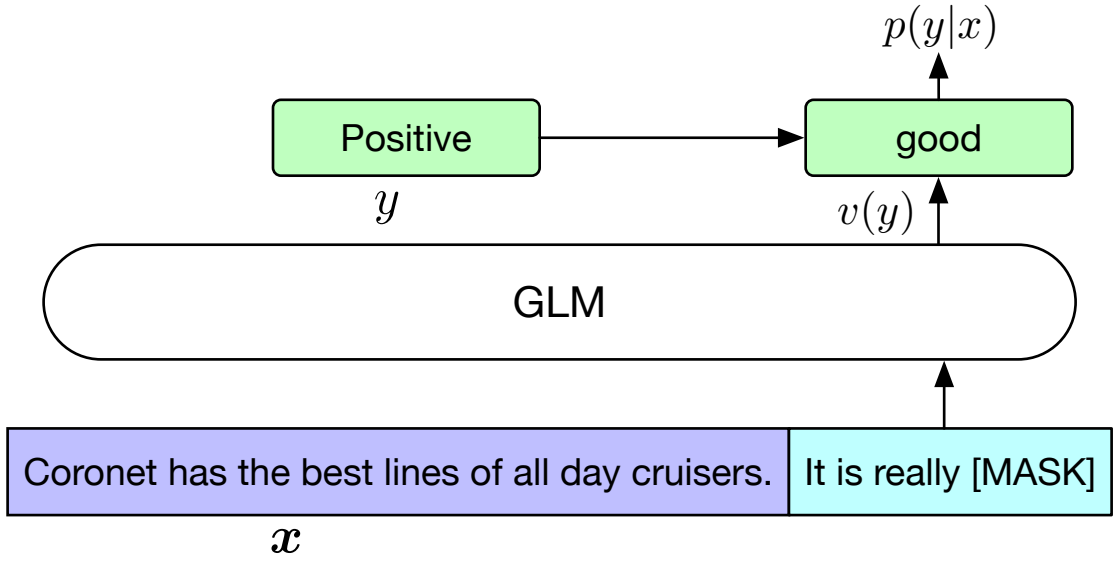
HW3 Tutorial

2022.12.20

HW3

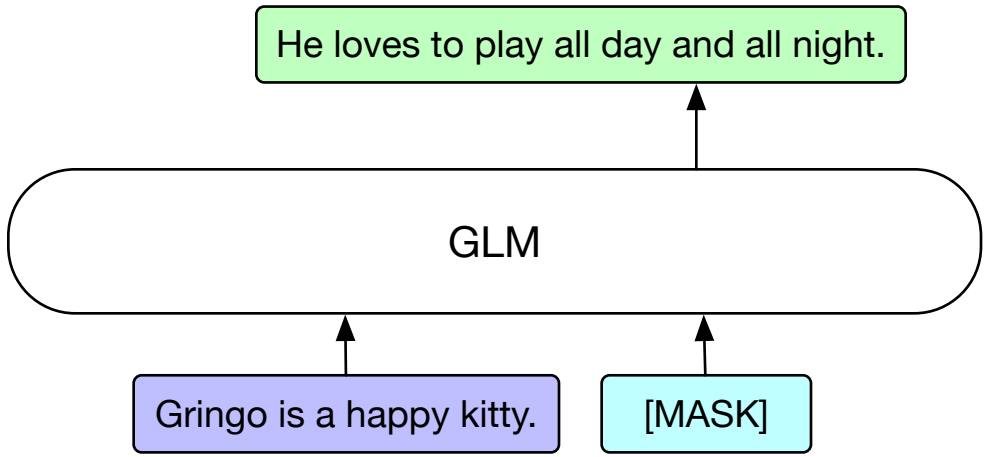
- In HW3, you need to implement
  - (50%) Fine-tuning GLM on NLP tasks: classification and generation tasks
  - (20%) Grid search for best hyperparameters & Analysis
  - (30%) Try different prompts & Compare with baselines
  - (Bonus) Implement parameter-efficient tuning
  - (Bonus) PR your codes to GLM's GitHub repository

Fine-tune GLM (Classification)



- Given a labeld example (x, y), we convert the input text x to a cloze question $c(x)$ via a pattern containing a single mask token [MASK].
- Each class y in the label space $\mathcal{Y}$ is also mapped to an answer to the cloze, called verbalizer $v(y)$.
- The predicted probability of the class y is proportional to $p(v(y)|c(x))$.

Fine-tune GLM (Generation)



- For text generations, we directly apply GLM as an autoregressive model.
- The given (prompted) context constitutes the Part A of the input, with a [MASK] token appended at the end. The model generates the text of Part B autoregressively.

Tips for STEP 1

- Multichoice:
  - Use cond_log_prob to calculate prob for each choice
  - loss = CrossEntropyLoss(choice_probs, label)
- Generation:
  - Concat context x and generation y and do a single forward
  - Loss = CrossEntropyLoss(prob_targets, y)
- All need batching to speedup!
- Checkout Huggingface's code for loss：
  https://github.com/THUDM/GLM#hugging-face-hub

Tips for STEP 2

- Grid search for best hyperparameters
  - Cf GLM's paper Appendix B
- **learning rate, batch size**
- lr warmup / weight decay …

B. Downstream Tasks

B.1. SuperGLUE

The SuperGLUE benchmark consists of 8 NLU tasks. We formulate them as blank infilling tasks, following (Schick & Schütze, 2020a). Table 7 shows the cloze questions and verbalizers we used in our experiments. For 3 tasks (ReCoRD, COPA, and WSC), the answer may consist of multiple tokens, and for the other 5 tasks, the answer is always a single token.

When finetuning GLM on the SuperGLUE tasks, we construct the input using the cloze questions in Table 7 and replace the blank with a [MASK] token. Then we compute the score of generating each answer candidate. For the 5 single-token tasks, the score is defined to be the logit of the verbalizer token. For the 3 multi-token tasks, we use the sum of the log-probabilities of the verbalizer tokens. Thanks to the autoregressive blank infilling mechanism we proposed, we can obtain all the log-probabilities in one pass. Then we compute the cross entropy loss using the groundtruth label and update the model parameters.

For the baseline classifiers, we follow the standard practice to concatenate the input parts of each task (such as the premise and hypothesis for textual entailment, or the passage, question and answer for ReCORD and MultiRC) and add a classification layer on top of the [CLS] token representation. We also implemented cloze-style finetuning for the other pre-trained models, but the performance was usually similar to the standard classifier, as we shown in the ablation study. Models with blank-infilling objectives, such as T5 and our GLM, benifits more from converting the NLU tasks into cloze questions. Thus for T5 and GLM, we report the performance after such convertion in our main results. For $\text{BERT}_{\text{Large}}$, $\text{RoBERTa}_{\text{Large}}$, and $\text{GLM}_{\text{RoBERTa}}$, we search the best hyperparameters on the dev set. The ranges of the hyperparameters for the grid search are: learning rate in $\{5e-6, 1e-5, 2e-5\}$ and batch size in $16, 32$.

B.2. Seq2Seq

We use the abstractive summarization dataset Gigaword (Rush et al., 2015) for model fine-tuning and evaluation. We fine-tune $\text{GLM}_{\text{LARGE}}$ on the training set for 10 epochs with AdamW optimizer. The learning rate has a peak value of $3e-5$, warm-up over the 6% training steps and a linear decay. We also use label smoothing with rate 0.1 (Pereyra et al., 2017). The maximum document length is 192 and the maximum summary length is 32. During decoding, we use beam search with beam size of 5 and remove repeated trigrams. We tweak the value of length penalty on the development set. The evaluation metrics are the F1 scores of Rouge-1, Rouge-2, and Rouge-L on the test set.

B.3. Language Modeling

We evaluate the model's ability of language modeling with perplexity on BookWiki and accuracy on the LAMBDA dataset (Paperno et al., 2016).

Perplexity is an evaluation criterion that has been well studied for language modeling. Perplexity is the exponentiation

Tips for STEP 3

- Promptsource: https://github.com/bigscience-workshop/promptsource
  - Many prompt templates

```
# Load an example from the datasets ag_news
>>> from datasets import load_dataset
>>> dataset = load_dataset("ag_news", split="train")
>>> example = dataset[1]

# Load prompts for this dataset
>>> from promptsource.templates import DatasetTemplates
>>> ag_news_prompts = DatasetTemplates('ag_news')

# Print all the prompts available for this dataset. The keys of the dict are the uuids the uniquely
>>> print(ag_news_prompts.templates)
{'24e44a81-a18a-42dd-a71c-5b31b2d2cb39': <promptsource.templates.Template object at 0x7fa7aeb20350>,

# Select a prompt by its name
>>> prompt = ag_news_prompts["classify_question_first"]

# Apply the prompt to the example
>>> result = prompt.apply(example)
>>> print("INPUT: ", result[0])
INPUT:  What label best describes this news article?
Carlyle Looks Toward Commercial Aerospace (Reuters) Reuters - Private investment firm Carlyle Group,
>>> print("TARGET: ", result[1])
TARGET:  Business
```

- T5: https://github.com/Shivanandroy/T5-Finetuning-PyTorch

GPU Resources

- Feishu Doc: https://bytedance.feishu.cn/docx/AqROdOpNhobCq2xdPDqcdbzTnKc
- Account will be distributed later

1. Download aisrc from Feishu Doc

```
aisrc config set
aisrc workspace create demo-workspace
aisrc dev_instance create demo-dev
aisrc dev_instance list
```