

# Usage of N-gram models to improve autocompletion in Pharo

Myroslava Romaniuk  
Ukrainian Catholic University  
romaniuk@ucu.edu.ua

## ABSTRACT

Ask Stephane if you should add an affiliation with Inria and RMoD. This abstract is too long. Consider splitting it into Abstract and Introduction. Code completion is one of the most essential features of any IDE that greatly improves a good code completion tool can greatly improve the developer experience, whereas poor autocompletion can hinder the progress and discourage developers. Thus, it is important to find a way to make it as effective as possible. Specifically, in the case of Pharo, same name for a Smalltalk-like language and an open-source programming language and IDE, in the last year and a half a lot of work has been carried out to improve code completion. However, there are more potential improvements from which developers can only benefit: this makes no sense, try to rephrase it: saving time and effort spent while coding remains an open question, and improving the relevance of suggestions and hence reducing time the developer takes to go through the list seem promising. In Pharo, the existing autocompletion implementation so far only allows for alphabetical sorting of results. The current implementation of code completion in Pharo sorts the proposed completions alphabetically. Mention that it makes use of the AST, class, etc. While this approach is mostly usable, it This approach completely ignores the potential of suggesting completion results based on code history and analysis. What analysis?, which is something that has been successfully adopted by other IDEs. Thus, in this work I intend to study how code completion in Pharo can be improved by using machine learning and statistical language models. In particular, the idea is to use the N-gram model, as it has been documented to be used for such a task, and, if successfully adapted to Pharo, might help enhance the quality of code completion. code completion quality in general and the relevance of proposed completions in particular. What is the difference between completion quality and relevance?

There are many approaches to handling the process of completing code. The classic strategies have been to use lexical models (completing based on the prefix the user is typing) or to use semantic models (use code structure analysis information to tailor completions). In the latest years statistical or machine learning models have also gained popularity. They are not the same thing. For the most part when using ML, source code analysis is also often disregarded in lieu instead of simply training the model on the code base and inferring all the

possible code dependencies from that. I don't understand this last sentence.

Don't write the title of every paper, simply cite them and titles will appear in references. My work was inspired by The famous paper "On The Naturalness of Software" (by Hindle et al. [2] served as a sort of catalyst for the following research, applying statistical models to Pharo. There, as well as in "On the localness of software" (Tu et al.), "Code completion with statistical language models" (Raychev et al.), "Are deep neural networks the best choice for modeling source code?" (Hellendoorn et al.) and "Code completion with neural attention and pointer networks" (Li et al.) [1, 3–5], the structural information is not taken into consideration and the problem of improving code completion is reduced to a natural language processing problem, where predicting the next token is treated as predicting the next word in a sentence.

The n-gram language model approach proposed by Hindle et al. is based on capturing high-level statistical regularity. What does it mean? at the n-gram level by taking n-1 previous tokens that are already entered into the text buffer, and attempting to guess the next token. Using this model, it is possible to estimate the most probable sequences of tokens and suggest the most relevant code completions to developers. Make it more clear that your work is different from the work of Hindle et al. and explain why it is different.

Therefore, in Pharo our idea is to have a sorting strategy based on an n-gram model, on top of the actual completion engine that works by analysing AST information of the source code, having actually utilised structural information. What is structural information and how do you utilise it? In this case, we are able to get semantically-appropriate results based on type whenever possible, and then sort the results by relevance. Will help which allows us to suggest much more accurate tokens for when the type is not known. (as Pharo is a dynamically typed language). You can mention that at the beginning of the sentence.

The goal of this will be to see The contributions of this work are the following:

- whether we can improve code completion in Pharo by sorting candidate completions with an n-gram language model
- whether we can build a tool based on a trained n-gram model that would propose completion fast enough to be used in an IDE
- how can we Propose a methodology to numerically evaluate the results of code completion produced by different completion strategies

## ACM Reference Format:

Myroslava Romaniuk. 2020. Usage of N-gram models to improve autocompletion in Pharo. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

Conference'17, July 2017, Washington, DC, USA

© 2020 Association for Computing Machinery.

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...\$15.00

<https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

## REFERENCES

- [1] HELLENDORF, V. J., AND DEVANBU, P. Are deep neural networks the best choice for modeling source code? In Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (2017), pp. 763–773.
- [2] HINDLE, A., BARR, E. T., SU, Z., GABEL, M., AND DEVANBU, P. On the naturalness of software. In Software Engineering (ICSE), 2012 34th International Conference on (2012), IEEE, pp. 837–847.
- [3] LI, J., WANG, Y., LYU, M. R., AND KING, I. Code completion with neural attention and pointer networks. arXiv preprint arXiv:1711.09573 (2017).
- [4] RAYCHEV, V., VECHEV, M., AND YAHAV, E. Code completion with statistical language models. In Acm Sigplan Notices (2014), vol. 49, ACM, pp. 419–428.
- [5] TU, Z., SU, Z., AND DEVANBU, P. On the localness of software. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (2014), pp. 269–280.