Analyzing programming languages by community characteristics on Github and StackOverflow

Samarth Tambad
Courant Institute of Mathematical Sciences
New York University
New York, NY, USA
svt258@nyu.edu

Rohit Nandwani
Courant Institute of Mathematical Sciences
New York University
New York, NY, USA
rhn235@nyu.edu

Suzanne K. McIntosh
Courant Institute of Mathematical Sciences
New York University
New York, NY, USA
mcintosh@cs.nyu.edu

***Abstract*—The choice of programming language is a very important decision as it not only affects the performance and maintainability of the software but also dictates the talent pool and community support available. To better understand the trade-offs involved in making such a decision, we define and compute popularity, demand, availability and community engagement of programming languages through online collaboration platforms. We perform our analysis using data from Github and StackOverflow, two of the most popular programming communities. We get data related projects, languages and developer engagement from Github and programming questions with answers along with language tags from StackOverflow. We compute metrics separately for the two data sources and then combine the metrics to provide a holistic and robust picture of the communities for the most popular programming languages.**

***Index Terms*—github, stackoverflow, analysis, community engagement, popularity, availability, demand**

I. Introduction

Software development and maintenance is a complex activity involving many important decisions that need to be made. The choice of programming language is one such decision. From the perspective of the managers of the software projects, this decision not only affects the performance of the software but also dictates the talent pool and community support available. From the perspective of the developer, it dictates the current job opportunities and their future career trajectory.

We analyse the popularity and the community friendliness of programming languages and estimate the availability and demand of developers proficient in them. For our analysis, we look at data from Github and StackOverflow, two of the most popular programming communities.

Github is a platform for collaborative software development. Data gathered from this platform is suitable for measuring the popularity of languages and availability/demand of developers. Particularly the information available about repositories such as languages used and contributions made by developers is useful.

StackOverflow is a popular online programming Q&A community providing its participants with rapid access to knowledge and expertise of their peers. The community support is a valuable tool for developers in any programming language. Therefore, a more open, welcoming and responsive (i.e. friendly) community is a good thing to have in order to be more productive as a developer. Data such as the questions asked and the quality and time-frame of the response is a good indicator of the "friendliness" of a particular programming community.

We combine the data gathered from the two sources to compute the metrics of popularity, community engagement, availability and demand. These metrics provide a holistic view of the pros/cons of different languages. We then use these metrics to compare different languages and help answer questions such as: which is the first language I should learn?, which language is most in demand right now?, suggest an alternative language because I work with x language but the community support is bad, etc.

The remainder of this paper is organised as follows: we describe our motivation in Section II followed by a survey of related work in Section III. In Section IV, we provide a detailed description of the datasets used. We describe our analytic in Section V followed by application design in Section VI. In Section VII we describe our experimental setup and analysis of results. In Section VIII we provide our conclusions and provide scope for future work in Section IX.

II. Motivation

Open source has been gaining popularity among the developer community. Increasingly, many companies are also realising the benefit of contributing to open source projects which may benefit their business directly or indirectly [1]. Also, developers are increasingly realising the benefit of contributing to open source. Therefore, analysis on the open source developer community is good proxy for the developer community in general.

While choosing a programming language to learn or build a project, it is important to understand the characteristics and strengths of the landscape of programming languages. At the same time, it is critical to have an active and cooperative community for the programming language under consideration to speed up the learning and building process. We find that there is a lack of research on the latter aspect, which combines data from multiple available sources.

The choice of language based on community has a massive impact on the levels of productivity for the developer and the company [2], performance of the applications, and the overall satisfaction of the development process [3]. It will also result in increased demand for the developers in the language with better community characteristics.

There are multiple studies mapping developer productivity and satisfaction, to the profitability of the company [3] [4]. Programming languages can also have a major impact in the career trajectories and overall satisfaction of developers.

III. Related Work

In this section, we describe the related work compiled from the literature.

A. Analysis using Github data

Ray, B. et al [5], perform a large scale study on the quality of code with respect to programming languages using text mining and regression techniques. They find that there is a significant correlation between the two.

Kalliamvakou et al [6] describe the perils on mining data on GitHub. They point that inactive account, invisible merges on pull requests, public activity on repositories could cause problems in analysis and how to overcome them.

B. Analysis using StackOverflow data

Jie Yang et al [7] study the characteristics of experts on StackOverflow. They give us important metrics such as the debatableness of a question and the utility of an answer.

Seyed Mehdi Nasehi et al [8] describe what makes a good code example on StackOverflow by analyzing the interactions with code examples.

Blerina Bazelli et al [9] describe the personality traits of successful contributors on StackOverflow including extroversion and negativity.

Gupta, R. et al [10] study reopened questions on StackOverflow, and suggest the editing questions / answers even after acceptance/closing is a good sign of expertise in the community.

Wang, S et al [11] study if the population on StackOverflow can be divided into givers or takers. They also model the types of questions asked using LDA.

C. Analysis using Github and StackOverflow combined

Lee, R. et al [12], compare the developer interests on Github and StackOverflow and suggest a high correlation between the two. This helps us know the differences in proportion of contribution on the two different collaboration platforms.

Badashian, A. S et al [13] provide methods and metrics to measure core contributions, editorial activities and influence on Github and StackOverflow.

Vasilescu, B. et al [14] show how activity on StackOverflow impacts the activity on Github and vice versa.

Tian, Y et al [15] measure the quality of individual contributors on the two platforms and combine the measures across both platforms for each individual.

IV. Datasets

There are two main datasets used in our analysis. Here we describe the datasets in detail along with their schema.

A. Github

Github is a collaborative software development platform that allows code sharing and version control. Developers can perform various activities such as creating, forking or committing to a repository, opening issues or submitting pull requests to contribute someone else's repository. The programming language used is tagged for each repository which is very helpful.

We collected the data from the GH Torrent project [16]. It is a dump of Github usage data over the period ranging from October 2013 to June 2019. The data is separated into multiple tables and stored as csv files. The list of tables is given in Table I and the descriptions for each with their complete schema is given in Tables II - VII.

TABLE I
GITHUB TABLES DATA

Tables	**Schema**	***Description***
projects.csv	Table II	Github project repositories
commits.csv	Table III	A list of all commits on Github.
pull_requests.csv	Table IV	List of pull requests for repos
pull_request_history.csv	Table V	Chronologically ordered list of events on a pull request
issues.csv	Table VI	Issues that have been recorded for a project
issue_events.csv	Table VII	Chronologically ordered list of events on an issue

TABLE II
PROJECTS: 138205530 ROWS

Columns	***Data Type***	***Min***	***Max***	***Distinct***
id	Integer	1	137611262	65296722
owner_id	Integer	1	51697268	10778579
language	String	1	24	376
year	Integer	2007	2019	13

TABLE III
COMMITS: 1353856359 ROWS

Columns	***Data Type***	***Min***	***Max***	***Distinct***
id	Integer	1	1415397637	1353856359
author_id	Integer	1	51697270	21406056
committer_id	Integer	1	51697269	18732264
project_id	Integer	1	137611262	73223832
year	Integer	2007	2019	13

B. StackOverflow

StackOverflow is the largest peer reviewed Q/A system for computer programming. All the data is open source and it is available here. It has data from the year 2008 to 2020. We are primarily interested in the 'Posts' dataset which is all

TABLE IV
PULL REQUESTS: 51730295 ROWS

Columns	*Data Type*	*Min*	*Max*	*Distinct*
id	Integer	6350	64121451	51730295
head_repo_id	Integer	3	137611190	13435966
base_repo_id	Integer	2	137611003	7099278
head_commit_id	Integer	23	1415397634	50517219
base_commit_id	Integer	14	1415395953	36332904
pull_request_id	Integer	1	231471	189365

TABLE V
PULL REQUEST HISTORY: 134649096 ROWS

Columns	*Data Type*	*Min*	*Max*	*Distinct*
id	Integer	15017368	152227176	134649096
pull_request_id	Integer	6350	64121451	51884523
action	String	6	11	6
actor_id	Integer	1	51697114	4484020
year	Integer	2010	2019	10

the questions, answers and the interactions with them. There are more than 15M posts with total size of 15 GB with over 500 programming languages. We filter the dataset top 50 most frequent programming languages on Github, narrowing the dataset to 12M posts. The dataset also has a score field which is the sum of upvotes and follows ranging from -146 to 24245 for the posts. We narrow our analysis to the columns listed in Table VIII.

V. DESCRIPTION OF ANALYTIC

In this section, we describe our analytic derived from each source and how we combine it.

A. Github

With the Github data, we used the relevant tables listed and described in Table I to compute the following intermediate metrics:

TABLE VI
ISSUES: 98076172 ROWS

Columns	*Data Type*	*Min*	*Max*	*Distinct*
id	Integer	2	110037555	98076172
repo_id	Integer	1	137611003	9498704
issue_id	Integer	0	337847	295187
year	Integer	1970	2019	22

TABLE VII
ISSUE EVENTS: 136108876 ROWS

Columns	*Data Type*	*Min*	*Max*	*Distinct*
event_id	Integer	2	2147483633	124838780
issue_id	Integer	2	110035665	31533578
action	String	6	24	35
year	Integer	1999	2019	21

TABLE VIII
POSTS: 13851898 ROWS

Columns	*Data Type*	*Min*	*Max*	*Distinct*
_Id	Integer	4	60472846	11746372
_OwnerUserId	Integer	1	12987310	2734937
_PostTypeId	Integer	1	1	1
_Score	Integer	-146	24124	1591
_Tag	String	1	16	270
_CreationYear	Integer	2008	2020	13
_AnswerCount	Integer	0	296	89

- **num_users -** total number of new users each year, grouped by associated language.
- **num_projects -** total number of new projects created each year, grouped by associated language.
- **num_commits -** total number of commits made to repositories each year, grouped by associated language.
- **num_pull_requests -** total number of pull requests opened to base repositories each year, grouped by associated language.
- **num_pending_issues -** total number of issues opened in their respective repositories each year that haven't been closed to date, grouped by associated language.

Using a combination of the above analytics, we computed the following metrics in Tableu:

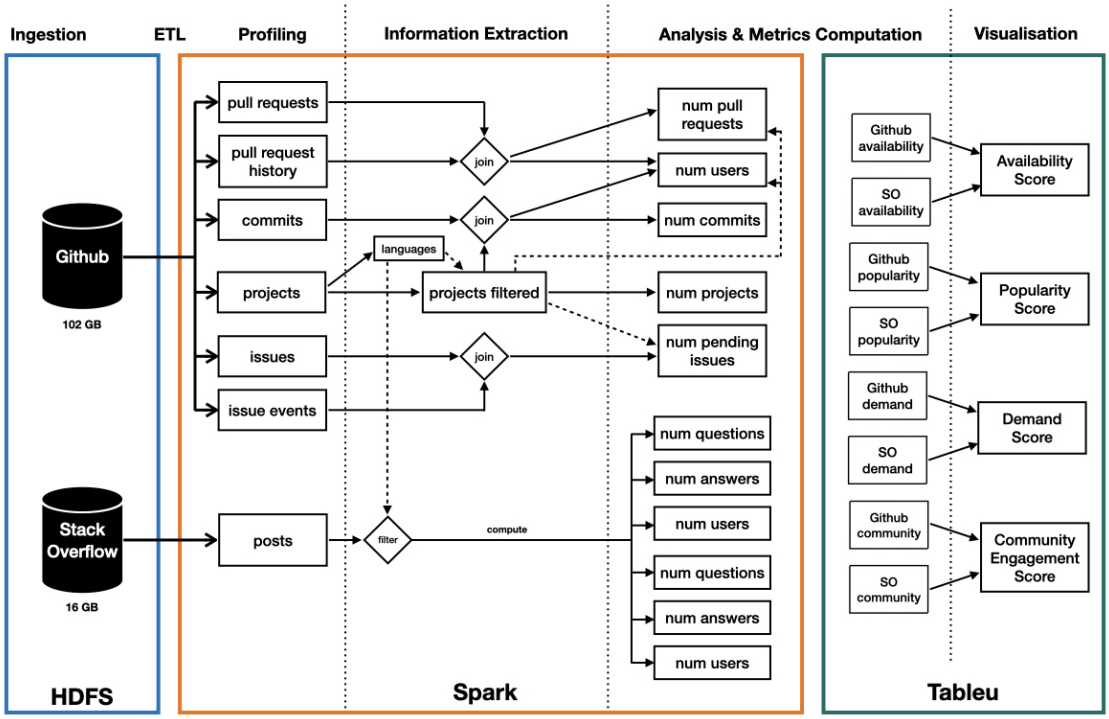
- **GH_Popularity =**
  Mean(num_repos, num_users)
- **GH_Availability =**
  Mean(num_pull_requests / num_projects, num_commits / num_projects)
- **GH_Demand =**
  Mean(num_pending_issues / num_projects)
- **GH_Community =**
  Mean(num_commits / num_projects, num_projects/ num_users, num_commits / num_users)

B. StackOverflow

With the StackOverflow data, we used the posts table described in Table VIII to compute the following intermediate metrics:

- **num_users -** total number of new users each year, grouped by associated language tag.
- **num_questions -** total number of new questions asked each year, grouped by associated language tag.
- **num_answers -** total number of answers to question each year, grouped by associated language tag.
- **total_score -** total score of all posts each year, grouped by associated language tag.
- **num_unanswered_questions -** total number of questions asked each year that haven't received a response, grouped by associated language tag.
- **avg_response_time -** average time taken (in hours) for questions asked to receive a response each year, grouped by associated language tag.

Fig. 1. Architecture design diagram



Using a combination of the above analytics, we computed the following metrics in Tableu:

- **SO_Popularity =**
  Mean(num_questions, num_users)
- **SO_Availability =**
  Mean(num_answers / num_questions)
- **SO_Demand =**
  Mean(num_unanswered_questions / num_questions)
- **SO_Community =**
  Mean(avg_response_time, total_score / num_answers, num_answers / num_users, num_questions / num_users)

C. Combined

We combine the above metrics computed separately for Github and StackOverflow. We get the following combined metrics:

- $Popularity =$
  $0.5 * GH\_Popularity + (1 - 0.5) * SO\_Popularity$
- $Availability =$
  $0.5 * GH\_Availability + (1 - 0.5) * SO\_Availability$
- $Demand =$
  $0.5 * GH\_Demand + (1 - 0.5) * SO\_Demand$
- $Community =$
  $0.5 * GH\_Community + (1 - 0.5) * SO\_Community$

Visualising these metrics provides valuable insights such as: the most popular programming languages, languages that have the most demand shortage, the languages with the most community engagement. It also highlights anomalies in the data that can be explained by real world events that happened in the past that had that effect. For example, we notice that swift rapidly increases in popularity while objective-c rapidly declines during 2013 - 2014. This could be explained due to the launch of swift in place of objective-c for iOS application development.

VI. Application Design

In this section, we describe the overall architecture of our application as shown in Fig. 1.

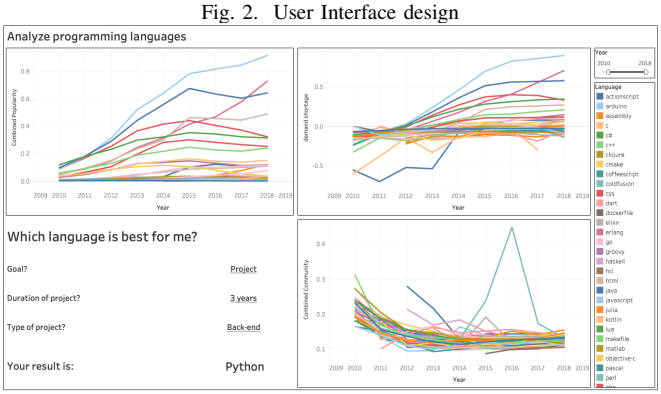
Our data sources are described in section IV. We ingest the data by downloading them into HDFS. The data is a raw dump and therefore we clean the data by dropping unwanted columns, handling null values and formatting special fields such as timestamps, etc. This is a very important step as it minimizes the possibility of run-time exceptions as well as significantly reduces the size of the data which speeds up processing.

After cleaning, we profile the data to understand the characteristics of each column. We identify the data type and measure minimum, maximum as well as number of distinct values. This information is important as it helps us identify which columns can be joined as well as reason if that column needs to be reduced before a join.

Once we understand the data, we proceed to compute the analytic as described in the previous section.

Once the processing is completed as illustrated in Fig. 1, we obtain the analytic separately for each data source. Since the size of this data is quite small, we load it directly onto Tableu for further refinement and visualisation. We then visualize the results in a Tableu dashboard. The final design of our user interface is shown in Fig. 2. We take the goal of selecting the programming language from the user. If the goal is to build a project, we ask the user the duration of the project. If it is learning a language based on the demand shortage, we ask the time horizon of looking for a job. We ask the user the general category of the programming language of interest. Based on this information and based on the metrics we computed, we give the user a ranked list of recommendations of programming languages.

Fig. 2. User Interface design



VII. Analysis

We work with the two data sources separately at first. For both datasets we clean and join the relevant tables together. The major challenge is in joining the separate tables over a common column. This is because the data is huge and directly joining the tables results in the job getting killed due to exceeding the memory limit. For example, joining the commits and projects table over the $project\_id$ column results in the job getting killed because the commits table is huge and the result of the join exceeds memory limit. We tackled this problem by reducing the commits column by $year$ and $project\_id$ which significantly reduces the number of rows. Also, smart use of caching speeds up the process.

We also filter the entire dataset based on the top 50 most popular programming languages found on GitHub.

After computing the intermediate metrics described in section V, we normalize the data using min-max normalisation procedure. We also stationarize the year on year data using first order differencing to remove the effect of increased usage of the Github and SO platforms over time.

We find the following insights based on the metrics computed and the vizualization:

Fig. 3. Languages Legend

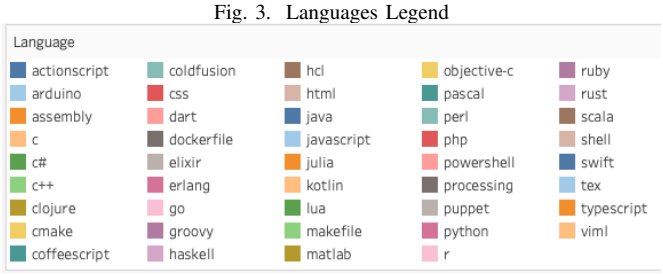
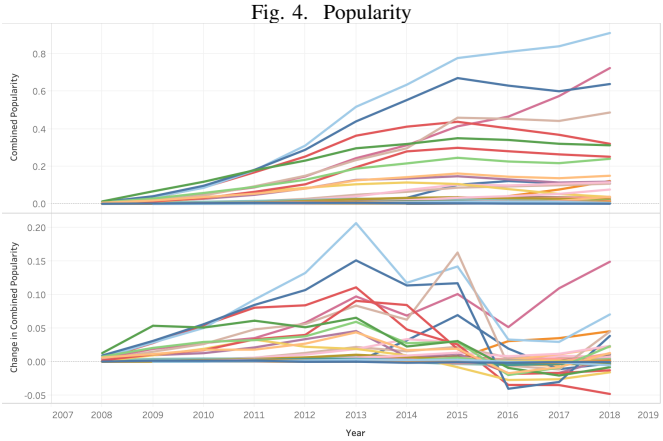


Fig. 4. Popularity



A. Popularity

Unsurprisingly, Javascript and Python are the most popular languages, with Python overtaking Java in 2017. Web Languages including Javascript, HTML, CSS along with Python are increasing the most in popularity. Go and Dart are the languages to watch out for.

B. Demand - Availability

The most popular languages such as JS, Python and Java are also the ones which have the biggest demand shortage on SO and Github. Amongst the top are PHP and C# whose popularity has been on a degrading over the last few years. Go and Dart have the biggest rise in demand shortage, along with Python and JS.

C. Community Engagement

Ruby, Go, Julia, Dart and Rust have the most engaged online communities.

VIII. Conclusion

We analyse community characteristics of programming languages. In order to do so, we use data from Github and StackOverflow. We compute metrics intermediate metrics grouped by programming languages over the years. We use a combination of these metrics as a proxy for measures for qualities such as popularity, availability, demand and community engagement. We then visualise these qualities to analyse trends and provide relevant recommendation.

For students just starting to learn programming, exploring popular web based languages such as JavaScript along with

Fig. 5. Popular programming languages

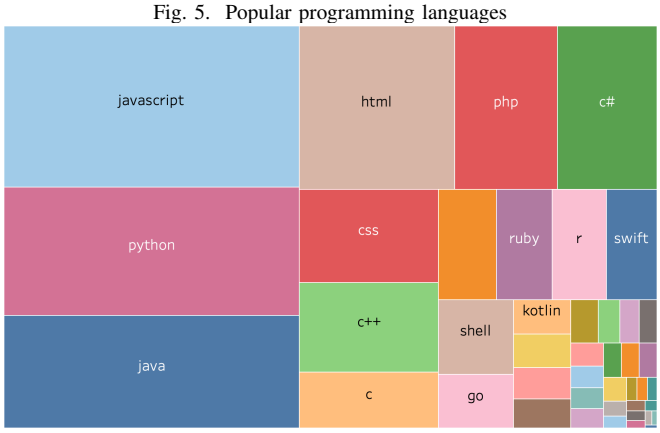
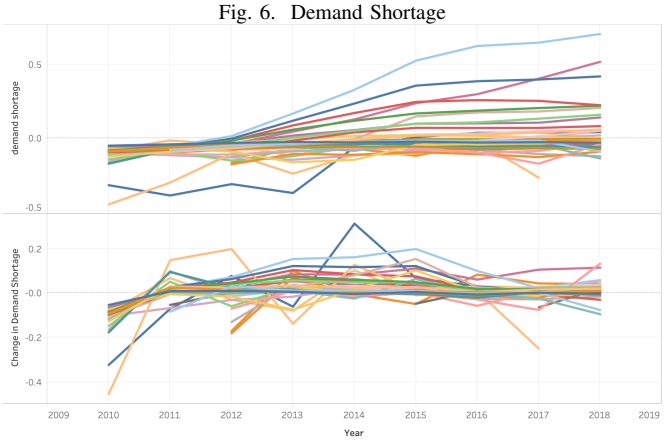


Fig. 6. Demand Shortage



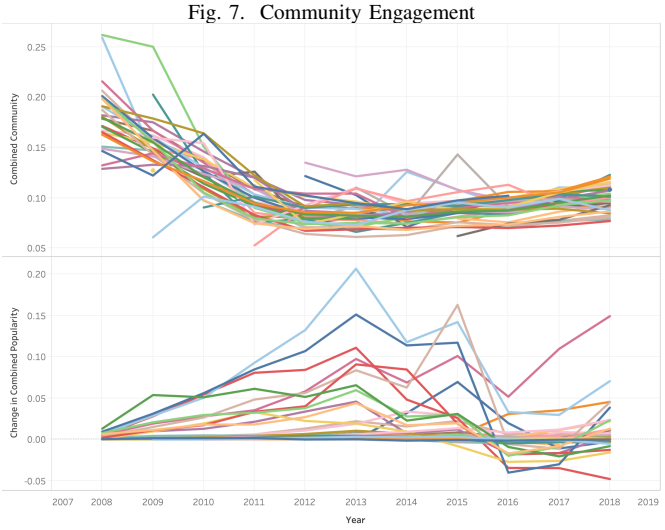
HTML, CSS will be beneficial due to higher demand shortage as well as good community engagement. Investing time in learning Dart and Go which have the most engaged online communities as well as the highest rise in demand shortage will provide a competitive advantage. Languages on the decline in popularity such as PHP or C# may not necessarily be a bad idea if looking for a job in the near future due to constant demand shortage.

For companies, working with the most popular languages gives easy access to talent, however, working with languages having high community engagement opens access to the top talent. Some of the best languages to consider are Ruby, Go, Julia, Dart and Rust.

IX. Future Work

As part of the future work, the application could be extended to identify the type of the programming language. For example, C and Go could be categorised as ”systems” as they are lower level and are well suited for systems development, Python could be categorised as back-end, etc. Then, the languages could only be analysed against languages belonging to the same category. This would enable a fairer comparison and would significantly improve the actuation of recommending languages or alternatives.

Fig. 7. Community Engagement



Acknowledgment

Many thanks to Professor Suzanne McIntosh for the teaching and facilitation of the course. We also thank the NYU HPC team for the seamless provision of highly powerful infrastructure and Tableau for the free academic licence.

References

[1] J. Lerner and J. Tirole, “The economics of technology sharing: Open source and beyond,” *Journal of Economic Perspectives*, vol. 19, no. 2, pp. 99–120, 2005.

[2] D. P. Delorey, C. D. Knutson, and S. Chun, “Do programming languages affect productivity? a case study using data from open source projects,” in *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS’07: ICSE Workshops 2007)*. IEEE, 2007, pp. 8–8.

[3] K. Edwards, J. Weststar, W. Meloni, C. Pearce, and M.-J. Legault, “Developer satisfaction survey 2014. summary report,” 2014.

[4] K. K. Bhatti and T. M. Qureshi, “Impact of employee participation on job satisfaction, employee commitment and employee productivity,” *International review of business research papers*, vol. 3, no. 2, pp. 54–68, 2007.

[5] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, “A large scale study of programming languages and code quality in github,” in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 155–165.

[6] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, “An in-depth study of the promises and perils of mining github,” *Empirical Software Engineering*, vol. 21, no. 5, pp. 2035–2071, 2016.

[7] J. Yang, K. Tao, A. Bozzon, and G.-J. Houben, “Sparrows and owls: Characterisation of expert behaviour in stackoverflow,” in *International conference on user modeling, adaptation, and personalization*. Springer, 2014, pp. 266–277.

[8] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, “What makes a good code example?: A study of programming q&a in stackoverflow,” in *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 2012, pp. 25–34.

[9] B. Bazelli, A. Hindle, and E. Stroulia, “On the personality traits of stackoverflow users,” in *2013 IEEE international conference on software maintenance*. IEEE, 2013, pp. 460–463.

[10] R. Gupta and P. K. Reddy, “Learning from gurus: Analysis and modeling of reopened questions on stack overflow,” in *Proceedings of the 3rd IKDD Conference on Data Science, 2016*, 2016, pp. 1–2.

[11] S. Wang, D. Lo, and L. Jiang, “An empirical study on developer interactions in stackoverflow,” in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1019–1024.

[12] R. K.-W. Lee and D. Lo, "Github and stack overflow: Analyzing developer interests across multiple social collaborative platforms," in *International Conference on Social Informatics*. Springer, 2017, pp. 245–256.

[13] A. S. Badashian, A. Esteki, A. Gholipour, A. Hindle, and E. Stroulia, "Involvement, contribution and influence in github and stack overflow." in *CASCON*, vol. 14, 2014, pp. 19–33.

[14] B. Vasilescu, V. Filkov, and A. Serebrenik, "Stackoverflow and github: Associations between software development and crowdsourced knowledge," in *2013 International Conference on Social Computing*. IEEE, 2013, pp. 188–195.

[15] Y. Tian, W. Ng, J. Cao, and S. McIntosh, "Geek talents: Who are the top experts on github and stack overflow?" 2019.

[16] G. Gousios, "The ghtorent dataset and tool suite," in *2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, 2013, pp. 233–236.