Jingsong Chen, jc3497

Link to my GitHub repository: https://github.com/Jingsong-Chen/Yoav-Lab-Application

(It includes a jupyter notebook file and python code)

The task seems to be building a discriminator for a Generative Adversarial Network (GAN). I approached as a simple binary classification problem.

Step 1 – Data Pre-processing:

Each translation block ends with an empty line. I used this feature to get 584 blocks in the training set and 174 blocks in the testing set. Each translation block has 5 lines:

- line 1: source (Chinese)
- line 2: translation1 (reference)
- line 3: translation2 (machine or human)
- line 4: quality (float)
- line 5: label (human/machine)

For simplicity, I only used the third line and the fifth line of each translation block. I encode the label 'H' as 0 and 'M' as 1. There are 312 human translations (not including the references) and 272 machine translations in the training set. There are 92 human translations and 82 machine translations in the testing set. So, the data seems to be pretty balanced.

Step 2 – Embedding:

I transformed each sentence in to a 768-dimentional vector using BERT, Google's pre-trained model.

Step 3 – Build a discriminator & evaluation:

I built a two-layer neural network with Keras library with a drop rate of 0.3 and a learning rate of 0.001. I trained the network with a batch size of 32 and 100 epochs (more details are included in my code). It took several hours for it to finish the training. However, the result was not satisfactory – the model always outputs 1 as prediction on the training set. Since the training set was balanced, and the history obviously didn't seem like the model was always predicting 1, I think the reason was probably due to the instability of Google Colab – it got disconnected after the training was done, and I had to load the trained model from my Google drive. But I ran out of time to repeat. To remedy the situation, I tried out two simple sklearn solutions.

The first thing I tried was a MLPClassifier with two hidden layers, each with 5 nodes. It got an f1 score of 0.97 in identifying human translations but only 0.15 in identifying machine translations.

I tried again with an SVM. Surprisingly, it achieved an f1 score of 0.828 for human translations and 0.811 for machine translations. The average f1 score is 0.820 which is pretty good given the cap is 1.

* The most-frequent class strategy won't work for the average f1 score because the f1 score for non-most-frequent class will be 0. Let's say there are a lot of human translations, so we only predict human. Then the f1 score for the machine translation class will be 0 as there's no true positive for machine identification, and the numerator of f1 score is true positive. This way, the average f1 score will be no greater than 0.5, which is not good.