X1: Stance Detection (cxl190012)

a) A figure of the neural architecture:

Diagram

Description automatically generated

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:

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:

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:

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

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:

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated Chart, line chart

Description automatically generated

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

[[-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?

Diagram

Description automatically generated

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, 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.