RepoSkillMiner: Identifying software expertise from GitHub repositories using Natural Language Processing

Stratos Kourtzanidis
Department of Applied Informatics,
University of Macedonia
54006 Thessaloniki, Greece
ekourtzanidis@uom.edu.gr

Alexander Chatzigeorgiou
Department of Applied Informatics,
University of Macedonia
54006 Thessaloniki, Greece
achat@uom.edu.gr

Apostolos Ampatzoglou
Department of Applied Informatics,
University of Macedonia
54006 Thessaloniki, Greece
a.ampatzoglou@uom.edu.gr

ABSTRACT
A GitHub profile is becoming an essential part of a developer's resume enabling HR departments to extract someone's expertise, through automated analysis of his/her contribution to open-source projects. At the same time, having clear insights on the technologies used in a project can be very beneficial for resource allocation and project maintainability planning. In the literature, one can identify various approaches for identifying expertise on programming languages, based on the projects that developer contributed to. In this paper, we move one step further and introduce an approach (accompanied by a tool) to identify low-level expertise on particular software frameworks and technologies apart, relying solely on GitHub data, using the GitHub API and Natural Language Processing (NLP)—using the Microsoft Language Understanding Intelligent Service (LUIS). In particular, we developed an NLP model in LUIS for named-entity recognition for three (3) .NET technologies and two (2) front-end frameworks. Our analysis is based upon specific commit contents, in terms of the exact code chunks, which the committer added or changed. We evaluate the precision, recall and f-measure for the derived technologies/frameworks, by conducting a batch test in LUIS and report the results. The proposed approach is demonstrated through a fully functional web application named RepoSkillMiner.

Tool Links:

Video, Code Repo, Application, Validation Dataset

CCS CONCEPTS
• Software and its engineering → Software creation and management → Software post-development issues;

KEYWORDS
Expertise; Frameworks; GitHub; Natural Language Processing; Software Project Management;

ACM Reference Format:

S. Kourtzanidis, A. Chatzigeorgiou, and A. Ampatzoglou. 2020. "RepoSkillMiner: Identifying software expertise from GitHub repositories using Natural Language Processing", In *Proceedings of ACM Automated Software Engineering Conference (ASE' 20). ACM, USA, 4 pages.* https://doi.org/

ASE '20, September 21–25, 2020, Virtual Event, Australia
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6768-4/20/09 $15.00
https://doi.org/10.1145/3324884.3415305

1. INTRODUCTION
Contemporary software development demands breadth and in-depth knowledge for a tremendously large set of technologies, tools and practices [1][2]. The challenge of retaining a competitive development team becomes even more pronounced considering the rapid evolution of software technologies and the continuous emergence of novel programming languages, frameworks and libraries. Considering that developers' skills also evolve over time, **project managers face the challenging task of tracking the kind and extent of knowledge that exists within the development team, to efficiently map people to tasks**. Quite often, a particular technology has been introduced into a project following rising trends, only to find out a few years later that few people in the team (or even none) still hold the corresponding knowledge to maintain the corresponding parts of the codebase. At the same time **identifying developers' expertise and skills is a topic of high importance among recruiters and human resource departments of software companies**. With the plethora of frameworks and technologies involved in software engineering (e.g., front-end technologies) a stated set of known skills is not sufficient to perform efficient hiring. Social platforms such as LinkedIn are gaining ground and constitute a major field for IT recruiters to find ideal candidates, but the information posted on such platforms may be inaccurate, out of date and subjective. To this end, identifying expertise from artifacts can be a useful alternative for organizations, not only for hiring purposes, but also for **managing their existing resources, maintaining and developing their skillset and monitoring any potential skill shortage**.

Artifact-based expertise identification offers a promising alternative to address such challenges. Software repositories such as GitHub or BitBucket can provide objective insights for an experienced technical eye. However, parsing through repositories to track individual contributions and then analyzing commits to derive the particular technologies that a developer is familiar with, is a tedious process urging for automation. At the same time, similar approaches are needed to automatically analyze the technologies used over time within a particular project and the diffusion of each technology within code. Related literature on identifying expertise from software development platforms, focuses on the programming languages used in each project [3][4][8]. However, considering only the dominant language of a project as an identifier can often lead to inaccurate results for today's multi-language projects. Thus, information about the multiple frameworks and technologies used in the context of a language is slipping under the radar.

In this paper we propose an approach (and tool) to advance the state-of-the-art on mining skill-related information from coding platforms with a hybrid approach aided by NLP. In particular, first we obtain an insight based on the files contained in each commit; and next we apply NLP on the actual code chunks included in the commits to identify the use of specific frameworks and technologies. The outcome of this approach is a report containing all the

programming languages, technologies and frameworks per contributor. To showcase the approach, we developed a web application, which performs the analysis and visualizes the results. In contrast to many existing approaches we use live data from GitHub, rather than relying on pre-existing curated datasets (e.g., GHTorrent)—increasing the applicability of the approach. The described tool constitutes our initial effort to develop a comprehensive dashboard that will provide to project managers an overview of actual technologies employed in a project as well as detailed reporting on the skills held by active developers, including the evolution of both over time. The envisioned platform will enable the early identification of gaps between people's skills and deployed technologies.

2. RELATED WORK

Relevant studies span across 3 different areas, namely mining software repositories, extracting software developer's expertise, and Natural Language Processing. Several studies introduce approaches and methods of extracting expertise from software repositories. Gousios et al. [3] propose a way to quantify developer's contribution by identifying specific developers' contribution types: e.g., adding code of good / bad quality, commits of new source files or directories, commits of code that generates / closes bugs and assign different weights to them. Constantinou et al. [4] examined the commit activity from the curated dataset GHTorrent [5] and proposed a way to extract developers' expertise in programming languages by considering the quantity and the continuity of contributions. The programming language in this case is the project's dominant language as identified by GitHub. In a different approach [6] the same authors proposed a way to identify contributors' expertise and roles, considering their contribution history across projects and technologies. The technologies are identified by the contents of the README file, assuming that this file typically contains description of the technologies used. Likewise, Greene et al. [7] combined commit details and readme files to extract similar information. A different path is followed by Montandon et al. [8], who focus on 3 JavaScript libraries to evaluate the performance of machine learning classifiers to predict expertise and to propose a method on clustering feature data from GitHub. Using NLP techniques on source code is a topic related mostly with static code analysis [9]. To the best of our knowledge there is no study about using NLP for expertise identification in source code.

3. BACKGROUND INFORMATION

The proposed approach relies on three main pillars: parsing (mining) of software repositories to seek code artifacts for analysis, NLP that treats code through statistical analysis; and Language Understanding to pull out of code the relevant information on the involved technologies.

Mining Software Repositories (MSR) has become an established field in empirical software engineering, focusing on extracting and analyzing data drawn from software repositories to reveal useful relations and information around software products, processes and people [10]. GitHub is by far the most popular social coding platform and most of the similar research efforts rely on GitHub, as reference source of data [11]. For the purpose of facilitating mining and increasing performance, curated datasets such as GHTorrent [5] and Boa are being maintained by research teams. The availability of a comprehensive API was one of the key reasons that render GitHub an appealing source for many software engineering research efforts [14]. However, there are several research studies pointing out the pitfalls of this process [15].

Natural language processing (NLP) leverages the power of machine learning and computational linguistics and is concerned with making computer systems learn the syntax and meaning of human language, process and understand the intent of it to perform meaningful tasks [16]. Applying NLP techniques on source code may sound unnatural, but there is scientific evidence supporting the validity of the approach. Hindle et al. [17] suggest that programming languages, in theory, are complex, flexible and powerful, but, "natural" programs that real people actually write, are mostly simple and rather repetitive; thus they have predictable statistical properties that can be captured in language models and leveraged for software engineering tasks. This approach has inspired us to use NLP on source code for expertise identification instead of crafting rules so that the proposed approach is scalable, benefit from the abundance of available data in software repositories to train the corresponding models and better cope with new situations.

Microsoft's Language Understanding Intelligent Service (LUIS) is based on work by Microsoft Research on interactive learning, and rapid development of language understanding models [18]. According to its creators it aims at enabling software developers to create cloud-based machine-learning language understanding models specific to their application domain, without ML expertise [19]. The model creator needs to provide a small set of utterances for each intent and the LUIS model is trained based on these and after it's published it is ready for use. Successful industry applications of LUIS include information chatbots, commerce chatbots and conversational IoT interfaces [20].

4. PROPOSED APPROACH

The proposed approach can be briefly outlined in the following steps:

Step 1. Data Collection. We used GitHub's REST API ver. 3 [21] to retrieve up-to-date data from GitHub repositories. Given a GitHub organization or repository we retrieve all commits as well as the authors of each commit. For every commit we retrieve the files included and the actual code chunks for these files. We preferred API ver. 3, over its successor ver. 4, because although the latest uses GraphQL, up to now there is no way to retrieve file contents using this version of the API.

Step 2. Identification of commits' programming language. For each retrieved commit we check the files included in the commit and in particular the file extensions to identify the employed programming languages. To do so, we use a slightly modified (removed duplicates and added a few more file extensions) version of the classification provided by GitHub Linguist [22], which is the library that GitHub uses for providing the language distribution information for the repositories. Both original and the modified classification file are available online[1,2].

Step 3. Identification of commit technologies with LUIS. We built a model in LUIS to identify three (3) technologies in the .NET framework domain, namely: (a) *Language-Integrated Queries (LINQ)* which are first-class language constructs that allow writing of queries against strongly typed collections of objects; (b) *Asynchronous Programming* that allows code in the form of sequential statements which however executes based on external resource allocation and according to the order of tasks; and (c) *Entity Framework* which is an object-database mapper. Moreover, LUIS is trained to identify two (2) front-end frameworks, namely: (d) *Angular* and (e) *React*. LUIS needs input utterances (i.e., inputs from the user that the model needs to interpret) to be provided for each target intent (technology/framework) in the training step. We note that an intent corresponds to a purpose or goal expressed in a user's utterance. To train

[1] https://github.com/github/linguist/blob/master/lib/linguist/languages.yml

[2] https://1drv.ms/u/s!Ak-X7Vy5IBQ3i_kwlqLNwVZS9_DO_A?e=Tn0Oqx

LUIS to extract intents and entities it is important to capture a variety of example utterances. Active learning, or the process of continuing to train on new utterances, is essential to machine-learned intelligence that LUIS provides. We have created 98 example utterances for the 5 intents using existing or slightly altered (mainly altered variable names) existing samples from the official Microsoft, Angular and React documentation. An example response from LUIS for the utterance `"var filteredResult = studentList.Where(s => s.Age > 12)"` is shown in Figure 1.

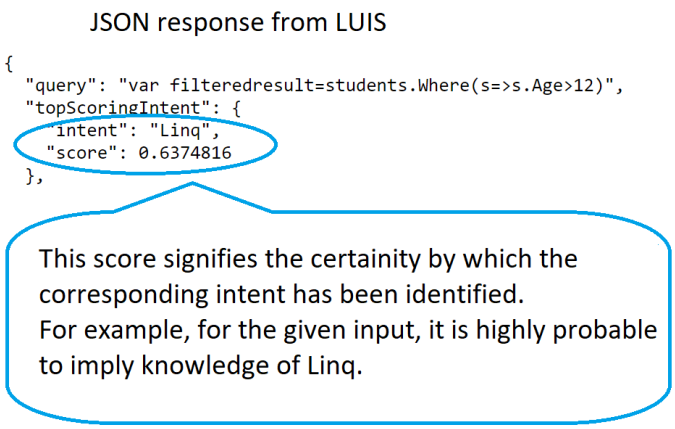

Figure 1: Example JSON response

Table 1 displays model statistics for the five (5) target intents (Linq, Entity Framework, Async Programming, Angular, React), whereas the scatterplot in Figure 2 shows the distribution of the test results for Linq. To perform the evaluation, we created a batch test in LUIS testing platform with 88 code snippets, which are available online[3]. Additionally, we note that the model has been exported and is available online[4].

Table 1: Model Statistics

Intent	*Precision*	*Recall*	*F-Measure*
Linq	1.00	0.82	0.90
Asynchronous Programming	0.94	0.94	0.94
Entity Framework	1.00	1.00	1.00
Angular	0.92	1.00	0.96
React	1.00	0.83	0.91

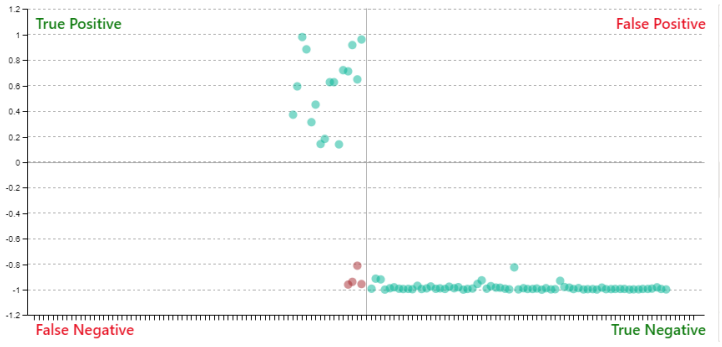

Figure 2: Model statistics for entity Linq

[3] https://1drv.ms/u/s!Ak-X7Vy5IBQ3i_kM4CLWuUg6_UzTsw?e=3lqg5c

[4] https://1drv.ms/u/s!Ak-X7Vy5IBQ3i_kLc8sc8wdPU6kLaA?e=NIXM5t

5. TOOL ARCHITECTURE

Figure 3 presents the architecture of the tool, namely **RepoSkillMiner**, consisting of a Blazor WebAssembly application, which consumes two services: GitHub API and LUIS. The user enters the search criteria (organization, repositories) to the Blazor WebAssembly app. The Blazor App makes the necessary calls to Github Rest API to gather the commit data (including authors and code chunks), processes them and extracts the employed technologies for each author. Then, it makes the calls to the LUIS services using the data collected during the previous step.

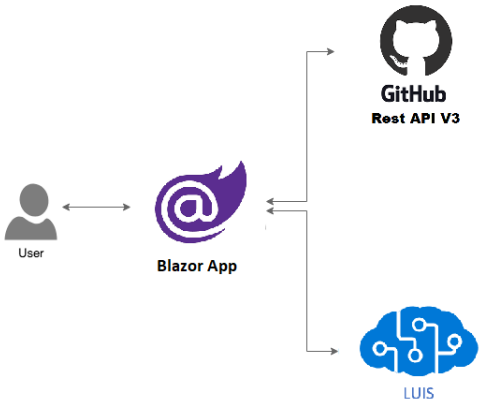

Figure 3: RepoSkillMiner Architecture

6. USAGE SCENARIO

A video demonstrating the functionality of RepoSkillMiner is available online[5]. The demonstration begins with the user entering the name of the GitHub organization to be analyzed in the search field shown in Figure 4. Next, he/she selects from the dropdown list the repository he/she wants to include for scanning and specifies that he/she wants to apply LUIS scanning and clicks the scan button. A table is populated containing the contributors of the organization and the technologies they used as shown in Figure 5. By clicking on the name of any of the contributors two graphs are populated showing the technologies distribution based on the number of commits for each technology and a detailed list of the technologies as shown in Figures 6-7.

7. CONCLUSIONS AND FUTURE WORK

RepoSkillMiner is a scalable and easy-to-use web application that can determine the knowledge (in terms of low-level skills—e.g., the command of specific programming techniques or frameworks), held by individual developers in an open-source software project. Since the tool is still in a prototype phase, the application has some limitations including browser support (Firefox is the only fully supported one) and API limitations (GitHub API sets a limit to 5000 requests per hour). We plan to include more detailed visualizations and in-depth insights from the collected data including the evolution of project technologies within an organization over time, the designation of technologies which are "at risk" because of lack of resources, the cross-tabulation of technologies and people's skills, etc. Developers' experience in terms of commits or years can also be derived by analyzing a projects' history. Once the tool is enhanced with the ability to detect more low-level technologies, we plan to evaluate its

[5] https://youtu.be/E_IeLprGlCc

accuracy against the actual skills held by developers in a selected company, using a questionnaire-based study.

ACKNOWLEDGMENT

Work reported in this paper has received funding from the European Union Horizon 2020 research and innovation program under grant agreement No. 780572 (project: SDK4ED).

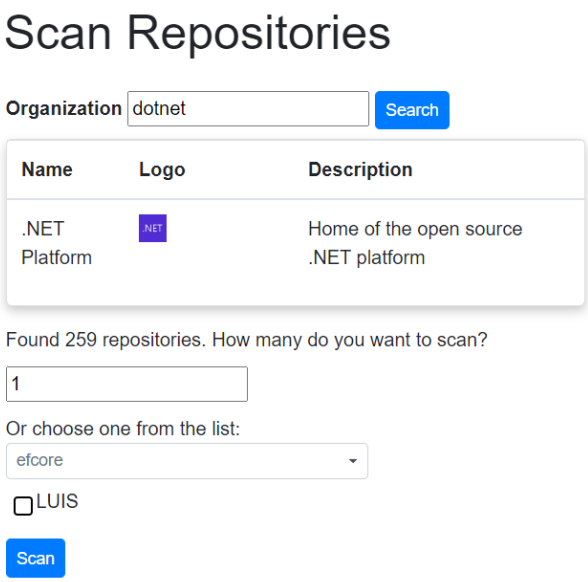

Figure 4: Search for a repository in an organization

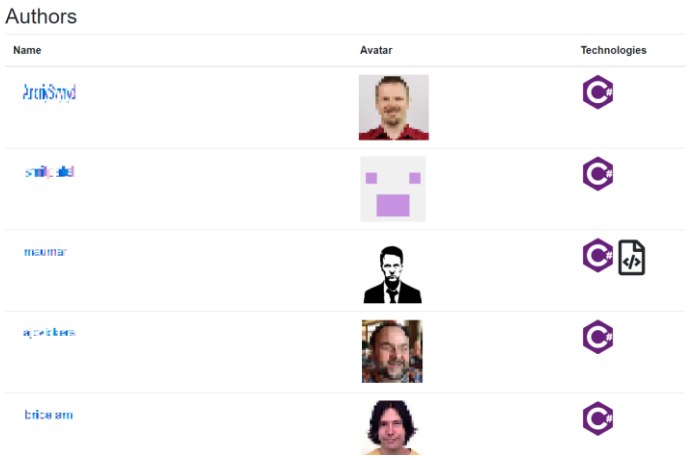

Figure 5: Search results

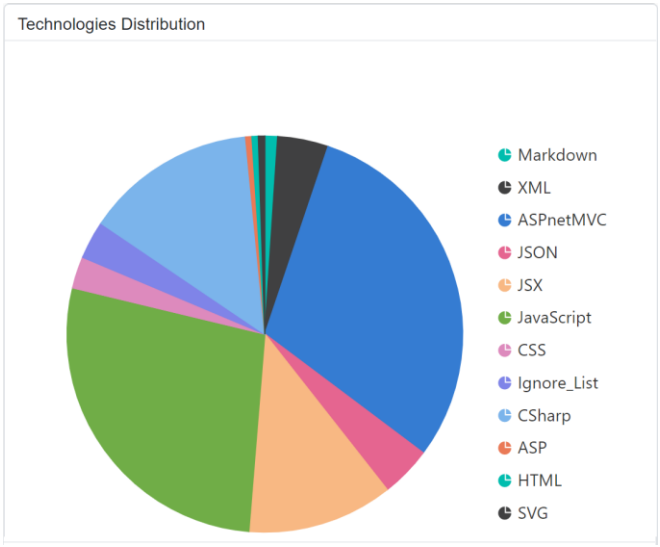

Figure 6: Pie chart of technology distribution

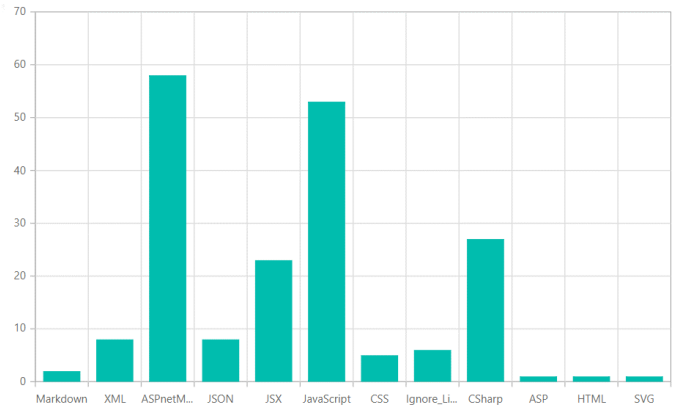

Figure 7: Bar chart of commits per technology.

REFERENCES

[1] A. Pano, D. Graziotin, and P. Abrahamsson. 2018. *Factors and actors leading to the adoption of a JavaScript framework. Empirical Software Engineering.* https://doi.org/10.1007/s10664-018-9613-x

[2] A. Pang, C. Anslow, and J. Noble. 2018. What programming languages do developers use? A theory of static vs dynamic language choice. *IEEE Symposium on Visual Languages and Human-Centric Computing*, https://doi.org/10.1109/VLHCC.2018.8506534

[3] G. Gousios, E. Kalliamvakou, and D. Spinellis. 2008. Measuring developer contribution from software repository data. In *Proceedings - International Conference on Software Engineering*: 129–132. https://doi.org/10.1145/1370750.1370781

[4] E. Constantinou and G. M. Kapitsaki. 2016. Identifying Developers' Expertise in Social Coding Platforms. In *Proceedings of the 42nd Euromicro Conference on Software Engineering and Advanced Applications, SEAA* 2016. https://doi.org/10.1109/SEAA.2016.18

[5] G. Gousios. 2013. The GHTorent dataset and tool suite The GHTorent Dataset and Tool Suite. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, June: 233–236. https://doi.org/10.1109/MSR.2013.6624034

[6] E. Constantinou and G. M. Kapitsaki. 2016. Developers Expertise and Roles on Software Technologies. In *Proceedings of the 23rd Asia-Pacific Software Engineering Conference (APSEC), Hamilton*, 2016, 365-368. https://doi.org/10.1109/APSEC.2016.061

[7] G. J. Greene and B. Fischer. 2016. CVExplorer: Identifying candidate developers by mining and exploring their open source contributions. ASE 2016 – In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*: 804–809. https://doi.org/10.1145/2970276.2970285

[8] J. E. Montandon, Luciana Lourdes Silva, and Marco Tulio Valente. 2019. Identifying experts in software libraries and frameworks among GitHub Users. In *Proceedings of IEEE International Working Conference on Mining Software Repositories 2019-May*: 276–287. https://doi.org/10.1109/MSR.2019.00054

[9] B. Lin, S. Scalabrino, A. Mocci, R. Oliveto, G. Bavota, and M. Lanza. 2017. Investigating the Use of Code Analysis and NLP to Promote a Consistent Usage of Identifiers. In *Proceedings of the 17th International Working Conference on Source Code Analysis and Manipulation*, Oct. 2017: 81–90. https://doi.org/10.1109/SCAM.2017.17

[10] T. Siddiqui and A. Ahmad. 2018. Data mining tools and techniques for mining software repositories: A systematic review, Advances in Intelligent Systems and Computing, volume 654. 717–726. AISC. https://doi.org/10.1007/978-981-10-6620-7_70

[11] M. A. F. De Farias, M. Colaço, M. Mendonça, R. Novais, L. P. D. S. Carvalho, and R. O. Spínola. 2016. A systematic mapping study on mining software repositories. *ACM Symposium on Applied Computing, 1472–1479.* https://doi.org/10.1145/2851613.2851786

[12] V. Cosentino, J. Luis, C. Izquierdo, and J. Cabot. 2016. Findings from GitHub: Methods, datasets and limitations. In *Proceedings - 13th Working Conference on Mining Software Repositories*, MSR 2016, 137–141. https://doi.org/10.1145/2901739.2901776

[13] R. Dyer, H. Anh Nguyen, T. N Nguyen, and H. Rajan. 2015. Boa: Ultra-Large-Scale Software Repository and Source Code Mining. ACM Transactions on Software Engineering and Methodology, Volume 25, Issue 1. https://doi.org/10.1145/2803171

[14] G. Gousios and D. Spinellis. 2017. Mining software engineering data from GitHub. In *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion*, ICSE-C 2017, 501–502. https://doi.org/10.1109/ICSE-C.2017.164

[15] C. Bird, P. C Rigby, E. T Barr, D. J Hamilton, D. M German, and P. Devanbu. The Promises and Perils of Mining Git. In *Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories*, 2009. https://doi.org/10.1109/MSR.2009.5069475

[16] A. Jain, G. Kulkarni, and V. Shah. 2018. Natural Language Processing. International Journal of Computer Sciences and Engineering 6, 1: 161–167. https://doi.org/10.26438/ijcse/v6i1.161167

[17] A. Hindle, E. T. Barr, M. Gabel, Z. Su, and P. Devanbu. 2016. On the naturalness of software. Communications of the ACM 59, 5: 122–131. https://doi.org/10.1145/2902362.

[18] J. D. Williams, N. B. Niraula, P. Dasigi, A. Lakshmiratan, C. G. J. Suarez, M. Reddy, and G. Zweig. 2015. Rapidly scaling dialog systems with interactive learning. Natural Language Dialog Systems and Intelligent Assistants: 1–13. https://doi.org/10.1007/978-3-319-19291-8_1

[19] J. D. Williams, E. Kamal, M. Ashour, H. Amr, J. Miller, and G. Zweig. 2015. Fast and easy Language Understanding for dialog systems with Microsoft Language Understanding Intelligent Service (LUIS). SIGDIAL 2015 - 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Proceedings of the Conference, September: 159–161. https://doi.org/10.18653/v1/w15-4622.

[20] Microsoft 2020. Language Understanding (LUIS) Retrieved from https://www.luis.ai/home

[21] GitHub REST API v3. https://developer.github.com/v3/

[22] Linguist. Retrieved from https://github.com/github/linguist/