Modeling Topic Exhaustion for Programming Languages on StackOverflow

Rao Hamza Ali and Erik Linstead
Fowler School of Engineering, Chapman University

Abstract

We apply latent Dirichlet allocation on StackOverflow questions, spanned across ten years, for Python, JavaScript, Java, C++, and R, in order to discover underlying topics of questions asked for these programming languages. We focus on topics that have exhausted over the years; topics that once peaked in terms of the number of questions being asked about them but are now in a decline. Studying these topics provides insight into a language's evolution and its cohesion with other programming languages, that may offer similar features. We also measure the average wait times to get answers for questions from exhausted topics, to highlight if the community also plays a role in making these topics exhausted.

Keywords: Latent Dirichlet Allocation; SOTorrent; Topic Modeling; Stack Overflow

1. Introduction

Stack Overflow (SO) has become one of the most popular platforms for programmers to ask and answer questions for a wide range of topics in software engineering [1]. Starting in 2008, the website has seen constant increase in questions asked everyday and the categories to which they belong [2]. The community, too, has been increasingly active and boasts an average reply time to a question of 11 minutes [3]. With the seemingly nonstop increase in questions asked and new categories sprouting up, we are interested in seeing how topics (many of which were of high interest to the developer community historically) have been performing with the influx of new topics. We are also interested in topics which have seen a peak of interest from programmers and now are in a decline. We term these topics as exhausted, in that they peaked in terms of questions asked about them, but are now seeing continuous decline in new questions being posted.

Questions on SO are assigned tags which categorize the field a question may belong to, but are not helpful in determining the topic of that question. The work in [4] described developer behavior for mobile application development, using tags assigned to SO posts. But two questions with an iOS tag could ask questions on app performance and creating a slide view, which are two relatively different topics. Here we use an unsupervised technique, latent Dirichlet allocation (LDA) [5], to extract topics from questions specific to individual programming languages and study their evolution over time. While there have been studies where LDA is applied to SO questions, the focus has mostly been on identifying topics across all languages [6] and looking at short-term or temporal trends[7] . Another novel approach to using LDA was presented by [8] to automatically categorize software from source code, hinting at the success of using LDA for the analysis of source code and related text using topic generation. Unlike this previous work, here we run LDA separately on questions for popular programming languages and observe the trends across the years. The ultimate goal is to identify which features of the language have been exhaustively discussed on SO and are no longer the main focus of the developer community. We also want to understand how the adoption of programming languages evolves based on the type of questions that are asked about them.

One important question that is raised when looking at topics that have been exhausted is: is the community no longer active in answering a question related to such topics? Long wait times to get an answer or failure to come up with a good answer in time, could dissuade developers from asking further questions about these topics. Once we identify topics that have been exhausted, we will also look at the community wide statistics about the time taken to answer questions across the years, and can then conclude the community's role in exhaustion of topics.

2. Data

We use the December 2018 data set of SOTorrent[9] for our study. It consists of over 42 million posts made on SO from August 2008 to December 2018, including question and answers from the community on different topics. Each question is manually tagged by the poster with a programming language and a related field. We use this attribute

DOI reference number: 10.18293/SEKE2020-107.

Table 1. Total number of questions per programming language

Language	Questions
JavaScript	1,723,695
Java	1,487,204
Python	1,068,646
C++	595,662
R	265,946

to subset the data for 5 programming languages: Python, JavaScript, Java, R, and C++. These languages are the most tagged on SO and comprise a large corpus of questions for our analyses. We are also using data beginning January 2009, to allow for uniform per-year stats.

We choose to use the text pertaining to the question asked instead of the question title. The question body provides greater detail and insight about the problem itself. We further remove any source code references from the question body as we are interested in discovering latent topics from the description of the programming language feature rather than from relevant code. From this, we are able to field more connections between words for LDA to utilize and get results which give a better understanding of the trends. For the community stats, we extract the first and accepted answer times for all questions. An accepted answer is selected by the user who posted the question, which they deem to be the best answer among all others. Table 1 describes the total number of questions, for each language, that were used in training the LDA model.

3. Method

We start by collecting all question posts for a programming language, and removing all stop-words, punctuation, and numbers from them. We extensively make use of the gensim [10] package in Python for topic modeling and natural language processing (NLP). Using built-in functions, we tokenize each question, and create a dictionary for each word and its occurrence count. We use spaCy [11], the NLP Python library, to lemmatize each token to reduce the word space to a common base form and discover more coherent topics. A bag-of-words model [12] is then generated from the dictionary. This step is crucial because we are not interested in the order of words for each question, but the word occurrences in all questions. This pre-processing step is done separately for all languages we run LDA on.

LDA is a probabilistic model for a collection of documents, where each document consists of a bag of words and is viewed as a mixture of different topics with varying probabilities. Using this explicit representation of documents in terms of topic probabilities, LDA identifies the topic that most represents each document. This is done by utilizing the mixture model we learn for each set of questions corresponding to a programming language. Given the vast vocabulary of words and the need to identify both high-level and low-level topics, we parameterize LDA to discover 30 topics for each programming language. This number also gave us the most coherent latent topics.

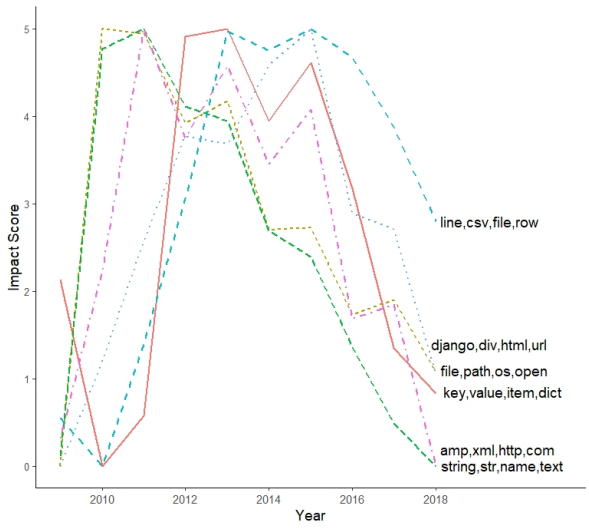

Figure 1. Trend of exhausted topics for Python

A topic coherence [13] for each LDA model is calculated which gives a measure of strength and consistency of all topics generated. Only models with a topic coherence greater than 0.5 are chosen. This is to ensure that, while 30 topics do cover all of the questions, the model has been successful in understanding the relationships in the corpus and the topics describe the data in the best possible way. The topics, identified by the model, can be joined back to the questions that they have the highest probability of belonging to. Next, we calculate the topic impact score, which is a rank of a topic's occurrence each year for a programming language in comparison to other topics utilized for the language within the same year. The impact score is scaled so that we are able to compare multiple exhausted topics at the same time and are able to observe common trends for them.

4. Results

For our results, we present the topics that have been exhausted, in terms of questions asked, across the years for the selected programming languages.By using a threshold of 30 topics for LDA, we were able to find not only top-

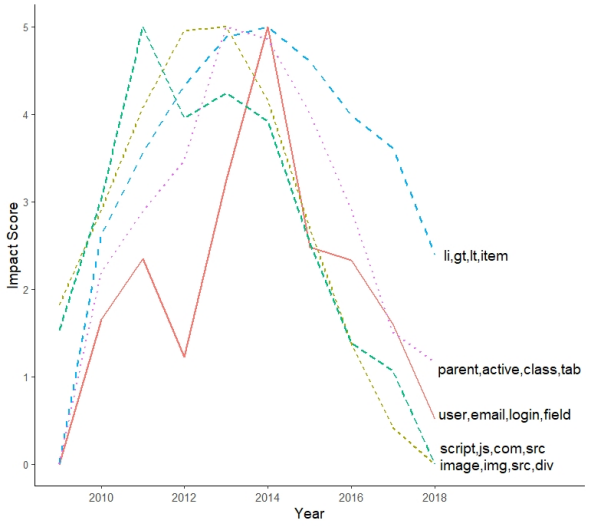

Figure 2. Trend of exhausted topics for JavaScript

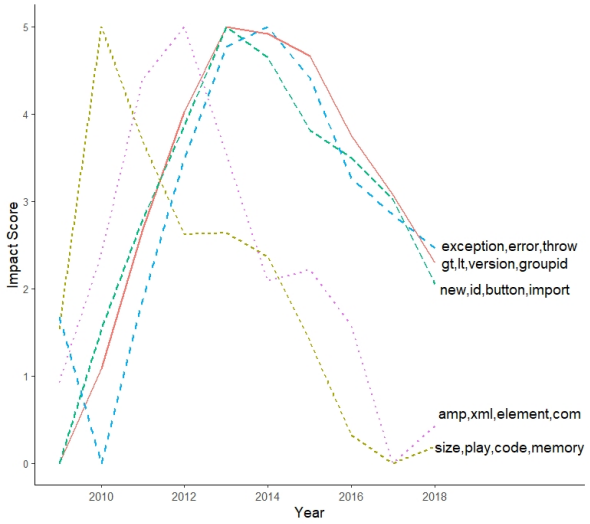

Figure 3. Trend of exhausted topics for Java

ics that represented a small set of questions, but also multiple topics that were part of a bigger theme and described different aspects of it. Each graph shows the yearly trend of exhausted topics, with respect to their impact score, for a different programming language. A set of words is assigned to each topic, extracted from the questions, and is used in the graphs to describe the overall theme of the topic. These words have the highest probability of belonging to that topic, among all words. They present an understanding of what kind of questions are being asked.

LDA has been extensively used in finding inherent topics for a corpus of questions asked on StackOverflow

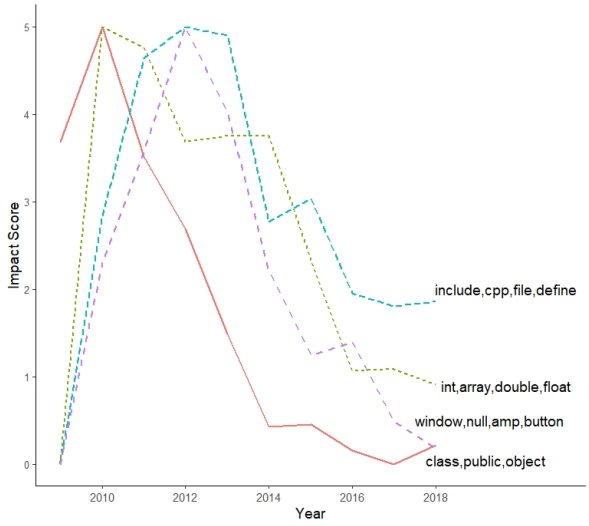

Figure 4. Trend of exhausted topics for C++

[14][15][16]. But the focus has remained on what kind of topics are discovered, or which topics have been becoming more popular or are in steady decline. Our research focuses on exhausted topics, that are not likely to have an impact anymore on the community, but need to be identified to further study a topic's trend over the years. It could be the case that adoption of a new programming language or an update in the functionality that the developer is interested in asking about, heavily saturates such topics and we no longer see them making the same impact on SO. Through LDA, we found these exhausted topics that cover features like file I/O, front-end and back-end development, and basic language functionality.

We start our discussion by identifying exhausted topics for Python. Figure 1 shows the trend of such topics between 2009 and 2018, given their impact scores. We only report topics that had an increase in impact after the inception of SO, but are now in a decline, in terms of questions asked. We note that topics related to HTML in Python are completely saturated and developers are no longer asking a lot of questions about them. Interestingly, we also find some topics related to Python's built-in functionality in a decline too. Python has more or less used the same functions for file IO and data structures, so it is apparent that developers have run out of new questions to ask about them, and their queries have most likely already been answered by a previous post. The decline in HTML topics is uniform across all platforms. It is possible that adoption of newer web scripting languages has moved developers away from using Python's web features.

Figure 2 describes the trends of exhausted topics for JavaScript (JS) on SO. We, here too, note that topics related to web development and visual programming are the ones

that have been most exhausted by the end of 2018. With no new features introduced to JavaScript related to these topics, developers have saturated the topics by asking all questions that could be asked, and are satisfied with the answers on questions already posted on SO, that are similar to their queries. Figure 3 shows the trend of exhausted Java topics across the years, with topics related to HTML again being the ones that have been exhausted. Since the introduction of Go, Rust, Kotlin, and Swift, the focus of developers interested in making visual and web based applications has switched over to these new programming languages, abandoning the functionality offered by the relatively older languages. This results in some topics for a language, that was popular a few years ago, no longer having the same impact as others. We see a similar trend for exhausted topics in C++ in Figure 4. Web application related questions are no longer being asked, and questions about the basic functionality of the language, which has not changed after many version updates, are also on a downward trend.

Another set of topics, we notice have been exhausted, are ones related to the basic functionality of a programming language. Since R is used for statistical modeling and has no applications in web development, all exhausted topics in R are related to the base functionality of the language. Questions about data I/O, plotting data points, and wrangling data of different types, are the main focus of the language. And all such topics have already been exhausted. Developers can independently develop packages for a programming language, to introduce new features, that core developers have not yet introduced, leaving base features to be the same across the years. Our analysis makes it apparent that questions for such features will not evolve over time and a developer looking to learn a language, will find answers to their questions already posted on the website. This points to the evolution of a programming language and the evolution of a developer using that language: how HTML based queries are on a decline and questions for big data are being asked more and more, or how people have a command over the basic features of Java and C++ and have now run out of questions to ask about them.

Modeling the trends of exhausted topics across different languages leads to more questions being asked about the role of the community in their decline. Given that no new functionality is added to a certain feature of a programming language, the community will eventually run out of new questions to ask and those seeking answers would get redirected to questions that have already been posted on SO. But if the number of such questions being asked declines, is the community also in decline in answering these questions? It could be possible that if the SO community takes a longer time to answer a question, or to give a suitable enough question that it is deemed 'accepted' by the original poster, that it deters developers from asking questions related to these

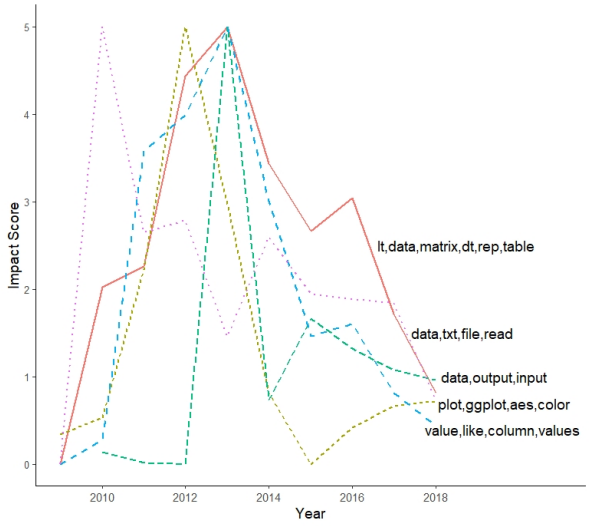

Figure 5. Trend of exhausted topics for R language on Stack Overflow

topics, and instead they try to find their solution in old posts.

To answer this question, we look at the average time taken, in hours, to post the first answer, and the accepted answer, for all questions asked for the exhausted topics. As the number of active users on SO increases over time, it is clear that the average wait time to get an answer will lower [17]. But only if the community is no longer interested in answering questions about exhausted topics, will we see an increase or even a plateau in the averages. Figures 6 and 7, respectively show the trend of average number of hours until a first answer is posted and an accepted answer is posted for the exhausted topics for all five programming languages between 2009 and 2018. With over a 100,000 questions being asked about these topics consistently since 2014, we observe a steady decline in the average wait time to get to a satisfactory answer. This shows that the community is is still active in answering questions, regardless of whether they belong to an exhausted topic or not.

We conclude that topic exhaustion of a programming language on SO is not due to the community's lack of interest in answering these questions, and instead, stems from the evolution of the language and its cohesion with other languages. Python still remains the programming language with the most questions asked. What has changed is that developers have found other, more efficient avenues to using some features of the language, via a different programming language. But with new functionality being added to languages, which leads to more questions being asked by developers, who want to learn about those functions, old topics become saturated and developers have exhausted the different type of questions they can ask about them.

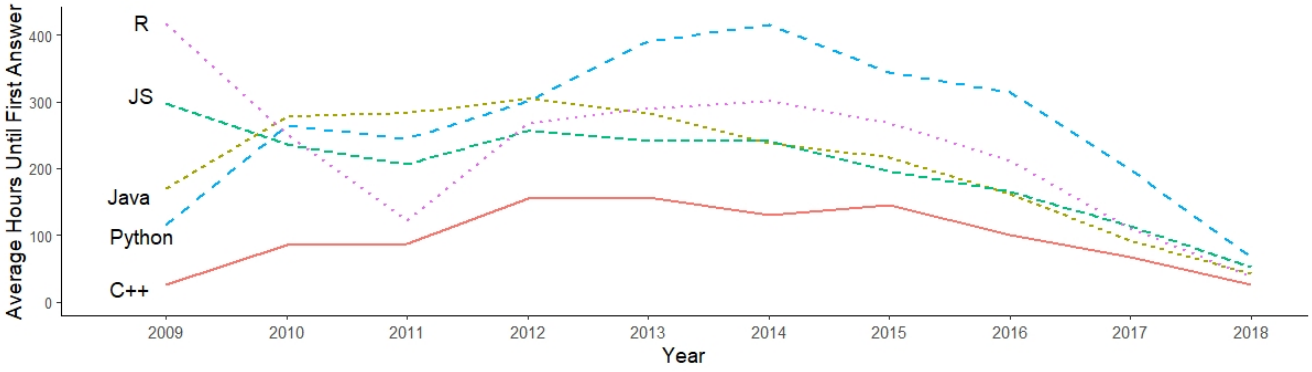

Figure 6. Average Hours until First Answer for Exhausted Topics

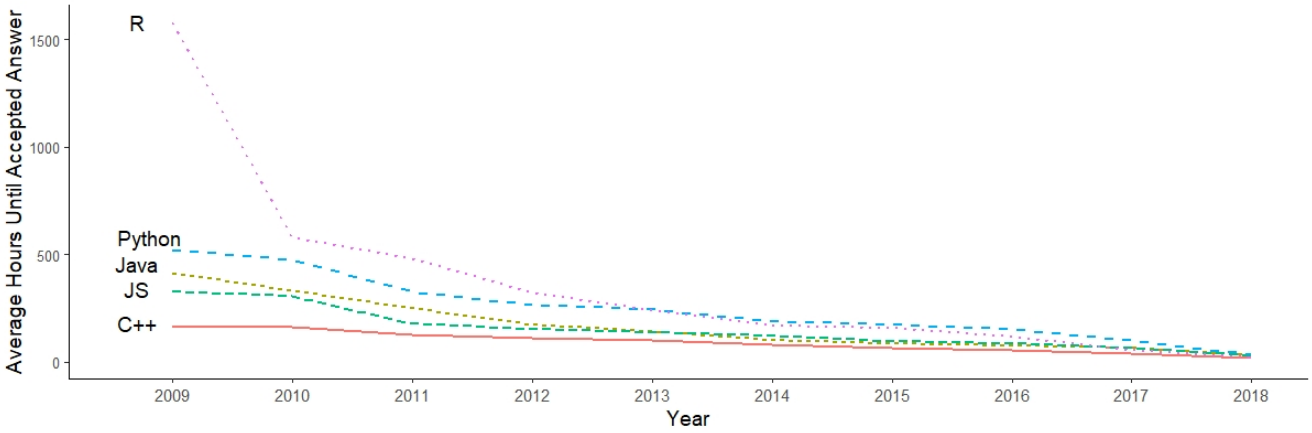

Figure 7. Average Hours until Accepted Answer for Exhausted Topics

5. Related Work and Future Directions

Since first applied to software in 2007 [18], LDA has become a staple for textual analysis of software artifacts. The work in [6] extracted topics from questions asked on SO using LDA and compared them with Java code tokens to find that some topics generated were either text or code identifier only. Mentioned earlier, [2] performed topic modeling on SO questions and answers, posted between 2008 and 2010, and highlighted main discussion topics, scores comparison of answer topics, developer interest, and change in interest in technologies over time. In [16], the authors looked into how the community answers posts on SO and calculated user stats over topics generated by an LDA model. Our paper focuses on exhausted topics across 5 programming languages, over the course of 10 years, and provides an insight into why they are no longer a big focus of developers.

Our research also relates to [19], which explores the evolution of features of a programming language across version updates, using topic modeling. The topics were generated using source code for large open source Java projects, and the trends showed a stark comparison of feature usage between version updates. In this paper, we focus on the vocabulary of questions asked about these features in hope that we can provide trends regardless of version updates, over a long period of time, and view the exhaustion of questions to ask as a measure itself of depletion of newer ways to use a language feature. Exhaustion of questions asked about a topic does not mean that the feature is less popular, nor does it mean that the feature is now deprecated and requires replacement. We instead focus on a community driven vantage point, which views such features as something that programmers have mastered, and are now looking at other prospects that are more challenging.

In the future, topic modeling SO questions for viewing trends for older languages would highlight the most common challenges programmers face for a language and analyzing the answers for such questions would also give a way of solving them. Investigating why certain topics have been exhausted in terms of questions asked, from a language developer point of view, is also a challenge worth tackling, which could provide new insight into how the community

asks questions for a highly documented feature, or how a major update can increase the hype around it. This information, in turn, can be leveraged by language developers to prioritize the integration of new features or even improve old ones.

References

[1] S. Baltes, L. Dumani, C. Treude, and S. Diehl, "Sotorrent: reconstructing and analyzing the evolution of stack overflow posts," in *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*, 2018, pp. 319–330.

[2] A. Barua, S. W. Thomas, and A. E. Hassan, "What are developers talking about? an analysis of topics and trends in stack overflow," *Empirical Software Engineering*, vol. 19, no. 3, pp. 619–654, 2014.

[3] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest q&a site in the west," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2011, pp. 2857–2866.

[4] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[6] M. Allamanis and C. Sutton, "Why, when, and what: analyzing stack overflow questions by topic, type, and code," in *Proceedings of the 10th Working Conference on Mining Software Repositories*. IEEE Press, 2013, pp. 53–56.

[7] K. Bajaj, K. Pattabiraman, and A. Mesbah, "Mining questions asked by web developers," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. ACM, 2014, pp. 112–121.

[8] K. Tian, M. Revelle, and D. Poshyvanyk, "Using latent dirichlet allocation for automatic categorization of software," in *Mining Software Repositories, 2009. MSR'09. 6th IEEE International Working Conference on*. IEEE, 2009, pp. 163–166.

[9] S. Baltes, C. Treude, and S. Diehl, "Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets," *CoRR*, vol. abs/1809.02814, 2018. [Online]. Available: http://arxiv.org/abs/1809.02814

[10] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[11] M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, 2017.

[12] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1-4, pp. 43–52, 2010.

[13] K. Stevens, P. Kegelmeyer, D. Andrzejewski, and D. Buttler, "Exploring topic coherence over many models and many topics," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012, pp. 952–961.

[14] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk, "An exploratory analysis of mobile development issues using stack overflow," in *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*. IEEE, 2013, pp. 93–96.

[15] X.-L. Yang, D. Lo, X. Xia, Z.-Y. Wan, and J.-L. Sun, "What security questions do developers ask? a large-scale study of stack overflow posts," *Journal of Computer Science and Technology*, vol. 31, no. 5, pp. 910–924, 2016.

[16] 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*. ACM, 2013, pp. 1019–1024.

[17] G. Hewgill, *Meta Stack Overflow Statistics Graphs*, 2010 (accessed May 4, 2020). [Online]. Available: https://meta.stackexchange.com/questions/38297/meta-stack-overflow-statistics-graphs

[18] E. Linstead, P. Rigor, S. Bajracharya, C. Lopes, and P. Baldi, "Mining eclipse developer contributions via author-topic models," in *Proceedings of the Fourth International Workshop on Mining Software Repositories*. IEEE Computer Society, 2007, p. 30.

[19] E. Linstead, C. Lopes, and P. Baldi, "An application of latent dirichlet allocation to analyzing software evolution," in *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*. IEEE, 2008, pp. 813–818.