First Milestone Report: Non-parametric Language Models for Natural Language to Code Generation

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

Mentored by Vincent Hellendoorn
Institute for Software Research

December 10, 2021

1 Progress Report

At this point, I have implemented and obtained results (see Table 1) for the "vanilla" non-parametric code generation model.[1] Specifically, this model is composed of a pretrained code generation model, BERT-TAE, the current state-of-the-art on the CoNaLa dataset [11], along with a non-parametric kNN-LM component that retrieves and references the nearest neighbors from an external datastore at each step of generation [9]. This borrows heavily from kNN-MT [8], which leverages the kNN mechanism for machine translation, a similar sequence-to-sequence task. Further, my implementation incorporates certain auxiliary augmentations to improve performance and efficiency [4], including adaptive retrieval (which ultimately was not used as it did not give good results), adding encoder context to the datastore representation, and reducing datstore size via PCA.

Current results are promising but not conclusive; while we do observe an increase in performance from adding the kNN component, this increase is fairly small in magnitude, at just +0.1 BLEU. Further, these improvements require a great degree of tuning to attain. Figure 1 shows the variation in model performance as we vary the number of nearest neighbors k, kNN distribution interpolation coefficient λ, and temperature τ. While the specific hyperparameters we choose yield improved performance, any slight change to them almost always drags performance below that of the original fully parametric model. Additionally, the non-parametric model is significantly slower than the parametric model due to the high cost of nearest neighbors search; while we use `faiss` to speed up search [7], due to the large size of the datastore (over 2 million entries), the non-parametric model remains roughly 3-5 times slower than the parametric model.

Additionally, by this point, I have completed my literature review of current approaches for code generation and understanding [1][2][11][13] as well as retrieval-augmented generation, both for code [12][14] and for NLP tasks in general [8][9][10]. Also of particular interest are methods for code representation learning [3][5][6][15], which may prove useful in improving kNN datastore representations.

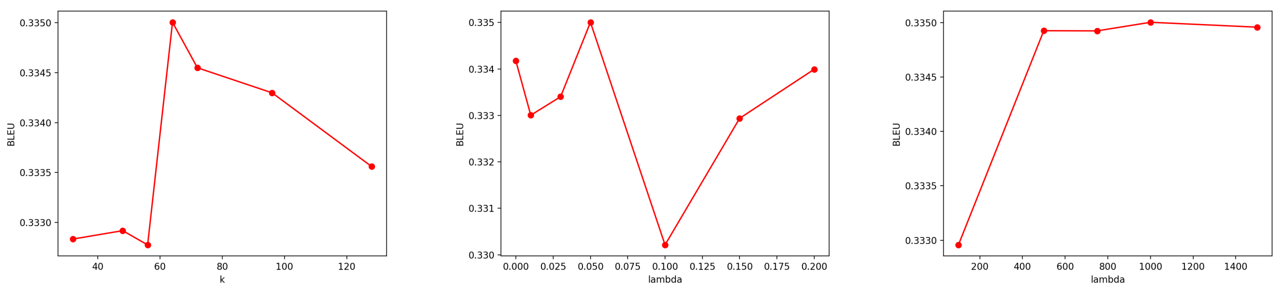
2 Reflection on Initial Plan

Overall, the path forward remains largely unchanged, and no 07-400 milestones need to be changed significantly. My results so far suggest that kNN models provide some advantage

[1] This work was done for my final project for another course that I am currently taking, 11-711. The full project report can be found here: `https://axie66.github.io/07400-project/11711report.pdf`

	BLEU	Exact Match
BERT-TAE	33.41	**3.4**
+ annotated datastore	33.17	3.4
+ mined datastore	33.41	2.6
+ encoder context	**33.50**	2.6

Table 1: Quantitative results and ablations for various data sources and context representations for our kNN datastore.



(a) $k \in \{32, 48, 56, 64, 72, 96, 128\}$ (b) $\lambda \in \{0, 0.01, 0.03, 0.05, 0.1, 0.15, 0.2\}$ (c) $\tau \in \{100, 500, 750, 1000, 1500\}$

Figure 1: Test set performance as we vary certain hyperparameters of our model.

over regular parametric models, albeit less than I might have expected. As such, I might look to dedicate additional time toward improving the "vanilla" kNN model with paired natural language/code data before moving onto kNN models using unpaired code-only data as originally planned. Specifically, one improvement to the vanilla kNN could be a lightweight mapping network that maps kNN queries to more appropriate embedding spaces. I will aim to get this work done over winter break or at the very start of next semester so as to leave enough time for the other objectives laid out in my project proposal.

I have achieved my primary aims for the first milestone. Specifically, as described earlier, I have implemented the kNN augmented model and obtained results that demonstrate its (modest) superiority over the original parametric model. In addition, I have set up but not run additional experiments incorporating the larger CodeSearchNet dataset (both in paired and unpaired form) into the datastore. This however, may require further modification and analysis as the code contained in CodeSearchNet is longer and more complex than that of CoNaLa, the primary dataset used for this project.

There have been no major surprises thus far in the project; while the model results are a bit lower than desired, they are not entirely unexpected. kNN models require extremely strong and expressive context representations, which can be attained for tasks such as language modeling and machine translation due to the relative maturity of models for those tasks. However, models for code generation, particularly on the CoNaLa dataset, do not perform as strongly and hence yield less informative representations, causing less relevant neighbors to be retrieved.

At this point in time, I believe that I have all the resources necessary for my 07-400 project; Professor Hellendoorn has graciously offered the usage of his lab machines next semester, which should provide more than enough computing power for my experiments.

References

[1] Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. Unified pre-training for program understanding and generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2668, Online, June 2021. Association for Computational Linguistics.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[3] Jian Gu, Zimin Chen, and Martin Monperrus. Multimodal representation for neural code search. *ICSME*, 2021.

[4] Junxian He, Graham Neubig, and Taylor Berg-Kirkpatrick. Efficient nearest neighbor language models. In *Proceedings of EMNLP*, 2021.

[5] Vincent J. Hellendoorn, Charles Sutton, Rishabh Singh, Petros Maniatis, and David Bieber. Global relational models of source code. In *International Conference on Learning Representations*, 2020.

[6] Paras Jain, Ajay Jain, Tianjun Zhang, Pieter Abbeel, Joseph E. Gonzalez, and Ion Stoica. Contrastive code representation learning. *arXiv preprint*, 2020.

[7] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[8] Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Nearest neighbor machine translation. In *International Conference on Learning Representations (ICLR)*, 2021.

[9] Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*, 2020.

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in*

Neural Information Processing Systems, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

[11] Sajad Norouzi, Keyi Tang, and Yanshuai Cao. Code generation from natural language with less prior knowledge and more monolingual data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 776–785, Online, August 2021. Association for Computational Linguistics.

[12] Md Rizwan Parvez, Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. Retrieval augmented code generation and summarization. In *EMNLP-Findings*, 2021.

[13] Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, 2021.

[14] Frank F. Xu, Junxian He, Graham Neubig, and Vincent J. Hellendoorn. Capturing structural locality in non-parametric language models. 2021.

[15] Daniel Zügner, Tobias Kirschstein, Michele Catasta, Jure Leskovec, and Stephan Günnemann. Language-agnostic representation learning of source code from structure and context. In *International Conference on Learning Representations (ICLR)*, 2021.