Zhuohan Yu NLP Individual Report

Introduction and My Work

Our project is working on glue dataset. We run XLNer, Bert, and Electra for all the tasks. Then, tuning parameters, modify electra, and do logistic regression.

I find and research the run_glue.py file, which allows us test different pretrained models and decide which one we want to work on. Then, since XLNet get higher score, I also read the XLNet and transformerXL papers to get some background knowledge about them. I also tried tuning the paremeters in pretrained models' config.json files. I trained some of them, but didn' t get better results and most of parameters cannot change due to contradiction between our setting and pretrained models.'

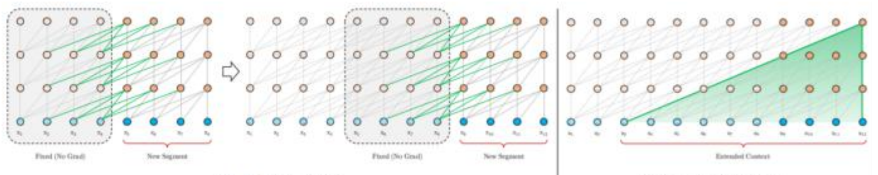
Besides, I helped my teammates to run some of models we tuned and give some suggestions about ensemble part. Finally, I tried to modify electra-base-discriminator but also didn't get better result.

XLnet

XLnet is another model we used. Although its architecture is similar to that of Bert, there are many novel techniques it uses to improve on performance..

TransformerXL

TransformerXL is another model that was published before XLnet which solved the issue of limited fixed-length context in the setting of language modeling by using the Segment-Level Recurrence with State Reuse.



(a) Training phase. (b) Evaluation phase.

Instead of learning dependency on a fixed length, segment-level recurrence allows the model to fix the previous segment into cache and reuse it as an extended context in the new segment. And, the recurrent mechanism is also faster to evaluate, because the representations from the previous segments can be reused instead of being computed from scratch.This is the basic idea of XLnet.

Permutation Language Modeling

The aim of using Permutation Language Modeling is to combine the advantages of BERT and AR language modeling as well as avoid their weaknesses. Because the model will only encounter text sequences with the natural order during fine tuning, permutation modeling will keep the natural sequence order and rely on a proper attention mask in Transformers to achieve permutation of the factorization order. This kind of modeling will gather information from all positions on both sides and maximize the likelihood of sequence.

Two-Stream Self-Attention

XLnet used two hidden representations to solve contradictions caused by using target-aware representation in standard Transformer architecture. First is the content representation which encodes both the context and token, and serves a similar role to the standard hidden states in Transformer. The second is the query representation, which only has access to the contextual information and the position but not the token itself.

Results

Throughout this project, we focused on using various methods to output the highest scores in all the GLUE tasks. As a result, our project can be split up into 3 sections: getting the benchmark values on the transformers we focused on, outputting the predictions for several transformers to use for further analysis and finally using ensemble techniques to maximize the metrics on all the glue tasks.

Getting a Benchmark

The benchmark we chose was the score of the bert_base_cased model that we got by running the run_glue.py file. BERT is the default pretrained model run_glue.py used, and it is one of the most widely-used and earliest pre-trained transformer-based language models. Since it has pre-trained for representation learning, BERT has a better understanding of the meaning of language within context and is able to predict results for various types of natural language processing tasks due to its framework.

To briefly explain the architecture as reasoning behind using it as a benchmark as well as in our final ensemble model, BERT uses MLM and NSP during pre-training. Therefore, it is able to understand the context of a sentence bidirectionally by predicting tokens in the pre-training corpus as well as predicting whether the context of the next sentence makes sense in reference to the previous sentence.

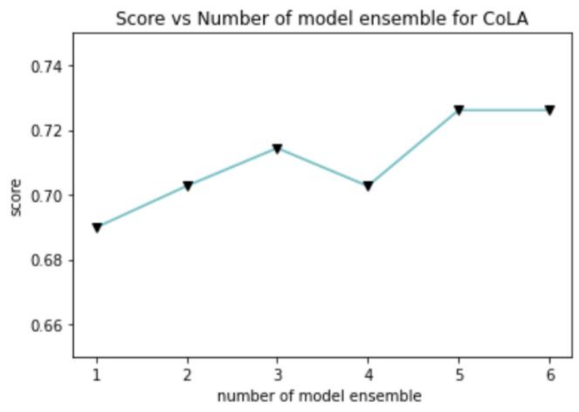
The table below shows our findings for all the GLUE tasks:

Task	**Metric**	**Bert-base-cased**	**Xlnet-base-cased**	**Electra-base-discriminator**
CoLA	Matthews corr	57.01	31.99	65.85
SST-2	Accuracy	92.43	94.15	95.41
MRPC	F1/Accuracy	89.46/84.8	89.76/85.29	90.88/86.76
STS-B	Pearson/Spearman corr.	88.82/88.5	86.73/86.56	90.31/90.35
QQP	Accuracy/F1	90.71/87.49	90.83/87.74	91.79/89.06
MNLI	Matched acc./Mismatched acc.	83.74/84.11	86.38/87.01	88.82/88.67
QNLI	Accuracy	90.32	92.02	92.77
RTE	Accuracy	64.98	64.98	72.56
WNLI	Accuracy	38.03	29.58	35.21

In order to find the best transformer that was applicable for all or most of the GLUE tasks, we tested using ELECTRA and several variations of BERT models including BERT, XLNet, roBERTa and DistilBERT as well as some of the size variations of each model (i.e. small, base, large). We found that electra-base-discriminator had a great performance on a majority of the tasks followed by bert-base-cased which was our benchmark and xlnet-base-cased, which performed slightly better than BERT. As a result, we used these 3 models as a foundation of what we built our ensemble model on.

Ensemble

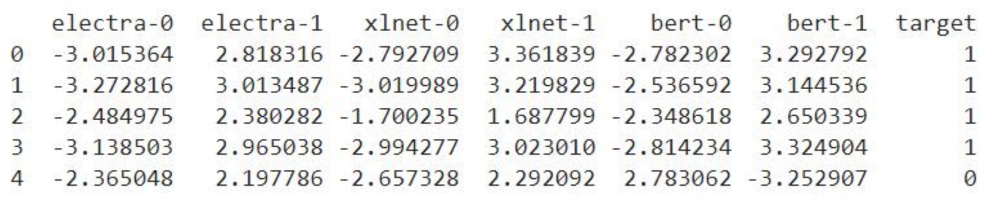
Before we began collecting the prediction data for our ensemble model, it was vital to understand how many models we use or we could risk getting even worse results than all the individual transformers. Therefore, we did an experiment on the first task, CoLA, to determine the accuracy scores of combining 1 through 6 models together for an ensemble. The results are shown in the graph below:



The transformer models we used for this graph were variations of ELECTRA, BERT and XLNet. In order to find the best results for this experiment, we decided to work backwards and start the ensemble at 6 models where the models that were repeated were the highest scoring models (ELECTRA and BERT). After that, we did several iterations of removing the lowest scoring model from the ensemble until there was one model left (ELECTRA).

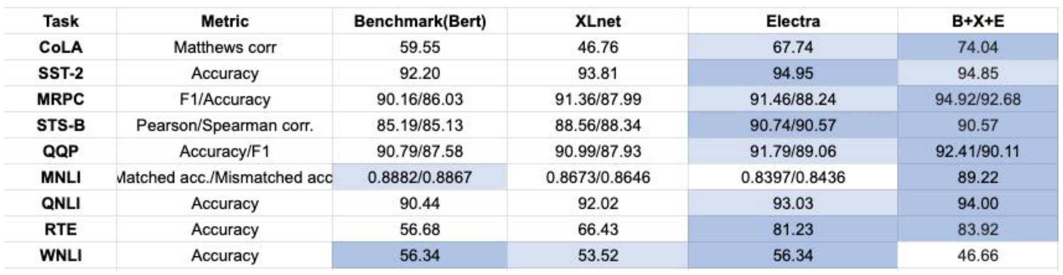
By doing this we were able to assure that the accuracy of the ensemble was reasonable and the limit to how many models we should use to collect prediction data for the GLUE tasks. We can see from the graph that when the number of models equals 3, there is a peak followed by a dip when the ensemble quantity is 4. The transition from 5 to 6 models in the ensemble was a massive increase, so we decided to stick with 3 models in our ensemble. To put into further context, since this experiment was only for 1 task, we decided to look at the individual scores of the other GLUE tasks and XLNet and BERT demonstrated 2nd highest scores in different tasks which explains why our final ensemble includes ELECTRA, BERT and XLNet.

In order to create a viable ensemble model, it was necessary to output the predictions of our models of interest to feed them into an ensemble classifier. These predictions were based on the validation set of each of the GLUE tasks. The figure below shows how we formatted these predictions into a new dataset. For each task, there is a target column indicating the actual target values in the validation set.


```
   electra-0  electra-1   xlnet-0   xlnet-1    bert-0    bert-1  target
0  -3.015364   2.818316 -2.792709  3.361839 -2.782302  3.292792       1
1  -3.272816   3.013487 -3.019989  3.219829 -2.536592  3.144536       1
2  -2.484975   2.380282 -1.700235  1.687799 -2.348618  2.650339       1
3  -3.138503   2.965038 -2.994277  3.023010 -2.814234  3.324904       1
4  -2.365048   2.197786 -2.657328  2.292092  2.783062 -3.252907       0
```

Due to the fact that most of the tasks contained binary or multi-class target variables, the ensemble classifier we used was Logistic Regression and Random Forest Classifier. After training and testing the predictions for each task on both these models, we found that the performance was extremely similar. STS-B was the only task that contained a continuous target variable, so we used Linear Regression as the basis of the ensemble model for that task. In order to test the performance of each ensemble, it was important we used the validation set as opposed to the test set.

The results of our ensemble model on each task is shown in the following table:



Task	Metric	Benchmark(Bert)	XLnet	Electra	B+X+E
CoLA	Matthews corr	59.55	46.76	67.74	74.04
SST-2	Accuracy	92.20	93.81	94.95	94.85
MRPC	F1/Accuracy	90.16/86.03	91.36/87.99	91.46/88.24	94.92/92.68
STS-B	Pearson/Spearman corr.	85.19/85.13	88.56/88.34	90.74/90.57	90.57
QQP	Accuracy/F1	90.79/87.58	90.99/87.93	91.79/89.06	92.41/90.11
MNLI	Matched acc./Mismatched acc	0.8882/0.8867	0.8673/0.8646	0.8397/0.8436	89.22
QNLI	Accuracy	90.44	92.02	93.03	94.00
RTE	Accuracy	56.68	66.43	81.23	83.92
WNLI	Accuracy	56.34	53.52	56.34	46.66

The results of our ensemble model were generally good in that none of the final metrics were below the lowest individual score of the 3 transformer models we used. Most of the final scores for the ensemble model were above or quite close to the highest scoring transformer model.

Summary and conclusions

Conclusion

Using the GLUE benchmark, we used transformers, state of the art language models, to fine-tune and get metrics for each task that consisted of single sentence prediction, similarity, paraphrase and inferencing. The primary model that we decided to focus on being ELECTRA, we tested out several other models that were mainly BERT framework based models and we ended up choosing BERT and XLNet as secondary models of interest.

Before training these models for predictions, we fine-tuned ELECTRA, BERT and XLNet by looking at the batch size, maximum sequence length, learning rate and number of epochs to get the best metrics possible. Since there were different metrics for each task, it was important to understand why the f1 score, Matthew’s Correlation Coefficient, Pearson-Spearman Correlation Coefficient and accuracy score were being used.

Finally, we began the ensemble process by first generating predictions from our 3 transformers on the validation set for every GLUE task followed by creating a new dataset for each task where the target was the actual label of the validation set and the independent variables were the predictions. We compared logistic regression and random forest classifier for the binary and multi-class label GLUE tasks and found that the results were very similar. For the GLUE task that contained a continuous target, we used linear regression to ensemble. Our results were predominantly positive in that the ensemble score was never lower than the lowest individual transformer score and most times similar or higher than the best individual score. Ultimately, using transformers and the ensemble technique to complete the GLUE benchmark was extremely useful in our understanding of transformer models and allowed us to gain a deeper understanding of the ELECTRA, BERT and XLNet architecture and framework.

Future Improvements

After completing this project, we noted several things that we could have done better to achieve a higher score or certain techniques we could have implemented in order to make our ensemble more accurate. Although our current results demonstrated the usefulness and function of ELECTRA, BERT and XLNet, we believe that we could go even further in the future to produce better results.

In the future, we plan to test the large versions of each of the transformers we selected as well as others. In doing this, we can run glue tasks on transformers that were pre-trained on a much larger corpus and contain an increased number of parameters by sacrificing computation speed. By using transformers that perform well individually, the predictions that they make will contribute to a better ensemble model. Furthermore, we can fine-tune more types of parameters. We found that there is a mode_config.json file that is in each transformer file on huggingface.com which contains parameters such as embedding_size, attention_heads, hidden_layers, etc. If we spend more time tuning these parameters, we would develop a better understanding of how each transformer architecture works at each layer and possibly output better metrics. The reason we decided to focus on the main parameters we listed (sequence length, batch size, etc.) is because the paper written by the creators of these transformers mentioned that the parameters values in the model configuration produced the best results.

During hyper parameter tuning, the maximum sequence length is defined as the number of tokens within each row of the text column. To get the proper parameter value that is unique to each GLUE dataset, we plan to create a histogram of the length of each row in the text column. From there, we can determine the average text length and initialize the proper max_seq_length parameter..

In order to create a better ensemble classifier, we thought it would be interesting to create an ensemble class that had each of our transformers of interest as layers. As opposed to running each model separately to collect the outputs, we could put the outputs of one model into the input of the next model by manipulating the number of neurons and hidden dimension sizes. As a result, the ensemble class could automatically output the predictions based on all 3 models combined. Another method that we considered was taking the average of the predictions we made from the 3 models and performing classification to train and test for accuracy. Finally, we would make predictions on the test set which we left unused in order to receive accuracy scores by uploading them to the leaderboard.

In this project, I learned how to use hugging face pretrained models, precess data into dataloader, use logistic regression to do ensemble, and modify the pretrianed models. Also, I further understand the XLNet and Electra, and learned how to use trainer class.

Copied Code percentage: 69.7%

References

Huggingface. (n.d.). Transformers/examples/pytorch/text-classification at master · Huggingface/Transformers. GitHub. Retrieved December 9, 2021, from https://github.com/huggingface/transformers/tree/master/examples/pytorch/text-classification.

Huggingface. (n.d.). Transformers/examples/pytorch/text-classification at master · Huggingface/Transformers. GitHub. Retrieved December 9, 2021, from https://github.com/huggingface/transformers/tree/master/notebooks.

Clark, M. Luong, Q. V. Le, and C. D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In ICLR.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, 32.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:1901.02860.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.