CS293S Course Project: XLNet

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. 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 experimental results shown in their paper, as the neural networks are 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 a reasonable amount of time but also retain as much as possible the original downstream task performance. Providing insights into the 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 autoregressive 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 the bidirectional context.

3 Method

The overall architecture of XLNet is shown in Figure 1. We shortly summarize the key points of XLNet methodology below.

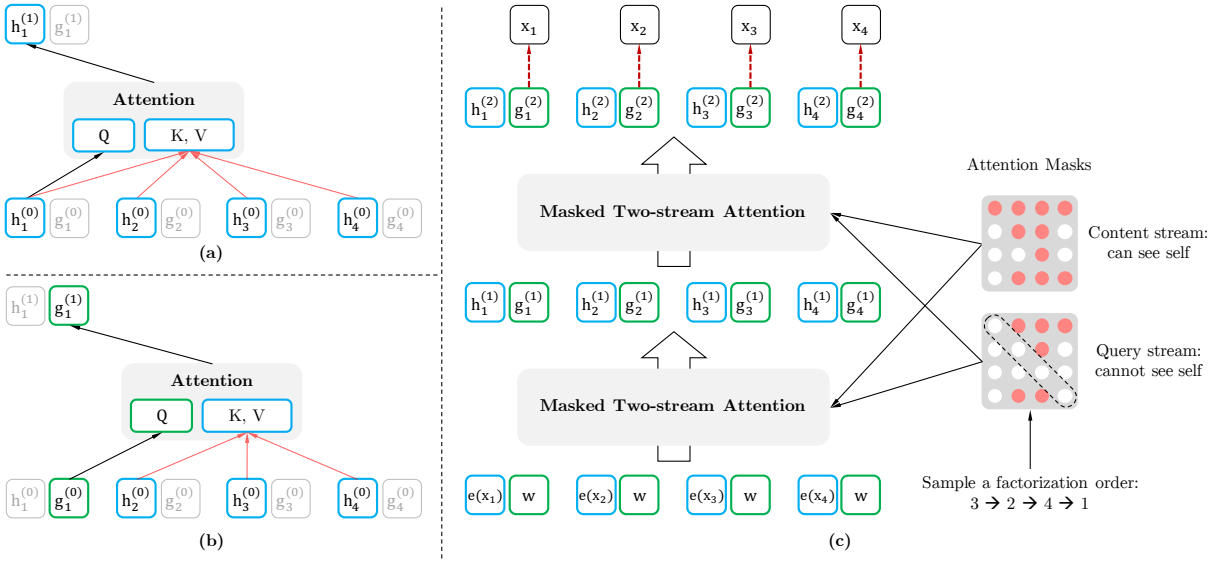

Figure 1: (a) Content stream attention. (b) Query stream attention. (c) Overview of the permutation language modeling training with two-stream attention.

3.1 Permutation Language Modeling

To capture bidirectional context, XLNet proposes an autoregressive objective based on permutations of different factorization orders:

$$\max_{\theta} \mathbb{E}_{\mathbf{z}\sim Z_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} | \mathbf{x}_{\mathbf{z}_{<t}}) \right] \tag{1}$$

Basically, given an input text sequence x, a random factorization order $\mathbf{z}$ is sampled from Z_T the set of all permutations of length T sequence. For example, $3 \to 2 \to 4 \to 1$ is sampled factorization order of original sequence $1 \to 2 \to 3 \to 4$. The likelihood $p_\theta(\mathbf{x})$ is then decomposed according to this factorization order. As we see, the modified autoregressive objective naturally captures bidirectional context.

3.2 Reparameterization

The standard Softmax parameterization of the next-token distribution $p_\theta(x_{z_t} | \mathbf{x}_{\mathbf{z}_{<t}})$ does not incorporate the position of the token, z_t, which makes it impossible to learn useful embeddings. XLNet overcomes this by reparameterize the distribution to be target position aware:

$$p_\theta(x_{z_t} | \mathbf{x}_{\mathbf{z}_{<t}}) = \frac{\exp(e(x)^\top g_\theta(\mathbf{z}_{<t}, z_t))}{\sum_{x'} \exp(e(x')^\top g_\theta(\mathbf{z}_{<t}, z_t))} \tag{2}$$

where $g_\theta(\mathbf{z}_{<t}, z_t)$ denotes the hidden representations that takes the target position z_t as an additional input.

3.3 Two-stream Attention

Since the hidden representations are fed as input to predict both the token itself and other tokens, one stream of $g_\theta(\mathbf{z}_{<t}, z_t)$ would lead to a contradiction. To resolve it, XLNet proposes two streams of hidden representations:

- Content stream $h_\theta(\mathbf{z}_{\leq t})$, abbreviated as h_{z_t}, that encodes both the context and x_{z_t} itself.
- Query stream $g_\theta(\mathbf{z}_{<t}, z_t)$, abbreviated as g_θ, that encodes the context $x_{\mathbf{z}_{<t}}$ and the target position z_t, but not the content x_{z_t}.

4 Experiments and Datasets

We focus on the fine-tuning of a pre-trained XLNet model based on the question answering task. In the following subsections, we explore three different components of XLNet to show how the model can be optimized with hardware that has limited computational power. We have used XLNet Github repository [10] for our initial starting point of coding.

4.1 Question Answering

In the architecture shown in Figure 2 [11], we have the inputs as a question and the paragraph. The representations of tokens in the output layer 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 is fed into another fully connected layers to find the best START and END token. CLASS tokens will be fed into another fully connected layer to predict the answerability of the question, given the paragraph.

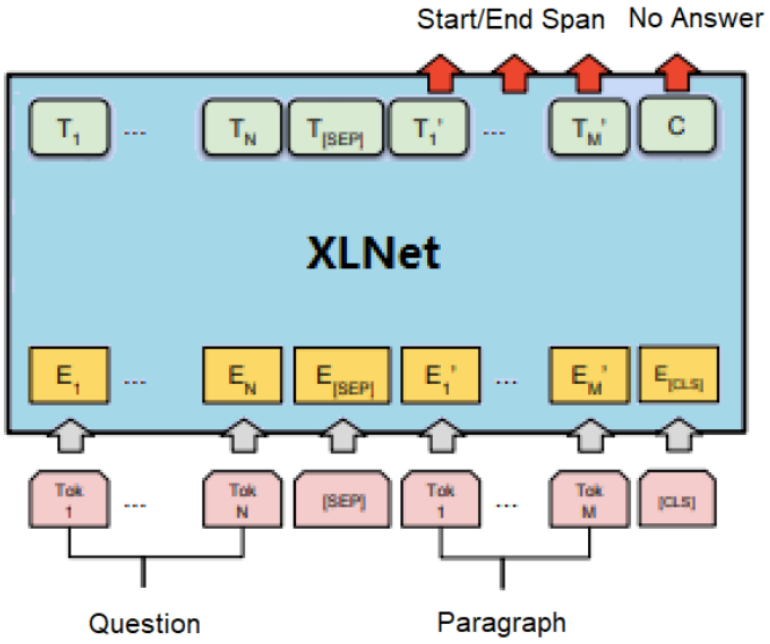

Figure 2: Question answering fine-tuning architecture.

4.2 Dataset

We use SQuAD v2 datasets for our experiments. Examples of both SQuAD v1.1 and SQuAD v2 are shown in Figure 3. In version 1.1, a paragraph and a question are presented, and the model predicts the answer from the paragraph. For example, for the question "Where do water droplets collide with ice crystals to form precipitation?", the model is expected to predict the START token as "within" and the END token as "cloud", leading to an answer "within a cloud". In the second version, a similar pair is presented, but now the question may not be answerable, meaning that the answer is not in the paragraph. We use the second version of our experiments. We have 100,000 answerable and 50,000 unanswerable questions.

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 3: Examples of SQuAD v1.1 and SQuAD v2.

4.3 Experiments

We explore the alternative settings to make the fine-tuning scalable with limited hardware (RTX 2070 Max-Q with 8GB Memory). We first decrease the batch size (from 8 to 4). Then, we vary three different components of XLNet and analyze their implications on the question answering performance.

- Sequence length.
- Fine-tuned layers.
- Downstream fully connected layers for START/END/Class tokens.

4.3.1 Experiment: Sequence Length

We try four different sequence lengths for comparison accuracy in terms of F1 score and running time in terms of seconds per iteration. The results are shown in Figure 6. Each experiment is only run once with 12000 train-steps. The result shows that larger sequence lengths lead to better performance and longer running time. We note that a sequence length of 64 has the worst performance, while a sequence length of 256 has the best performance, with significantly more running time. The best choice of sequence length depends on the available resources and we decide to use the sequence length of 192 for the next experiments.

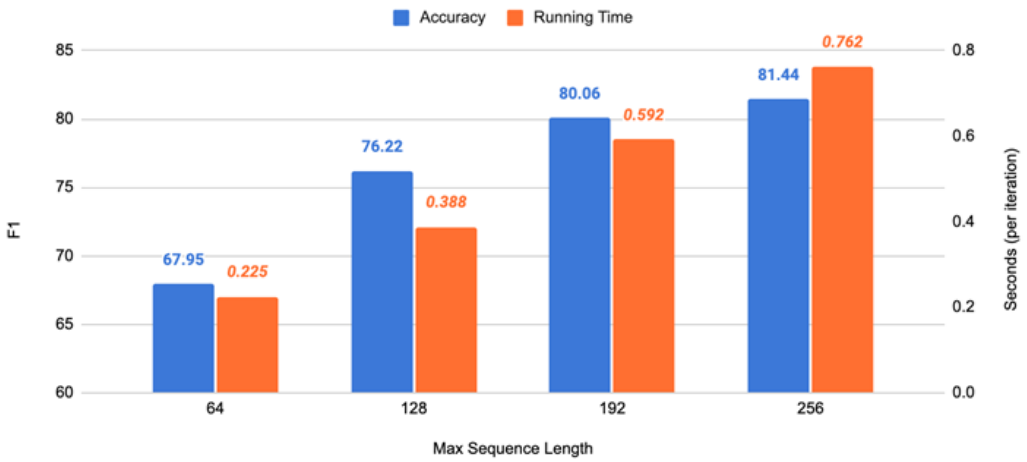

Figure 4: Accuracy and running time when max sequence length varies.

4.3.2 Experiment: Partial fine-tuning

We then evaluate the effect of partial fine-tuning, and the results are shown in Figure 5. The original XLNet model comes with 24 pre-trained layers and all of them are fine-tuned for specific downstream tasks. As an alternative, We fine-tune a proportion of them, leaving other layers with their pre-trained weights. As expected, when the number of fine-tuned layers increases, the performance increases along with the running time. Again, it all depends on the availability of memory and time to decide the best option, but for our hardware, we will continue with the 14 layers option.

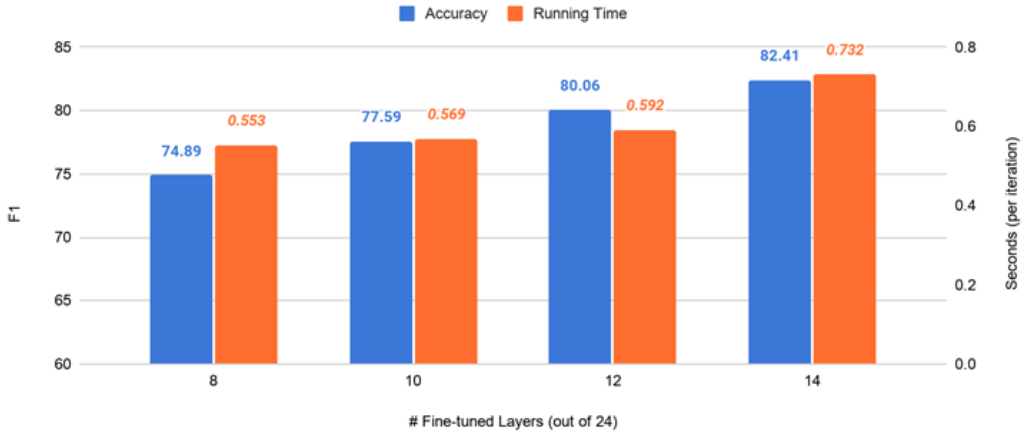

Figure 5: Accuracy and running time when the number of fine-tuned layers varies.

4.3.3 Experiment: Output Layer

Thirdly, we try analyze different output layer architectures. The results are shown in Figure 2. Two options are explored for both START and END tokens and CLASS tokens:

- Deep:
  - START and END: $1024 \rightarrow 512 \rightarrow 384 \rightarrow 1$
  - CLASS: $1024 \rightarrow 256 \rightarrow 1$

- Shallow:
  - START and END: 1024 → 1
  - CLASS: 1024 → 1

As expected, deep architectures produce better results with longer running time. For example, on the rightmost 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 fastest training.

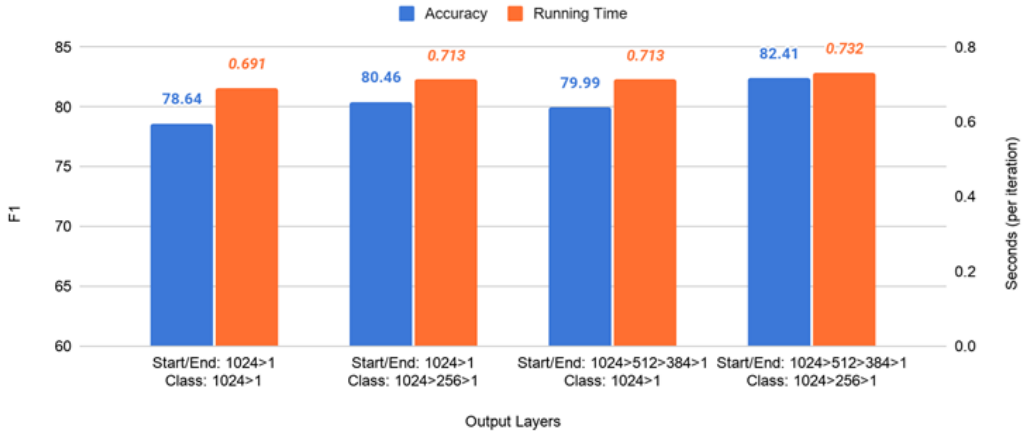

Figure 6: Accuracy and running time when output layers vary.

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.

As our contribution, 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.

Our project is closely related to the course materials in that we explore XLNet, the state-of-the-art method of neural word embedding, which is a notable follow-up work of word2vec and BERT that have been introduced in the lectures. Intrinsically, high-quality word embeddings could significantly increase the performance of information retrieval tasks, such as question answering, which has been illustrated with our experiments.

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.