CSCI 5541 - Homework 1

Avinash Akella, Ryan Devera, Yash Travadi, Hung-Da Wen
University of Minnesota
Minneapolis, MN 55414, USA

1 Introduction

For this homework, we chose to fine-tune an existing classifier, RoBERTa (Liu et al., 2019), using HuggingFace (Option 2). We chose the Question Answering task with the SQuAD 2.0 dataset (Rajpurkar et al., 2018). In the following, we will go over details of the pre-trained model, SQuAD 2.0 dataset description, model training and evaluation, and any areas for future investigation.

2 Task and Model

2.1 Task/Dataset Description

The SQuAD 2.0 dataset (Rajpurkar et al., 2018) consists of the following components for each observation: question, context, and answer. Question is usual one-sentence long and context typically consists of a passage of several sentences. Answer, on the other hand, is more complicated. Since we are posing Question Answering as a classification task, the answer(s) can only be from specific segments of the context. Since it is possible for multiple, somewhat equivalent answers to exist in the same context, we could potentially have more than one answer. Each answer contains two components: the exact strings and the starting/ending indices of the answer from the context. An added level of complexity of SQuAD 2.0, as compared with SQuAD 1.0 (Rajpurkar et al., 2016), is that unanswerable questions are now included. This means that not only does the model need to get the correct answer(s) from the context, it also has to identify situations where the context does not provide an answer.

2.2 Model Description

We used the RoBERTa (Liu et al., 2019) model for this particular task. This model is essentially a large BERT model (Devlin et al., 2018) with an optimized pre-training process. Liu et al. (2019) identified that the BERT model can be further optimized by training longer with bigger batches of data, using dynamic masking in sentences, and removing the next sentence prediction objective originally used for the BERT model. Empirically, they show these modifications help achieve state-of-the-art results on GLUE, RACE Lai et al. (2017) and SQuAD (Rajpurkar et al., 2016; 2018) benchmarks, without multi-task fine-tuning for GLUE (Wang et al., 2018) or additional data for SQuAD.

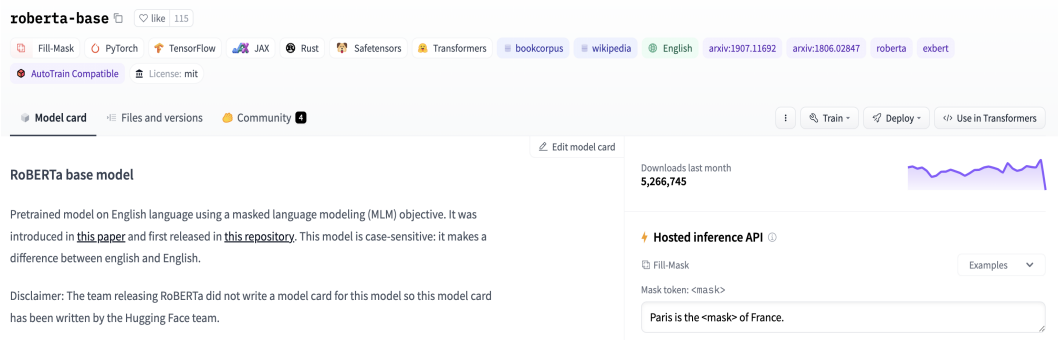


Figure 1: HuggingFace Roberta Base Model Card

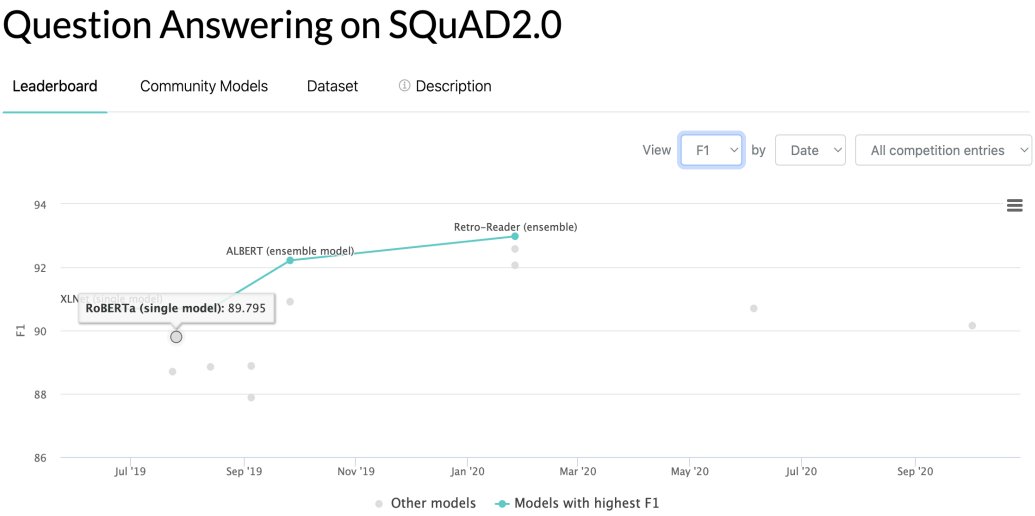


Figure 2: Roberta Model Performance on SQUAD-2.0

3 Model Training

3.1 Hardware

We're training our model on an NVIDIA GeForce RTX 2080 Ti GPU. This supposedly has 4352 CUDA cores and a clock speed of 1540 MHz.

3.2 Training and Inference Time

Overall, it takes us roughly 3 hours to run 3 training epochs of our model.

3.3 Hyperparameters

To extract features from the dataset we used the in-built `squad_convert_examples_to_features` method by setting `max_seq_length`=384, `doc_stride`=128, and `max_query_length`=64. For the model training process we used the Adam optimizer (Kingma and Ba, 2015) with a `learning_rate`=2e-5, and a `batch_size`=4 for both training and validation. We ran it for 3 epochs in total.

3.4 Learning Curve

The training loss plunges in the first few iterations and gradually reduces further, as shown in Figure 3. It seems the losses within each epoch quickly stabilize after the first few iterations. Losses can likely be further reduced from further increase in the number of epochs as the between-epoch drop in losses seem non-trivial.

4 Model Evaluation

4.1 Evaluation Metrics

The SQuAD dataset makes use of two different metrics to assess the performance of a model on the benchmark. The first one being an Exact Match metric, which measures the percentage of predictions that match exactly with any one of the ground truth answers. The second metric, the F1 score, is regarded as a looser metric that measures the average overlap between the prediction and the ground truth answer.

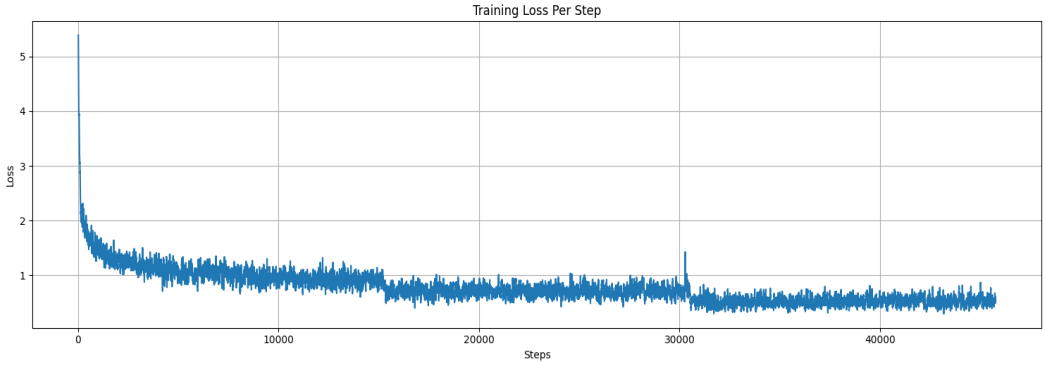


Figure 3: Training loss over multiple steps

4.2 Test set performance vs. Scores from Leadership Board

Epoch Number	Exact Match	F1	Number of Obs	Has An Exact Answer - Exact Match	Has An Exact Answer - F1	Number of Obs - Has Answer	No Exact Answer - Exact Match	No Exact Answer - F1	Number of Obs - No Answer
1	78.15	81.15	11873	70.18	76.18	5928	86.11	86.11	5945
2	78.72	82.04	11873	74.16	80.82	5928	83.26	83.26	5945
3	76.77	80.39	11873	77.67	84.92	5928	75.88	75.88	5945

Figure 4: Performance Per Epoch

Based on Figure 4, we see that the second epoch gives the best overall F1 score at 82.04. It also gives the best exact score, at 78.72. This is despite that the the first and third epoch give a better F1 score for no exact answer and exact answer, respectively. Like discussed earlier, if we were to train a few more epochs, we expect to see a more stabilized result.

On the leadership board, we see an exact match score and F1 score of 86.820 and 89.795, respectively. Given more computing resources and power, we think we can match the results from the leadership board to a better extent.

4.3 Hypothesis for Incorrect Prediction

Our model might struggle with certain types of samples more than others. First, if a question is not precisely phrased, or put in another way, if it is potentially subject to more than one interpretation, then our prediction might be off. Perhaps humans who label the answer(s) interpret the question slightly differently from the model due to nuances in the English language.

Another source of confusion for the model comes from questions corresponding to no answers. Even for humans, verifying that a passage does not contain a valid answer at all take more effort than to identify the correct answer, if present.

Lastly, the model might be prone to identify an answer from the section of the context that contains many of the same words from the question, which is also what humans tend to do. While this behavior does have an empirical basis, it does not work as well when it comes to passages written in more convoluted forms.

Examples of 10 incorrectly predicted samples can be found at `https://github.com/rydeveraumn/text-mining-titans/blob/main/results/incorrect-samples.csv`.

5 Conclusions and Future Work

This project allows us to get familiarized with using a pre-trained model from HuggingFace and fine-tuning it with the relevant dataset using a corresponding model tokenizer for the question answering task. While an ideal goal would be to replicate as closely as possible to the published evaluation metrics from the RoBERTa model on the leadership board, we acknoweldge that some result discrepancies between ours and published ones are unavoidable. This is in part due to the level of unseen data pre-processing, computational constraints, and model hyperparameter choices such as optimizer, scheduler, batch size...etc.

There are a few things we can do to further improve model performance. First, we can fine-tune the same RoBERTa model with other QA datasets similar to SQuAD. Second, additional pre-processing on SQuAD dataset, such as oversampling certain types of QAs, might yield better predictive power. Since the output of a model for question answering can come in so many forms, identifying a more comprehensive evaluation metrics might be helpful too. We leave these as future research topics to consider.

All the codes for this project can be found at `https://github.com/rydeveraumn/text-mining-titans`.

6 Contribution

Avinash Akella: Writing - Sections 2.2, 3.1-3.3, 4.1. Coding - helped with hate speech detection task and result analysis with the Roberta-base model.

Yash Travadi: Writing and coding for hate speech detection task(Details in Appendix A).

Hung-Da Wen: writing - Sections 1, 2.1, 3.4, 4.2, 4.3, 5. Coding - worked on a preliminary version of the hate speech task in parallel with the QA task.

Ryan Devera: Worked through the question and answering code, set up a team Github repository and also helped provide content for the final write up.

7 Resources

The resources used for the project are as follows:

- HuggingFace Tutorials
- Tutorials from class
- Tutorials on QA Systems: Building a QA System with BERT on Wikipedia, Evaluating QA: Metrics, Predictions, and the Null Response
- SQuAD 2.0 Data Description: Dataset - SQuAD2.0
- Leadership Board Ranking: Question Answering on SQuAD2.0
- Official implementation for hate-speech-detection: Github link

References

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th International AAAI Conference on Web and Social Media*, ICWSM '17, pages 512–515, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. RACE: large-scale reading comprehension dataset from examinations. *CoRR*, 2017.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, 2016.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, 2018.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, 2018.

A HATE SPEECH DETECTION TASK

As an alternative to the question-answering task, we also worked on the hate speech task in parallel. We used the `hate_speech_offesive` dataset (Davidson et al., 2017) on HuggingFace. The training strategy was also to fine-tune a pre-trained RoBERTa model. The task is to classify a given tweet into three classes: 'hate-speech', 'offensive-language', or 'neither'. The key challenge for the task is in form of an imbalance in the label distribution in the training data. The number of samples per class in the training set after separating a 10% hold-out evaluation set is 1291, 17296, and 3717 respectively. The lack of samples labeled as 'hate-speech' makes the training task more challenging. Initially, we tried approaching the problem similar to a general text classification problem, by using the standard cross-entropy loss for fine-tuning the RoBERTa model.

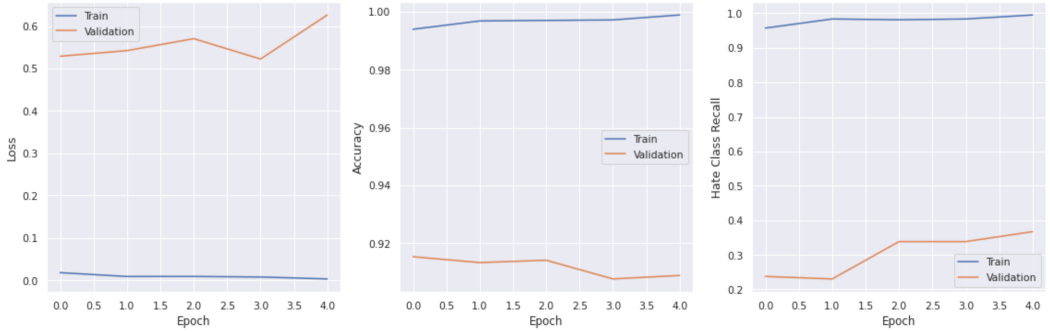


Figure 5: Evolution of loss, accuracy and recall for the 'hate-speech' class(minority) over training and evaluation sets while using standard cross-entropy loss

In Fig. 5, we notice that the training loss is decreasing to 0 and training accuracy is increasing to 1 over 5 epochs. However, performance on the validation set measured in terms of loss and accuracy values is getting worse. However, in an imbalanced learning setting, accuracy can be a misleading metric. We also track recall for the 'hate-speech' class over the training epochs and find that it is improving over training as well as validation sets. However, the final recall value on validation set is pretty poor. To address the imbalance in label distribution, we try to use weighted cross-entropy loss to assign more weights to the less frequent classes during the training process. In particular, we weighted each sample by the inverse of the corresponding class frequency in the training data.

In Fig. 6, we can observe that compared to standard cross-entropy loss, although the overall validation accuracy is worse, the minority class recall is much better. Since there is an inherent trade-off between precision and recall, we used F1 score measure to pick the final best model.

In Fig. 7, we compare the performance of our model to Davidson et al. (2017). We find that the results of our weighted cross-entropy(WCE) based model are closest to the benchmark, however

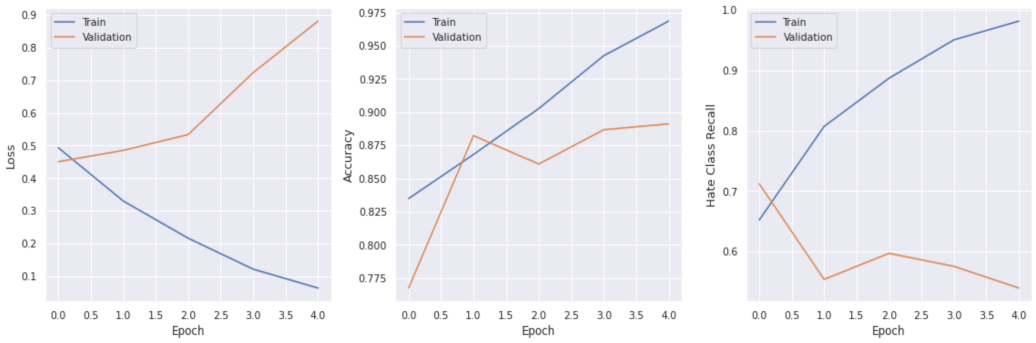
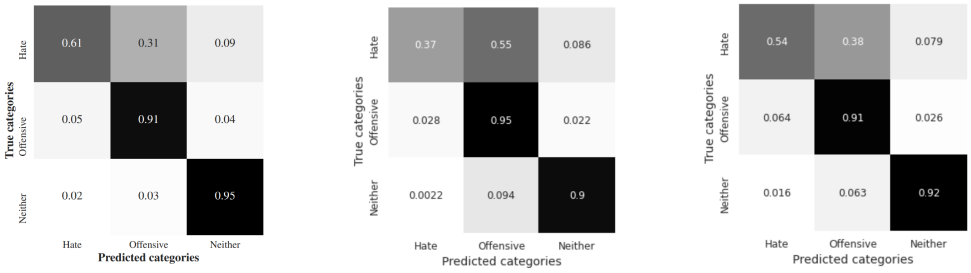


Figure 6: Evolution of loss, accuracy and recall for the 'hate-speech' class(minority) over training and evaluation sets while using weighted cross-entropy loss



(a) Davidson et al. (2017) (b) Our CE (c) Our WCE

Figure 7: Comparison of our results with Davidson et al. (2017) in terms of the confusion matrix

performance in terms of predicting the minority class 'Hate' is a bit worse for our method. Our standard cross-entropy(CE) model is able to perform a bit better on the majority class 'Offensive' by sacrificing the minority class performance, which is not very desirable in practice.