Seventh Milestone Report: Nearest Neighbor Code Generation and Summarization

Alex Xie
https://axie66.github.io/07400-project/

Mentored by Vincent Hellendoorn
Institute for Software Research

April 18, 2022

1 Major Changes

I have added code language modeling, an unconditional sequence generation task, as an additional focus of the project. Specifically, I am investigating the use of kNN retrieval for domain adaptation of language models of source code.

2 Progress Report

2.1 Accomplishments

A major problem when fine-tuning or adapting pretrained models is catastrophic forgetting, in which the model overfits to new data and loses its knowledge of data seen during its pretraining. kNN retrieval offers a possible solution to this problem; by performing no (or very little additional training) and simply outfitting a pretrained model with a datastore of examples in the new domain, we can imbue it with knowledge of the new domain without sacrificing performance on previous tasks. I investigate this through the lens of two tasks, code summarization and code languag modeling.

For the task of lanugage modeling, I first trained a 12 layer decoder transformer, initialized from the GPT2 (natural language only) pretrained checkpoint, on the CodeSearchNet corpus. Specifically, in order to evaluate domain adaptation, the model was initially trained for 100k steps on data from five of the six programming languages in CodeSearchNet, then retrained for a couple thousand steps on a "held-out" language, Ruby (i.e. the new domain). Afterwards, certain model checkpoints were then outfitted with a kNN-LM component to measure the impact of retrieval at various points in retraining.

Table 1 indicates that adding kNN helps most while the model is still severely underfitting the Ruby data; after training on Ruby for 1000 steps, the improvements become fairly marginal. While the perplexity decreases are promising, I am at this point unsure how practically meaningful the current results are. kNN seems to be more or less equivalent to training for 100 more steps; while this retraining hurts performance on other languages, the decreases do not seem to drastic. It is possible that the benefits might be clearer on a large model such as Polycoder [1] (or with a larger datastore, though obtaining it may not be possible at this point given the time constraints).

The code summarization experiments are currently in progress; initial models were biased due to having been pretrained on the new domain, leading me to retrain the entire model from scratch, excluding the data.

# retraining steps	Ruby	Java	Python
0	34.5	5.1	6.1
0 + kNN	26.8	-	-
100	27.9	5.1	6.2
200	20.5	5.1	6.3
200 + kNN	17.3	-	-
300	17.2	5.2	6.6
400	15.4	5.3	6.8
1000	11.0	5.7	7.7
1000 + kNN	10.5	-	-
1500	9.3	6.0	8.3
2000	8.9	6.2	8.7
2500	8.6	6.4	9.0

Table 1: Perplexity for various languages over the course of retraining on Ruby (smaller is better). The model at step 0 is the original pretrained model, which has never seen Ruby code.

2.2 Previous Milestone Goals

My main goals from the last milestone were investigating the applications of kNN language models for domain adaptation and unsupervised code summarization (a special case of adaptation). I have made significant progress toward the former, while I am waiting on models to train for the latter.

2.3 Surprises

The improvements from kNN-LM on language adaptation were somewhat surprising; in particular, I did not expect the step 0 model, which had not seen Ruby code, to benefit so greatly from kNN-LM.

3 Next Steps

3.1 Looking Ahead

As the project draws to a close, I plan to wrap up my remaining experiments on kNN retrieval for domain adaptation - namely, code summarization on CodeSearchNet and code language modeling with Polycoder, a large (2.7 billion parameter) model. Additionally, I will collect my results, analyses, and reports throughout the semester to complete my final report and poster.

3.2 Revisions to Future Milestones

No future milestones need to be revised at this time.

References

[1] Frank F Xu, Uri Alon, Graham Neubig, and Vincent J Hellendoorn. A systematic evaluation of large language models of code. *arXiv preprint arXiv:2202.13169*, 2022.