly focuses on exploring effects brought by incorporating topic extraction into stance detection system to see if there is any improvement on prediction accuracy.
As the situation of lacking annotated dataset for topic extraction on current training tweets, two methods are experimented, k-means clustering and LDA (Latent Dirichlet Allocation), to generate dataset initially. Then a fine-tuned BERT model is introduced to make topic prediction over testing tweets. Once results on testing tweets are predicted and written to the file, it is further used in the stance detection model where it will be merged with stance training files according to its tweet's id. Each tweet's topic along with tweet's text is embedded together to form a new input for the stance model. After training model and testing on the testing set, it suggests that the stance detection with enrichment of topic extraction is able to improve the accuracy of stance prediction over the testing set.
Below is a series of architecture used in this part, including topic extraction and stance detection.

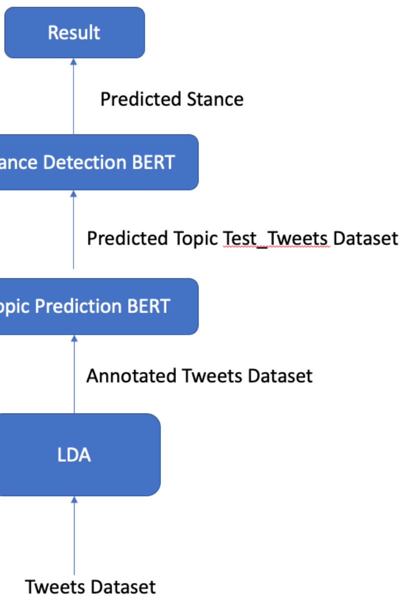
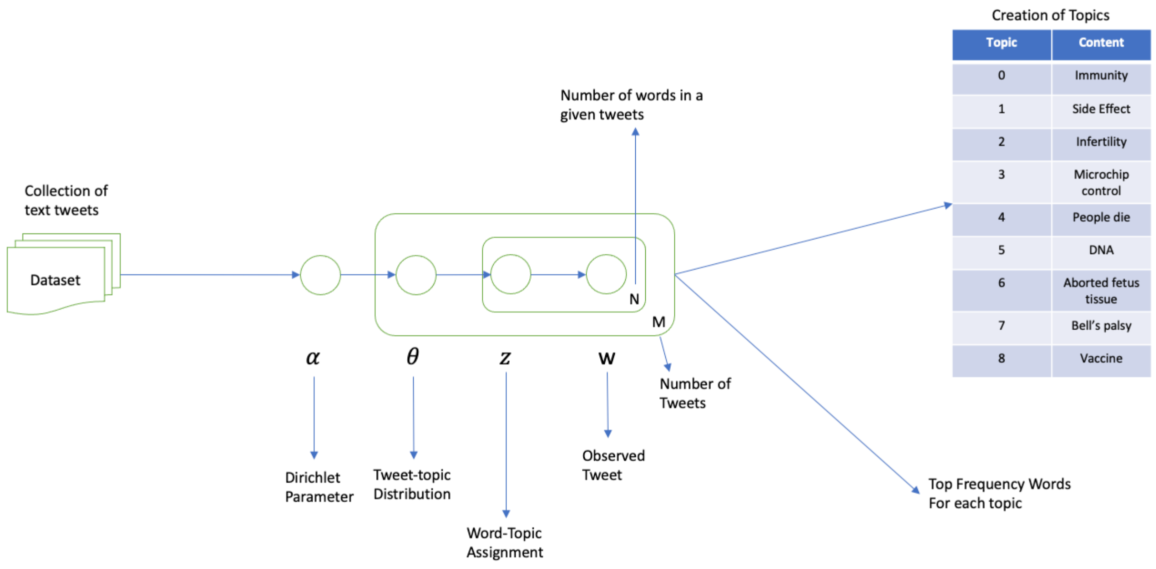


Figure 14: Overview of Topic-Stance Architecture

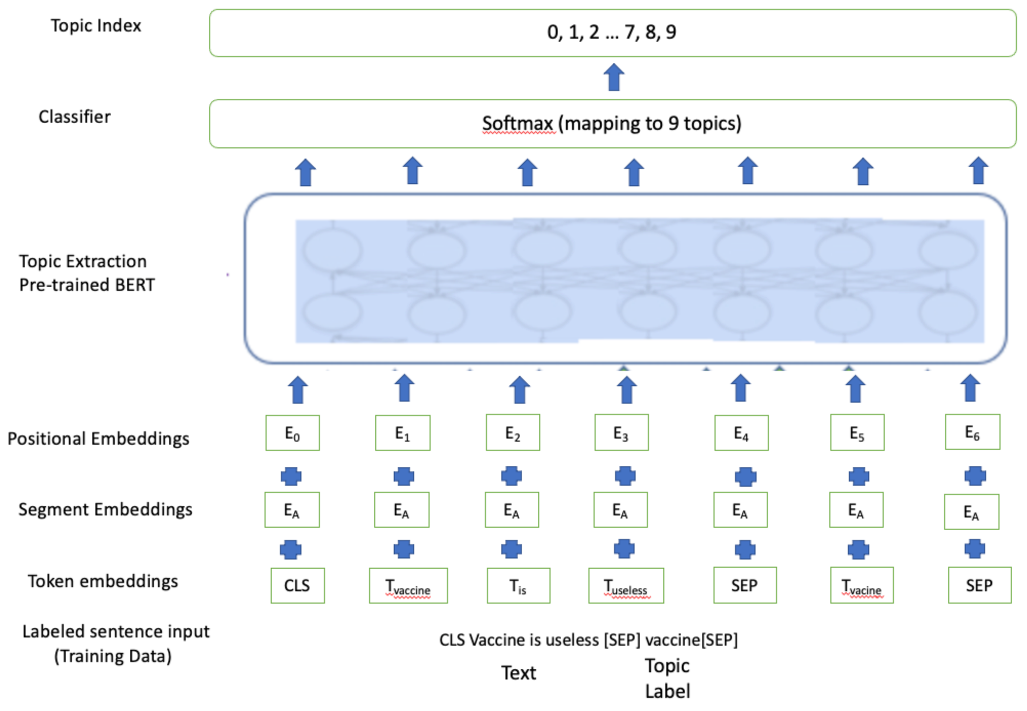
Above is an overview of topic-stance architecture. In terms of input and output. Starting from bottom, it is raw tweets information including tweet_id and text. After LDA procedure, each raw tweet is labeled a topic which is interpreted by top 8 words from each topic. Annotated tweets dataset is fed into Topic Prediction BERT model, where annotated tweets dataset is divided into training and testing sets. At the end of Topic Prediction BERT model, tweets that are supposed to be predicted will be mapped to a topic through softmax where there are 9 topics. This way, rather than using LDA to learn, it utilizes BERT neural net to predict. To incorporate stance-detection model finished in the part a, topic learned previously will be concatenated with tweet text and then be tokenized by the pre-trained stance-detection BERT model. In this way, it will be performed as add one extra feature (topic) in the stance detection model. More details on LDA, topic extraction BERT model and enriched stance detection BERT model are presented below.



Topic	Content
0	Immunity
1	Side Effect
2	Infertility
3	Microchip control
4	People die
5	DNA
6	Aborted fetus tissue
7	Bell's palsy
8	Vaccine

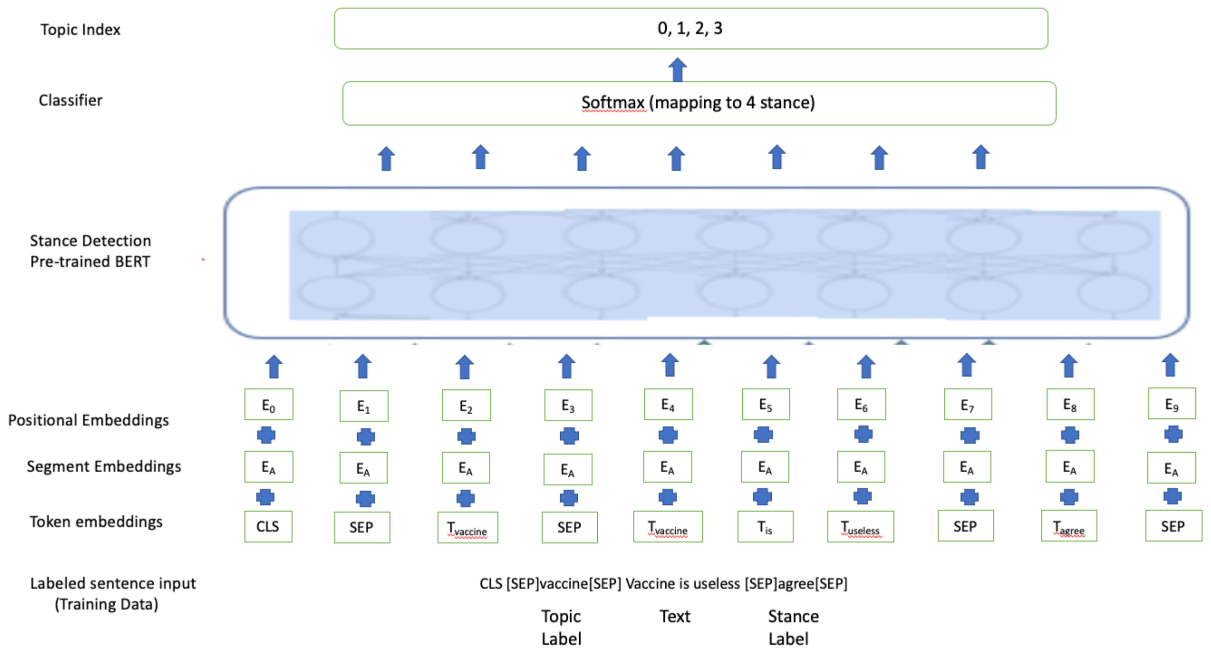
Figure 15: LDA Model Architecture

Member: Yibo Cheng(yxc190039), Chaoran Li(cxl190012)



Topic Index
0, 1, 2 ... 7, 8, 9
Classifier
Softmax (mapping to 9 topics)
Topic Extraction Pre-trained BERT
Positional Embeddings
E0 E1 E2 E3 E4 E5 E6
Segment Embeddings
EA EA EA EA EA EA EA
Token embeddings
CLS Tvaccine Tis Tuseless SEP Tvacine SEP
Labeled sentence input (Training Data)
CLS Vaccine is useless [SEP] vaccine[SEP]
Text
Topic Label

Figure 16: Topic Extraction BERT Model Architecture



Topic Index
0, 1, 2, 3
Classifier
Softmax (mapping to 4 stance)
Stance Detection Pre-trained BERT
Positional Embeddings
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9
Segment Embeddings
EA EA EA EA EA EA EA EA EA EA
Token embeddings
CLS SEP Tvaccine SEP Tvaccine Tis Tuseless SEP Tagree SEP
Labeled sentence input (Training Data)
CLS [SEP]vaccine[SEP] Vaccine is useless [SEP]agree[SEP]
Topic Label
Text
Stance Label

Figure 17: Stance Detection Bert Model with Topic Architecture

Member: Yibo Cheng(yxc190039), Chaoran Li(cxl190012)

Like mentioned above, the process of learning which topic on the training set is mapped into each topic is done firstly in the LDA model. LDA is allowed user to tune number of topics it will be clustered and output the one with highest probability. Meanwhile, top words from each topic can be retrieved and left to the user who will decide the descriptive topic content for each topic. As several trials, it suggests that 9 topics is informative and words in each topic is meaningful. This is how topic labels are assigned to each tweet in the training set. Then this training set is able to be trained and learned in the topic extraction BERT model since now input data has topic labels. The way to map topic label to each tweet is finished by softmax activation function which will output probability over entire topic labels. Then, using function np.argmax() to transform probability into an topic id which represents a label of topic extraction.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where:

$$\sigma = softmax$$
$$\vec{z} = input\ vector\ (output\ of\ BERT\ model)$$
$$e^{z_i} = standard\ exponential\ function\ for\ input\ vector$$
$$K = number\ of\ classes\ in\ the\ multi-class\ classifier\ (9\ topics)$$
$$z_j = standard\ exponential\ function\ for\ output\ vector$$

The way to collaborate with student 01 is to first determine how two neural architectures can be incorporated together. It is decided that results from topic extraction BERT model could be concatenated with input data for stance detection model. Then its concatenated result will be further tokenized and embedded and create a new input dataset. It turns out that this way, combining topic extraction model and stance detection model, can improve the overall accuracy and has better performance on prediction.

Conclusion

In a nutshell, the team is able to deliver the project on time and achieve a high score in the accuracy competition using emotion detection and stance detection model. Each team member plays an important role and is collective in the course of collaboration. The final accuracy is 68.5% in validation data. However, there are still some improvements the team can work on. For example, the team could explore a new pre-trained BERT model and test to see if different models could affect training accuracy. Also, pre-cleaning process on tweets could be more systematic and comprehensive to lower errors and noise in texts.