NLP HW3 Report

## Environment

| Running environment | Colab |
| --- | --- |
| Python version | Colab |
| GPU(s) used | T4 GPU |

## Discussions

**Which (pre-trained) model do you use? Why choose the model? (5%)**

I used the **google-bert/bert-base-uncased** pre-trained model from Hugging Face's Transformers library. And here are the reasons why I chose this model:

* According to my research, this model performs well, has high generalization and is good for multi-task learning as it outputs high-dimensional embeddings
* Most importantly, the model is not too large, making it efficient for training and inference
* And as the name, “uncased” suggests, tokenization ignores case sensitivity, which can reduce noise
* Additionally, this model performs well in many benchmarks

**Compared with models trained separately on each of the sub-task, does multi-output learning improve the performance? (8%)**

Yes and no, multi-output learning has its advantages and disadvantages.

**Advantages**:

* Efficiency: as it is much more efficient to train one models for multiple tasks than to train separate models for each tasks
* If tasks share some features, they could help the model learn better. For example, understanding relationships (classification task) may help in better scoring relatedness (regression task)
* Learning multiple tasks simultaneously acts as a regularizer, helping to reduce overfitting on individual tasks

**Disadvantages**:

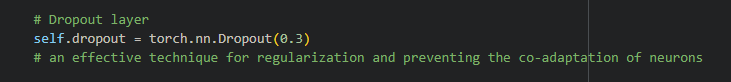
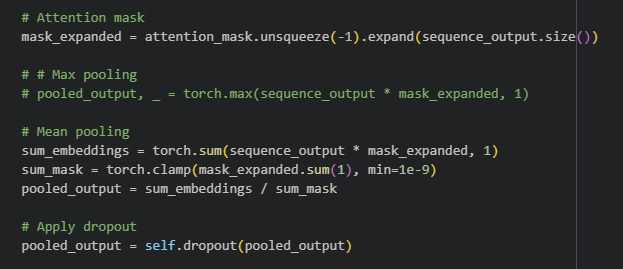
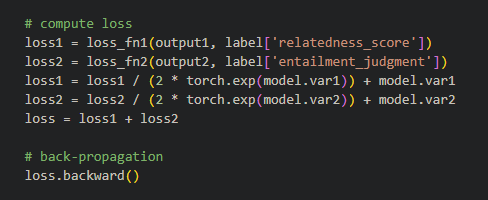
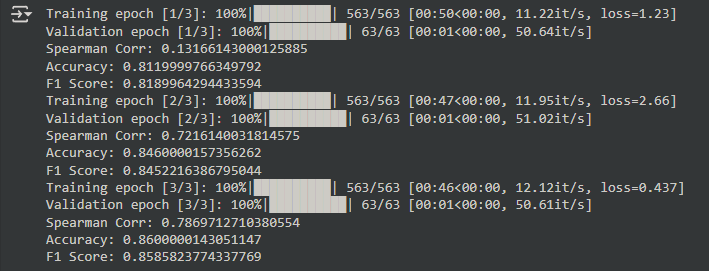
* On the other hand, if tasks are not closely related, learning one task may interfere with the other

**Why does your model fail to correctly predict some data points? Please provide an error analysis. (8%)**

**Possible error causes**:

* Upon evaluating my model, it seems it didn’t perform well in some neutral vs. entailment cases. Seems like the model didn’t learn their differences well enough or there may be unclear/ambiguous labels in the dataset
* Perhaps the dataset dominates in one label (e.g. neutral), which causes the model to struggle to learn other labels
* Maybe didn’t tune to model to meet these specific tasks

**How do you improve your model performance? (9%)**

* First, I noticed that my original model converges quickly and seems to overfit. Hence, I added a **dropout layer** (searched on pytorch doc) in my model. This dropout layer sets some neuron activations to zero (probability=0.3 in my case works best) during training. This can effectively reduce model overfitting.  
  Model construction:  
  
* And for similar reasons, I implemented a **pooling** step (searched online) in the forward function, in this case, mean pooling. I did this to help retain the most important information while discarding less important details.   
  Forward pass function:  
  
* Moreover, when training, I noticed that the loss seems very inconsistent. Perhaps different tasks may have losses on different scales (e.g., MSE might dominate over cross-entropy if not balanced). So I searched online for a method to **compute loss with dynamic weight**, which helps multi-task learning to balance the losses of different tasks dynamically.  
  During model training:  
  
* Here is the result after applying the methods mentioned above, and after playing around with some parameters:  
  

## 

## Anything that can strengthen your report (10%)

**Insights**

* As mentioned in the “Error Analysis”, the model struggles most with the "neutral" class, likely due to overlap with "entailment" examples. Adding more training data for "neutral" could probably solve this issue.
* As mentioned above, I added a dropout layer for regularization. And playing around with the probability, I noticed that the model performs best when dropout rates are within 0.1~0.5, and increasing from 0.1 to 0.3 reduced overfitting and improved validation performance by 3%
* Replacing [CLS] pooling with mean pooling: Regression Spearman correlation increased by 2%.
* Without dynamic weighting: Accuracy decreased by 5%, and regression loss became unstable.

**Future Improvements**

* Data Augmentation: Add paraphrases or adversarial examples
* Task-Specific Layers: Use task-specific transformer layers to enhance task specialization.
* Better loss function for multi-label learning

## References

[BERT](https://huggingface.co/docs/transformers/model_doc/bert)

[Multi-label classification for beginners with codes](https://medium.com/data-science-in-your-pocket/multi-label-classification-for-beginners-with-codes-6b098cc76f99)

[Dropout — PyTorch 2.5 documentation](https://pytorch.org/docs/stable/generated/torch.nn.Dropout.html)

[【NLP】BERT的五种Pooling方法-CSDN博客](https://blog.csdn.net/fengdu78/article/details/128059894)

ChatGPT was used to help demonstrate the use of pytorch library and python syntax