CS288 Natural Language Processing:

Homework4 Report

Name Diyun Lu SID 3039752224

1 Part a: Fintuning

(Because a few more runs after my report has been written so the results shown in the jupyter notebook are not totally the same with the data here)
The final accuracy on the test set is about 45.85%. I calculated the fraction of different kinds of errors and there were only 10 SQL queries that failed to execute. The rest of them (159) are all due to wrong answers. From my observations, the wrong answers are frequently seen under situations like: SQL queries are complicated, with nested select or order by structure; a confusion between state and city. In other words, when SQL is simple and does not involve many functions, it will be predicted more easily.
For different accuracy metrics: since different SQL may have the same answer, checking the execution accuracy seems to make more sense to me, the method for checking the answers for accuracy can show the capability of the model. But an exact match also somehow makes sense since that is the goal that the model tried to achieve.
Another interesting aspect is the use of the prompt. I found that if I use "Write a SQL query based on the question" before my question, the result turned out worse than without this instruction.

2 Part b: Prompting

With the long input prompts, the parameter `max_length` has to be increased to a larger number. That results in a longer running time. But even if it is 512, I still get many inputs longer than 512 which affects the results of predictions. There are about 20% of the data have failed to execute, and I think the long prompt is one of the reasons.
It also showed bad performance on nested select sentences.

3 Experiments

For finetuning, I adjusted the parameters: increase the learning rate and the number of training epochs. The results turned out to be much worse with only about 33% execution accuracy. The training loss was 0.88 at the checkpoint, but it was only 0.02 on the previous try. So increasing the batch size did not seem to be a good idea, and also the learning rate seemed too large.
For prompting, I tried to use cosine metric to calculate the similarities for generating the few shot data. I think the way I calculated the similarity may be wrong since it is not much better than random sampling.