COMPSCI591NR/691NR Neural Networks: Neuroscience & Engineering **Spring 2023**

Final Project Milestone Report

Collaborators: *Quoc Anh Bui, Anshumaan Chauhan* **Github**

1 Technical Progress

1.1 Domain-Specific Model Training

We have successfully loaded and processed the training data (metadata of the All Beauty Products in Amazon Product Review dataset from the UCSD repository) for our Domain-Specific model. BERT Models can be pretrained either using Masked Language Modelling (MLM) or Next Sentence Prediction (NSP) approach. In the project we are using the MLM approach, therefore we used `BertForMaskedLM` - a pretrained model, which is trained under self-supervised manner to enhance the model's semantic and syntactical information on the domain of the downstream task that we are interested later on. Weight Decaying Adam optimizer (AdamW) was used for the task of gradient updates during training - was done only for 10 epochs (Figure 1), because Large Language Models (LLMs) are susceptible to overfit when trained for large number of epochs.

```
Epoch 0: 100%|██████████| 1823/1823 [28:55<00:00,  1.05it/s, loss=0.11]
Epoch 1: 100%|██████████| 1823/1823 [28:49<00:00,  1.05it/s, loss=0.0825]
Epoch 2: 100%|██████████| 1823/1823 [28:51<00:00,  1.05it/s, loss=0.0656]
Epoch 3: 100%|██████████| 1823/1823 [28:50<00:00,  1.05it/s, loss=0.0594]
```

Figure 1: Training of the Domain-Specific Model

1.2 Classification Model Training

Using the Domain-Specific Model that we trained from the previous section, we are able to fine-tune it into "Matching Chemical Ingredients and Cosmetic Products" classification problem. The original model has the training objective for MLM and therefore, additional classification layer is added on top of the BERT Domain-Specific model to make it suitable for classification. The model is then trained in a supervised approach on the aggregated version of cosmetic.csv and sephora_website_dataset.csv. The resulting dataset had 10K input-output pairs, which were later split into train (9000 pairs) and test datasets (1500 pairs). The model was trained using the CrossEntropy objective as the loss function on the train dataset for 5 epochs (Figure 2).

```
Epoch 0: 100%|██████████| 1125/1125 [04:28<00:00,  4.19it/s, loss=2.36]
Epoch 1: 100%|██████████| 1125/1125 [04:27<00:00,  4.21it/s, loss=1.54]
Epoch 2: 100%|██████████| 1125/1125 [04:26<00:00,  4.22it/s, loss=2.6]
Epoch 3: 100%|██████████| 1125/1125 [04:26<00:00,  4.22it/s, loss=2.24]
Epoch 4: 100%|██████████| 1125/1125 [04:26<00:00,  4.22it/s, loss=2.72]
```

Figure 2: Training of the Task-Specific Model

2 Intermediate / Preliminary Results

The model achieves 0.12% accuracy in the classification problem.

3 Discussion

We are currently seeing a very low accuracy in the performance of the model. Reasons for the low performance are as follows:

1. **Less number of training epochs**: We have trained the model on the downstream task for only 5 epochs, considering the huge number of parameters (110M parameters) of the BERT model it was unable to learn the mapping in such few epochs.
2. **Few data samples**: The train dataset contains only 9K samples, and there are a total of 149 label classes - on an average making 60 entries per label. We cannot increase the dataset size, but one possible approach to solve this issue is later discussed.
3. **Sub-optimal architectural change**: BertForMaskedLM model has a decoder layer at the end - output is the prediction scores of the language modeling head, on top of which we added the classification layer. However, this is not an appropriate approach since we are only interested in the hidden embeddings of the input. Decoder layer output is of dimension [batch size, max length, vocabulary size] (Figure 3) - this is quite large considering we are using only a single classification layer to learn mapping.

```
    (cls): BertOnlyMLMHead(
      (predictions): BertLMPredictionHead(
        (transform): BertPredictionHeadTransform(
          (dense): Linear(in_features=768, out_features=768, bias=True)
          (transform_act_fn): GELUActivation()
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
        )
        (decoder): Linear(in_features=768, out_features=30522, bias=True)
      )
    )
  )
  (classifier): Linear(in_features=30522, out_features=149, bias=True)
)
```

Figure 3: Architecture of the Task-Specific Model

Considering the above limitations and drawbacks, we are now implementing the following changes to overcome them.

1. **More training epochs**: We will be training the Task Specific model for more number of epochs so that proper mapping is learned.
2. **Merging of Labels**: There are several redundant labels that can be merged into one single category, by doing this the model will have more samples for each label to learn from.
3. **Model Architecture**: Updates in the architecture we will be removing the decoder layer from the architecture during task specific training. Furthermore, the classification part would now be consisted of a few Dense and Dropout (for regularization) layers, and will be added after the Encoder.