An Empirical Study on Developer Interactions in StackOverflow

Shaowei Wang, David Lo, Lingxiao Jiang
Singapore Management University
{shaoweiwang.2010,davidlo,lxjiang}@smu.edu.sg

ABSTRACT

StackOverflow provides a popular platform where developers post and answer questions. Recently, Treude et al. manually label 385 questions in StackOverflow and group them into 10 categories based on their contents. They also analyze how tags are used in StackOverflow. In this study, we extend their work to obtain a deeper understanding on how developers interact with one another on such a question and answer web site. First, we analyze the distributions of developers who ask and answer questions. We also investigate if there is a segregation of the StackOverflow community into questioners and answerers. We also perform automated text mining to find the various kinds of topics asked by developers. We use Latent Dirichlet Allocation (LDA), a well known topic modeling approach, to analyze the contents of tens of thousands of questions and answers, and produce five topics. Our topic modeling strategy provides an alternative perspective different from that of Treude et al. for categorizing StackOverflow questions. Each question can now be categorized into several topics with different probabilities, and the learned topic model could automatically assign a new question to several categories with varying probabilities. Last but not least, we show the distributions of questions and developers belonging to various topics generated by LDA.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management

General Terms

Management; Human Factors

Keywords

Developer Interaction Mining; Developer Forum Mining; Latent Dirichlet Allocation (LDA)

1. INTRODUCTION

Software development and maintenance are complex activities that often involve many concepts and reference documents. Many aspects of the software, such as the interface defined, APIs used, bugs fixed, and even architectural designs, may be changed over time. In order to work with so many aspects and details involved in a software project, developers often need helps from one another. A widely used way is for developers to ask questions and/or answer them in various online forums. StackOverflow is one of the most popular question and answer sites [1] for developers. Millions of questions are posted there, and many of these questions are answered by various developers [16].

Understanding how a question and answer site in general and StackOverflow in particular is used by developers could help us in better improving developer experience in using these sites. It would also help us to better understand how information within StackOverflow could be useful to various software development activities, and the community of knowledge is formed to benefit individual developers.

To better understand StackOverflow, Treude et al. manually analyze 385 questions and assign them to 10 categories [16]. While the previous study is useful to aid the understanding of StackOverflow, there are questions that require more investigation and we would like to answer some of these questions in this paper. Also, Treude et al. employ a manual process to analyze the content of questions. In this work, we would like to automate the process by using a text mining solution that could automatically assign categories to questions.

In this work, we propose a framework which takes data from StackOverflow as input and eventually answers a number of research questions. We first investigate the distribution of questioners and answerers. We also dive deeper into the behaviors of questioners and answerers. For example, we investigate if developers tend to reciprocate favors received from other developers, and whether the questioners and answerers on StackOverflow are two different sets of people. Finally, we investigate tens of thousands of questions from StackOverflow and infer the different topics that developers ask by using topic modeling. The topic model could be used to assign topics to new questions whose topic labels are unknown.

To achieve the above we perform both text and graph analysis. We first create a graph that links developers based on the questions that they ask and those that they answer. We then analyze the links (or edges) in the graph to investigate the behaviors of questioners and answerers. Aside from graph analysis, we also perform text analysis on the content of the questions. Every question could contain two parts: normal text and code snippet. We need to perform a separate processing for the normal text and for the code snippet. For the code snippets, we extract identifier names from them. The processed word tokens are then send to a topic modeling tool, namely Latent Dirichlet Allocation (LDA), which would report the different topics (or groups of words with possibly similar semantics) that exist in the questions and their answers.

The contributions of this work are as follows:

1. We investigate the distribution of questioners and answerers and investigate their behaviors.
2. We analyze the role bias of people in StackOverflow: whether most people are predominantly questioners or answerers.
3. We employ topic modeling to assign topics to tens of thousands of questions from StackOverflow.

The structure of this paper is as follows. In Section 2, we describe preliminary information on StackOverflow and topic modeling (Latent Dirichlet Allocation). In Section 3, we describe the methodology we use in this paper. Section 4 describes our empirical evaluation. We describe related work in Section 5. Finally, we conclude with future work in Section 6.

2. PRELIMINARIES

In this section, we introduce StackOverflow and topic modeling.

2.1 StackOverflow

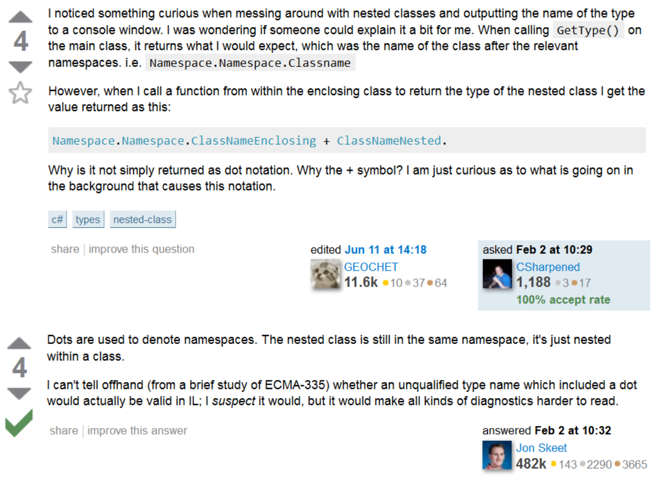
StackOverflow allows users to register, post questions, and answer posted questions. Since users are registered, one could track the questions he or she posts, and answers he or she makes.

For each posted question, a user can include textual description of the problem. The user can also include code snippets. Code snippets are often separated from other normal texts. Other users can answer the posted question. Multiple answers could be given by various people. The one posting question could then either post a comment or indicate one of the answers as correct. Other people could also rate whether they like either the questions and/or the answers. A snapshot of the StackOveflow page showing a question and its corresponding answer is shown in Figure 1.

2.2 Topic Modeling

Topic modeling is a way to unsupervisedly group a pool of words into groups. It is unsupervised as there is no need for users to provide a labeled training data containing words assigned to predefined labels. Typically the number of groups is decided by the user. The topic modeling approach would then generate at most the specified number of topics. Each topic is simply a set of words. A topic model would assign a topic to each word in a document. A document would then be represented as a distribution of topics.

There are various topic modeling approaches, such as Latent Dirichlet Allocation (LDA) [5], hierarchical LDA [4], Locally-consistent Topic Modeling [8], Discriminative Topic Modeling [11], etc. All of these approaches build a statistical model which is used to group related words together. In this study we use LDA as it is a popular one and has been shown to be effective to solve software engineering problems [14, 2, 18], and an implementation is available[1].


4

I noticed something curious when messing around with nested classes and outputting the name of the type to a console window. I was wondering if someone could explain it a bit for me. When calling `GetType()` on the main class, it returns what I would expect, which was the name of the class after the relevant namespaces. i.e. `Namespace.Namespace.Classname`

However, when I call a function from within the enclosing class to return the type of the nested class I get the value returned as this:

```
Namespace.Namespace.ClassNameEnclosing + ClassNameNested.
```

Why is it not simply returned as dot notation. Why the + symbol? I am just curious as to what is going on in the background that causes this notation.

c# types nested-class

share | improve this question

edited **Jun 11 at 14:18**
GEOCHET
11.6k ● 10 ● 37 ● 64

asked **Feb 2 at 10:29**
CSharpened
1,188 ● 3 ● 17
100% accept rate

4

Dots are used to denote namespaces. The nested class is still in the same namespace, it's just nested within a class.

I can't tell offhand (from a brief study of ECMA-335) whether an unqualified type name which included a dot would actually be valid in IL; I *suspect* it would, but it would make all kinds of diagnostics harder to read.

share | improve this answer

answered **Feb 2 at 10:32**
Jon Skeet
482k ● 143 ● 2290 ● 3665

Figure 1: A Sample Question (Top) and Answer (Bottom) on StackOverflow

3. METHODOLOGY

The framework for our empirical study is shown in Figure 2. The framework takes as input data from StackOverflow and eventually answers a number of research questions that we have. It has two processing components: data processor and text mining. We describe the input of our framework, and the two processing components in the following paragraphs.

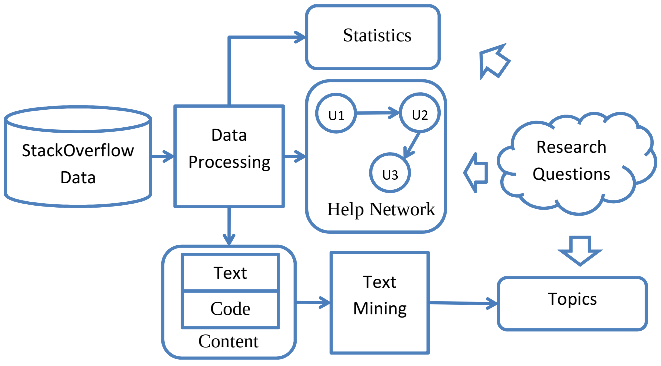


Figure 2: Empirical Study Framework

We take in a set of questions posted in StackOverflow and their answers. For every question we know the content of the question as well as the developer that posts the question. For questions which are answered, we also have the list of answers and for each answer we have the information of the developer who answers it. This data is the input to our framework.

Our first processing component, data processor, analyzes the input and generates three kinds of outputs. First, we

[1]http://jgibblda.sourceforge.net/

generate some statistics. These include the number of times each developer answers questions, posts questions, and the proportion of a developer's posts that are questions. Second, we build what we refer to as a help graph. A help graph is a directed graph. Each node in the graph is a developer. A directed edge from developer D_1 to D_2 in the graph denotes that developer D_1 helps developer D_2 (i.e., D_1 answers D_2's questions). To understand the reciprocity relationship between developers, we simply analyze who help developer D and who get favor from D by analyzing the edge linked to node D in help graph. Third, we also extract the contents of the questions. Each question contains two kinds of content: normal text and code. We can separate normal text from code contents since StackOverflow clearly separates text and code using a special notation.

Our second processing component analyzes the content of the questions. We have two sub-steps: content pre-processing and topic modeling. The content pre-processing step takes in both normal text and code, performs tokenization, stop word removal, and stemming. Tokenization breaks a paragraph into word tokens. Stop word removal removes commonly used words like: is, are, I, you, etc. Stemming reduces a word to its root form, e.g., reading to read, etc. For the code, we remove reserved keywords such as: if, while, etc., curly brackets, etc, and extract identifiers and comments. These are then subjected to tokenization, stemming, and stop word removal too.

After the contents have been pre-processed, we employ Latent Dirichlet Allocation (LDA), to do topic modeling. Topic modeling analyzes a set of documents (i.e., the questions) and produces a set of topics. Each topic is a set of words. For each topic, LDA also returns the set of most representative words.

The outputs of the data processor and text mining components are used to answer the various research questions that we describe in the next section.

4. EMPIRICAL EVALUATION

In this section, we first present our dataset and the set of research questions we address. Next, we present our findings that provide answers to the research questions. We also present some threats to validity.

4.1 Dataset

We extract the first 100,000 questions using StackOverflow API[2]. For each day, we can download from an IP address at most 30,000 questions. For every call, the API would return 100 randomly (or pseudo-randomly) selected questions. Thus, some of the 100,000 returned questions are duplicates of another (they have the same question identifier). We apply a filter to remove duplicates and have in total 63,863 unique questions. We use these questions and their answers in our study.

4.2 Research Questions

In this study we are interested in the following research questions:

[2]https://api.stackexchange.com/2.0/questions

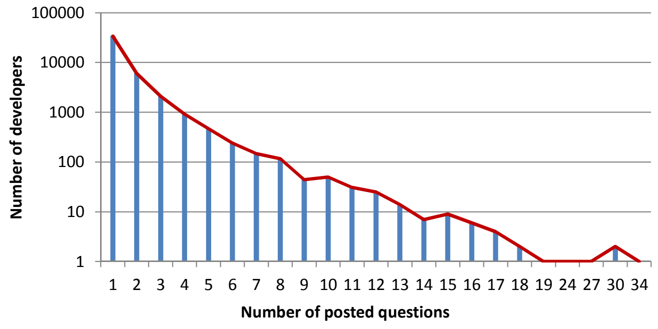


Figure 3: Histogram of Questioners

RQ1	What are the distributions of developers that post questions?
RQ2	What are the distributions of developers that answer questions?
RQ3	Do developers that ask questions answer questions too?
RQ4	Do developers receiving help returns the favor?
RQ5	What topics do developers ask about and what are the distributions of the topics?

4.3 RQ1: Distribution of Questioners

We plot the histogram of questioners in Figure 3. The graph shows the number of developers that ask a given number of questions, and its y-axis is in log-scale. From the graph, we notice that most developers (33,907 out of 44,087 developers) only ask one question. Only about 23.1% of the developers ask two or more questions. The number of developers that ask questions reduces exponentially as we consider a higher number of posted questions. Only 1.6% of the developers ask more than 5 questions.

The result shows that there are few "regular" questioners on StackOverflow. This is possibly because many questions have already been asked before and users could find answers to them by just looking into the various pages on StackOverflow or other question and answer sites via search engines.

4.4 RQ2: Distribution of Answerers

We plot the histogram of answerers in Figure 3. The graph shows the number of developers that answer a given number of questions and its y-axis is also in log-scale. From the graph, we notice that most developers (28,578 out of 44,087 developers) only answer one question. About one thousand developers (2.3%) do not answer any questions. Only about 35.2% of the developers answer two or more questions. The number of developers that answer questions reduces exponentially as we consider a higher number of answers. Only 7.8% of the developers answer more than 5 questions. The highest number of questions a developer answers in our dataset is 178. There is only one developer that answers this many questions.

Compared with the distribution of questioners, the distribution of answers have several differences. First, some developers answer zero question while no developer asks zero question. Also, interestingly, the number of developers that answer a substantial number of questions (> 5) is more than the number of developers that ask a substantial number of questions (> 5)—3,424 (7.8%) versus 701 (1.6%). This may

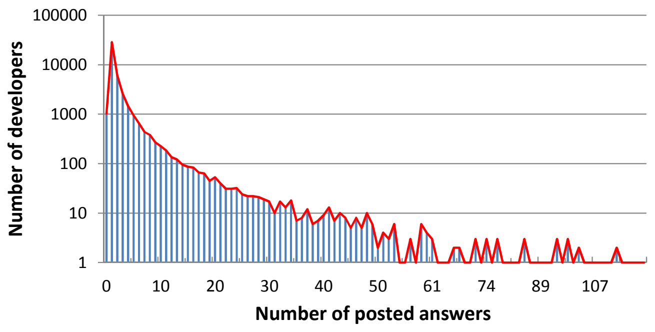


Figure 4: Distribution of Answerers

Table 1: Reciprocity in StackOverflow

Helper	Helpee that Reciprocate in the Future
429349	934123
243225	1259881
15055	9530
1077364	744859
1077364	544504
18 Other Pairs	

imply that many developers on StackOverflow are interested in contributing to the community and are not solely interested in getting his or her questions answered.

4.5 RQ3: Segregation of StackOverflow Community

To answer this research question, we investigate the proportion of posts that various developers make that are answers to some questions. We show this in Figure 5. We notice that a majority of developers only ask questions but do not answers them (83.2%, 36,672 developers). Thus, we could divide the StackOverflow community into two groups: people that only ask questions, and those that answer one or more questions. The first group is the majority.

Interestingly, we do not find any developer in our pool that does not ask any question—the number of developers whose posts are 100% answers is zero. We also note another peak in Figure 5: These are developers (2,956 of 44,087 developers), with 50-59% of posts being answers. These correspond to ideal developers that contribute answers to the community as much as requesting answers from the community.

4.6 RQ4: Reciprocity in StackOverflow

To answer the fourth research question, we investigate the help graph. A help graph is a directed graph, where each developer is a node, and the node corresponding to a developer D_1 is linked to that of D_2 if D_1 answers a question posted by D_2. We would like to investigate how often two developers D_1 and D_2 are connected by two edges, one from D_1 to D_2 and the other from D_2 to D_1. We find that there are only a few of such developers (23 pairs). We highlight a few in Table 1. The table contains the identifiers of the helpers and helpees that reciprocate. From the result, we hypothesize that developers tend to help anyone no matter if they have helped him or her before and StackOverflow tends to benefit the community as a whole.

4.7 RQ5: Topics Developers Ask About

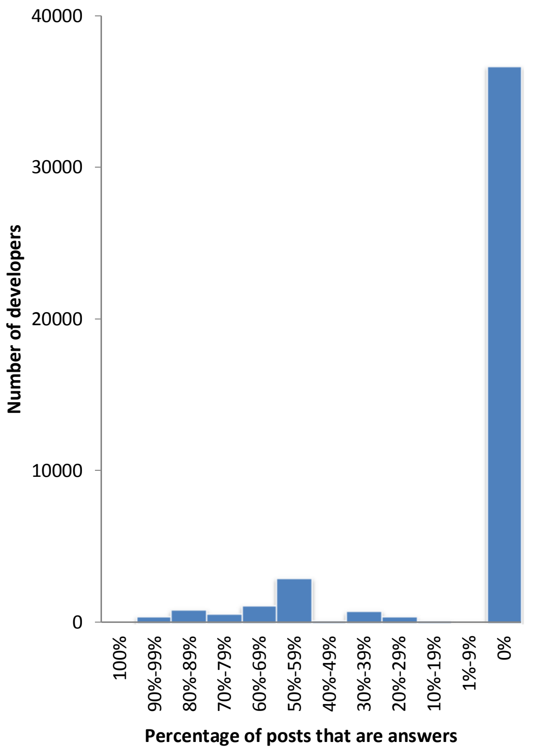


Figure 5: Post Proportion

To answer this research question, we run a topic modeling technique—LDA on the text and code contents of the questions that developers ask. We set the number of topics to be 5, and after LDA completes running, it outputs 5 topics: each topic is a set of words sorted in terms of their likelihood of belonging to the topic. LDA does not generate a meaningful label for each topic; We manually study the words in each topic and related questions, and assign a label to the topic.

Based on the above analysis, Table 2 shows the five topics with our manually assigned labels, some representative words in each topic, and the identifier of an example question that has a high probability of belonging to the topic. The five topics are: user interface, stack trace, large code snippet, web document, and miscellaneous. User interface topic is characterized by words such as view, image, button, etc. Many questions strongly related to this topic ask questions on how to render an image or fix a bug in a user interface. Questions strongly related to the stack trace topic typically contain a stack trace which is provided by developers to explain these questions clearly. A stack trace is often associated with words: error, java, org, server, etc. The words java and org are often repeated many times in the stack trace. Questions belonging to the large code snippet topic typically contain a large code snippet. A large code snippet contains keywords such as string, new, class, etc. The web document topic contain words related to the various web documents, e.g., HTML, etc. Many questions strongly related to this topic contain HTML snippets. The miscellaneous category contains many different kinds of questions.

Also, we plot the proportions of questions that belong to each category. We plot the proportions in two approaches. In the first approach, we assign only one topic with the

Table 2: Topics, Related Words, and Questions

Topic	Words	Question
User Interface	view, image, button, etc.	10934198
Stack Trace	java, error, org, server, etc.	2521468
Large Code Snippet	code, string, new, object, class, etc.	10934762
Web Document	href, page, html, php, etc.	10930491
Miscellaneous	us, can, strong, like, would etc.	10881333

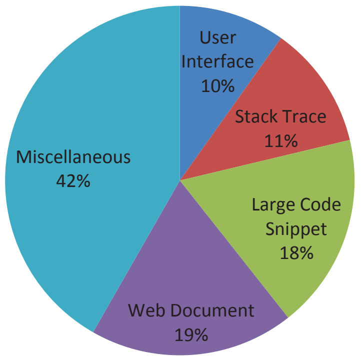


Figure 6: Document Distribution (Approach 1: Top-1)

highest probability to a question based on the topic probabilities assigned by LDA to the question and count how many questions belong to each topic. In the second approach, we assign multiple topics to a question: If a question has 0.2 probability of belonging to topic 1 as decided by LDA, we increment the counter for topic 1 by 0.2. The pie charts in Figures 6 & 7 show the distribution of documents into various topics by following the first and second approaches respectively. In Figure 7, we notice that each topic has similar number of questions. The miscellaneous topic has the most number of questions followed by web document, large code snippet, stack trace, and user interface.

In addition, we show the distribution of developers involved in each topic category. We count the number of unique developers involved in all questions in each topic category and plot the histogram in Figure 8 and 9 by following the two approaches used above to assign one topic or multiple topics to a question. Following the first approach, the most popular topic category is miscellaneous, followed by web document, large code snippet, stack trace and user interface in Figure 8. Following the second approach, miscellaneous is the most popular topic, followed by large code snippet, web document, stack trace and user interface.

4.8 Threats to Validity

Threat to internal validity often refers to experimenter biases. In this study, most of our process is automated. The only manual process is the assignment of labels to the 5 topics that LDA returns. We assign the labels based on the top-50 words and most similar questions for each topic. A similar strategy is performed in other studies that use topic modeling techniques too, e.g., [18].

Threat to construct validity often refers to the appropriateness of the metrics used. In this paper, we only study

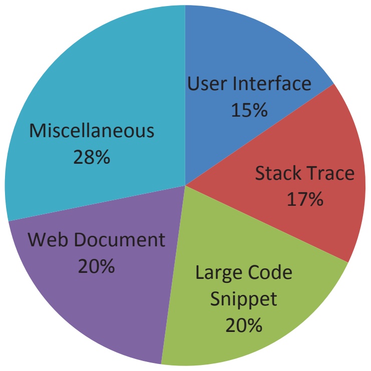


Figure 7: Document Distribution (Approach 2: Weighted)

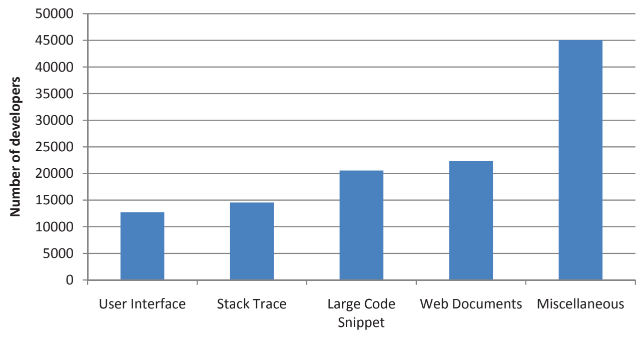


Figure 8: Histogram of Developers Per Topic Category (Approach 1: Top-1)

the five high-level categories for the StackOverflow questions. When we assign topics differently to the questions, our analysis results may vary. In future work, we will investigate the effects of various topic modeling techniques and various topic settings on our results.

Threat to external validity refers to the generalizability of our findings. We analyze tens of thousands of questions and answers. StackOverflow API returns questions and answers from different time in a single response. Thus we have injected an amount of randomness in our data which depends on how random the returned sets of StackOverflow questions are. Having a sizable randomly picked sample questions reduces our threat to external validity. Still in the future, we would like to extend this study to include more questions and answers from both StackOverflow and other web sites to reduce the threat even further.

5. RELATED WORK

Treude et al. is the first to analyze StackOverflow question and answer site [16]. In their study, they manually investigate a few hundred questions and assign them into 10 categories. They also investigate how tags are used in StackOverflow—what tags are common and how to group tags into categories. Treude et al. also investigate the distribution of questions that receive zero or more answers. They also look into what kinds of questions receive more answers. In this study, we extend the work of Treude et al. by in-

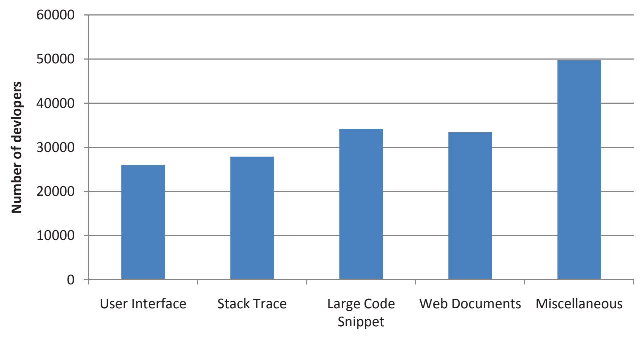


Figure 9: Histogram of Developers Per Topic Category (Approach 2: Weighted)

vestigating additional research questions. We look into the distribution of questioners and answerers. We employ Latent Dirichlet Allocation, a state-of-the-art and well-known topic modeling technique, to semi-automatically infer topics from questions and assign a question to several topics with some probabilities.

There are other studies that analyze the use of social media for software engineering. Some others propose tools that could help users to leverage social media for software development. Bachelli et al. develop an Eclipse plugin which allows users to get information from StackOverflow seamlessly and integrate information from StackOverflow with code in Eclipse [3]. Pagano and Maalej investigate how software developers blog [13]. Bougie et al. and Tian et al. investigate the micro-blogging site Twitter and see how it is used to distribute microblogs related to software engineering [6, 15]. Dabbish et al. analyze the social coding site GitHub and investigate its impact on developers by conducting a number of interviews [10]. Monperrus and Menzini provide an approach to semi-automatically extract frequently asked questions from various sources during software development, such as email lists and discussion forums, to provide reference documents for developers [12]. Breu et al. analyze questions asked in Mozilla and Eclipse bug reports to propose some ways to improve bug tracking systems [7].

There are a number of software engineering studies that employ topic modeling. Asuncion et al. use LDA for software traceability [2]. Wang et al. investigate the effectiveness of many topic modeling approaches for concern location [17]. Chen et al. use topic modeling to find defect prone topics from Mozilla Firefox, Eclipse, and Mylyn [9].

6. CONCLUSION AND FUTURE WORK

In this work, we investigate developer interactions on StackOverflow. Treude et al. recently analyze 385 questions manually and label them into 10 categories [16]. They also investigate the usage of tags in StackOverflow. In this work, we investigate the nature of developer interactions on StackOverflow. We find that most developers only answer or ask one question. Few developers answer and ask many questions. There are around 8% of developers that answer more than 5 questions. Most developers only ask questions but never answer any. On the other hand, there are some developers who ask and answer a similar number of questions. We notice that questions posted by developers could be grouped into 5 categories based on the topic modeling technique that we use: user interface, stack trace, large code snippets, web documents, and miscellaneous. The topics are supported by a similar numbers of questions.

In the future, we plan to extend this study by investigating more questions from StackOverflow and from other question and answer web sites. Our current study only investigates the 5 high-level topics. We plan to try various numbers of topics and various topic modeling techniques, and investigate lower level topics in future work.

7. REFERENCES

[1] http://stackoverflow.com/.

[2] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor. Software traceability with topic modeling. In *ICSE (1)*, 2010.

[3] A. Bacchelli, L. Ponzanelli, and M. Lanza. Harnessing stack overflow for the IDE. In *In Proceedings of RSSE 2012: 3rd International Workshop on Recommendation Systems for Software Engineering*, 2012.

[4] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies. *J. ACM*, 57(2), 2010.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[6] G. Bougie, J. Starke, M.-A. Storey, and D. German. Towards understanding twitter use in software engineering: Preliminary findings ongoing challenges and future questions. In *International Workshop on Web 2.0 for Software Engineering*, 2011.

[7] S. Breu, R. Premraj, J. Sillito, and T. Zimmermann. Information needs in bug reports: improving cooperation between developers and users. In *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, CSCW '10, pages 301–310, New York, NY, USA, 2010. ACM.

[8] D. Cai, X. Wang, and X. He. Probabilistic dyadic data analysis with local and global consistency. In *ICML*, 2009.

[9] T.-H. Chen, S. W. Thomas, M. Nagappan, and A. E. Hassan. Explaining software defects using topic models. In *MSR*, 2012.

[10] L. A. Dabbish, H. C. Stuart, J. Tsay, and J. D. Herbsleb. Social coding in github: transparency and collaboration in an open software repository. In *CSCW*, 2012.

[11] S. Huh and S. E. Fienberg. Discriminative topic modeling based on manifold learning. *TKDD*, 5(4), 2012.

[12] M. Monperrus and M. Mezini. Semi-automatically extracting FAQs to improve accessibility of software development knowledge. In *ICSE*, 2012.

[13] D. Pagano and W. Maalej. How do developers blog?: an exploratory study. In *MSR*, pages 123–132, 2011.

[14] K. Somasundaram and G. C. Murphy. Automatic categorization of bug reports using latent dirichlet allocation. In *ISEC*, pages 125–130, 2012.

[15] Y. Tian, P. Achananuparp, I. N. Lubis, D. Lo, and E.-P. Lim. What does software engineering community microblog about? In *MSR*, 2012.

[16] C. Treude, O. Barzilay, and M.-A. D. Storey. How do programmers ask and answer questions on the web? In *ICSE*, pages 804–807, 2011.

[17] S. Wang, D. Lo, Z. Xing, and L. Jiang. Concern localization using information retrieval: An empirical study on linux kernel. In *WCRE*, 2011.

[18] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In *ECIR*, 2011.