Categorization of process names

Zamira Kholmatova
Innopolis, Russia
E-mail: z.kholmatova@innopolis.university

Abstract. The categorization of processes according to their functionality is a key point to understand, how the user spent his or her time working with a computer. This task is not trivial due to the limited textual information available with applications. The existing approaches of this area commonly use predefined categories or clusterization techniques for applications, not processes. These techniques also do not describe the choice of these particular categories. In this paper, we describe our approach for the categorization of process names and explain, why we choose the set of categories we used. We evaluated the proposed approach on our collected data and got 81% correctly classified processes.

1. Introduction
Nowadays, the biggest software repositories are hosting thousands of different software systems [1]. With the growth of software products, it is important to guarantee that we still can differentiate given applications, since the categorization can be very useful in a variety of tasks:

- finding the software. We can group similar software systems [2, 3] in one category to make them easily found;
- finding CPU utilization objectives [4]. We can understand, which category consumes more energy [5, 6, 7, 8, 9];
- informing developers about related software systems [10]. Developers can learn the “best practices”;
- software maintenance [11]. Developers could recognize what problems or bugs of applications from the same category can appear.

Also, categorization can help to understand user preferences, improve personalized services. In our case, we are more interested in the investigation of the time spent on each category by different users. To do this, we need to be able to categorize the processes that were executed. In the past, the categories were assigned manually, but now with the huge amount of applications, it is almost impossible to do. This is based on software metrics and the existing wide approaches in their analysis [12, 13, 14, 15, 16, 17, 18, 19] and it is particularly important in the agile industry, where we aim at gathering such information without additional effort from developers [20, 21, 22, 23, 24, 25, 26, 27].

In this paper, we propose a method for categorization of processes executed during the user’s interaction with a computer. It is based on finding the longest matching subsequence of two elements: the process name and element from the list of applications with categories. Our results are promising: we can correctly categorize 81% of process names we collected.
The paper is organized as follows. Section 2 gives brief information about what has already been done in this area. In Section 3 we described the data we collected for our study and explained our choice of categories. Section 4 reports about our proposed approach. In Section 5 one can find the descriptions of experiments and evaluation of a proposed approach. Discussion of the results is in Section 5. Finally, possible improvements and future work is described in Section 6.

2. Related works
Before constructing the method, it would be better to know, what are we expecting from the categorization system. Kawaguchi et al. (2006) [28] described requirements for an ideal automatic software categorization system and proposed their MUDABlue method based on Latent Semantic Analysis. With the help of Latent Semantic Analysis, they retrieved categories and then determined what software systems belong to those categories. Maletic et al. (2000) [29] also used Latent Semantic Analysis to the source code, but for finding similarities in software systems. Source code analysis for identifying topics was also discussed in works Kuhn et al. (2007) [30], Maskeri et al. (2008) [31].

Instead of using source code, Zhu et al. (2012) [32] proposed a classification approach for mobile applications using contextual information about them. Their method captures context patterns. Applications, which have similar context patterns, will go to the same class. Yoon et al. (2017) [33] considered only application titles as textual information and suggested a method for automatic classification of applications without predefined categories. To improve classification accuracy, Radosavljevic et al. (2016) [34] proposed a method without predefined categories, which takes into consideration both the textual description and title of an application.

Singla et al. (2018) [35] showed, that using application images with contextual information can also help to obtain better classification accuracy.

3. Data collection
We have two sets of data:

(i) applications with predefined categories;
(ii) process names, for which we need to define the category.

The difference between the application names and process names is that the process names have extensions like .exe or the path to the executable file.

The first set of data is the set of application names, with which we will find the similarity of a given process. Since there are a lot of sources containing application names with categories, we decided at first to construct a mapping from processes to applications and then for each application define the category. The second set is the process names themselves.

Based on the works of researchers and tools (like RescueTime [36]) in the related fields, we defined the most popular categories that we will use in our work:

(i) utilities - downloaders, OneDrive;
(ii) entertainment - games, music and video players;
(iii) development - IDEs and development tools;
(iv) communication - chats, messengers;
(v) management - calendar, project management and tracking tools;
(vi) education - readers, text editors;
(vii) design&creativity - photo and video editors, design applications.
Each of these categories we filled with the names of applications taken with a help of SourceForge \cite{37}. Overall, we have 125444 popular application names in both open source and commercial usage \cite{38, 39, 40, 1, 41, 42, 43}.

The data without categories comes from a non-invasive measurement tool, “Innometrics” \cite{44, 45, 46, 47, 48, 49, 50, 51, 52}, tracking the application on focus. Each record of the data contains the process name.

First of all, we preprocessed these process names. We deleted the extensions and the path to the executable file, leaving only the main part of the process name. For example, if we have “Telegram.exe”, we will use only “telegram”. After the preprocessing procedure, we labeled this data manually in order to test our proposed approach \cite{53}.

4. Proposed approach

We wanted to create a simple method, which will take a process name as an input and give the category as an output. We saw from works described in Section 2, that the classifiers, which works with contextual information, can give good results. Based on this, we decided to use only the process name in our approach. Since in most cases, the process names are very similar to application names, we decided to use this fact as a cornerstone of our approach.

The challenging thing in such an approach is how to measure the similarity between two strings. The application name can have more symbols than the process and vice versa. Also, the process name could be an abbreviation of an application name, so weed solution, that will capture this information.

We considered different ways of measuring the similarity, starting from the most popular distances between two words (like Levenshtein \cite{54} and Hamming \cite{55} distances) and finishing with built-in sequence matching functions in Python. For our task, we choose “SequenceMatcher” class from python module “difflib” \cite{56}. It is a class for comparing pairs of sequences of any type. The main idea is to find the longest contiguous matching subsequence that does not contain unnecessary elements. SequenceMatcher tries to compute a difference between two sequences that is more familiar for human eyes.

![Proposed method diagram](image)

**Figure 1.** Proposed method

We can see the scheme of our method work in Figure 1. It takes a process name from the Innometrics system. After that, the process name is going to be preprocessed. At this stage, all the extensions and paths will be removed. Then we will calculate the similarity between “clear” process name and applications from the first dataset described above. The application with the highest value will be chosen as ground truth and its category will be assigned to the given process name.
5. Results and discussion
During the evaluation of our method, we set the threshold: if the ratio of similarity is less than 0.5, we consider our process name as uncategorized. These uncategorized processes we will consider separately since it could be an executable file of operating system interior process.

Also, in our experiments, we did not classify the browsers, since their category strongly depends on the considered information through it.

For our experiments, we have different process names taken from the Innometrics system. Our proposed approach can correctly assign the category for 81% of process names. 14.3% of considered processes were labeled as “Uncategorized”. In this case, we had some names, for which it is difficult (abbreviations from application names) or impossible (executable files of operating system interior process) to find applications. And finally, only 4.7% of processes were classified incorrectly.

6. Conclusion and future work
The ability to categorize applications in our own way encourages us to move ahead in this direction. We want to use this categorization approach to understand the time spent for each category by a computer user. After it, we want to extend our classifier, so it could classify the browser titles based on the same categories.

Also, we would be most happy to share the data that we have collected with other researchers interested in this area, for a comprehensive progress of the discipline.

Acknowledgments
This research project is carried out under the support of the Russian Science Foundation Grant № 19-19-00623.

References
[1] J. Clark, C. Clarke, S. De Panfilis, G. Granatella, P. Predonzani, A. Sillitti, G. Succi, and T. Vernazza, “Selecting components in large cots repositories,” Journal of Systems and Software, vol. 73, no. 2, pp. 323–331, 2004.
[2] J. Escobar-Avila, M. Linares-Vázquez, and S. Haiduc, “Unsupervised software categorization using bytecode,” in 2015 IEEE 23rd International Conference on Program Comprehension. IEEE, 2015, pp. 229–239.
[3] V. Ivanov, A. Kruglov, A. Sadovykh, and G. Succi, “Scenarios for the evaluation of the energy efficiency of mobile applications,” in 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). IEEE, Oct. 2019. [Online]. Available: https://doi.org/10.1109/iemcon.2019.8936235
[4] P. Ciancarini, S. Ergasheva, Z. Kholmatova, A. Kruglov, G. Succi, X. Vasquez, and E. Zuev, “Analysis of Energy Consumption of Software Development Process Entities,” Electronics, vol. 9, no. 10, p. 1678, 2020.
[5] L. Corral, A. Sillitti, G. Succi, A. Garibbo, and P. Ramella, “Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm,” in Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, ser. Onward! 2011. New York, NY, USA: ACM, 2011, pp. 181–183.
[6] L. Corral, A. B. Georgiev, A. Sillitti, and G. Succi, “A method for characterizing energy consumption in Android smartphones,” in Green and Sustainable Software (GREENS 2013), 2nd International Workshop on. IEEE, May 2013, pp. 38–45.
[7] ———, “Can execution time describe accurately the energy consumption of mobile apps? An experiment in Android,” in Proceedings of the 3rd International Workshop on Green and Sustainable Software. ACM, 2014, pp. 31–37.
[8] D. Atonge, V. Ivanov, A. Kruglov, I. Khomyakov, A. Sadovykh, D. Strugar, G. Succi, X. Z. Vasquez, and E. Zuev, “The development of data collectors in open-source system for energy efficiency assessment,” in IFIP Advances in Information and Communication Technology. Springer International Publishing, 2020, pp. 14–24. [Online]. Available: https://doi.org/10.1007/978-3-030-47240-5_2
[9] S. Ergasheva, D. Strugar, A. Kruglov, and G. Succi, “Energy efficient software development process evaluation for MacOS devices,” in *IPIP Advances in Information and Communication Technology*. Springer International Publishing, 2020, pp. 196–206. [Online]. Available: https://doi.org/10.1007/978-3-030-47240-5_10

[10] J. Dinkelacker, P. K. Garg, R. Miller, and D. Nelson, “Progressive open source,” in *Proceedings of the 24th International Conference on Software Engineering*, 2002, pp. 177–184.

[11] M. Linares-Vásquez, C. McMillan, D. Poshyvanyk, and M. Grechanik, “On using machine learning to automatically classify software applications into domain categories,” *Empirical Software Engineering*, vol. 19, no. 3, pp. 582–618, 2014.

[12] A. Valerio, G. Succi, and M. Fenaroli, “Domain analysis and framework-based software development,” *SIGAPP Appl. Comput. Rev.*, vol. 5, no. 2, pp. 4–15, Sep. 1997.

[13] P. Musilek, W. Pedrycz, N. Sun, and G. Succi, “On the Sensitivity of COCOMO II Software Cost Estimation Model,” in *Proceedings of the 8th International Symposium on Software Metrics*, ser. METRICS ‘02. IEEE Computer Society, June 2002, pp. 13–20.

[14] W. Pedrycz and G. Succi, “Genetic granular classifiers in modeling software quality,” *Journal of Systems and Software*, vol. 76, no. 3, pp. 277–285, 2005.

[15] M. Ronchetti, G. Succi, W. Pedrycz, and B. Russo, “Early estimation of software size in object-oriented environments a case study in a cmm level 3 software firm,” *Information Sciences*, vol. 176, no. 5, pp. 475–489, 2006.

[16] R. Moser, W. Pedrycz, and G. Succi, “Analysis of the reliability of a subset of change metrics for defect prediction,” in *Proceedings of the Second ACM-IEEE International Symposium on Software Engineering and Measurement*, ser. ESEM ’08. ACM, 2008, pp. 309–311.

[17] B. Rossi, B. Russo, and G. Succi, “Modelling Failures Occurrences of Open Source Software with Reliability Growth,” in *Open Source Software: New Horizons - Proceedings of the 6th International IFIP WG 2.13 Conference on Open Source Systems, OSS 2010*. Notre Dame, IN, USA: Springer, Heidelberg, May 2010, pp. 268–280.

[18] L. Corral, A. Sillitti, and G. Succi, “Software Assurance Practices for Mobile Applications,” *Computing*, vol. 97, no. 10, pp. 1001–1022, Oct. 2015.

[19] S. Ergasheva, I. Khomyakov, A. Kruglov, and G. Succi, “Metrics of energy consumption in software systems: a systematic literature review,” *IOP Conference Series: Earth and Environmental Science*, vol. 431, p. 012051, Feb. 2020. [Online]. Available: https://doi.org/10.1088/1755-1315/431/1/012051

[20] J. Kivi, D. Haydon, J. Hayes, R. Schneider, and G. Succi, “Extreme programming: a university team design experience,” in *2000 Canadian Conference on Electrical and Computer Engineering. Conference Proceedings. Navigating to a New Era (Cat. No.00TH8439)*, vol. 2, May 2000, pp. 816–820 vol.2.

[21] G. Succi, W. Pedrycz, M. Marchesi, and L. Williams, “Preliminary analysis of the effects of pair programming on job satisfaction,” in *Proceedings of the 3rd International Conference on Extreme Programming (XP)*, May 2002, pp. 212–215.

[22] G. Succi, E. Benedicenti, and T. Vernazza, “Analysis of the effects of software reuse on customer satisfaction in an RPG environment,” *IEEE Transactions on Software Engineering*, vol. 27, no. 5, pp. 473–479, 2001.

[23] A. Sillitti, T. Vernazza, and G. Succi, “Service Oriented Programming: A New Paradigm of Software Reuse,” in *Proceedings of the 7th International Conference on Software Reuse*. Springer Berlin Heidelberg, April 2002, pp. 269–280.

[24] W. Pedrycz, B. Russo, and G. Succi, “A model of job satisfaction for collaborative development processes,” *Journal of Systems and Software*, vol. 84, no. 5, pp. 739–752, 2011.

[25] ——, “Knowledge Transfer in System Modeling and Its Realization Through an Optimal Allocation of Information Granularity,” *Appl. Soft Comput.*, vol. 12, no. 8, pp. 1985–1995, Aug. 2012.

[26] A. Sillitti, G. Succi, and J. Vlasenko, “Understanding the Impact of Pair Programming on Developers Attention: A Case Study on a Large Industrial Experimentation,” in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE ‘12. Piscataway, NJ, USA: IEEE Press, June 2012, pp. 1094–1101.

[27] I. D. Coman, P. N. Robillard, A. Sillitti, and G. Succi, “Cooperation, collaboration and pair-programming: Field studies on backup behavior,” *Journal of Systems and Software*, vol. 91, pp. 124–134, 2014.

[28] S. Kawaguchi, P. K. Garg, M. Matsushita, and K. Inoue, “Mudablue: An automatic categorization system for open source repositories,” *Journal of Systems and Software*, vol. 79, no. 7, pp. 939–953, 2006.

[29] J. I. Maletic and A. Marcus, “Using latent semantic analysis to identify similarities in source code to support program understanding,” in *Proceedings 12th IEEE Internationals Conference on Tools with Artificial Intelligence. ICTAI 2000*. IEEE, 2000, pp. 46–53.

[30] A. Kuhn, S. Ducasse, and T. Girba, “Semantic clustering: Identifying topics in source code,” *Information and software technology*, vol. 49, no. 3, pp. 230–243, 2007.

[31] G. Maskeri, S. Sarkar, and K. Heafield, “Mining business topics in source code using latent dirichlet allocation,” in *Proceedings of the 1st India software engineering conference*, 2008, pp. 113–120.
