NLP Final Project - Fact Check
Group 1

1. Briefly explain your choice to do evidence sentence extraction and compare the results of choosing different parameters(if has) in your method.(15pt.)

    To do evidence sentence extraction, we extract the content from the JSON files of the corresponding premise articles given the claims. There are multiple premise articles for each claim. To extract the information from the article, we implement a tokenizer ourselves. We tokenize the content and the claim by removing the stopwords and then find the lines in the content that have most intersections with the claim. For each claim, we take at most 50 extracted lines, which were sorted by the intersection length since we assume that the sentence with more intersection has more importance. After that, the extracted lines were written into a CSV file.

    In order to make the process of reading input files easier during the training and testing phase, we merge all the extracted files into three CSV files: train.csv, valid.csv, and test.csv. The train.csv and valid.csv include column names: id, claim, text and label, while the test.csv include column names: id, claim, and text. The special characters in the extracted lines are deleted. The extracted lines of each file are then merged into a paragraph, removing characters which are not numbers or alphabets, with each sentence separated by the [SEP] token. Finally, the three CSV file is generated.

2. Describe how you implement your sequence model, including your choice of packages, model architectures, loss functions, hyperparameters (learning rate, epochs, etc.)etc.(15pt.)

    Since the task aims to classify an article with a claim, it fits perfectly with the AutoModelForSequenceClassification from HuggingFace. We use the trainer API from HuggingFace and set the hyperparameters in the TrainingArgument class. We set batch size to 12, learning rate to 3e-5, warm up steps to 150 and 10000 on logging steps and save steps. The metrics we used is the macro f1, which is the same as the metrics evaluated on Kaggle. After the training is finished, we store the last model and get the predictions by calling trainer.predict(). Finally, write them to a CSV file.

3. All of the methods you have tried, and compare with your best method.(10pt.)

    At first, we substitute the special symbols, like comma, dot or other punctuations to a space character, which creates a blank space between the same word. However, these spaces make the words lose their meaning. Thus, we delete the special symbols directly. Then, we found out that the data is unbalanced. There are 7000 data for type 1 and 2 while there are only 2894 data for type 3. In order to balance the data, we doubled the number of data from type 3. With the changes above and changing the larger batch size, our score on Kaggle improved by 0.06687.

4. Do error analysis. You can use a confusion matrix to illustrate the whole model performance, or try to analyze the dataset's problem like unbalanced number of labels could lead what difficulty on training and how to overcome it. Share anything you observe. (10pt.)

    We can observe that the number of each class is unbalanced. Namely, class 2 is far fewer than either class 0 or class 1, which leads to make the performance on class 2 worse than others. We can check by the confusion matrix below. Also, As another result, the ratio is relatively lower.

Confusion Matrix

	0	1	2
0	0.26525424	0.11694915	0.04152542
1	0.11101695	0.23008475	0.08262712
2	0.02288136	0.05762712	0.07203390

5. Reproduce the results
    - Environment Setting:

        Install the HuggingFace module by entering `pip install transformers` in the terminal.

    - To reproduce the training part:

First, put the data provided by TA in the directory “extract” and store them under your working directory. Second, run compose.py by `python3 compose.py` to preprocess the data. The results will be stored under your working directory.

Then, use the command `python3 train.py` to start the training. After the training is finished, the model will be stored in the “model” directory under your working directory.

- To reproduce the result

  First, run the training process and the model will be stored under your working directory. Then, run the test by entering `python3 test.py` in your terminal.

  Finally, the predict results will be stored under your working directory as “output.csv”

6. Others

- Dataset path: ./extract/
- Model path: ./model