Optimizing Duplicate Size Thresholds in IDEs

Konstantin Grotov,[*][†] Sergey Titov,[†] Alexandr Suhinin,[§] Yaroslav Golubev,[†] Timofey Bryksin[†]

[§]*JetBrains*, [†]*JetBrains Research*, [*]*Constructor University*

{konstantin.grotov, sergey.titov, alexandr.suhinin, yaroslav.golubev, timofey.bryksin}@jetbrains.com

Abstract—In this paper, we present an approach for transferring an optimal lower size threshold for clone detection from one language to another by analyzing their clone distributions. We showcase this method by transferring the threshold from regular Python scripts to Jupyter notebooks for using in two JetBrains IDEs, Datalore and DataSpell.

I. INTRODUCTION

While clone detection is a well-established field of research, its practical application still faces open questions, in particular, the minimal size of clones that need to be detected to filter out trivial, universal code, with researchers often selecting different thresholds for the same techniques [1], [2]. Several works even researched the idea of using many different thresholds at once [3], [4]. This problem manifests itself in IntelliJ-based IDEs [5], such as PyCharm [6], where exact clones [7] are underlined in the editor. To do this, the current threshold was established empirically back when the system was developed. At the same time, different languages vary in their verboseness and thus may require different thresholds. Therefore, for the younger IDEs, we decided to establish new, more fitting, thresholds. However, firstly, it is very time-consuming to carry out a separate user-studies and A/B tests for each IDE and language. Secondly, existing users already got used to the current default thresholds, and changing the IDE's behavior drastically might interfere with their workflow. Thus, rather than implement a brand new threshold, the IDE development team wanted to *update* and *optimize* the threshold for a new language so that the amount of detected clones is roughly the same. In this paper, we describe how we did this for two new JetBrains IDEs — Datalore [8] and DataSpell [9] — via comparing regular Python code and Jupyter notebooks.

II. APPROACH, EVALUATION, AND FUTURE WORK

Approach. Our core idea is to use distributions of all the detected clones within two languages or ecosystems to find such a threshold that would detect the same percentage of clones. In this work, we consider only exact clones within a single file, since they are the ones highlighted in the IDE.

To find all clones within one file, we employed the popular suffix tree based approach that is used in the platform [10]. While the suffix tree is usually applied to strings, it is possible to build it for any arbitrary sequence, therefore, we use it directly on the elements of the platform's concrete syntax tree (CST) [5]. Having obtained these elements from parsing, we iteratively go through all possible sizes for duplicates, from 3 tokens up to the half of the file's length. As a result, we get a list of all duplicates for each given size for the analyzed file.
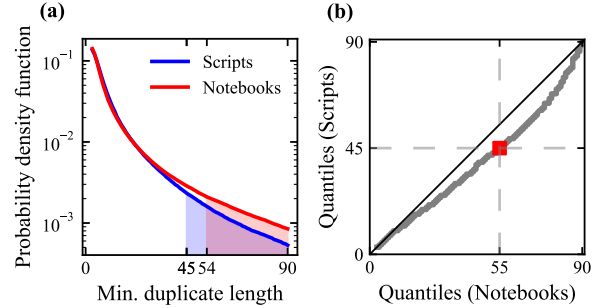

Fig. 1. (a) Probability density function of duplicate count in notebooks and scripts. (b) Quantile-Quantile plot of notebook and script clone distributions. Red square indicates the intersection of quantiles, corresponding to the same quantile's marker of different distributions.

Firstly, we take the language, for which we know the optimal threshold, collect a dataset for it, and search it for clones as described above. This results in the distribution of a mean number of duplicates in a file with each tested minimal threshold. On this distribution, we can find the bin corresponding to our optimal value and calculate its quantile rank. In order to find the optimal threshold for another language, we need to repeat the same process and find the threshold that corresponds to the same quantile rank in its distribution.

Evaluation. We applied this approach to find the optimal value for Jupyter Notebooks for two new IDEs — Datalore [8] and DataSpell [9]. Research shows that code clones are frequent in notebooks [11] and that the code in regular Python and notebooks is different [12]. We sampled 10,000 Python scripts and 10,000 Jupyter notebooks with permissive licences and 10+ stars on GitHub from our previous work [12]. Next, we applied our algorithm, and calculated that the default PyCharm's threshold of 45 CST elements corresponds to the 95th percentile of the distribution, meaning that 5% of all potential Python clones are underlined. The area under curve of the probability density functions (Figure 1a) shows that to highlight the same percentage in notebooks, the threshold needs to be higher, at 54 elements, since Jupyter notebooks have a "heavier tail" of larger clones. The same can be visualized as a QQ-plot (Figure 1b). You can find additional technical details and figures in our online appendix [13].

Future work. Currently, the obtained threshold is being evaluated by the development teams of Datalore and DataSpell, we plan to carry out UX studies with users to compare the threshold's comfortability. We believe that our pipeline, while simple, can be useful for various practical applications of clone detection. It can be used for any other family of languages, for example, for JVM-based languages.

REFERENCES

[1] T. Wang, M. Harman, Y. Jia, and J. Krinke, "Searching for better configurations: a rigorous approach to clone evaluation," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, 2013, pp. 455–465.

[2] C. Ragkhitwetsagul, J. Krinke, and D. Clark, "A comparison of code similarity analysers," *Empirical Software Engineering*, vol. 23, no. 4, pp. 2464–2519, 2018.

[3] I. Keivanloo, F. Zhang, and Y. Zou, "Threshold-free code clone detection for a large-scale heterogeneous Java repository," in *2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. IEEE, 2015, pp. 201–210.

[4] Y. Golubev, V. Poletansky, N. Povarov, and T. Bryksin, "Multi-threshold token-based code clone detection," in *2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2021, pp. 496–500.

[5] Z. Kurbatova, Y. Golubev, V. Kovalenko, and T. Bryksin, "The IntelliJ platform: a framework for building plugins and mining software data," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*. IEEE, 2021, pp. 14–17.

[6] "PyCharm," https://www.jetbrains.com/pycharm/, [Online; accessed 16-March-2023].

[7] C. K. Roy and J. R. Cordy, "A survey on software clone detection research," *Queen's School of computing TR*, vol. 541, no. 115, pp. 64–68, 2007.

[8] "Datalore," https://datalore.jetbrains.com/, [Online; accessed 16-March-2023].

[9] "DataSpell," https://www.jetbrains.com/dataspell/, [Online; accessed 16-March-2023].

[10] H. Liu, Z. Ma, L. Zhang, and W. Shao, "Detecting duplications in sequence diagrams based on suffix trees," in *2006 13th Asia Pacific Software Engineering Conference (APSEC'06)*. IEEE, 2006, pp. 269–276.

[11] A. P. Koenzen, N. A. Ernst, and M.-A. D. Storey, "Code duplication and reuse in Jupyter notebooks," in *2020 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2020, pp. 1–9.

[12] K. Grotov, S. Titov, V. Sotnikov, Y. Golubev, and T. Bryksin, "A large-scale comparison of Python code in Jupyter notebooks and scripts," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 353–364.

[13] "Online appendix and replicating package," https://github.com/JetBrains-Research/jupyter-python-clones, [Online; accessed 16-March-2023].