CS293S Course Project
XLNet: Generalized Autoregressive Pretraining for Language Understanding

Mert Kosan (mertkosan@ucsb.edu)
Zexi Huang (zexi_huang@ucsb.edu)

1 Introduction

High-quality word embedding plays an increasingly important role in various Natural Language Processing (NLP) and Information Retrieval (IR) downstream tasks. Through this course, we have learned multiple word embedding techniques, such as word2vec and BERT. A further exploration of the state-of-the-art word embedding algorithms is the topic of this course project. Specially, we focus on XLNet [1], a recent autoregressive language model that has shown consistently superior performance to BERT in standard benchmarks.

1.1 Challenge

The main challenge we identify in XLNet is its scalability with limited hardware, which consists of two parts: memory and time. More specifically, as the authors of XLNet have pointed out, 128 GB of GPU or TPU memory is needed to reproduce the experiment results shown in their paper, as the neural networks is deep and the paragraph lengths are large. This level of hardware is usually beyond reach for most researchers and practitioners who want to make take advantage of this state-of-the-art technique. In addition, even with the required hardware, the training process usually takes days to finish, which greatly increases the efforts needed for early prototyping and testing.

1.2 Objective

The objective of this project is to explore alternative experimental settings to handle the scalability challenges discussed above. We want to reproduce the experiments with limited hardware and within reasonable amount of time but also retain as much as possible the original downstream task performance. Providing insights of trade-off between time and accuracy is the main contribution of this project.

2 Related Work

The pioneering work of neural word embedding is word2vec [2], which is based on word co-occurrence pairs. GloVe [3] and fastText [4] use similar techniques to generate context-free word embeddings.

Contextualization has been shown to improve the word embedding quality. One thrust of contextualization is based on autoregression, such as ELMo [5] and GPT [6]. Another thrust is based on autoencoding, and the most impactful one is BERT [7], which has several notable extensions, such as RoBERTa [8] and ALBERT [9].

However, both autoregression and autoencoding based methods have some limitations: the autogressive models don't capture the bidirectional context, while the autoencoding models suffer from pre-train fine-tune discrepancy due to the artificial noise and also don't account for the dependence between predictions. XLNet overcomes both limitations and provides an autoregressive model that captures bidirectional context.

3 Framework

4 Experiments and Datasets

We focus on question answering tasks with fine-tuning. In the following subsections, we explain how we try to contribute XLNet by doing additional experiments. We check the sensitivity of three parameters to show

how the model can be optimized in a setting where you don't have too much computational power. We have used XLNet Github repository [10] for our initial starting point of coding.

4.1 Question Answering: Fine Tuning

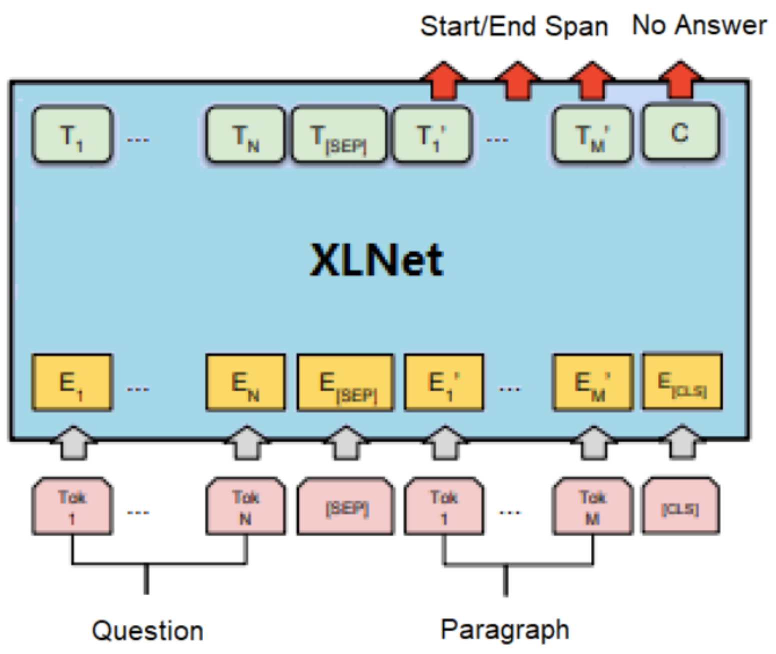
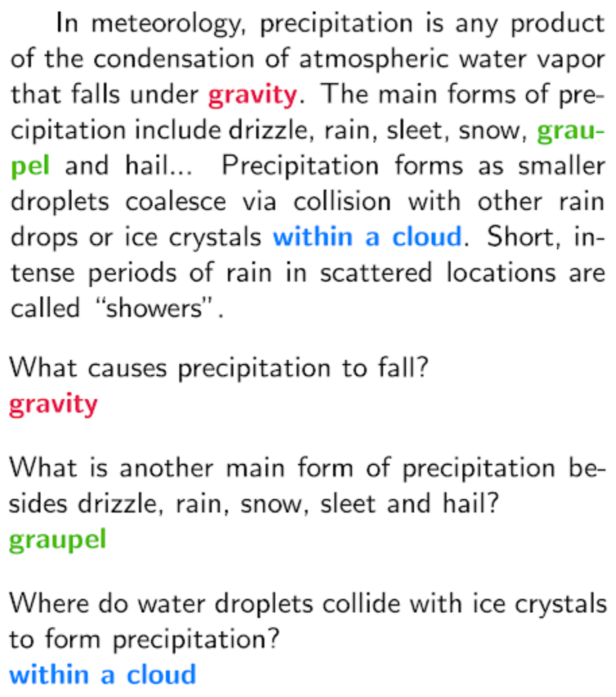


Figure 1: In this structure [11], we will have inputs as a question and the paragraph. The representation of tokens in the output layer will give us information about the answer to the question. We need to find START and END positions to answer the question. Output representation of tokens will feed into another Fully Connected layers to find the best START and END token. Class token will feed into another fully connected layer to tell us if the question has an answer or not.

4.2 Dataset

For the dataset, we use SQuAD v2 dataset for our experiments, there are two main versions of SQuAD.



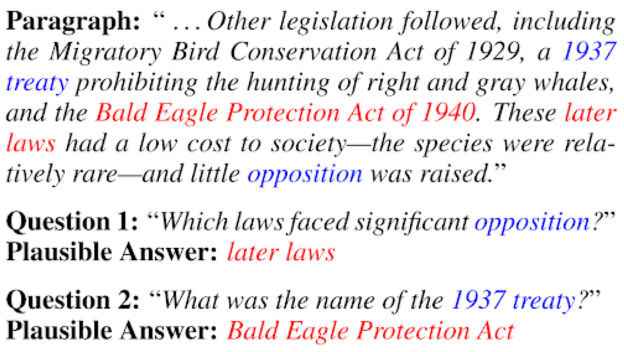
In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

(a) SQuAD v1.1 [12]



Paragraph: " *. . . Other legislation followed, including the Migratory Bird Conservation Act of 1929, a 1937 treaty prohibiting the hunting of right and gray whales, and the Bald Eagle Protection Act of 1940. These later laws had a low cost to society—the species were relatively rare—and little opposition was raised.*"

Question 1: "*Which laws faced significant opposition?*"
Plausible Answer: *later laws*

Question 2: "*What was the name of the 1937 treaty?*"
Plausible Answer: *Bald Eagle Protection Act*

(b) SQuAD v2 [13]

Figure 2: In version 1.1, there is a paragraph and the questions, the model will predict the answer. For example for the question "Where do water droplets collide with ice crystals to form precipitation?", START token is "within", END token is "cloud". So the answer is "within a cloud". In the second version, we have a similar format, but now the question may not be answerable meaning the answer is not in the paragraph. We use the second version of our experiments. We have 100.000 answerable and 50.000 not answerable questions.

4.3 Experiments

We explore the alternatives to make the fine-tuning scalable with limited hardware (a RTX 2070 Max-Q with 8GB Memory). We first decrease the batch size (from 8 to 4). We also decrease the max sequence length (512 to 256) to have less input size. We train only some portion of the network so that we can gain from time and memory. Then, we try to analyze how the change in fully connected layers for START/END/Class tokens will affect the performance.

4.3.1 Experiment: Sequence Length

We try four different sequence length for comparison of F1 score and running time.

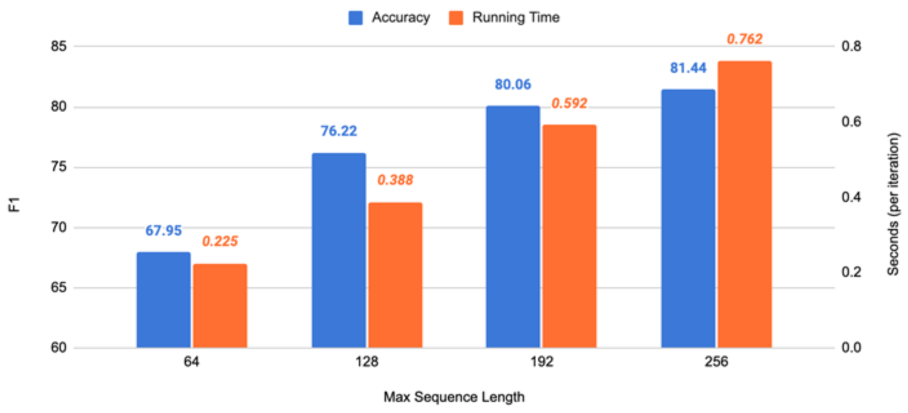


Figure 3: The figure shows the change in F1 score and running time when max sequence length varies. Each experiment only runs once with 12000 train-steps. The result shows larger sequence length will give us better performance and longer running time. As we can see here sequence length 64 has a very bad result. Even though 256 has a better result, it has a larger running time than 192. The best result is depending on the resources you have so we decided to use 192 sequence length for the next experiments.

4.3.2 Experiment: Partial fine-tuning

For the second experiment set, we check the effect of partial fine-tuning.

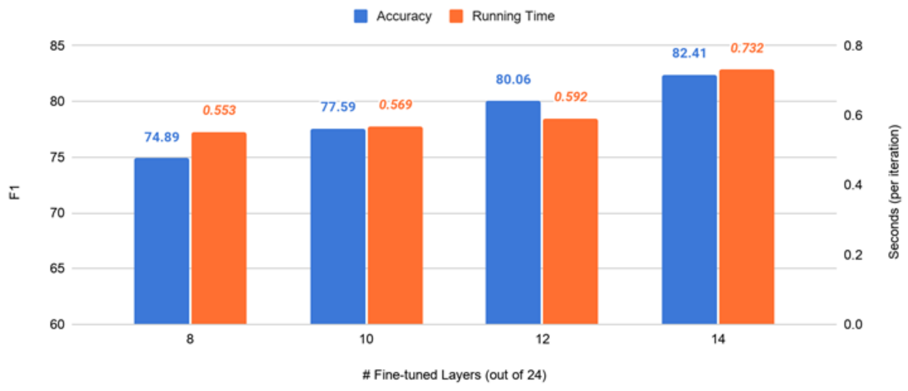


Figure 4: The figure shows the change in F1-score and running time when the number of trained layers varies. Each experiment only runs once with 12000 train-steps and max sequence length is set to 192. The model has 24 layers. However training all layers has a scalability problem, so we decided to train only a portion of the layers. Remember we still use the other layers with constant weights as they are pre-trained. As expected, when the number of fine-tuned layers increases; the performance is also increasing, however, the running time increasing as well. It is hard to see which option is better, but it all depends on the availability of memory and time. So we continue with 14 layers option.

4.3.3 Experiment: Output Layer

Thirdly, we try analyze different output layer structures.

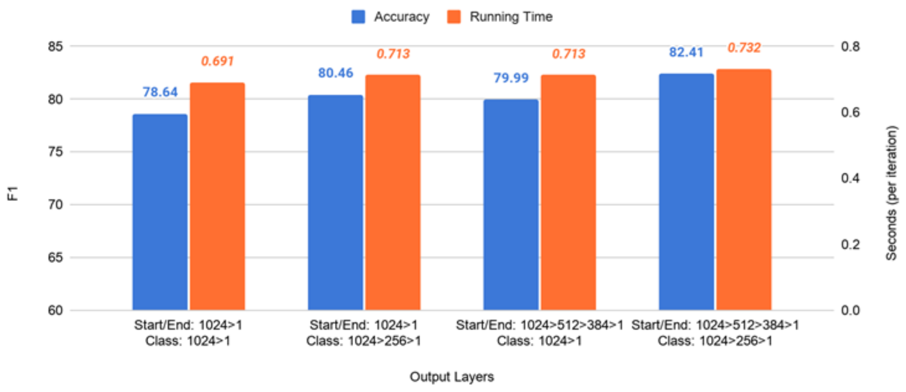


Figure 5: The figure shows the change in F1-score and running time when output layers vary. Each experiment only runs once with 12000 train-steps, max sequence length is set to 192 and the number of trained layers is 14. We use deep, 1024 $\rightarrow$ 512 $\rightarrow$ 384 $\rightarrow$ 1 for START and END, 1024 $\rightarrow$ 256 $\rightarrow$ 1 for Class. and shallow network 1024 $\rightarrow$ 1 for START, END and Class tokens. As expected deep network model produces better results with longer running time, for example on the most right when we have deep START/END/Class, we have better performance but again slower training. The difference between shallow START/END + deep Class and Deep Start/End + shallow Class is minimal. Also, as expected, shallow START/END/Class has worse performance whereas faster training.

5 Conclusion

XLNet is an autoregressive model that naturally captures bidirectional context, and has shown superior performance to autoencoding models, such as BERT and RoBERTa. We explore scalable alternatives to the question-answering fine-tuning to tackle the memory and time challenges of XLNet. We have provided insights into tradeoffs between performance and running time with different sequence lengths, partial fine-tuning, and different output layers.

We show our efforts and contribution to XLNet throughout the section Experiments and Dataset. We use one of the neural networks approaches XLNet for information retrieval applications we have learned during the class. Specifically, the class had talked about BERT and we wanted to explore in that direction which XLNet is one of the paper shows BERT's weaknesses and proposes a better approach where we explain throughout this report.

References

[1] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119, 2013.

[3] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[4] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[5] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2227–2237. Association for Computational Linguistics, 2018.

[6] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical report, OpenAI*, 2018.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[8] 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. *arXiv preprint arXiv:1907.11692*, 2019.

[9] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[10] Zihang Dai. https://github.com/zihangdai/xlnet, 2018.

[11] Steve Zheng. https://github.com/stevezheng23/xlnet_extension_tf, 2019.

[12] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[13] Pranav Rajpurkar, Jia Robin, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *arXiv preprint arXiv:1806.03822*, 2018.