Software Forums Tagging Automation

[Making use of general trends of software forums for automation]

Diksha Gohlyan
Computer Science Department
NC State University
Raleigh, NC, 27606
dgohlya@ncsu.edu

Ayush Gupta
Computer Science Department
NC State University
Raleigh, NC, 27606
agupta25@ncsu.edu

Rajat Shah
Computer Science Department
NC State University
Raleigh, NC, 27606
rshah6@ncsu.edu

ABSTRACT

Traditional information retrieval techniques perform poorly on software forums as the complexity of software systems leads to use of several jargons in posts and duplicate content. Manually filtering relevant answers in these long threads is tedious and confusing. Searching for relevant answers is particularly difficult in software forums as the source code complexity can have a variety of issues because of the way they are expressed. Moreover, many a times users paste excerpts from different internet resource leading to duplicate content.

By incorporating tagging in these software forums, the process of retrieving relevant information can be expedited. Software forum tagging is an efficient way to efficiently retrieve information from a complex resource. It is lightweight and greatly improves different social and technical aspects of software development. The data present in large software forums often becomes more structured and searchable if the posts are associated with tags. With the help of tags, the underlying search system can rank the results in way that it favors posts having similar tags over other posts. In most of the implementation at present, tagging is done manually. This means that whenever a new post is made in a forum, the user needs to search for the corresponding tag and add it to the post. Manually tagging different software forums is time consuming and often prone to errors. Moreover, users have the tendency to avoid adding all the relevant tags in the interest of time and hence this makes the available tags as well insufficient. This makes automatic tagging of post in software forums a very important task. Once the existing tag vocabulary is analyzed, tags for incoming work items could be suggested. Strong candidates for suggestions are tags that have extensively been used in the near past such as lifecycle related tags and tags that have been applied to work items in the same category. There are also many challenges associated with it, since it requires correct parsing of the forums and determining accurately the list of associated tags. This report summarizes the techniques that can be used to automate the process of tagging software forums.

Keywords

Software Tagging; Forums; Data Mining; Feature Extraction; Technical Forum

1. INTRODUCTION

An online forum is a web application for holding discussions. Users post questions, usually related to some specific problems, and rely on others to provide potential answers. Within a forum, there are many threads. And in each thread, there are many posts. Software forums contain a wealth of knowledge related to discussions and solutions to various problems and needs posed by various developers and users. Therefore, mining such content is desirable and valuable

Forum thread usually consists of an initiating post and a number of reply posts. The initiating post usually has several questions and the reply posts usually contain answers to the questions and perhaps new questions or clarifying information or some kind of feedback. The threads at times grow long either because the issue is hard to solve or new independent initiating questions are posted in the same thread. The new initiating questions will have more replies and the process continues. We refer to a sequence of posts related to a particular initiating question post as a question-answer

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

conversation. Within a thread, there could be multiple independent question-answer conversations each starting with a different initiating question post.

Tagging is a popular concept in Web communities. With tagging, an external metadata is applied to Web blogs, Webpages, and other Web objects. This can then be used to search data, data description, identification, bookmarking, etc. By tagging a software forum, faster and efficient retrieval of information can be done. This eventually helps developers' communication and bridges the gap between social and technical aspects. Tagging provides a simpler method to annotate, in which a developer could tag his artifacts so that others could search and find them. Developers can also see the broad categories of the tagged artifacts in terms of matching subjects, purposes, or functions. Tagging is also used for artifact organization. Despite its importance in supporting informal activities in software development, there is little research on the crucial automated supports for software tagging. Tagging has been shown to be a lightweight and useful mechanism for improving the communication and bridging the gap between social and technical aspects in software development. Auto-tagging has also been investigated in the Web research and social computing communities. However, there has been little research on auto-tagging for evolving software. The introduction of tags into software development raises the question of how the informality of tagging affects the process of developing software and how a typical software development process can take advantage of the characteristics of tags. Tagging is not a new concept to software engineering. Tags have been used for decades for annotating check-in and branching events in software version control systems, as well as for documenting bugs in bug tracking systems. Due to these inconsistencies on the term tagging, Christoph Treude, et. al defined a tag as follows: *A tag is a freely chosen keyword or term that is associated with or assigned to a piece of information. In the context of software development, tags are used to describe resources such as source files or test cases in order to support the process of finding these resources.*

To automatically search for relevant answers, typically developers search via a standard search engine (e.g. Google) or specialized search engines in software forums. The former approach will return many webpages many of which are often not relevant to answering questions. The later approach would return specialized pages from software forums that often contain answers to various question. However, even in the second approach returned answers could be numerous.

One of the key findings of previous study (as elborated later) implies that Software developers use tags for three main purposes: 1) categorization of artifacts in different broad concerns, goals, subjects, functions in the system, 2) organization of artifacts in the project, and 3) support for searching of the artifacts. This is consistent with prior findings. Interestingly, many work items that have the similar concerns, goals, subjects, or functions should have additional tags to further and better characterize or describe them with regard to those aspects. Importantly, there is a large percentage of work items that did not have any tag. According to prior research [2], developers recognize the important roles of tags in software development. Therefore, a more accurate and efficient auto-tagging tool that takes into account the contents of work items would improve the current state of software tagging.

To leverage the wealth of information in software forums, [23] propose a framework to find relevant answers from software forums. Their framework consists of two main steps. First, they propose an engine that automatically classifies and tags posts in software forums as: answers, clarifying answers, clarifying questions, feedbacks - both positive and negative, and junk e.g., "today I am happy". Users could then focus on reading only the questions/answers including those buried deep inside long threads rather than the entire posts. Some questions with correct answers (based on the positive or negative feedbacks) could also be identified. Second, Swapna Gottipati, et.al built a semantic search engine framework that uses the inferred semantic tags to recover the relevant answers from the threads

In one of the case studies by Christoph Treude, et. al, they show how tagging can help bridge the gap between social and technical aspects in software development. Their findings indicate that lightweight informal tool support, prevalent in the social computing domain, may play an important role in improving team-based software development practices. They examine how a software development team uses tags for work items, both for individual use and for collaboration. They gathered data through the inspection of the project repositories and by conducting interviews with developers on the team. Their main contribution is the identification of different ways in which tags support informal processes in software development and thus fill the gap between social and technical aspects of artifacts. The main advantages of using tags in software development are their flexibility and their lightweight, bottom-up nature. While fields such as operating system, milestone, level of effort, or cross-cutting concerns could be part of fixed schemata, this would add overhead. Tags add the same functionality without implying administrative changes. Tags can also be implemented on a fine-grained level, e.g. for methods and fields. This would enable tagging across different types of content and thus would further support collaborative organization of artifacts. Furthermore, they examine how tagging was adapted by software developers to suit their needs and they identify potential tool enhancements.

This paper is organized as follows: Section II describes the background and related work. Section III details about other studies similar to current experiment. Section IV is about out commentary and section V is about the results. Section VI and VII describe future work and conclusion respectively.

2. BACKGROUND AND RELATED WORK

2.1 Usage patterns of collaborative tagging systems

Collaborative tagging describes the process by which many users add meta-data in the form of keywords to shared content. Recently, collaborative tagging has grown in popularity on the web, on sites that allow users to tag bookmarks, photographs and other content. There are some regularities in user activity, tag frequencies, kinds of tags used, bursts of popularity in bookmarking and a remarkable stability in the relative proportions of tags within a given URL. In a study of Usage patterns of collaborative tagging systems Data was analyzed to uncover patterns among users, tags and URLS. The analysis was performed on two sets of Delicious data, which were retrieved between the morning of Friday, June 23 and the morning of Monday June 27, 2005. Both of the

datasets were created by first retrieving public RSS feeds and then crawling a portion of the website. The combined tags of many users' bookmarks give rise to a stable pattern in which the proportions of each tag are nearly fixed. Empirically, usually after the first 100 or so bookmarks, each tag's frequency is a nearly fixed proportion of the total frequency of all tags used. This stable pattern can be explained by resorting to the dynamics of a stochastic urn model originally proposed by Eggenberger and Polya to model disease contagion. A number of now prominent web sites feature collaborative tagging. Typically, such sites allow users to publicly tag and share content, so that they can not only categorize information for themselves, they can also browse the information categorized by others. There are therefore at once both personal and public aspects to collaborative tagging systems. In the paper Usage patterns of collaborative tagging systems [1] the authors analyzed the structure of collaborative tagging systems as well as their dynamic aspects. Specifically, through the study of the collaborative tagging system Del.icio.us, they were able to discover regularities in user activity, tag frequencies, kinds of tags used and bursts of popularity in bookmarking, as well as a remarkable stability in the relative proportions of tags within a given URL. They also presented a dynamic model of collaborative tagging that predicts these stable patterns and relates them to imitation and shared knowledge. They concluded with a discussion of potential uses of the data that users of these systems collaboratively generate. It was observed that collaborative tagging users exhibit a great variety in their sets of tags; some users have many tags, and others have few. Tags themselves vary in frequency of use, as well as in what they describe.

2.2 How tagging helps bridge the gap between social and technical aspects in software development

Another study by Christoph Treude, Margaret-Anne Storey helped us to understand how Tagging helps bridge the Gap between Social and Technical Aspects in Software Development. Their main contribution is the identification of different ways in which tags support informal processes in software development and thus fill the gap between social and technical aspects of artifacts. Furthermore, they examined how tagging was adapted by software developers to suit their needs and we identify potential tool enhancements. Software development is recognized to be one of the most challenging management tasks performed by humans [22]. The larger systems get and the more complicated the compositions of the developing teams are, the more obstacles there are on the way to the release of a software system. Since most software systems are developed by teams, effective coordination and communication are crucial to the success of software projects.

The concept of tagging, as it is currently used, comes from the social computing domain. Social computing technologies, sometimes referred to as Web 2.0, are seeing rapid adoption by emergent communities on the web. Key examples include Facebook (facebook.com), YouTube (youtube.com) as well as community based recommender systems such as CiteULike (citeulike.org), TripAdvisor (tripadvisor.com) and flickr (flickr.com). Tagging is used by many of these systems and is often referred to as social bookmarking.

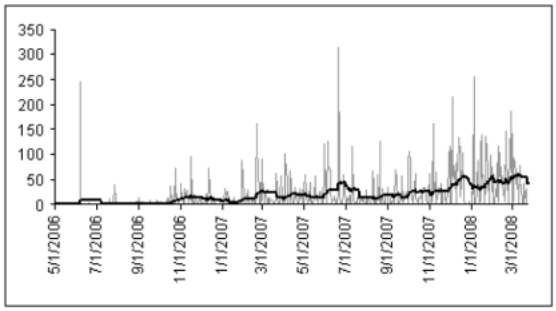

Figure 2. Tag uses per day

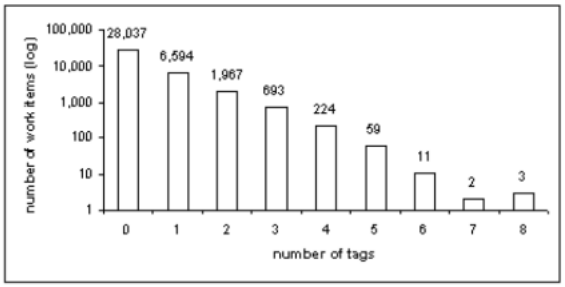

Figure 3. Distribution of tags (log scale)

The success of tags is closely related to their bottom-up nature: tags do not have to be pre-defined, every user can choose their own tags, and the number of tags per item is arbitrary. Based on these characteristics, tags are used to classify items in an informal way, and they stand in contrast to formal top-down classification mechanisms. Golder et al. and Hammond et al. provide overviews of tagging systems and classify the main reasons for user tagging. A more detailed study was done with the photo sharing website flickr . A common finding across these studies is that users tag to provide information on an artifact (e.g. what an artifact is or to refine a category) and for organizing artifacts. Part of the success of tagging comes from allowing users to define their own vocabulary. Information retrieval is also enhanced by community tagging. The predominant reason for the use of work item tags is categorization. While the Jazz interface already provides an opportunity to categorize work items (Filed Against field in Figure 1), tags are more flexible. The category tree can be altered, but this would change the available categories for the whole team and is still only one-dimensional and thus not appropriate for cross-cutting concerns. Tags play an important role in collaboration. In almost half of all the cases of applying a tag to a work item, the developer who applied the tag is not the one who created the work item, and in only less than one third of all cases, the developer applying the tag owns the work item. In other words, most tags are created for work items owned by somebody else.

2.3 Fuzzy set approach for automatic tagging in evolving software

Another study by Al-Kofahi, A. Tamrawi, T. Nguyen, H. Nguyen, and T. Nguyen, gave us an idea about Fuzzy set approach for automatic tagging in evolving software. Their goal was to identify the key components in an automatic tag-

ging tool for software artifacts, especially an accurate tool that is efficient in the evolutionary process of software development. With that goal in mind, they first investigated the current usages of tags in a real-world software project. We conducted an empirical study on the IBM's Jazz repository and focused on software tags for work items. Jazz's repository contains the real-world development data from IBM for more than 3 years. A work item is a generalized concept of a development task. It contains a summary, a description, a tag, and relevant software artifacts including source code, requirements, test cases, discussions, etc. In the empirical study, the key questions include 1) what are the general purposes and types of tags used in software development, 2) what are the common characteristics of artifacts that share assigned tag(s) from developers, 3) with the current tagging, whether or not the similar work items with the same/similar characteristics/purposes share any tag(s), and 4) with current tagging supports, whether or not the tags are sufficient to distinguish them and to serve those general purposes. The results of this study showed that developers use tags for three main purposes: 1) categorization of artifacts in different broad concerns, goals, subjects, functions in the system, 2) organization of artifacts in the project, and 3) support for searching of the artifacts. With the motivation from this empirical study, the authors developed an accurate, automatic tag recommendation tool, TagRec, that is able to take into account users' feedbacks on tags, and is very efficient in coping with software evolution. The core of TagRec is an automatic tagging algorithm for textual artifacts that is based on the fuzzy set theory.

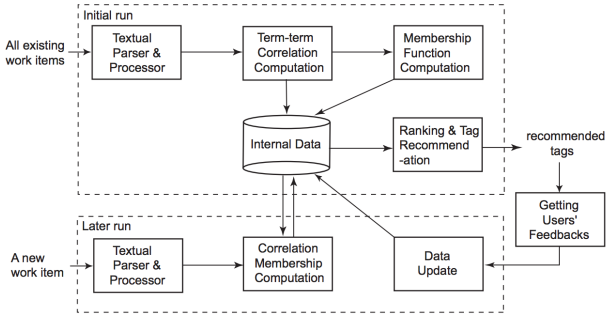

Fig. 1. TagRec Architecture

They also conducted an empirical evaluation of TagRec in tag recommendation using the IBM's Jazz data. They ran TagRec on Jazz's work items and requested human subjects to evaluate the results from TagRec to see if the resulting tags are good descriptions in terms of both goals, subjects, concerns, or functions of the work items. The same experiment was also carried out to evaluate how well TagRec recommends additional tags to already tagged work items. The results show that TagRec is very time efficient and provides useful recommendation tags with high precision and recall. The third evaluation was conducted in which TagRec was trained in a set of work items and used to recommend for the other set of items in the same project. TagRec also gave high quality results in term of both precision and recall.). This study confirms the three purposes of tag usages: categorization, organization, and searching. Work items that share tags are often related to the same/similar categories (concerns), goals, or subjects. The work items with the same or similar categories, goals, or subjects generally share the

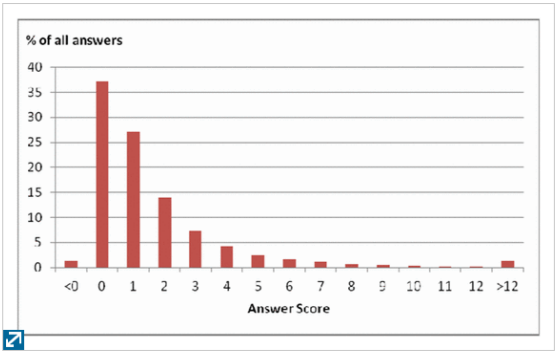

Figure 1. Distribution of answer scores on SO

same tags (with few exceptions). It was found that there are several missing tags in artifacts and more precise tags are desired. This paper introduces an accurate, automatic tag recommendation tool that is able to take into account users' feedbacks on resulting tags, and is very efficient in coping with software evolution. The core technique is a fuzzy set based automatic tagging algorithm. The unique features of this algorithm from existing ones include its ability 1) to efficiently handle the tag recommendation for new work items introduced during software development, 2) to recommend to a work item with tags that do not even occur within its own textual content, 3) to improve the future tagging by taking into account the users' feedbacks on the current resulting tags, and 4) to assign tags to items while maintaining the already assigned tags. So we can say that TagRec is time efficient and well-suitable for interactive uses in daily development. Importantly, it is very accurate with high recall (76%) and precision (50%).

2.4 Finding relevant answers in Software Forum

So, the next thing that comes into mind from tagging, as far as technical community is concerned, is, finding relevant answers in Software Forums. Software forums are web applications where the programmers look for solutions to their coding problems. A very widely used example of a software forum is www.stackoverflow.com where some programmers post their problems and others provide solutions to those. Traditional information retrieval techniques perform poorly on software forums as the complexity of software systems leads to use of several jargons in posts and duplicate content. Manually filtering relevant answers in these long threads is tedious and confusing. To address these issues, a paper by Swapna Gottipati, David Lo, Jing Jiang, 2011, proposes a semantic search engine framework which infers semantic tags of posts and uses them for effective search results. Forum thread usually consists of an initiating post and a number of reply posts. The initiating post usually has several questions and the reply posts usually contain answers to the questions and perhaps new questions or clarifying information or some kind of feedback. The threads at times grow long either because the issue is hard to solve or new independent initiating questions are posted in the same thread. The new initiating questions will have more replies and the process continues. We refer to a sequence of posts related to a particular initiating question post as a question-answer conversation. Within a thread, there could

be multiple independent question-answer conversations each starting with a different initiating question post. Another interesting observation is that the majority of software forum threads contain questions and answers rather than junks or irrelevant contents. It is different from social networking site like Twitter [30]. In a thread, the first post is likely a question and the following posts may contain related information like clarifying posts or potential solutions and feedbacks. Thus, software discussion boards contain rich information similar to organized QA services like Yahoo! Answers that are designed specifically for question answering purpose. Unfortunately, this wealth of information is often hidden in many threads of posts, some of which are very long. In this study, they work towards analyzing software forums by proposing an approach to retrieve relevant answers to natural language queries, which are embedded in the mass of posts in software forums, via a tag inference approach. In this study, as the first step of our framework we propose an engine to automatically infer tags for posts in software forums. Their system could be used to aid other works requiring the tagging of software forum posts. For example, Hou et al. extracted questions from news groups and tagged them manually [11]. But their system could potentially be used to provide the tags automatically. Treude and Storey shows the usefulness of tagging in software engineering activities to link technical and social aspects in managing work items [29]. Al-Kofahi et al. infer tags from software artifacts in IBM Jazz [1]. Duan et al. propose methods to extract the questions semantically close to a queried question [8] and, Jeon at al. retrieve similar questions [14]. In the paper by Swapna Gottipati, David Lo, Jing Jiang, they infer tags from software forums and utilize the inferred tags to effectively retrieve relevant answers to user queries expressed in natural language. To evaluate their overall search engine framework, we conduct a user-assisted experiments where users are tasked to label returned answers to 17 technical software-related queries shown in the Table VII that are expressed in natural language returned by a standard information retrieval toolkit and our proposed framework. In this paper, they present a new approach to find relevant answers from software forums by leveraging an engine that automatically infers tags of posts in software forum threads. This tag inference engine automatically assigns 7 different tags to posts: question, answer, clarifying question, clarifying answer, positive feedback, negative feedback, and junk. We build a semantic search engine by leveraging the inferred tags to find relevant answers. Our experiments shows that our tag inference engine could achieve up to 67% precision, 71% recall, and 69% F-measure. The best result is obtained when using the SSA (Stop-Stem-Author) feature extractor and context dependent classification model implemented by Hidden Markov SVM. ‘ There is a lot of scope of future work in this field. The extracted tags can be further analyzed to extract answers with positive feedback and same can be sent to experts along with the questions with no answers. There is also a scope in future to detect noise in human written communication by looking into spelling variations. An automated approach to cluster the forum posts in hierarchical fashion is also a possible future extension.

2.5 What makes a good coding example? A study of programming QA in StackOverflow

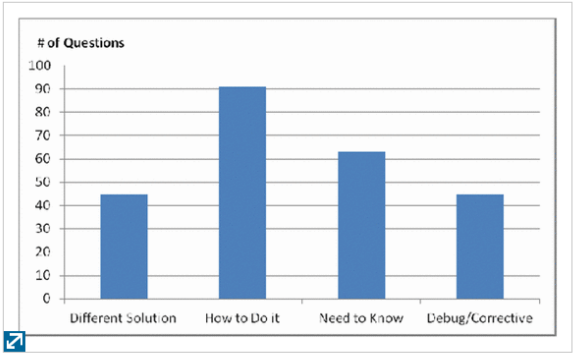

Figure 2. Distribution of question types in our sample

Now, moving on to the software forums, we studied various papers which analyze different aspects of asking questions and answering those on software forums. One of the most widely used such platform is Stackoverflow.com and most of the papers that we studied used this platform as a base for their research as well due to the huge user group of Stackoverflow.com. In a study by Jonathan Sillito et.al, they try to understand what makes a good code example, by study of programming QA in StackOverflow.com. Programmers learning how to use an API or a programming language often rely on code examples to support their learning activities. However, what makes for an effective ode example remains an open question. Finding the characteristics of the effective examples is essential in improving the appropriateness of these learning aids. To help answer this question they have conducted a qualitative analysis of the questions and answers posted to a StackOverflow.com. On StackOverflow answers can be voted on, indicating which answers were found helpful by users of the site. By analyzing these well-received answers we identified characteristics of effective examples. We found that the explanations accompanying examples are as important as the examples themselves. Their findings have implications for the way the API documentation and example set should be developed and evolved as well as the design of the tools assisting the development of these materials. There are other sources to find answers containing code, besides other developers. A questioner can refer to documentations, books, tutorials, etc., or search online resources available on the web. The first step in finding answers is to formulate a question. It can be articulated in natural language when the audience is a human being, or has to be expressed in some kind of query language to retrieve relevant examples. However, even if a developer is able to find a relevant example, then they are still bound by the quality of that example. In other words, finding a relevant example is necessary but not sufficient to ease the developer’s task; however, finding good examples can do the trick. StackOverflow.com is probably one of the most successful software Q/A platforms. Mamykina et al. conducted a statistical study of the entire SO corpus on the usage patterns to find out what is behind the immediate success of itSome software developers believe that SO has replaced web search/forums as their main source of finding answers to their programming problems [16]. These findings show that a majority of the questions will receive one or more answers (above 90%) and very quickly (with a median answer time of 11 minutes). Since this study was done in

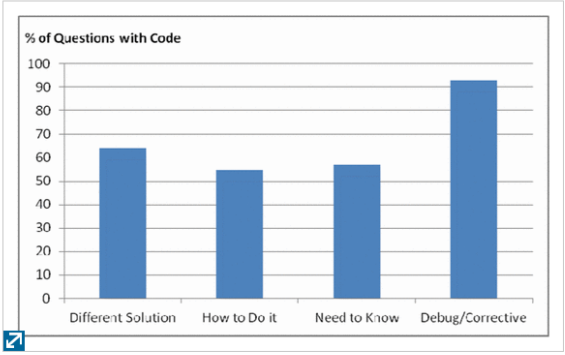

Figure 3. Percentage of questions containg code for each question type

View All

August 2010 and the SO web site has a rapid growth rate, we decided to recalculate some of the statistics presented in that paper using the publicly available data dump query tool. The mentioned study also lacks any statistical analysis of answer votes which is a main metric in our study design and analysis. The following statistics were calculated at the end of March 2012. There are some one million registered users on SO.

They have posted 2.78 million questions, 5.77 answers and 10.5 million comments. The average score of answers is 1.8 (ĬČ=6.2), of questions is 1.5 (ĬČ=5.9). Fig. 1 shows the distribution of answer scores. From this figure it is obvious that higher scores are quite rare. In fact, 87% of answers have a score of 3 or less, and only 2.3% of answers have a score of 10 or greater. Most of the questions receive at least one answer (91.8%) and receive it rather quickly (with median time of 15 minutes to receive the first answer). In the first hour, 70% of questions receive their first answer, and in the first day, 89% of them. Since accepting an answer is optional, the ratio of questions with an accepted answer is not very high (only 62.5% of questions with one or more answers have an accepted answer with the average answer score of2.94). The median time of accepted answers being posted is 24 minutes. 63% of accepted answers are posted in the first hour after the question been posted and 87% in the first day. APIs play a major role in developing software (65% of the questions in our sample ask a question related to some API). There are some studies addressing issues in learning APIs. Ko et al. found six learning barriers that end-user programmers would face. They found that programmers can overcome some of these barriers by finding relevant examples. However, they might face some other barriers when trying to adapt these examples to their needs . This is not often the case with examples on QA web sites, since the answer code is usually tailor-made to the questioner's needs. Robillard's findings based on a survey from professional developers show that API learning resources have a similar importance as the API design on the ease of learning for developers. He also found that developers want some well-structured and complete documentation and a comprehensive set of examples that show usage best practices for different scenarios. A study of discussions in a programming forum categorized three major types of questions (asking for a solution, using a wrong solution, and using a solution incorrectly) and found several obstacles imposed by the API for each category, for instance improper default solutions and hard to find elements. Finding relevant QA threads could be a challenge, especially in large forums. Gottipati et al. developed a text classification method on different types of entries in software forums and showed that their tool would outperform normal search engines in recovering relevant answers to user queries. Using the web as a main source for programmers to gather information has been the focus of some research studies. Programmers use the information available on the web for three main reasons: just-in-time learning, clarification, and remembering difficult things. They use different types of queries (natural language, code, or a mixture of both) based on their main objective. QA web sites can be a good source to provide the needed information to programmers. An analysis on how programmers conduct QA on SO resulted in several question categories (how-to, review, error, conceptual, etc.) and that questions containing code are very common for review questions. Besides the question type, other factors, such as the question posting date and time, the identity of the questioner, the technology in the question, the length of the question, and the availability of code in the question affect receiving good answers. The web can also provide alternative means of API documentation. A study showed that blogs cover 87.9% of a specific API's methods and provide tutorials and personal usage experiences. They also found that these sources are the main way of support for some niche communities. Using examples is a main source of help for both professional and end-user programmers. Both groups are trying to solve a problem by reusing those examples. For novice developers, having more knowledge about the architecture of a framework will be a major factor in adapting framework usage examples more efficiently, so it was suggested that learning a framework should start with teaching novices about the design of that framework. There are also tools designed to extract code from source repositories to provide them as examples. Some of them use mining technique to locate features; others extract common API usage patterns. These tools rely on queries consisting of keywords to find examples. Holmes et al. developed a tool that uses the current development context (i.e. types used in code under development) as the query to find similar code snippets that might be reused. The extracted code by all of these tools does not necessarily have a familiar context and is not always accompanied with helpful explanation. To overcome some of these shortcomings, Stylos et al. developed a tool that extracts some common usage scenarios using web search (such as object creation) and injects them into the API documentation; albeit for very limited scenario types. Documentation is also a main source of information for developers. Dagenais and Robillard studied the creation and evolution of the documentation, and found that different types of documentation are more appropriate for libraries and frameworks. They also found that mailing lists could be considered as bug reports and be used in the evolution of the documentation. Dagenais and Ossher built a tool that helps developers to create light-weight documentation while using a framework by defining different steps of a task and adding elements to each step and thus creating a set of guides. Nontrivial and infrequent knowledge about how to use elements of an API could be hidden under tons of API documentation text. Two studies defined different types of these hidden elements (called directives). Dekel also built a tool integrated to an IDE to present these important elements to developers while

they are using the API and showed that it improved the outcome of documentation reading for their study subjects.

2.6 Answering Questions about Unanswered Questions of Stack Overflow

In another study - "Answering Questions about Unanswered Questions of Stack Overflow" by Muhammad Asaduzzaman, Ahmed Shah Mashiyat, Chanchal K. Roy, Kevin A. Schneider[6], they have manually investigated a random sample of unanswered questions.

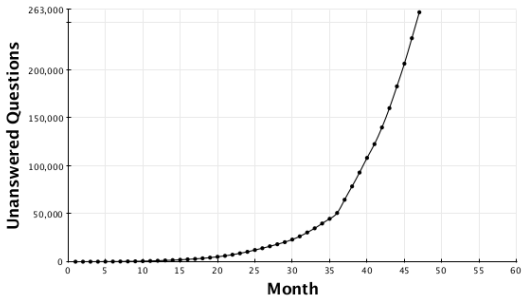

Fig. 1. Number of unanswered questions after each month

They use both question content and comments to understand why the questions were unanswered. Finally, they consider how this information can be used to better support the SO site and build classifiers for predicting the number of days a question will remain unanswered. Prior work on community QA services focused on studying the dynamics of community activities. Anderson et al.(Discovering value from community activity on focused question answering sites: a case study of stack overflow) used such knowledge to predict the long lasting value of a question and to decide whether a question has been sufficiently answered. They randomly sampled 400 unanswered questions, selecting 100 questions from each of the four years for which we have data3 . Commenting on unanswered questions is an indication that users are attempting to answer the question or are providing feedback. It is expected that all questions may not be answered in a crowdsourced QA site. However, unanswered questions in SO are still being viewed 139 times on average and this motivated us to investigate why such questions were remaining answered. Their qualitative analysis identified some characteristics of unanswered questions, which they use as a taxonomy for classifying unanswered questions.

2.7 Towards Discovering the Role of Emotions in Stack Overflow

Next, we studied a paper by Nicole Novielli et.al. , which deals with discovering the Role of Emotions in Stack Overflow. Today, people increasingly try to solve domain-specific problems through interaction on online Question and Answer (QA) sites, such as Stack Overflow. The growing success of the Stack Overflow community largely depends on the will of their members to answer others' questions. Recent research has shown that the factors that push members of online communities encompass both social and technical aspects. In this paper, they have described the design of an empirical study aimed to investigate the role of affective lexicon on the questions posted in Stack Overflow. Pubic data dump from Stack Overflow was used in the study. This includes all the questions, answers, votes and tags between 2008 to 2014. This data contains: a. 7,214,802 questions with 58% having accepted answers. b. 12,609,623 answers and 36,923 tags. Due to its wide range of potential applications, emotion recognition is an increasingly important research area in social software engineering and, more in general, in social computing. Instead, a gap exists in literature about the role of emotions and their expression in Stack Overflow. This research is also expected to have practical implications in terms of the definition of new guidelines for practitioners and other researchers who intend to improve emotional interface design. In particular, such enhancements would enable: (i) better user experience and engagement; (ii) the development of new tools for embedding emotional intelligence into online QA communities to support both community members and managers.

2.8 Mining Successful Answers in Stack Overflow

In another study named Mining Successful Answers in Stack Overflow by Fabio Calefato et. Al., theye argue that also the emotional style of a technical contribution influences its perceived quality. In this paper, they investigate how Stack Overflow users can increase the chance of getting their answer accepted and focus on actionable factors that can be acted upon by users when writing an answer and making comments.

They found evidence that factors related to information presentation, time and affect all have an impact on the success of answers. Dataset was from the official SO data dump, updated on September 2014. SO official datasets always report the reputation score of users when the dump is created. The SO reputation system assigns users to the following categories, based on the reputation score gained: New Users (score<10), Low Reputation Users (score in [10, 1000[), Established Users (score in [1000, 20k[), and Trusted Users (score ÂŢ 20k). Since SO allows users to gain at most 200 reputation points per day, it is reasonable to assume that the reputation category of the largest majority of users stays unvaried over a month [3]. Therefore, dataset was built for analysis by considering the answers to the questions posted during the last month of the dump (14th Aug.-14th Sept.). To enlarge the dataset, they also considered answers to the questions posted from the 5th of April to the 5th of May, corresponding to the last month of the previous dump of May 2014. Out of all the four variables considered in the affective dimension, the positive sentiment expressed in the answerer's comments has the most significant impact on success, immediately followed by the negative sentiment within comments that, conversely, negatively influences the probability of getting an answer accepted. Among the 311,733 answers in our dataset, only 64,043 (21%) originate a follow-up discussion involving also the answer's author. Still, the impact of sentiment expressed in comments is significant.

TABLE II. RESULTS OF LOGISTIC REGRESSION FOR PRESENTATION QUALITY, AFFECT AND SOCIAL FACTORS

VARIABLE	COEFFICIENT ESTIMATE	ODDS RATIO	p < 0.05
Intercept	-2.17	0.1142	*
Presentation Quality			
Number of URLs	0.0760	1.0790	*
Presence of Code Snippet	0.2800	1.3231	*
Length	0.0004	1.0004	*
Uppercase Ratio	-0.2933	0.7458	
Affect			
Answer Positive Sentiment	0.0214	1.0216	
Answer Negative Sentiment	0.0434	1.0444	*
Comment Positive Sentiment	0.0906	1.0948	*
Comment Negative Sentiment	-0.0377	0.9630	*
Reputation			
Answerer's Reputation Score	2.19E-06	1.0000	*
Answerer's Number of Badges	0.0002	1.0002	*
Asker's Reputation Score	4.74E-06	1.0000	*

To further investigate this phenomenon, they analyze the top 100 answerers' comments with the highest positive and negative scores, respectively. This analysis reveals that comments are very rich in affective lexicon. We observe that SO users express a wide variety of affective states in comments. This might occur because comments are seen as a âĂŸfree zone' since reputation mechanisms do not apply to comments. The findings of this study provide a set of user-driven guidelines for contributors who intend to increase the chance of getting their answers accepted. In particular, potential contributors should be prompt in replying, as also emphasized by previous research. Another recommendation is to comply with the SO presentation quality standard by including code snippets, as well as URLs for providing contextual information. Besides, contributors should avoid a negative attitude towards information seekers. These findings inspire directions for future work. First, the wide variety of affective states expressed in comments recommends a more fine-grained investigation of the role of emotions in SO since different affective states might be relevant to different contexts and tasks. Besides, it would be interesting to investigate to what extent emotions and personality traits shown in questions influence the answering behavior. Finally, there is a need for tuning state-of-the-art resources for sentiment detection by adapting them to domain-dependent use of lexicon.

3. OTHER STUDIES

Automatic tagging is also popular in Web and social computing areas. In those areas, there exist literature surveys on different tagging systems and classifications of user tags [21], [24]. Song et al.[25] use spectral recursive embedding clustering and a two-way Poisson mixture model for real-time tagging of Web documents. TagAssist [22] is an automatic tagging tool for new Web blog posts by utilizing existing tagged posts. It performs lossless compression over existing tag data. P-TAG [16], a method which automatically generates personalized tags for Web pages, produces keywords relevant both to textual content and to the data residing on the surfer's Desktop using Semantic Web technology. Brook et al. [4] propose auto-tagging with hierarchical clustering. They show that clustering algorithms can be used to reconstruct a topical hierarchy among tags.

4. OUR COMMENTARY

We can see that how usage patterns of collaborative tagging system evolved [21] and many awesome research on it has led to our final paper mining successful answers in stack overflow. We have observed that collaborative tagging users exhibit a great variety in their sets of tags; some users have many tags, and others have few. Tags themselves vary in frequency of use, as well as in what they Describe.

The prevalence of tagging with a very large number of tags and according to information intrinsic to the tagger demonstrates that a significant amount of tagging, if not all, is done for personal use rather than public benefit. Nevertheless, even information tagged for personal use can benefit other users. Apart from this, Tags are one way to look at the informal side of software development in a team setting. Through understanding how developers use tags in their daily work, we can extend our knowledge on informal aspects of software development and further-more understand how a social computing technology, such as tagging, is adapted by software developers.

Our research has shown how the social computing mechanism of tagging has been adopted and adapted by a large software development team.[6] Not only is tagging used to support informal processes within the team, it has also been adapted to the specific needs of software developers.Despite the importance of tags, there exists limited support for automatic tagging for software artifacts.

We found an algorithm to support this. The core technique is a fuzzy set- based automatic tagging algorithm.[10] The unique features of this algorithm from existing ones included its ability to efficiently handle the tag recommendation for new work items introduced during software development, to recommend to a work item with tags that do not even occur within its own textual content, to improve the future tagging by taking into account the users' feedbacks on the current resulting tags, and to assign tags to items while maintaining the already assigned tags.

We then used this algorithm to find relevant answers in software forums.[23] This tag inference engine automatically assigns 7 different tags to posts: question, answer, clarifying question, clarifying answer, positive feedback, negative feedback, and junk.

We further studied programming Q/A in StackOverflow to find out what makes a good coding example.[3] We found that code examples and the accompanied explanation are two inseparable elements of recognized answers. We also found attributes of these two elements and techniques being used to shape these elements, such as making concise examples, shaping the explanation based on the questioner's expertise level, and making use of the question context to decrease the cognitive distance. We further extended our study to determine unanswered questions in Stack Overflow. 7.5% of SO′s total questions are unanswered.

We provide a taxonomy for this minority group to help understand why there is a delay in answering questions and whether a question requires further information.[20] Using the taxonomy, post metrics and information about askers, we built a classifier that predicts how long a question will remain unanswered in SO. Moderators can use the classifier to predict questions likely to have delayed responses and can consider taking actions to promote an earlier response, such

as editing the question or routing the question to an expert member. Identifying the quality of a question can also help in this regard, which we plan to consider in a future study.

We extended our study towards discovering the role of emoticons in stack overflow.[15]Due to its wide range of potential applications, emotion recognition is an increasingly important research area in social software engineering and, more in general, in social computing. Instead , a gap exists in literature about the role of emotions and their expression in Stack Overflow.

Finally we move to mining successful answers in stackoverflow.[7] the success probability of an answer in a logistic regression framework since it allows us to reason about the significance of one factor given all the others. We use the acceptance vote as the dependent variable and presentation quality, affective, temporal, and social metrics as independent variables.

5. NEW RESULTS

Prior work on community Q&A services focused on studying the dynamics of community activities. Anderson et al.[1]used such knowledge to predict the long lasting value of a question and to decide whether a question has been sufficiently answered. Admaic et al.[11]leveraged community knowledge to predict the best answer. Harper et al.[8] compared answer quality of community-based question answering sites and found that topic, type, and payment are the significant contributors for getting the best answers. Yang et al.[12] applied a machine learning technique to predict not-answered questions in Yahoo! Answers. However, the policy that governs not-answered questions is not the same for Yahoo! Answers. These were the study on answers and site dynamics. However, the focus of answering questions about unanswered questions focused on unanswered questions.

The topics of questions have been used in post categorization [2] [5] and in studies attempting to predict the answer probability and quality associated to a given post [20] [8]

Tagging, as defined here, has not been extensively re searched in a software engineering context. Some systems support social bookmarking, for example Code Snippets (bigbold.com/snippets) and ByteMycode (bytemycode.com). They support social tagging of source code, but require the user to post code fragments on public servers before tags can be applied. A recent tool that intersects social tagging with software development is described by Storey et al. [13]. Their tool TagSEA (Tags for Software Engineering Activities) is a collaborative tool to support asynchronous software development and uses the ideas of social tagging to support coordination and communication. A case study [14] showed that TagSEA provides the user-defined navigation structures that are lacking in traditional task annotations. Opposite to this bottom-up approach, the tools Concern Graphs [18] and ConcernMapper [19] enable developers to associate parts of source code with high level Concerns.

6. FUTURE SCOPE/OUR RECOMMENDATION

Tagging helps in searches, our recommendation for the same is to make the search more personalised according to the user's preferences. Informal language is not taken care of properly, so need to incorporate the fact that informal language is used in Software Forums. There have still not been any proceedings of using Natural language processing techniques to improve the quality of data and hence the result. Program analysis can further be used to enrich the semantic relations between source code and work items.

The paper usage patterns of collaborative tagging system talks about manual way of tagging, however the study results provided can further be used to automatically tag some context with some machine learning paradigm. And this tagging system can further be used for categorization. In most of the papers, the experiment done are limited with limited data available for the results. The experiment done should be on variety of languages and variety of forums with diverse platform and diverse set of data.

The generalizability of our findings could be regarded as another limitation. However, most findings in paper what makes a good answer have some overlaps with some other studies [9] [17] [5] which suggests that our results could be generalized to some extent. At least our findings can be used as hypotheses to be examined in later studies.

Interview, or survey with moderators or the well reputed members of the forums can help understand the overall problem better . In most of the papers, reputation score of the users is considered. However, growing score of the user seems more relevant considering new users.

Further role of tagging, sentiment analysis and emotions can be studied to have better findings of software forums which can help automate making these forums better by automated tagging, automated rankings, as well as further automated notifications and answers.

7. CONCLUSION

We concluded that tagging is an important concept in various places. It is highly used in machine learning to automate a lot of things. From determining good answers in stack overflow to finding charateristics and also determining unanswered questions. It gives a lot of opportunity in making software forums more robust by giving guidelines to the top developers and contributors ways of them making their answers better and the forums helpful. We also see that in future all these findings can be used to determine the best answers and boost up the rankings for the same. Moreover, it can further be used to automatically answer questions once a pattern is determined and through the machine learning algorithms used in the various papers.

8. REFERENCES

[1] J. K. A. Anderson, D. Huttenlocher and J. Leskovec. Discovering value from community activity on focused question answering sites: a case study of stack overflow. *KDD, pages 850âĂŞ858*, September 2008.

[2] C. C. H. D. C. D. C. J. Bosu, A. and Kraft. Building reputation in stackoverflow: An empirical investigation. *10th IEEE Working Conference on Mining Software Repositories*, 2013.

[3] J. S. F. M. S. M. N. C. Burns. What makes a good code example?: A study of programming qa in stackoverflow. *IEEE Press Piscataway, NJ, USA*, September 2012.

[4] s. C. H. Brook and N. Montanez. âĂIJimproved annotation of the blogo sphere via autotagging and hierarchical clustering. *15th international conference on World Wide Web*, 2006.

[5] O. B. C. Treude and M. Storey. How do programmers ask and answer questions on the web? *ICSE 2011, New York, NY, USA, 2011, pp. 804-807*, 2011.

[6] M.-A. S. Christoph Treude. How tagging helps bridge the gap between social and technical aspects in software development. *IEEE Computer Society Washington, DC, USA Âl'2009*, May 2009.

[7] M. C. M. F. Calefato, F. Lanubile and N. Novielli. Mining successful answers in stackoverflow. *In Proceedings of the 12th Working Conference on Mining Software Repositories (MSR)*, May 2015.

[8] S. R. F. M. Harper, D. Raban and J. Konstan. Predictors of answer quality in online qa sites. *CHI, pages 865âĂŞ874*, 2008.

[9] D. Hou and L. Li. Obstacles in using frameworks and apis: An exploratory study of programmers' newsgroup discussions. *ICPC 2011, 2011, pp. 91-100*, 2011.

[10] T. T. N. H. A. N. T. N. N. Jafar M. Al-Kofahi, Ahmed Tamrawi. Fuzzy set approach for automatic tagging in evolving software. *IEEE Computer Society Washington, DC, USA Âl'2010*, September 2010.

[11] E. B. L. Adamic, J. Zhang and M. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. *WWW, pages 665-674*, 2008.

[12] Q. L. X. W. D. H. Z. S. L. Yang, S. Bao and Y. Yu. Analyzing and predicting not-answered questions in community-based question answering services. *AAAI, pages 1273âĂŞ1278*, 2011.

[13] I. B. M.-A. Storey, L.-T. Cheng and P. Rigby. Shared waypoints and social tagging to support collaboration in software development. *CSCW âĂŹ06: Proc. of the 2006*, 2006.

[14] J. S. M. M. D. M. M.A. Storey, L.-T. Cheng. How programmers can turn comments into waypoints for code navigation. *IEEE Intl. Conf. on Software Maintenance*, 2007.

[15] F. L. Nicole Novielli, Fabio Calefato. Towards discovering the role of emotions in stackoverflow. *SSEâĂŹ14, Hong Kong, China*, November 2014.

[16] W. N. P.-A. Chirita, S. Costache and S. Handschuh. P-tag: large scale automatic generation of personalized annotation tags for the web. *WWW âĂŹ07: 16th Int. Conference on World Wide Web*, 2007.

[17] M. P. Robillard. What makes apis hard to learn? answers from developers. *IEEE Softw., vol. 26, pp. 27-34*, November 2009.

[18] M. P. Robillard and G. C. Murphy. Concern graphs: finding and describing concerns using structural program dependencies. *24th Intl. Conf. on Software Engineering*, 2002.

[19] M. P. Robillard and F. Weigand-Warr. Concernmapper: simple view-based separation of scattered concerns. *OOPSLA workshop on Eclipse technology eXchange*, 2005.

[20] M. A. A. S. M. C. K. R. K. A. Schneider. Answering questions about unanswered questions of stack overflow. *IEEE Press Piscataway, NJ, USA*, May 2013.

[21] B. A. H. Scott A. Golder. Usage patterns of collaborative tagging systems. *Journal of Information Science*, April 2006.

[22] S. C. Sood and K. Hammond. Tagassist: Automatic tag suggestion for blog posts. *International Conference on Weblogs and Social*, 2007.

[23] D. L. Swapna Gottipati and J. Jiang. Finding relevant answers in software forums. *IEEE Computer Society Washington, DC, USA Âl'2011*, November 2011.

[24] B. L. T. Hammond, T. Hannay and J. Scott. Social bookmarking tools (i): A general review,. *D-Lib , vol. 11, no. 4*, 2005.

[25] H. L. Q. Z. J. L. W.-C. L. Y. Song, Z. Zhuang and C. L. Giles. Real-time automatic tag recommendation,âĂİ in sigirâĂŹ08: Acm conference on research and development in information retrieval. *515âĂŞ522. ACM*, 2008.