Assignment 3

Machine Learning, 2022 Fall

Due: 2023-01-08 11:59 PM. You should submit a PDF format report (in NeurIPS template, no more than 4 pages except for reference and appendix; we appreciate quality instead of length) and code for this assignment to **Tsinghua Web Learning**. This assignment accounts for 20 points in your final score (1 for 2), as we do not have a HW4 due to the pandemic. Please finish the HW **independently**. Plagiarism is strictly prohibited, or you will fail the course.

1 Introduction

We have learned various types of pretraining architectures including autoregressive models (e.g., GPT), autoencoding models (e.g., BERT), and encoder-decoder models (e.g., T5) in previous lectures. NLP tasks are different in nature, with three main categories being classification, unconditional generation, and conditional generation. However, none of the pretraining frameworks performs the best for all tasks, which introduces inconvenience for model development and selection.

1.1 GLM: General Language Modeling as Autoregressive Blanking Infilling

GLM (General Language Model) is a novel pretraining framework to address this challenge. GLM is based on autoregressive blank-filling pre-training method. It randomly blank out continuous spans of tokens from the input text, following the idea of autoencoding, and train the model to reconstruct the spans, following the idea of autoregressive pre-training. To learn both the bidirectional and unidirectional attention mechanism in a single framework, the input text is divided into two parts, where the unmasked tokens can attend to each other, but the masked tokens cannot attend to subsequent masked tokens. GLM also use a 2D positional encoding technique to indicate inter- and intra- span position information. Figure 1 illustrates our pre-training objective. As a result, GLM learns both contextual representation and autoregressive generation during pre-training.

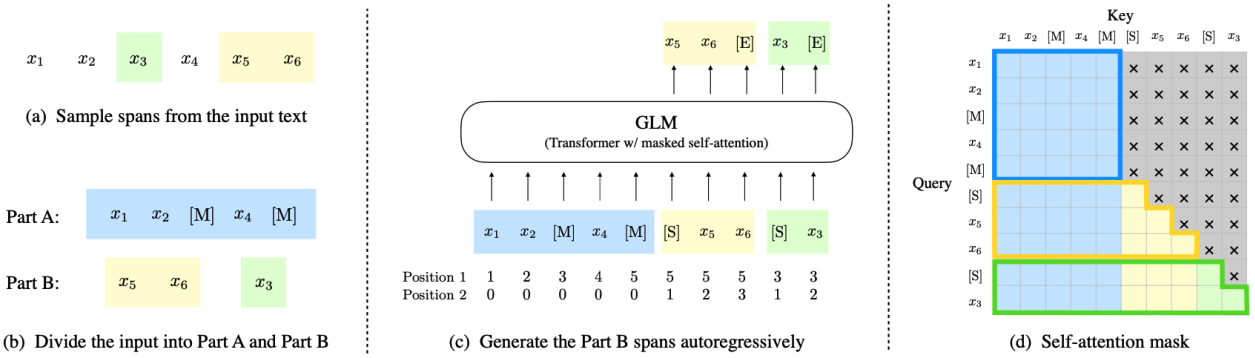

Figure 1: GLM pre-training framework. (a) The original text is $[x_1, x_2, x_3, x_4, x_5, x_6]$, and two spans $[x_3]$ and $[x_5, x_6]$ are sampled. (b) Replace the sampled spans with [MASK] tokens to form Part A, and shuffle the sampled spans to form Part B. (c) GLM is trained to autorergessively generate Part B. Each span is prepended with a [START] token as input and appended with a [END] token as output. 2D positional encoding is used to represent inter- and intra- span position. (d) Self-attention mask controls the attention mechanism, where the gray areas are masked out. Part A tokens can attend to A (the blue-framed area) but not B. Part B tokens can attend to A and their antecedent tokens in B (the yellow- and green-framed areas denote the tokens which the two spans in B can attend to). [M], [S], and [E] represents [MASK], [START], and [END] respectively.

1.2 Formulating All NLP Tasks As Prompted Generation

When fine-tuning GLM on downstream tasks, those tasks are reformulated as blank-filling generation tasks, as shown in Figure 2. Each task is associated with a human-crafted cloze question, and the model predicts the answer to the cloze. For example, a sentiment classification task is reformulated as filling the blank in "[SENTENCE]. It's really". The prediction of "good" or "bad" indicates the sentiment being positive or negative. With such formulation, GLM benefits from the consistency between pre-training and fine-tuning, because both

pre-training and fine-tuning involves training the model to generate text given context. As a result, GLM is more suitable for downstream classification tasks compared to BERT-like models.

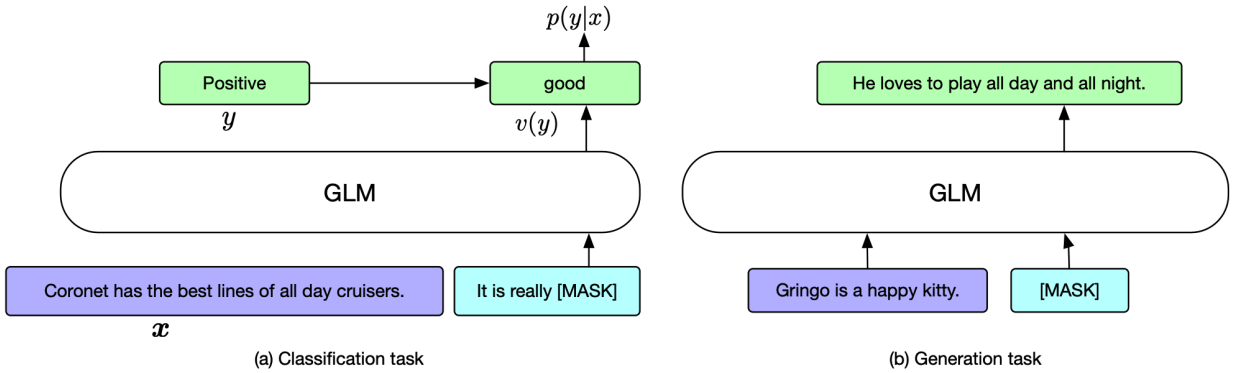

Figure 2: GLM fine-tuning framework. (a) Formulation of the sentiment classification task as blank infilling with GLM. (b) GLM for text generation given the context. This can be the language modeling in the zero-shot setting, or seq2seq with fine-tuning.

Compared to previous work, GLM has three major benefits: (1) it performs well on classification, unconditional generation, and conditional generation tasks with one single pretrained model; (2) it outperforms BERT-like models on classification due to improved pretrain finetune consistency; (3) it naturally handles variable-length blank filling which is crucial for many downstream tasks. Empirically, GLM substantially outperforms BERT on the SuperGLUE natural language understanding benchmark with the same amount of pre-training data. Moreover, GLM with 1.25× parameters of BERTLarge achieves the best performance in NLU, conditional and unconditional generation at the same time, which demonstrates its generalizability to different downstream tasks.

2 Your Tasks

In this assignment, you need to explore the use of the GLM model for different NLP tasks. Detailed requirements are as follows:

1. **Goal 1: Fine-tuning GLM on NLP tasks (50%)**: Select one classification task (excluding datasets from glue/super_glue/tweet_eval benchmarks) and one generation task for this assignment. We strongly recommend you to choose tasks from promptsource[1] since they already provide several prompts for each task. Use the `glm-roberta-large` model on Hugging Face[2] to fine-tune tasks of your choice. It's not allowed to add a additional linear classifier for the fine-tuning of NLU tasks. Our criterion is that the final performance should be reasonable. We provide a basic jupyter notebook **GLM-examples.ipynb** for you, which implements the necessary functions and framework of GLM. Please write down the dataset you decide to implement in the Tencent Sheet (https://docs.qq.com/sheet/DUEVsR1VES0t6bWNI?tab=BB08J2), and do not select datasets that has been chosen; first-come, first-choose.

2. **Goal 2: Grid search for best hyperparameters & Analysis (20%)**: Fine-tuning often sensitive to hyperparameter settings (e.g. learning rate and batch size). For each task, do a grid search for different hyperparameter combinations on the validation set, and report the results on the test set if available. Analyze the hyperparameter sensitivity and other discoveries, and maybe the comparison between conventional linear head-based fine-tuning and prompt-based fine-tuning from your experiments. The analysis' insightfulness would matter to your score in this part.

3. **Goal 3: Try different prompts & Compare with baselines (30%)**: Downstream performance of same model can vary significantly with different prompts. Try at least three prompts for different and compare their results after fine-tuning. Compare GLM's results with at least 2 similar-scaled model's performance (e.g., RoBERTa, T5-large, BART) on the same dataset. Codes for baselines need to be submitted, too.

4. **(Bonus, 2 points) Implement parameter-efficient tuning**: Parameter-efficient tuning, which only tunes continuous prompts with a frozen language model, substantially reduces per-task storage and memory

[1] https://github.com/bigscience-workshop/promptsource
[2] https://huggingface.co/BAAI/glm-roberta-large

usage at training. Implement at least one parameter-efficient tuning method (e.g. P-tuning v2[3]) and compare its downstream performance with fine-tuning.

5. **(Bonus, maximum 10 points) PR your codes to GLM's GitHub repository**[4]: We are more than delighted to invite you to become a contributor of GLM. If you make an *accepted (i.e., satisfy the PR rules)* pull request of a classification dataset (excluding datasets from glue/super_glue/tweet_eval benchmarks) to GLM, you will get 0.5 bonus point in your final score; if it is a generation dataset, you will get 1 bonus point in your final score. **(Deadline: 2023.01.15)**

The classification and generation task implementations account for 60% and 40% in Goal 1/2/3's scores, respectively.

3 Resources

We have contacted GPU sponsors for this assignment. The detailed GPU access tutorial will be out very soon.

4 References

- GLM paper: https://aclanthology.org/2022.acl-long.26/ (Sec 2.4 for fine-tuning GLM)
- Hugging Face GLM model: https://huggingface.co/BAAI/glm-roberta-large

[3] https://arxiv.org/abs/2110.07602
[4] https://github.com/THUDM/GLM/tree/main/examples