Data Technologies and Applications

A survey on mining stack overflow: question and answering (Q&A) community

Arshad Ahmad, Chong Feng, Shi Ge, Abdallah Yousif,

Article information:

To cite this document:
Arshad Ahmad, Chong Feng, Shi Ge, Abdallah Yousif, (2018) "A survey on mining stack overflow: question and answering (Q&A) community", Data Technologies and Applications, https://doi.org/10.1108/DTA-07-2017-0054
Permanent link to this document:
https://doi.org/10.1108/DTA-07-2017-0054

For Authors

About Emerald www.emeraldinsight.com

A survey on mining stack overflow: question and answering (Q&A) community

Arshad Ahmad, Chong Feng, Shi Ge and Abdallah Yousif
Computer Science and Technology, Beijing Institute of Technology, Beijing, China

Received 22 July 2017
Revised 8 November 2017
Accepted 9 December 2017

Abstract

Purpose – Software developers extensively use stack overflow (SO) for knowledge sharing on software development. Thus, software engineering researchers have started mining the structured/unstructured data present in certain software repositories including the Q&A software developer community SO, with the aim to improve software development. The purpose of this paper is show that how academics/practitioners can get benefit from the valuable user-generated content shared on various online social networks, specifically from Q&A community SO for software development.

Design/methodology/approach – A comprehensive literature review was conducted and 166 research papers on SO were categorized about software development from the inception of SO till June 2016.

Findings – Most of the studies revolve around a limited number of software development tasks; approximately 70 percent of the papers used millions of posts data, applied basic machine learning methods, and conducted investigations semi-automatically and quantitative studies. Thus, future research should focus on the overcoming existing identified challenges and gaps.

Practical implications – The work on SO is classified into two main categories; "SO design and usage" and "SO content applications." These categories not only give insights to Q&A forum providers about the shortcomings in design and usage of such forums but also provide ways to overcome them in future. It also enables software developers to exploit such forums for the identified under-utilized tasks of software development.

Originality/value – The study is the first of its kind to explore the work on SO about software development and makes an original contribution by presenting a comprehensive review, design/usage shortcomings of Q&A sites, and future research challenges.

Keywords Mining, Software development, Information retrieval, Text mining, Software repositories, Stack overflow

Paper type Literature review

1. Introduction

1.1 Background

With the advent of rapid advancement and rise in the use of diverse social media technologies (Wang *et al.*, 2017; Khusro *et al.*, 2017; Jafari *et al.*, 2012; Borjigen, 2015; Udanor *et al.*, 2016), e.g., Q&A forums/communities, blogs, and Wikis, the software engineering research community has also started realizing and utilizing it for software development. Software development is a knowledge-intensive activity (Ahmad and Khan, 2008; Alnuem *et al.*, 2012; Khan *et al.*, 2012; Khan *et al.*, 2011) and the recent advancements in social media technologies have further pushed software developers to leverage it for knowledge sharing, learning, and collaborating with others. Besides, the innovations in social media technologies have also changed the shapes and ways of developing software, challenging old conventions about how developers learn and work with others. These different social media technologies thus serve not only as software repositories having structured/unstructured data about software development life cycle (SDLC), but also serve as a professional user base (Storey *et al.*, 2014; de Souza *et al.*, 2016; Storey, 2015; Parnin *et al.*, 2013; Pagano and Maalej, 2011; Tian *et al.*, 2012; MacLeod *et al.*, 2015).

The authors would like to acknowledge the support provided by the National 863 Project, China, under Research Grant No. 2015AA015404.

In recent years, the software engineering research community has endeavored to enhance software development through deeply mining and assessing software repositories, e.g., e-mail archived communication, source code changes, bugs repository, execution logs (Chen *et al.*, 2015; Godfrey *et al.*, 2008; Hassan, 2008), and online Q&A forums (Ponzanelli *et al.*, 2014a; Ponzanelli *et al.*, 2015; Treude and Robillard, 2016). These different research efforts depict the possibility of obtaining useful and real-world results via mining these repositories. Thus, enabling software developers and project managers to comprehend their software systems and eventually enhance the quality of their end products in a more timely and cost-efficient way (Chen *et al.*, 2015; Tichy, 2010).

Recently, specific achievements have been reported with mining and investigating the available structured data in these software repositories (Chen *et al.*, 2015). These achievements are gained due to research community interests, analyzing effort level required, the presence of useful knowledge, different tools support to mine/analyze, the nature of data (structured) itself, and availability of such repositories. Structured data refer to information that is organized following some specific data model or known form/structure. For example, source codes parse trees, call graphs, inheritance graphs, execution logs, and traces are all structured data software repositories.

However, the recent exponential surge in availability and forms of unstructured data in software repositories has also pushed the software engineering research community to mine and analyze the useful knowledge present in such repositories, i.e., different versioning systems, e.g., Git[1], SVN[2], CVS[3], archived communications, e.g., mailing lists, chat logs, and online forums, e.g., Q&A websites, mobile app stores, software artifacts, online video-sharing websites, e.g., programming tutorials shared on YouTube[4], and slide hosting services, e.g., technical presentations shared on SlideShare[5] (Bavota, 2016). Unstructured data refer to natural language text or information that is not organized by following some specific data model or known form/structure (Chen *et al.*, 2015). Despite the rise in availability of unstructured data (approximately 80-85 percent in software repository) and researchers focus, there are still some associated challenges, i.e., the lack of automated techniques for mining and understanding, which are believed to be an impediment for researchers and practitioners to efficiently utilize these repositories for software development (Chen *et al.*, 2015; Hassan, 2008; Blumberg and Atre, 2003).

The recent years have also witnessed enormous growth in developing various software development tools, languages, and platforms for various purposes due to diverse demands from developers and users. With this demanding pace of evolution, software developers also need to be skilled not only with their existing tools, languages, and platforms but also with every newer versions or feature released. Currently, most of the software developers use several online development communities to solve their problems by posing/discussing Q&A with other professionals in the community, e.g., Quora[6], Stack Exchange (SE)[7], Reddit[8], and GitHub[9]. Among all these communities, SE is deemed to be the most popular Q&A community due to the number of registered users, daily visits, and above all the satisfaction level of users. The SE Q&A community itself has a diverse set of communities covering different topics in mathematics, statistics, computer science/programming, and education. In total, there are 150 + Q&A communities on SE including stack overflow (SO) founded in 2008, the prominent site for software developers to discover and post Q&A about the entire software development spectrum.

The importance and abundance of unstructured data in software repositories for software engineering researchers can be realized from these sample statistics, e.g., GitHub[10] hosted about 46 K software projects in year 2009, 1 M in 2010, 10 M in 2013, 27 M in 2015, and in total, more than 47 M projects hosted so far as of 2016 (software projects growth ratio is about 586 times in around six years duration). On SE[11], every hour about 102 K users search for help, and about 6 K hours of fresh video contents are shared on YouTube[12].

Similarly, SO Q&A community is aimed to serve diverse software development topics on tools, platforms and other related software development issues. The SO[13] community itself serves more than 40 M professionals and novice programmers every month and has more than 6 M registered users, approximately 12 M questions, 20 M answers, 51 M comments, and 46 K tags on various issues/topics of software development.

The questions posted by software developers on SO usually are long unstructured natural language texts containing code snippets and weblinks, which makes it more challenging for researchers/practitioners to automatically mine and analyze those SO posts. Some of the frequently reported associated challenges in automatically mining/analyzing such natural language texts are vagueness, impreciseness, grammatical mistakes, spelling errors or typos, noisiness, synonyms, and unknown acronyms (Carreno and Winbladh, 2013; Chen *et al.*, 2014; Iacob and Harrison, 2013; Chen *et al.*, 2015). We illustrate below some sample chunks of SO posts (Q&A and comments) to know their characteristics and associated challenges in automatically mining, analyzing and understanding them.

In following Question No. 1, the poster asks on SO about a feature possible in Android Studio similar to Xcode feature having title "pragma mark equivalent in Android Studio."

Question No. 1: "XCode have a feature called pragma mark It's very util and I'm looking for anything similar into Android Studio it can be native or a plugin[14]."

Answer X: "In Android Studio you can add regions using the steps […]."

Comment: "Cool, I wish it showed in the Structure view (CMD + 7) in bold like it did in the Xcode dropdown but there's always going to be development tool differences."

In Question No. 1, there is some vagueness present, as reported in Chen *et al.* (2015). The word used "util" is incomplete and imprecise, hence, processing such kinds of vague words used in SO questions is quite challenging to automatically mine and analyze.

The following Question No. 2 is posted on SO about adding/requesting a feature in Android Studio having title "Add unimplemented methods."

Question No. 2: "In the Eclipse IDE there is a great feature allows you to add (implement) all of the required methods of the particular class. I'm looking for this feature in the Android Studio IDE, but without success so far. Is there something similar? For me it is one of the key-features and can't live without[15]."

Answer X: "Of course there is. It is called Implement methods or Override Methods. The default shortcut is CTRL-I and CTRL-O. See descrption of Implementing Methods and Overriding Methods."

Comment: "Ok, but this is not what I'm asking for. I don't want to choose methods to implemet. I want IDE to do it for me like Eclipse were doing. For example when I clicked "Add unimplemented methods" inside any Activity extented class all of these onCreate() onPause() onResume() were generated."

Comment: "the answer below by pbespechnyi is the right one."

Comment: "Yup 'ALT+ENTER' should be the right answer not 'CTRL-O.' "

Answer Y: "Alt + Enter - on class definition; Ctrl + I – in class body to show list of unimplemented methods. Ctrl + O – in class body to show list of override methods."

In Question No. 2, many difficulties exist for processing such texts. For instance, there are several grammatical, typos or spelling errors (descrption, implemet, extented), as reported in Chen *et al.* (2015). Other difficulties include the text in the comments or answers refers to some weblinks or answers. For example, in one of the above comments, the user refers the poster to see "pbespechnyi," so it is quite challenging to mine, analyze, and understand such kind of posts whether "pbespechnyi" is a person name or some other word having different meanings.

The following Question No. 3 is posted by software developer about a missing feature in Android Studio having title "Android Studio is missing permissions."

Question No. 3: "I recently updated Android Studio to 2.2.2. The IDE is missing permissions from both Manifestclass and AndroidManifest.xml (code suggestion). I knew

I can add the permission anyways, but the code suggestion helps me to make sure I typed it correctly and check what permissions are available[16].”

Comment: “AFAIK, code suggestions is based entirely on your compileSdkVersion. You might file a feature request to extend code suggestions to include things that were removed from the SDK.”

In the comment of Question No. 3, there is a feature request proposed by Android Studio Software Company to the poster of Question No. 3 to file an official feature request on Android Studio forum.

The aims of the examples above are twofold: on the one hand, it raises the issues of “SO design and usage” perspective, such as question quality and the practice of including proper tags in the posted question. On the other hand, it depicts the value that is shared in such posts for software development on SO, i.e., “SO content applications.” In these examples, the developers are seeking help for coding, solution/knowledge or requesting for the missing feature in software development tools. Besides, it is also evident that the above posts are unstructured and have various levels of vagueness, impreciseness, and grammatical mistakes (spelling errors or typos), as reported in Chen *et al.* (2015). Thus, the essence of these examples is to show the characteristics of SO posts and the associated challenges for mining these posts to get the value of it. Besides, highlighting the issues of posting quality questions or the lack of adopting quality questions guidelines and finally the type of knowledge that is searched/shared by developers regarding finding the code solutions or suggesting the missing features in existing software development tools. The SO repository is piled with thousands of such unstructured posts every day, which makes the manual analysis extremely hard and almost impossible and thus, it ultimately demands to have more efficient automated techniques for mining, analyzing, understanding, and exploiting such rich unstructured data efficiently for software development.

Thus, the challenge for researchers and practitioners remains to be addressed as for how such useful knowledge present in SO posts can efficiently be mined, extracted, analyzed, and used not only by software developers but also by software development companies to improve the quality of software cost efficiently. Besides, mining this useful knowledge will also enable the future Q&A community providers to know the shortcomings of design/usage features in existing Q&A forums or communities. Since mining and analyzing will enable the researchers and practitioners to know or will get answers of what, why, when, and how do software developers ask for in SO posts about software development?

The recent research studies have started to focus mining the pool of knowledge present on Q&A websites in the form of Q&A, comments, and discussion threads which are quite unstructured, e.g., on SO (Barua *et al.*, 2014) and Yahoo Answers (Gyongyi *et al.*, 2007). There are many studies which primarily investigated Q&A websites aimed for different purposes, e.g., design features of Q&A websites (Mamykina *et al.*, 2011), determining the quality of questions post (Shah and Pomerantz, 2010), and determining various patterns of user behavior (Gyongyi *et al.*, 2007).

However, in this paper, we provide an initial literature survey on the work that has been done so far on mining SO explicitly addressing/related to real software development, i.e., methods, tools, and techniques. According to Sommerville (2010), SDLC is comprised of mainly four activities or stages, namely, software specification, software designing and coding, software validation, and software evolution. According to our knowledge, we are the first to present a comprehensive literature survey which has covered almost all the aspects of SDLC. There are some previous literature surveys available (Section 2.1), but all of them did not cover the whole specific software development work on SO.

The primary intent of our paper is to show that how academics/practitioners can get benefit from the valuable user-generated content shared on various online social networks, specifically from Q&A community SO for SDLC. This motivation enabled us to identify, explore, and address several aspects which are essential but unnoticed before. For instance, what are the Q&A community characteristics specifically SO like its purpose of use and design features? How can the future Q&A community be better designed so that users can get the most benefit out of it efficiently while seeking help? What kind of knowledge is searched and shared by software developers on SO? How Q&A community SO is helping software developers primarily in solving software development issues like coding issues, fixing bugs, and automatic comment generation? For what purposes are SO posts used and utilized in software development lifecycle? How can SO help software developers in utilizing the identified under-utilized tasks of software development lifecycle? What are the tools and methods used to mine SO repositories for software development? What are the challenges associated with mining SO repositories for software development? Our main contributions are: providing a first comprehensive literature survey of its kind on mining SO related to SDLC, systematically classifying and mapping the research on SO in relation to software development in to two broad categories "SO design and usage" and "SO content applications" having further eight topics/themes, the broad categorization informs Q&A forum providers how can such forums be improved in future and recommends software developers to utilize such forums for the identified under-utilized tasks in SDLC, providing an overview of the tools and techniques used to mine SO, the amount and type of data used to evaluate, discussing essential research done on SO aimed to help both academia and practitioners to understand the overall current research trends in the field of mining SO for software development and finally identifying research directions for future research.

Paper structure. The remainder of this paper is organized as follows: Section 2 describes the related work. The research methodology is explained in Section 3. In Section 4, we discuss the primary classification/categorization of mining work performed on SO related to software development. Section 5 lists and discusses the validity threats of the study. Finally, Section 6 concludes the paper along with highlighting directions for future research endeavors.

2. Related work

We categorized the related work into "previous surveys" and "other related areas" as discussed below.

2.1 Previous surveys

Baltadzhieva and Chrupała (2015b) reviewed questions quality posted on different community question answering (CQA) websites like SO. They highlighted metrics on which question quality can be determined, which by in large affect questions quality. Later on, Chen *et al.* (2015) presented a comprehensive survey on the use of topic models applied to mine software engineering repositories by selecting 167 research articles. The significant finding of their literature review is that the majority of the research conducted revolves around a very few number of software engineering tasks, utilized and applied elementary topic models, i.e., lacking entirely investigating their fundamental suppositions and values of the parameters.

Bavota (2016) presented a survey in the area of application of mining unstructured data (MUD) in software engineering. Also, their research work signified numerous types of unstructured data along with identifying some essential mining approaches to explore the data. Afterwards, a detailed outline of the current applications of various MUD in software engineering is given with an emphasis on explicit textual content present in numerous software repositories and code constituents. Similarly, de F Farias *et al.* (2016) performed a systematic mapping study by assessing 107 research articles published in MSR conferences. They mainly focused on gathering empirical evidence in MSR conference papers about the goals of software analysis performed, data sources used, assessment methods performed, and finally how the area is expanding over the passage of time. The findings revealed that MSR methods were utilized for numerous goals, i.e., primarily for code/defects comprehension, assessment of software developers' participation/behavior, and understanding the evolution of software. Besides, structured data repositories are comparatively more investigated than the unstructured ones. However, the number of methods utilizing unstructured data repositories source has shown a rise in the last three years.

All these survey papers have primarily focused on the following: questions quality in CQA forums, mining software repositories, and use of different topics models while mining software repositories. Besides, all these surveys did not cover the SO in full, and none of them surveyed mining SO specifically for software development. After thoroughly reading all these surveys, we felt the missing aspect of surveying SO literature specifically for software development which has remained the primary focus of our literature survey paper. We are the first to survey SO literature specifically for software development and classify/map the whole work done systematically.

2.2 Other research areas

This section summarizes other research areas work done on SO by focusing on the following: software parts of speech tagging, sentiment detection, measuring the willingness of developers to concern SO for help and so on. However, we do not claim that "other research areas" provide a complete review of all the SO works, but instead highlight and discuss some of the closely related research areas.

Tian *et al.* (2014) developed a software-specific WordNet like repository called WordSimSE DB, through utilizing the textual content in SO posts. The proposed system calculates the similarity level of the words by calculating the resemblances of the weighted co-occurring words with three kinds of semantic words in the textual repository. They assessed the effectiveness of the proposed system on a set of software-specific words and compared through a user study with state-of-the-art WordNet-based method referred as WordNetres, aimed to yield top-k most similar words. The output revealed that the proposed system performed better compared to WordNetres, by accomplishing average Likert score and aggregated discounted cumulative gain more than 50 and 66 percent, correspondingly. It was also found that WordNetres retrieved nothing for 55 percent of the searched queries and the rest of the queries; thus, WordNetres gave considerably more inferior results compared to WordSimSE DB.

Novielli *et al.* (2015) investigated the applicability of an innovative sentiment analysis tool for identifying sentimental expressions in SO. They also investigated to confirm the construct validity of selecting the polarities (positive and negative) of the sentiments and their strength as a most suitable way to functionalize affective conditions in empirical studies on SO. They also specified the dire necessity to cope the shortcomings brought by the use of restricted domain which ultimately may lead to yield inconsistent outputs.

Squire (2015) investigated and specified the critical reasons of support that motivate developers to move to SO. Afterwards, he gathered and assessed the data from a set of software development projects that already adopted it with the aim to depict the anticipated quality level of support which was practically accomplished. The output depicted two crucial quality indicators: developer involvement and response time considerably depicted enhancement on SO compared to mailing lists and forums. Nevertheless, they found that numerous software projects discontinued SO, regardless of accomplishing these anticipated enhancements.

Ye, Xing, Li and Kapre (2016) developed a software-specific parts-of-speech (S-POS) tagger to process the textual posts on SO. They first characterized a POS tagset capable of explaining the available software engineering knowledge. Then, choose a repository, built a custom tokenizer, performed data annotation, and proposed features aimed to supervise model training. The output depicted that the preciseness of tagging done by S-POS performed excellently compared to the Stanford POS Tagger for tagging any software text contents. Later on, Ye, Xing, Foo, Ang, Li and Kapre (2016) developed efficient software-centric named entity recognition (S-NER) system, applicable to software engineering social and professional content like SO. S-NER is applicable to the field of software engineering since it covers a wide range of areas of software entities. They assessed their proposed machine learning-based S-NER compared to the rule-based baseline system, which revealed S-NER performed remarkably well.

3. Research methodology

We performed a literature review to identify and evaluate relevant published research work on software development in SO. The overview of research methodology is depicted in Figure 1. The literature review was conducted by a team of four researchers, i.e., three PhD students and one academic staff member. All team members took participation in all the phases of the literature review. To minimize the personal bias and to improve the literature review process, inter-rater reliability tests were conducted at initial and final selection phases of the literature review process.

3.1 Scope of the survey

The SO analysis literature included research studies that analyzed data mined from SO. We are explicitly focused on the studies that addressed the work on technical aspects of software development, i.e., methods/tools and SDLC in SO. Our survey did not have any specific research question and followed an ad hoc literature selection procedure; hence, our survey is not a systematic literature review (SLR) as defined by Kitchenham (2004). In contrast, an SLR is a methodologically rigorous review of research by following an agreed review protocol having specific research questions. We believe that the area of mining SO is still evolving; however, it has not achieved a level of maturity at which specific research questions can be selected and posted. Thus, we aim to outline, gather, and curate the dissimilar literature and reasoning and to indicate that there does essentially exist a coherent field of research that can be called as SO analysis for software development. We expect that this will mostly turn out to be an enabling research study for conducting SLRs in this field in the near future. Our literature review execution is done by following these steps.

3.2 Search strategy

We selected suitable search terms to ensure that no relevant article is missed in our study. We validated the selected keywords in research databases and the following synonyms found to be relevant to the topic.

Stackoverflow = Stackoverflow, Stack overflow.

Software development = Software development, software engineering.

We used Boolean operators to combine major search terms and constructed the following search terms after validating them through already known relevant research papers.

(Stackoverflow OR Stack overflow) AND (mining, text mining, software development or software engineering, analysis, survey).

We selected ACM Digital Library, IEEE Xplore, Springer Link, and Google Scholar academic databases from the identified list of Brereton *et al.* (2007) for software engineering

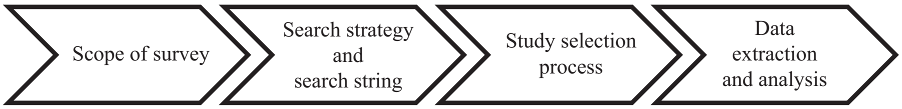
Scope of survey
Search strategy and search string
Study selection process
Data extraction and analysis

Figure 1. Overview of the research methodology

domain, based on the availability of academic research databases. We believe that the list provided by Brereton *et al.* (2007) is sufficient and has been used in a number of software engineering systematic literature review studies (Kitchenham and Charters, 2007; Niazi *et al.*, 2016). We performed a search in the selected available academic databases to identify relevant articles published between 2008 and June 2016. Since these databases have considerably different search criteria and capability, we accordingly adopted our search terms.

3.3 Inclusion and exclusion criteria

The authors applied the following inclusion criteria:

- the research article is purely related to software development in SO and may have actionable consequences for practitioners or researchers; and
- the research article is related to SO and the data mined/used from SO have full or partial linkage with the technical aspects of software development, i.e., SDLC, methods, and tools.

The authors applied the following exclusion criteria:

- The research paper focuses on software development but does not extend to SO explicitly.
- We have included all those papers which purely mined data from SO. However, we have not included all those research studies whose data are synthetic, though based on some guidance from SO.

3.4 Study selection, extraction, and analyzing the data

The selection criterion of relevant papers is done in two stages: initial selection and final selection. In the initial selection criteria, the title and abstract of the papers are read. All those publications are removed which are entirely irrelevant explicitly from the name of their titles. The abstract is read, and a decision is made whether the article is relevant or irrelevant based on the scope of the study defined in Section 3.1. In the final selection, from the initially selected list of papers the decision is made upon reading the entire papers thoroughly that meet the selection criteria, i.e., (a) whether the article fulfills the significant requirements as defined in the scope of the survey (Section 3.1), or (b) is very close to our field, hence, cannot be ignored in order to place the primary literature into context. Thus, all those research papers are included which match the requirements of (a) or (b) or both of them in our literature survey. These two stages study selection criteria were performed by one of the authors of this paper. The total number of papers retrieved after performing a search using the defined search strings in the selected academic databases is shown in Table I. For Google Scholar, we considered only top 500 retrieved results.

For a paper to be included in our study, we primarily assessed the content of the paper thoroughly, issues stated in the paper, findings of the paper, and referencing of the paper.

Table I.
Search results

Resource	Total results	Initial selection	Final selection
IEEE Xplore	492	181	79
ACM Digital Library	721	165	56
Springer Link	462	85	14
Google Scholar	500	98	17
Total	2,175	529	166

We finally selected 166 papers which met our selection criteria, as shown in Table I. For our survey paper, the extracted data from the selected papers were as follows: authors, type of publication, paper title, no. of posts, investigation method, technique/tool/algorithm, data source, publisher, and date of publication.

We performed inter-rater reliability test to minimize the researcher's study selection bias. In inter-rater reliability test process, three independent reviewers randomly chose ten publications each from the "total results" list and carried out the initial selection process. Likewise, the three independent reviewers also chose randomly ten publications each from the "initial selection" list and carried out the final selection process.

We employed non-parametric Kendall's coefficient of concordance (W) (Eye and Mun, 2006) to assess the inter-rater agreement between three independent reviewers. Kendall's coefficient of concordance (W) value has a scale between 0 and 1, where 0 depicts full disagreement, and 1 depicts full agreement. In our study, Kendall's coefficient of concordance (W) for the ten randomly chosen papers from "total results" was 0.83 ($p = 0.007$), which depicts a strong level of agreement between the results generated by the primary researchers and the three independent reviewers. Besides, Kendall's coefficient of concordance (W) for the ten randomly chosen papers from the "initial results" list was 0.91 ($p = 0.003$), which also depicts a strong level of agreement between the results generated by the primary researchers and the three independent reviewers. Based on the depicted significant agreement between the authors and the independent reviewers, we did not change the "total results" and "initial results."

Card sorting has been efficiently applied by researchers and practitioners in various fields for defining categories, due to its capability to recognize and explain the structures present in an information space (Hannah, 2005). We have used open card sort technique (Hannah, 2005) to categorize 166 SO research studies into different themes/topics. We summarize the abstract of each paper separately in a table. The short summary of each paper served as a card in the applied open card sort technique. Three of the authors individually performed open card sort method. During the open card sorting process, each of the individual authors analyzed each paper, determined suitable topics/groupings, and finally labeled each of the emerged groupings/topics. It was interesting to see that each of us came up with a different number of groupings/themes. Finally, all the individual authors emerged groupings/topics were discussed in a group meeting involving all of the four authors. The authors' group discussion analyzed the content of each theme thoroughly, renamed some of the themes, merged/optimized further the similar or overlapping themes, and finally mutually agreed on categorizing 166 SO articles into two main broad categories having further eight themes/topics.

4. Mining SO for software development

In this section, we will discuss and summarize the selected 166 research articles done on mining SO in the context of software development. The whole content and summarized discussions on mining SO in the context of software development is categorized and grouped by applying open card sorting (Section 3.4) into two main categories "SO design and usage" and "SO content applications," having further eight topics/themes. It will enable the readers to get a systematic overview of the work in general and specifically to know future research avenues in each of the eight topics/themes. The selected 166 research articles are discussed in detail in the next subsections (Sections 4.1-4.2) and are chronologically summarized in Tables III-X.

The systematic mapping, categorization, and grouping of selected SO publications into two main categories having eight further themes/topics related to SDLC are presented in Table II, which basically illustrate the presence of software development activities on SO.

	Software specification	Software designing and coding	Software validation	Software evolution
SO design and usage				
Asking right questions or question quality		✓	✓	
Reputation and reward system		✓	✓	✓
Knowledge sharing, learning, and searching	✓	✓	✓	✓
Exploration of gender and experts or expertise		✓	✓	
SO content applications				
Extracting, analyzing and improving source code		✓	✓	✓
Automatic comment generation		✓		
Application programming interface (API) usage		✓		✓
Topics or issues on stack overflow	✓	✓	✓	✓

Notes: The first and second leftmost column shows the categorization of SO literature into two main categories, having further eight themes/topics. The topmost rows depict different activities of SDLC and the "tick marks" shows the mapping or presence of each SO theme/category literature to the respective activity/stage of SDLC

Table II.
Systematic mapping of selected SO publications to SDLC

It can be observed in Table II that the majority of research done is in the area of software coding and design. It confirms several facts: SO community is serving the programming community considerably well, most of the posts or discussions programmers make are about software development tools/platforms usage, processes and seeking coding help (i.e., retrieving code snippets and fixing problems in code), and inform the tools/platform developers to provide better support or documentation. The other two areas which are addressed almost equally in SO studies are related to software validation and evolution. All the research studies in these areas revolve around fixing bugs, improving existing systems based on the modification in code and user requirements. The most under-researched area is software specification which is the first and foremost important step in SDLC. Some studies highlighted and investigated the presence of user features or user requirements or bugs discussed in SO posts about some existing software development tools or platforms. We strongly encourage that this area needs further investigation to identify useful requirements/features present on SO posts about software development tools, i.e., development platforms, designing and testing, which can be utilized by tool developers to improve their tools according to the demands suggested by the real tool users.

The growth, trend and further breakdown of SO selected studies year wise in each of the eight topics/themes are presented via a histogram in Figure 2.

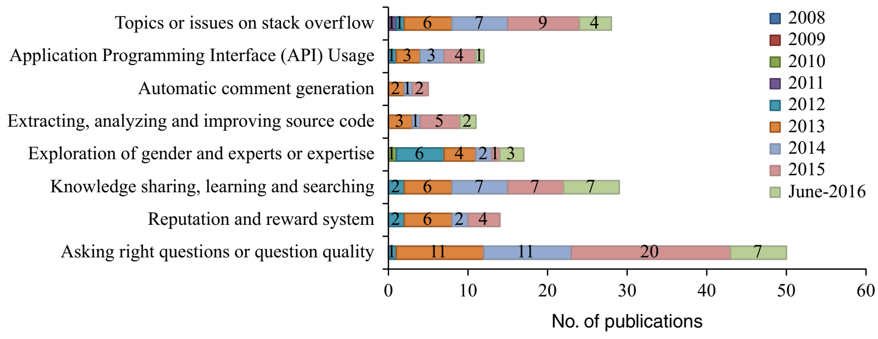

Figure 2.
Histogram showing topics/themes trends during the period from 2008 till June 2016

The histogram in Figure 2 illustrates the eight themes categorically on SO for software development and the number of studies year wise from 2008 to June 2016. If we carefully observe the histogram, it is evident that more studies are reported starting from the year 2013 to June 2016. More specifically, the categories of "asking right questions or question quality," "knowledge sharing, learning and searching," and "topics or issues on stack overflow" are leading the contribution on SO during 2013 till June 2016 for software development. Regarding the characteristics of SO, these statistics depict us that developers are seeking help on SO related to software development and the trend is increasing due to the fact they are satisfied with what they get from SO. For practitioners and researchers, they did studies starting from what kind of topics and knowledge are discussed and shared mostly, which in parallel lead them to start investigating the quality of questions (how, what) and characteristics of professionals (gender, age, and skills) on SO. Then, the focus shifted to recommending tags for posing questions, automatically generating comments, fixing issues/bugs, and integrating IDE for Q&A site retrieval. Thus, the histogram mainly illustrates the readers to get a systematic breakdown of studies year wise, know the trends and importance of each theme, and can advise them research gaps yet to be explored.

The total distribution or percentage of the number of studies belonging to each of the eight topics/themes identified on SO for software development are depicted via a pie chart in Figure 3.

The pie chart in Figure 3 illustrates each theme/topic research studies percentage wise on SO for software development. It is evident that work done on "asking right questions or question quality" is 30 percent, highest among all the topics/themes. It shows that the research community has emphasized the importance of posing right or quality questions on SO, since, this will increase the chance of getting precise and useful answers from the community. Other leading works are on "knowledge sharing, learning and searching" and "topics or issues on stack overflow" having 18 and 17 percent research studies, respectively. All these statistics show not only the interests of researchers but also the traits of SO professionals. The other aspect is the abundance of data availability, which has been mined for knowing solutions to the problems and further yet to be explored especially for extracting features or bugs about development tools/platforms. The trend of knowledge sharing on SO is increasing, because the users can solve their problems efficiently. This trend ultimately has created an opportunity for researchers to thoroughly investigate what is shared, how it is shared and how it can be further mined/analyzed and utilized for improving software development. Besides, SO community is on track not only for solving developers' problems but also serves as a platform to learn and discuss numerous ideas. Furthermore, these percentages illustrate

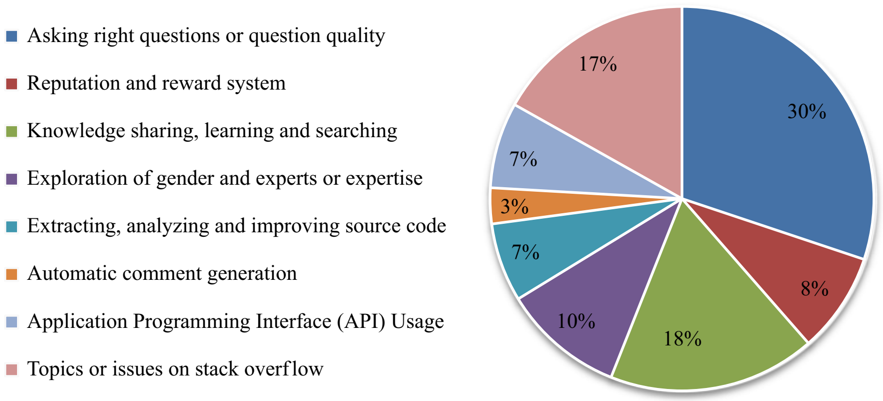

Figure 3.
Pie chart showing the overall subtopics distribution percentage during the period from 2008 till June 2016

the readers, the areas which are explored in detail, highlighted the importance of each theme/topic, and the areas that need further investigations.

The whole trend of research conducted (166 publications) on SO for software development during the period from 2008 to June 2016 is depicted via a graph in Figure 4.

It is quite evident in Figure 4 that the number of studies on SO has seen an enormous growth, starting from the year 2010 and kept on increasing except a slight decrease in the year 2014. After the inception of SO in 2008, no relevant studies have been reported till 2009. SO was not popular at its start, so fewer professionals knew about it, no such huge activity was done, so there were fewer posts data to be analyzed. The trend of a low number of research studies remained until 2010-2011, and then a sudden rise in 2012 having 13 research papers was reported. The rising trend kept steadily increasing with 41 research papers reported in 2013 and then a negligible downfall in 2014 having 34 research papers. Then, a sudden rise was seen in 2015 having the highest number of 52 research studies, and 24 research studies till June 2016 (many studies are expected to be published the same year). The reasons for the rapid increase in a number of studies are due to advent of newer technologies, i.e., Q&A forums/websites, and rise in the use of these technologies by software developers due to the capability these technologies provide in solving problems, knowledge sharing, and learning. It has led to the growth in software repositories, and consequently got attention from researchers/practitioners to automatically mine, analyze, and utilize these software repositories for improving software development. We expect that this trend will increase in future due to the availability of sufficient rich data, i.e., code snippets, bugs reports, documentation, feature requests, and tutorials, on SO for developers. Hence, it will create opportunities for practitioners/researchers to deeply mine and analyze both structured and unstructured content present on Q&A site SO for improving software development.

The size of SO data used in all of the 166 surveyed papers is illustrated in Figure 5.

The graph in Figure 5 primarily illustrates the approximate data size used in all the research articles included in our literature survey. The x-axis represents the data size in log scale, whereas the y-axis represents the number of papers. The approximate mean data size used in the research studies lies in the range of 10^5-10^8. We can also see that some of the papers did not use SO data, but they accessed the website directly for their research without specifying the size of data. The approximate mean data size range of 10^5-10^8 gives insights to future researchers/practitioners for selecting a suitable data size to get useful results. However, using such a large-scale unstructured data also has some perils, i.e., requires technical expertise, extra effort, additional cost, and tools to utilize the content efficiently. Thus, a balance is needed based on the needs of the research study.

4.1 SO design and usage

In this section, we included all those studies which focused on SO platform design and guidelines for using SO efficiently. Thus, the research studies investigated the existing

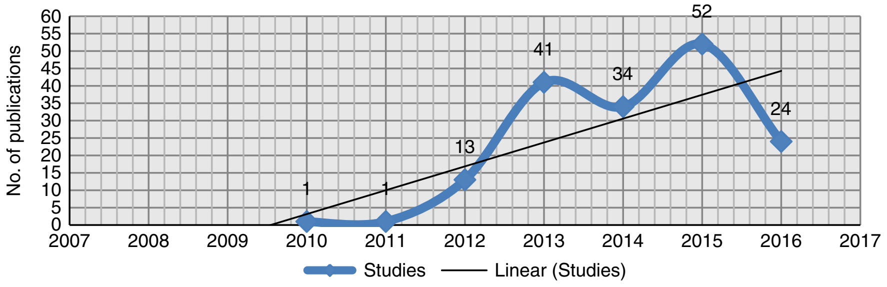

Figure 4.
Graph showing the trend of work done in mining SO during the period 2008 till June 2016

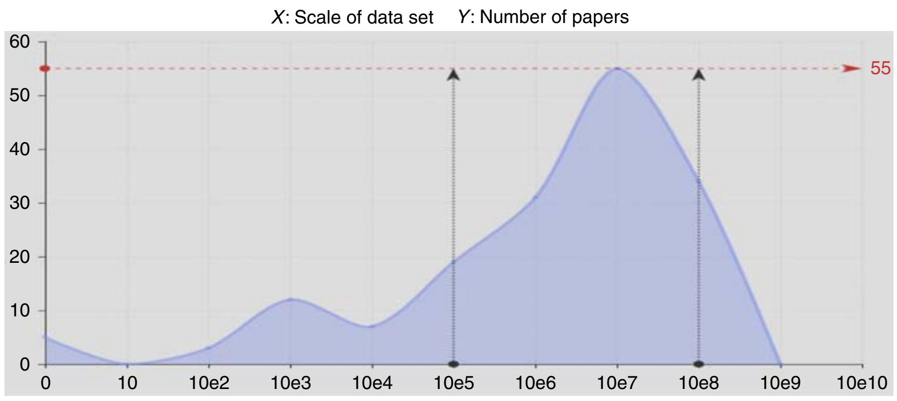

Figure 5.
Scale of SO data used in surveyed papers

features provided by Q&A community SO. By assessing and knowing the existing features of SO will enable the platform providers/designers to improve the design of existing Q&A communities effectively in future. The research on investigating SO usage/guidelines will also enable software developers to use and utilize SO Q&A community efficiently for seeking help.

4.1.1 Asking right questions or question quality. This section included all those papers that discussed and analyzed how questions are posted by developers about various software development topics/issues on CQA website SO. The assessment then leads toward identifying reasons why questions are left unanswered or closed and suggesting methods to post right questions that possess specific characteristics like presence of code fragments, title, length, the presence of tags, predicting a suitable best answer for a newly posted question, and so on. Since all these factors are influential in questions quality, and thereby, affect to have right answers from the community professionals. This section has some overlap with "extracting, analyzing and improving source code;" nevertheless, we think they are different because this section is focused on how to ask quality questions or predict suitable answers on SO, whereas "extracting, analyzing and improving source code" is focused on how to improve the quality of code snippets extracted from SO posts for its program. The majority of these research papers investigated automatically or semi-automatically huge amount of SO data (minimum 163 posts, maximum 25 M) for their works, as briefly summarized in Table III.

Nasehi *et al.* (2012) manually performed a qualitative assessment to investigate the important features of precise code examples in answers of 163 SO posts. They revealed that some additional descriptions with code examples are as useful as the code examples deemed useful themselves in the answers. Similarly, Yao *et al.* (2013) investigated quality prediction of both Q&As on SO. The output revealed that answer quality is strongly positively associated with that of its question. Later on, Tian *et al.* (2013) developed a method to forecast the most suitable answerer for a new question posted on SO. The developed method deems the relevancy of both user interest and user expertise to the topics of the posted question. Their proposed approach outperformed the baseline approach TF-IDF. Romano presented an approach weighted votes (WV) metric to give various weights to the vote's subject on how many answers were already there the time when the vote is done. The main intent of WV is to highlight all those answers that obtain the majority of the votes at the time when most of the answers were already posted on SO (Romano and Pinzger, 2013).

Ginsca and Popescu (2013) investigated specific characteristics of a user's profile in SO, which can be utilized to highlight high-quality contributions. The output revealed that answer rankings acquired using a user model outperformed the baseline method based on

Paper	Investigation	Technique/tool/algorithm	Data source	No. of posts
Nasehi *et al.* (2012)	Manual	Open coding	SO	163
Romano and Pinzger (2013)	Semi-automatic	Paired cliff's delta effect, *t*-tests	SO	4,392,956
Tian *et al.* (2013)	Automatic	LDA, collaborative voting	SO	99,166
Ginsca and Popescu (2013)	Automatic	Ranking SVM (RSVM)	SO	6,468,635
Yao *et al.* (2013)	Semi-automatic	Preprocessing, Pearson correlation coefficient, Root mean square error	SO	5,392,181
Saha, Saha and Schneider (2013)	Automatic	SVM, performed preprocessing steps	SO	1.3 M
Stanley and Byrne (2013)	Automatic	Performed certain preprocessing steps	SO	1,468,485
Xia *et al.* (2013)	Automatic	TF-IDF, Multi-label learning, Binary relevance, NB classifier	SO and FreeCode	SO – 47,668 FreeCode – 39,231
Asaduzzaman *et al.* (2013)	Manual	J48, RF	SO	400
Correa and Sureka (2013a)	Semi-automatic	SVM, NB, LR, Stochastic gradient boosted trees (SGBT)	SO	102,993
Saha, Saha and Perry (2013b)	Automatic	Information gain/ratio ranking algorithm, J48 DT, K-NN, RF, NB	SO	10,311,875
cG Galina and Kuznetsov (2013)	Automatic	RF, SVM, Vowpal Wabbit, TF-IDF, LDA	SO	3,664,927
Correa and Sureka (2014)	Automatic	LIWC2007, Ada-boost classifier, DT	SO	14.5 M
Lal *et al.* (2014)	Semi-automatic	K-NN, DT, Ada-Boost, Gaussian NB	SE (five sites)	38,609
Rekha *et al.* (2014)	Automatic	SVM, Performed preprocessing steps	SO	50,000
Wang, Lo, Vasilescu and Serebrenik (2014)	Semi-automatic	L-LDA, POS Tagger, Performed certain preprocessing steps,	SO, Ask Ubuntu, Ask different, FreeCode	SO – 47,668/Ask Ubuntu – 37,354/ Ask different – 13,351, FreeCode – 39,231
Bhat *et al.* (2014)	Automatic	Logarithmic binning, LR, SVM with linear/radial kernel, DT	SO	10.2 M
Yang, Hauff, Bozzon and Houben (2014)	Semi-automatic	Logistic regression based classifier	SO	5 M
Gkotsis *et al.* (2014)	Automatic	Alternate decision trees	SE (21 sites)	19 M
Ponzanelli, Mocci, Bacchelli, Lanza and Fullerton (2014)	Automatic	Linear quality function, GA	SO	5,648,975
Squire and Funkhouser (2014)	Semi-automatic	Flesch-Kincaid reading ease metric	SO	10.1 M
Novielli *et al.* (2014)	Semi-automatic	Logistic regression model	SO	7 M
Ponzanelli, Mocci, Bacchelli and Lanza (2014)	Semi-automatic	DT, Genetic algorithm	SO	5,648,975
Arora *et al.* (2015)	Automatic	Multinomial naive bayes (M-NB), BM25, LM, Term Frequency-inverse document frequency (TF-IDF)	SO	8 M
Gantayat *et al.* (2015)	Automatic	*t*-Tests, binary logistic regression	SO	8 M

(*continued*)

Table III. Chronological summary of asking right questions or questions of quality-related SO literature showing the paper, investigation, technique/tool/algorithm, data source, and the number of posts evaluated in the study

Paper	Investigation	Technique/tool/algorithm	Data source	No. of posts
Baltadzhieva and Chrupała (2015a)	Semi-automatic	Performed certain preprocessing steps, Ridge regression model	SO	1,713,400
Duijn *et al.* (2015)	Manual	Preprocessing steps, Decision trees (DT), LR, Random forests)	SO	521,530
Wang, Xia and Lo (2015)	Semi-automatic	L-LDA, POS Tagger, Performed certain preprocessing steps	SO, Ask Ubuntu, Ask Different, FreeCode	SO – 47,668/Ask Ubuntu – 37,354/ Ask Different – 13,351, FreeCode – 39,231
Ercan *et al.* (2015)	Semi-automatic	L-ROUGE, Token Similarity algorithm	SO	300 K
Anand and Vahab (2015)	Automatic	LDA, Linear regression (LR), OpenCalais tool, SPARQL	SO	100,000
Li *et al.* (2015)	Automatic	Negative binomial regression models, LDA	SO	15,933,529
Joorabchi *et al.* (2015)	Automatic	SVM, K-NN, Bayes Network, DT, RF, Bagging RF, Random committee RF, RF + All features except F9/F2	SO	20 M
Zhu *et al.* (2015)	Automatic	SVM, Performed preprocessing steps	SO	7,990,787
Bhat *et al.* (2015)	Automatic	LR, SVM, DT, SVM Linear Kernel, SVM radial basis kernel function	SO	10.2 M
Latorre *et al.* (2015)	Automatic	SODA	SO	SO Data visualize
Beyer and Pinzger (2015)	Semi-automatic	Performed certain preprocessing steps	SO	7,990,787
Ganguly and Jones (2015)	Automatic	Preprocessing, LDA, PLDA, SPLDA	SO	21 M
Mo *et al.* (2015)	Automatic	Performed certain preprocessing steps, DT, Heuristic Greedy Matching		366,717
Romero *et al.* (2015)	Automatic	SVM, NB, Preprocessing steps,	SO	3 M
Rahman and Roy (2015)	Automatic	J48, LR, NB	SO	8,057
Slag *et al.* (2015)	Semi-automatic	SentiStrength, k-means clustering	SO	25 M
Calefato *et al.* (2015)	Automatic	SentiStrength, Delong's test	SO	348,618
Zhang *et al.* (2015)	Semi-automatic	LDA, WVTool	SO	2 M
Ahasanuzzaman *et al.* (2016)	Semi-automatic	Preprocessing, BM25, LR, Stanford named entity recognizer, WS4J	SO	25 M
Gupta and Reddy (2016)	Automatic	DT	SO	445,000
Singh and Simperl (2016)	Automatic	SPARQL, Wikipedia-miner, OpenCalais, PageRank, HITS	SO and Reddit	43 M SO, Reddit – 19,000
Xia *et al.* (2016)	Semi-automatic	Naïve Bayes multinomial algorithm, NB, RF, Ada-boost classifier	SO	470,096
Beyer and Pinzger (2016)	Semi-automatic	Preprocessing steps, Clustering, Walktrap community	SO	7,990,787
Boudaer and Loeckx (2016)	Automatic	Labeled LDA	SE	24,748
Jiarpakdee *et al.* (2016)	Automatic	RF, Scott-Knott clustering	SO	7,990,787

Table III.

ranking in the sequential order of answers. Similarly, Anand and Vahab (2015) investigated a different mechanism to represent user expertise to efficiently evaluate the quality level of posts that are posted in SO. The initial experiments revealed that taking into account the extra features in the aspect of user expertise provides a rise in the precision of classification despite neglecting features measurable within 24 hours only. Likewise, Novielli *et al.* (2014) examined the role of emotional lexicon on the questions posted on SO by evaluating approximately over 7 M posts. Specifically, they claimed that the technical posts on SO do have an emotional style, which ultimately affects the prospects of receiving a substantial quality answer as well as the time to respond.

Saha, Saha and Schneider (2013) addressed the problem of accurately suggesting “tags” for the questions on SO. They used discriminative model approach to propose question tags automatically and assist a developer questioner in proper tags selection for receiving good solution/answer. Similarly, Stanley and Byrne (2013) also proposed a model named ACT-R inspired Bayesian probabilistic model, capable of forecasting the tags used while posing questions on SO. The outcome revealed that the developed model gives 65 percent correct results in a situation where one tag prediction is needed on average. Besides, the work of Xia *et al.* (2013) and Wang, Xia and Lo (2015) also focused on developing a technique called TagCombine, aimed to propose tags automatically which examine objects in software information websites. The output of the conducted experiments revealed that TagCombine outperformed the available tag recommendation methods.

Later on, the work of Rekha *et al.* (2014) come up with proposing a hybrid auto-tagging system based on the content in questions posted by the developers on SO. The proposed system is capable of proposing tags to the questioner as soon as the question is posted on SO based on the content present in the posted question. Wang, Lo, Vasilescu and Serebrenik (2014) extended prior work of Xia *et al.* (2013) by proposing another technique called EnTagRec that aimed to address the same issue. The EnTagRec integrates two components, namely, a Bayesian inference component (BIC) that makes use of Labeled-LDA, and an improved Frequentist Inference Component that aims to exclude the unassociated through a parts-of-speech (POS) tagger and identifies the related tags through a spreading activation algorithm. The results revealed that EnTagRec accomplished better performance compared to TagCombine on three data sets, namely, SO, Ask Ubuntu, Ask Different, and EnTagRec performance is similar to that of TagCombine on FreeCode data set. Similarly, Romero *et al.* (2015) investigated the problem of predicting tags to be assigned to the questions posted on SO. The classification is multi-class and multi-tag which ultimately means that a question posted can be allocated to different topics and can possess several tags. They come up with the proposal of a five-way multi-class classifier system to overcome the problem. The outcome of the experimentation was quite impressive by attaining F1 scores ranging from 0.59 to 0.76.

The focus of Bhat *et al.* (2014) work remained on investigating and estimating the response time of questions posted on SO. The output revealed that tag-associated factors, for instance, their “popularity” and the total number of their “subscribers,” reveal quite stronger indication compared to factors not associated with the tags. Later on, Bhat *et al.* (2015) extended their prior work (Bhat *et al.*, 2014) by assessing all those factors which have a close association with question response time. The outcome revealed that tag-related factors, e.g., popularity and number of subscribers, have more influence on response compared to the factors not belonging to tags. Besides, tag-based features can also be used for predicting the response time of questions posted, which shows the worth of appropriate tags selection on SO.

Joorabchi presented automatic mapping of user tags to similar Wikipedia concepts. They used only 1,256 tags from SO posts to assess the level of mapping of the proposed system. The outcome revealed that F1 outperformed the others by achieving 99.6 percent mapping (Joorabchi *et al.*, 2015).

The research focus of Zhu *et al.* (2015) remained on addressing the problem of providing semantic relationships between tags on SO. They derived taxonomy from the tags on SO and come up with a learning method having novel features to make ontology capable of capturing the hierarchical semantic structure of the tags. The outcome revealed the excellent quality of the tags taxonomy by beating the baseline methods in achieving good accuracy.

Ganguly and Jones (2015) examined the retrieval of a set of questions (documents), which are closely associated with a newly posted question. They propose a supervised partially labeled topic model (SPLDA) to approximate the distributions of tags per topic. The experiments revealed that SPLDA performed best in improving retrieval performance and outperformed the simple baseline approach of standard language model query likelihood and other topic model smoothed extensions of language model, including LDA, SLDA, and PLDA.

Mo *et al.* (2015) proposed a new tagging-based technique to link identity among software communities known as tagging-based approach to identity linkage. The output revealed that the proposed method is considered viable and suitable for linking software developer identities. Latorre *et al.* (2015) developed a tool SODA to assist practitioners in deeply understanding the current trend of highly discussed topics dependent upon the tags on SO. SODA is capable to efficiently visualize the trends of discussed topics with the evolution of time on SO. The output revealed that SODA is capable of identifying specific useful discussions in SO over the passage of time.

Ponzanelli, Mocci, Bacchelli and Lanza (2014) developed approach to do automatic categorization of questions based on their quality. They investigated how can the quality of a posted question be forecasted and modeled by taking into consideration the following features: the textual (readability metrics) and non-textual (fame of a user in the community). Similarly, Jiarpakdee *et al.* (2016) investigated to recognize the influence of several properties/features on question quality and which of them has the most impact by creating prediction models that can forecast whether a question is expected to get no answer. The assessment revealed that two most important aspects which play a pivotal role in the detection of question quality are as follows: community-based and affective features. Similarly, Squire and Funkhouser (2014) investigated to gain a deep understanding of whether the inclusion of source code (and its ratio) essentially will make the "best" SO posts. Besides, to know whether the non-code portions of text can also potentially be amongst the "best" SO posts. The output suggested the "bit of" source code should be in higher ratios in answers text (1:3) compared to the text of the question (1:9).

Gkotsis *et al.* (2014) developed a novel approach to identify and suggest best answers through utilizing only textual features. They examined the discrete characteristics of answers accepted in the community and suggested a strategy for classification to accomplish this prediction proficiently. In the same line of research but with a different goal, Ponzanelli, Mocci, Bacchelli, Lanza and Fullerton (2014) developed an approach to enhance the existing automatic system for identifying low-quality posts on SO. The approach assessed not only the content of posts comprised of usual textual features and complex readability metrics but also the SO-associated characteristic comprised of the reputation of the member in the forum. The output of the proposed method thus considerably lessens the magnitude of the review queue efficiently and eliminates the posts misclassified as having good quality.

Yang, Hauff, Bozzon and Houben (2014) presented new insights into the study of collaborative question answering platforms by thoroughly exploring the editing attitude of users. The approach is comprised of identifying whether a question needs editing or not and if it needs editing then identify which aspects need to be edited to make the quality question. The assessment of editing actions revealed that it came up with appropriate reformulation proposals. Similarly, Li *et al.* (2015) investigated a specific design decision in Q&A sites;

thus, enabling Wikipedia-like collaborative modification on Q&A. The output revealed that the gains of collaborative editing are more significant than its risks, which support the claim that collaborative editing is beneficial. Ercan *et al.* (2015) explored quantitatively all those questions having code fragments and examined the effect of explaining these code pieces effectively on time to respond. The results revealed that it lead to a five σ (single-tail significant) rise in precision compared to other baseline prediction times. Thus, recommend the use of proposed approach as an "edit suggestion," i.e., all questions posted having a low score could alarm the user to describe the incorporated code effectively.

Baltadzhieva and Chrupała (2015a) predicted question quality and thoroughly examined the elements of questions which affect it. They evaluated the effect of the following: tags, title and body length, incorporating code snippet, the reputation of users and various terms used to frame the question. The output revealed that terms forecasting high-quality questions are as follows: stating excitement, negative experience or terms regarding exceptions. The terms forecasting low-quality questions are the terms comprised of spelling errors or specifying off-topic questions and interjections. Similarly, Duijn *et al.* (2015) presented a new approach to assess the quality of questions posted based on the different aspects of information on CQA sites. They investigated the presence of code snippets in questions posted on SO and employed a novel way of assessing the questions quality never explored prior. The output revealed similar performance to a classification dependent on a varied set of metrics, thus possibly accomplished a comparatively better classification.

Arora *et al.* (2015) developed a method to solve the issue of automatically categorizing questions. They extracted same questions formerly posted in the same CQA sites, and finally made use of the text from these formerly posted similar questions to forecast the quality level of the question posted. The results enabled them to increase the prediction accuracy level of the question quality by approximately 2.8 percent and recall of negatively scored posted questions by around 4.2 percent.

Gantayat *et al.* (2015) investigated the relationship between acceptance of an answer and votes given to the accepted answers by the users. The results revealed that 81 percent questions having several answers also got accepted answers and highly voted as well. It suggests that some votes given in a post by users effect the decision of accepting them. They also investigated the situations where the asker's selection and the public opinion are found entirely disagreed.

Boudaer and Loeckx (2016) assessed the effect of adding personal tag histories when tags are assigned to the posts posted on SE. The attribution of tags is normally done via using natural language processing (NLP) only or collaborative filtering techniques. The outcome revealed that incorporating content-based text features with the personal profile enables to trade-off the accuracy of forecasts for the recall. It also enhances "exact match" in multi-label setting from a reference point of 18.2 percent text-only to 54.3 percent.

Beyer and Pinzger (2015) examined manually to understand the possible ways how synonym pairs of SO are constructed. The outcome of experiments revealed that tag synonym suggestion tool (TSST) tool has an accuracy of 88.4 percent for finding the perfect tags synonyms, and for 72.2 percent of the tags the accurate synonym is within the top ten ranked suggested list. Furthermore, the TSST was also applied to ten arbitrarily chosen Android-related tags and assessed through an online survey of the proposed synonyms. The outcome revealed that in 80 percent of their ratings, the TSST tool gave appropriate tag synonym suggestions. Later on, Beyer and Pinzger (2016) further extended their prior work (Beyer and Pinzger, 2015) by proposing a technique to group the tag synonym pairs to important topics. They experiment several input graphs and configurations for the algorithms. Consequently, revealed that the walktrap community detection algorithm fits the needs when applied thoroughly. The output revealed that by keeping synonym pairs with a ranking of less than or equal to 0.55 as input graph and configuring the step size of the walktrap community algorithm to three, one can get effortlessly meaningful tag communities.

Asaduzzaman *et al.* (2013) manually investigated via a qualitative study the reasons why questions are left unanswered on SO. The significant reasons revealed are as follows: unable to find experts, small length and unclear questions, propriety technology questions, without code snippets examples, off-topic questions and so on. They also experimented to detect and predict for how long time a question posted on SO will remain unanswered. cG Galina made a classifier aimed to predict whether a posted question will be closed or not on SO. cG Galina and Kuznetsov (2013) also identified reasons why a specific question is closed on SO. Similarly, Correa and Sureka (2013a) did an empirical study on the characterization of "closed" questions on SO via examining the contents of the question, patterns of the answer, and chronological analysis of the closed question. They use ensemble-based machine learning framework to make a predictive model to predict closed question on SO and results revealed to have 73 percent precise predictions. They also assess the features and depict the useful features to discriminate the closed questions from the un-closed on SO. Later on, Rahman and Roy (2015) performed an exploratory study with the intent to examine the reasons behind a question is left unanswered on SO. They assess four pivotal dimensions of those questions, their answers given and the respective members that somewhat describe the witnessed scenario. They evaluate the method by conducting experiments which revealed that the model could predict the unresolved questions by accomplishing a precision of 78.70 percent and a 76.10 percent recall value. Similarly, Saha, Saha and Perry (2013) empirically investigated why questions are left unanswered via applying a specific mixture of statistical and data mining methods. The results revealed that there exist some topics that were never addressed to be answered and most of the questions that were left unanswered were due to less interest shown by the community to answer.

Lal *et al.* (2014) presented an in-depth characterization of migrated questions on five famous SE sites. They illustrate the output of the chronological distribution, the structure of CQA sites, reputation of the owner, the quantity of discussion made, and the fame of the migrated questions. They detected specific discriminative features of migrated questions and suggested a framework built upon machine learning techniques, aimed to predict questions migration on SE sites. The output of the evaluation on five SE sites revealed that the suggested model is efficient by accomplishing a precision of 73 percent in predicting the questions migration.

Correa and Sureka (2014) focused on characterization of the deleted questions and development of a prediction model. They observed that recent years had witnessed an increase in question deletion on SO. They developed a predictive model to identify the potential deleted question at the time it is posted on SO. They experimented with 47 features based on the member profile, community, content, and stylistic features. The output revealed that the prediction model was able to identify approximately 66 percent accuracy precisely in predicting the deletion of questions . Later on, Xia *et al.* (2016) extended the work of Correa and Sureka (2014) by developing a hybrid two-stage technique called DelPredictor. The developed method integrates both text processing techniques and classification techniques to forecast the deleted questions from SO. The output of the experiments on five years SO deleted questions data revealed that DelPredictor enhances F1 scores considerably over the previous baseline method (Correa and Sureka, 2014) and the text-based methods by accomplishing 29.50, 9.34, and 28.11 percent, correspondingly.

Calefato *et al.* (2015) focused on investigating the impact of emotion style on perceived quality of a contribution on SO. They target to identify the driving factors that can be practiced by community users while contributing on SO. The results revealed that the factors which influence the success of answer are as follows: presenting information in an appropriate, precise, timely and effective manner.

Slag *et al.* (2015) investigated a group of users called one-day flies to know why these groups of users are not interested in contributing anymore on SO. The output revealed that one-day flies group users are not possible because of posting duplicate questions and making use of specific unusual tags or having fewer views of their questions. However, the questions they post are removed quite frequently by themselves or by moderators. Thus, most probably they do not get an answer to their posted question. Though these two factors possibly give an idea why one-day flies take part less in the community, still the whole picture of reasons is not clear which needs further research.

Zhang *et al.* (2015) developed a method called DupPredictor aimed to detect duplicate questions on SO. The proposed method computes the resemblance of two questions via matching titles, descriptions of questions, tags, and finally, the latent features conforming to the topic allocations that are recognized from natural language explanations of the posted questions. The output revealed that DupPredictor has the potential to gain recall-rate@5, recall-rate@10, and recall-rate@20 by accomplishing scores of 42.3, 53.3, and 63.8 percent, correspondingly. Furthermore, they also compared DupPredictor against the standard search engine of SO, and other approaches to identify duplicate bug reports. The output revealed that DupPredictor has considerably enhanced these baseline methods by approximately 10.2 ~717.9 percent. Later on, Ahasanuzzaman *et al.* (2016) extended the work of Zhang *et al.* (2015) by proposing a new technique called Dupe, aimed to identify the duplicate questions on SO. First of all, they conducted a manual examination to know the reasons behind the users submitting duplicate questions on SO, which ultimately lead to suggesting Dupe. The results revealed that Dupe is capable of identifying duplicate question with considerable precision. They also compared Dupe with state-of-the-art methods like DupPredictor (Zhang *et al.*, 2015). However, the results revealed that Dupe performed well by accomplishing good recall rate compared to DupPredictor (Ahasanuzzaman *et al.*, 2016).

Gupta and Reddy (2016) investigated the issue of suggesting edits to reopen the prior closed questions. The intent is to propose users with edit features that will enable them to enhance the text of their previously posted closed question. The edit features could efficiently be proposed through knowing the editing style of the available skilled users on SO. The non-established users are having fewer skills of editing, thus have considerably fewer chances of reopening the questions (4.4 percent) in comparison to the established users (6.4 percent) on SO. The reasons being the established users have good skills of editing. They proposed editing to the community users through learning the distinguishing features of editing extracted from the edited questions of the community established users.

Singh and Simperl (2016) developed a system known as Suman system, aimed to detect answers for the unanswered questions posted on SO and Reddit. The system utilizes a combination of keywords-based semantic search along with the traditional text-based search to identify answers to the unanswered questions. The Suman system is also capable of recommending the expert members who are proficient in answering those questions, thus help to reduce the queue of unanswered questions. The evaluation of Suman system revealed that the keywords produced by Suman turned out to be at a higher rate than the original keywords from the sites. The assessment also depicts that the participants approved the algorithm rating given to the answers for unanswered questions by the proposed system.

Summary and future work: in this section, the majority of the research studies have focused on particular characteristics of questions posted on SO. The quality of question plays a vital role in getting the precise help from the professionals on SO. The research studies discussed several missing characteristics of questions ultimately lead to delay in response or no response, questions deletion, and imprecise response. The research studies recommended several features that need to be adopted by the users of Q&A sites while posting questions. Some of the recommended features that every questioner must adapt while posting a question on SO are as follows: inclusion of tags, precise titles/length of titles, adding code snippets, adding examples, and so on. All these recommendations are essential for software developers while posting questions since it will increase the chance of getting accurate and useful answers from the diverse professionals on SO. In future, the research should focus on the following specific issues:

- Currently, no specific tools or support could automatically recommend and assist software developers in formulating quality questions not only on SO but also on other several Q&A websites.
- There is a dire need for more rigorous methods aimed to predict good-quality answers on Q&A websites.
- Recognition of emotion is quite a hot and essential area of research in social software engineering and specifically in social Q&A websites. The current Q&A websites lack tools for embedding emotional intelligence to facilitate community members.
- The majority of the research studies mainly focused on posts related to Java programming language to assess the quality of questions posted on SO. In future, the researchers should consider more diverse languages for assessing questions quality or proposing guidelines for posting valid questions on SO.

4.1.2 Reputation and reward system. This section includes research about investigating the role of reputation and reward system practiced on SO platform, which is the design characteristic of SO. This area seems to be non-technical in software development. However, we consider it relevant since it is the social aspect of professionals on SO which is quite useful to assess, especially regarding their knowledge contribution on SO about software development tools/methods. The users gain rewards and badges on SO based upon their level of knowledge contribution and reputation. This information is very relevant for developers while seeking help, since knowing such attributes of professionals is quite useful in deciding which solution to follow about tools/methods. Another aspect is that more professionals are attracted and motivated to contribute for gaining fame in the community. Thus, the papers included here mostly investigated the behavior of professionals, the earning of badges and its impact on SO community, the earning of rewards in driving professionals to contribute and so on, as chronologically summarized in Table IV. We can observe that all the research papers included here have assessed semi-automatically millions of SO posts/users for their works.

Li *et al.* (2012) investigated the influence of badges on user engagements through utilizing econometric models in SO. The output revealed that a large number of badges apparently proved to encourage users to engage and give more in all kind of activities performed on SO. Unexpectedly, the results depicted that the required activities not listed to be fulfilled for earning a badge are also influenced by the status of attaining the badge.

Paper	Investigation	Technique/tool/algorithm	Data source	No. of posts/users
Lotufo *et al.* (2012)	Semi-automatic	Non-Parametric Spearman's test	SO	5.2 M Data, 400 K user
Li *et al.* (2012)	Semi-automatic	*t*-test, Econometric analysis	SO	58,479 SO users
Anderson *et al.* (2013)	Semi-automatic	Experiments, Optimization	SO	3.2 M
Grant and Betts (2013)	Automatic	MySQL, Visualization	SO	1,295,620 users
Sinha *et al.* (2013)	Semi-automatic	LR Model	SE	10.1 M
Movshovitz-Attias *et al.* (2013)	Semi-automatic	PageRank, Singular value decomposition	SO	10,311,875 posts, 1,295,620 users
Bosu *et al.* (2013)	Semi-automatic	Gephi, Metrics (accepted ratio, unanswered ratio, no-response ratio, etc.)	SO	1.3 M users
Bazelli *et al.* (2013)	Semi-automatic	LIWC, ANOVA and Tukey's HSD test	SO	15 M
Halavais *et al.* (2014)	Semi-automatic	Paired *t*-test, Generalized est. equation	SO	15 M
Hart and Sarma (2014)	Semi-automatic	Survey (34 participants)	SO	556,276
Cavusoglu *et al.* (2015)	Semi-automatic	Cognitive evaluation, Organismic integration Theory, *t*-test, Difference-in-Differences method, Propensity scoring	SO	46,571 users
Sinha *et al.* (2015)	Semi-automatic	Latent variable modeling, Quadratic assignment procedure (QAP)	SO	22.5 M posts, 3.7 M users
Jin *et al.* (2015)	Semi-automatic	Gamification-influenced metrics (Rapid response, Accepted answer, Tag score)	SO	1,223,088 posts, 101,291 users
Marder (2015)	Semi-automatic	Regression analysis, Delta method	SO	2 M users

Table IV. Chronological summary of reputation and reward system-related SO literature showing the paper, investigation, technique/tool/algorithm, data source, and the number of posts/users used in the study

Lotufo *et al.* (2012) examined the utilization of gamification concepts in SO, and the possibility of transferring and applying these concepts to bug tracking systems. The output revealed that SO gamification techniques are possible to be used for tackling such issue via encouraging the participants to contribute more quality content, through filtering out the significant contents, and by creating a swift and reliable self-control system. Besides, it was revealed that majority of these mechanism are possible to be applied to specific bug tracking systems. Similarly, Cavusoglu *et al.* (2015) also depicted a solid empirical proof, the significance of the badges and the efficacy of gamification in steering voluntary activities and contributions. Later on, Jin *et al.* (2015) investigated certain gamification-inspired metrics associated with the response time of a posted question. The output of experiments revealed that approximately 92 percent of the users have a few rapid responses. Also, the accepted answers association with rapid responses is not very clear. Nevertheless, it was apparent that rapid responses considerably possess tags which did not obey normal tagging tendencies. Similarly, the different works of Anderson *et al.* (2013), Grant and Betts (2013), Marder (2015) investigated salient aspects of user behavior after earning badges, and specifically for assessing the means in which badges can drive users to alter their behavior on SO. The output of these works confirmed that badges drive user behavior in the ways entirely consistent as predicted, i.e., an increase in user activity was noticed before a badge is earned compared to the duration onwards. Besides, the users who receive badges, a considerable amount of change is observed regarding their contribution compared to those who have not earned any badge.

Bosu *et al.* (2013) investigated SO data from four aspects to know the steering factors of reputation building on SO. They give some recommendations to new SO users, who aim to get high reputation scores quite faster. Bazelli *et al.* (2013) investigated SO to identify programmer personality characteristics through utilizing linguistic inquiry and word count (LIWC) technique. They examined the personality characteristics of SO members through classifying them into diverse classes based upon their level of reputation. The results revealed that well-reputed users are more extroverted in comparison to medium- and low-reputed member. Also, the users who received up-voted on their posts show considerably less negative emotions compared to the users who received down-voted on their posts.

Sinha examined the "activeness" of users in 36 forums of SE network specifically from the aspect of posting Q&A, durability in the forums, and the influence of badges and reputation score. They also thoroughly assessed how forums' reward mechanism had facilitated user's participation and contribution in the community. They also examined how users have spread to other SE network forums with the evolution of time, thus exploring and contributing to new forums of the network (Sinha *et al.*, 2013). Movshovitz-Attias *et al.* (2013) investigated via graph analysis approach the reputation and reward system and the contribution shapes of both high- and low-reputation users on SO. The results revealed that the participation of very high reputation SO users specifies that they are the main source of answers on the forum, and particularly compose very high-quality answers.

The work of Hart and Sarma (2014) aimed to analyze the role of social reputation and other characteristics that play in the process of selecting answers from SO by the novice users. The results depicted that novice users judge and select the quality of information depending upon the essential qualities of the answer. The results indicate that social reputation system has considerably no influence on the technical novices to select and filter out the content available on Q&A forums.

Halavais *et al.* (2014) investigated the individual user badge earning process driven by revelation to the achievements of other users. The output depicted that the impact of friends on badge choice is quite weak. However, it does have an impact specifically for topically restricted badges known as tags on SO. Finally, Sinha *et al.* (2015) investigated via latent variable and predictive modeling the dynamics of user reputation with the evolution of time on SO. They showed the application of network analytic techniques, i.e., quadratic assignment procedure (QAP) that is capable of assisting in assessing and suggesting how users in certain Q&A sites incline to fall into same reputation categories with the evolution of time.

Summary and future work: in this section, the research studies focus remained on exploring the role of reputation and reward system practiced on SO platform. The users of SO gain specific rewards and badges based upon their knowledge contribution to the community. The reputation gained by professionals on SO community is very vital to know for others while deciding to accept or reject a solution for a specific issue. The majority of papers focused on investigating characteristics of professionals, the overall impact of earning rewards or reputation in the community, and the impact of rewards or reputation on professional's motivation to contribute to SO community. There is a strong correlation found between registered members earning badges/reputation and the level and quality of their contribution on SO. Some of the future areas worth investigating are as follows:

- There is a lack of research explicitly investigating how the introduction of gamification mechanism affects the natural practices of users, and the quality of the content shared on SO and other Q&A forums?

- It is also worth investigating the norm of users rapid responses and comparing it with a different source of reputation, e.g., earning votes on posts in SO with the aim to assess posts quality.
- The current incentive mechanism offered on Q&A websites/forums has mostly affected the quality of the content shared. Thus, in future, how to design an incentive mechanism which could help in sustaining the Q&A community efficiently, i.e., without compromising on the contents' quality?

4.1.3 Knowledge sharing, learning, and searching. This section summarizes one of the most important aspects of searching relevant knowledge on SO about the issues faced by developers while coding. Software development is a knowledge-intensive activity, and it becomes more critical especially in SO, since SO is all about knowledge searching, sharing, and learning. The area of knowledge sharing, learning, and searching on SO has covered various aspects, such as possibility of automated knowledge sharing, knowledge recommendations, investigating the level of knowledge available on SO, methods for integrating SO search in programming IDE, knowledge accessibility, improving knowledge creation on SO, association and comparison of knowledge sharing/learning on SO with other similar communities, correlating SO with other search engines, identifying success factors for useful knowledge sharing on SO, and so on, as chronologically summarized in Table V. We can observe that these research papers used a wide range of data sources, e.g., SO, GitHub, R-mailing lists, Android Issue Tracker, and SE. Besides, the majority of these research papers examined semi-automatically or automatically the massive size of data ranging (minimum 540 posts, maximum 25 M of posts) for their works.

Ponzanelli *et al.* developed an Eclipse plugin called Seahawk, aimed to leverage the diverse knowledge of SO and to aid developers to seamlessly utilize the existing knowledge of SO without switching the context in their IDE. It enabled users to automatically articulate search queries and retrieve the matching Q&A from SO by presenting an ordered and interactive list of results, associate the relevant discussions to the source code in Eclipse, allow users to import code snippets examples in discussions via drag and drop and attach detailed comments to the link (Bacchelli *et al.*, 2012; Ponzanelli *et al.*, 2013a, b). Later on, Ponzanelli *et al.* further extended their work (Ponzanelli *et al.*, 2013a, b) by proposing a new system called Prompter, an IDE plugin aimed to assist developers in retrieving discussions made on SO without leaving IDE. Their association using a multi-faceted ranking model is evaluated, and when a particular confidence level is exceeded, it ultimately notifies the programmer (Ponzanelli *et al.*, 2014a, b, 2015).

Similarly, Rahman *et al.* (2013, 2014) developed an Eclipse IDE-based web search solution known as SurfClipse, which gathers data from different web search application programming interfaces (APIs), namely, Google, Yahoo, Bing, and SO. The search outputs are given in the IDE considering not only the elements of the particular error into the account but also the context of the problem, reputation, and the search engine suggestions of the outcome links.

Anderson *et al.* (2012) explored the dynamics of activities performed in the community that forms a collection of answers, together with how answers and voters reach with the passage of time and how this affects the output. They found a significant assortativity in the fame of co-answerers, associations amongst reputation and the speed of the answer. Also, the possibility of an answer selected as the suitable one heavily relies on the chronological features of the answer arrivals.

Gómez *et al.* (2013) examined the practice of sharing web link on SO to know how developers determine and distribute innovations. The output revealed that the practice of

Paper	Investigation	Technique/tool/algorithm	Data source	No of posts/ issues/tags/errors
Anderson *et al.* (2012)	Semi-automatic	Different metrics used for measuring the prediction of question long lasting value and whether question sufficiently answered	SO	3.8 M
Bacchelli *et al.* (2012)	Automatic	Apache Solr, SQLite, TF-IDF	SO	6 M
Gómez *et al.* (2013)	Semi-automatic	Coding schemes, Manual classifications	SO	15.1 M
Vasilescu *et al.* (2013)	Semi-automatic	Split-and-Compare approach, ANOVA, Multiple contrast procedure-T, Gini Index	SO GitHub	1,295,622 SO 397,348 Github
Schenk and Lungu (2013)	Semi-automatic	Yahoo geolocation API, Quicksilver tool	SO	15.1 M
Ponzanelli *et al.* (2013b)	Automatic	Preprocessing, Apache Solr, SQLite, TF-IDF	SO	6 M
Ponzanelli *et al.* (2013a)	Automatic	Preprocessing, Apache Solr, SQLite, TF-IDF	SO	6 M
Rahman *et al.* (2013)	Semi-automatic	PageRank, SimHash, User Study, 8 Metrics	SO	25 errors
Rahman *et al.* (2014)	Semi-automatic	PageRank, SimHash, User Study, 8 Metrics	SO	75 errors
Ponzanelli *et al.* (2014b)	Semi-automatic	User studies, Different metrics Used, TF-IDF	SO	
Ponzanelli *et al.* (2014a)	Semi-automatic	User studies, Different metrics Used, TF-IDF	SO	
de Souza *et al.* (2014b)	Semi-automatic	Preprocessing steps, LR, Naïve Bayes (NB), Multilayer perceptron (MLP), SVM, J4.8 DT (J4.8), RF, K-Nearest neighbors (K-NN)	SO	119,832
Vasilescu *et al.* (2014)	Semi-automatic	Text mining, qualitative survey	SO, R-help	r-help–344,854 SO – 67,248
Vasilescu (2014)	Semi-automatic	Text mining, qualitative survey	SO, R-	r-help-344,854 SO – 67,248
Ye *et al.* (2014)	Semi-automatic	Preprocessing steps, Information Gain algorithm (InfoGain), Bayesian logistic regression (BLR), LR, Lucene search	SO	4,286 posts
Zou, Ye, Lu, Mylopoulos and Zhang (2015)	Semi-automatic	BLR, LR, Lucene search engine, TF-IDF	SO	27,258 posts
Rekha and Venkatapathy (2015)	Manual	Survey (57 participants)	SO	
Wang, Yin, Wang, Yang and Zou (2015)	Semi-automatic	Preprocessing steps, VSM, LSI, LDA	SO and Android IssueTracker	1,169,415 posts, 151,815 Android issues
Yang *et al.* (2015)	Semi-automatic	Tag Co-occurrence Graph, Normalized discounted cumulative gain (nDCG), Kendall Tau, Pearson rank correlation	SO	18 M Posts 35.2 K tags
Singh *et al.* (2015)	Semi-automatic	LDA, PageRank, Student survey	SO	25 M

(*continued*)

Table V. Chronological summary of knowledge sharing, learning and knowledge searching-related SO literature showing the paper, investigation method, technique/tool/ algorithm, data source, the number of posts/sentence and the number of issues/tags/ errors used/evaluated in the study

Paper	Investigation	Technique/tool/algorithm	Data source	No of posts/ issues/tags/errors
Caalefato *et al.* (2015)	Semi-automatic	Different metrics, LDA	SE	300 K
Ponzanelli *et al.* (2015)	Semi-automatic	User studies, different metrics Used, TF-IDF	SO	
Zagalsky *et al.* (2016)	Semi-automatic	Text mining, Qualitative survey	SO R-help	R-Mailing List – 315,297, SO – 67,013
Ponzanelli, Bavota, Mocci, di Penta, Oliveto, Hasan, Russo, Haiduc and Lanza (2016)	Semi-automatic	Island Parser, H-AST, Tesseract-Ocr, BoofCV, VSM, YouTube API, Google2Srt, Snowball stemmer, MoJo Measure, Lucene	SO	540
Ponzanelli, Bavota, Mocci, di Penta, Oliveto, Russo, Haiduc and Lanza (2016)	Semi-automatic	Island Parser, H-AST, Tesseract-Ocr, BoofCV, VSM, YouTube API, Google2Srt, Snowball stemmer, MoJo measure, Lucene	SO	540
Chen, Gao and Xing (2016)	Automatic	Association rule mining (ARM), Continuous skip-gram model, Word embedding	SO	9,970,064 posts, 36,197 tags
Ye, Xing and Kapre (2016)	Semi-automatic	OpenCoding, Correlation profile, Strict power law detection, Jensen-Shannon divergence	SO	24,053,291
Chen and Xingg (2016)	Semi-automatic	Different metrics, Cross-correlation method, Dice coefficient	SO, Google Trends	SO 7.2 M, Google Trends 185 terms
Joorabchi *et al.* (2016)	Automatic	Text mining, Wikipedia minner, Gephi	SO	188,548

Table V.

sharing link is very beneficial on SO, since SO is an essential source for distributing software development innovation and is used for seeking information/help about software development/tools.

Vasilescu *et al.* (2013) examined the associations between SO activities and the development process shown by the modifications in the code committed on coding repository GitHub. The output depicted that more frequent GitHub committers post fewer questions and comparatively give more answers than others. It also revealed that frequent questioners on SO share their work comparatively in a reduced systematic way than developers that do not post questions.

Schenk and Lungu (2013) investigated SO forum from the aspect and level of use based on geographical regions. They summarized knowledge sharing from lower level users to the higher levels, i.e., geographical regions. They came to know that Europe and North America are the main contributors at an equal level, Asia comes after them having more contributions from India, and Oceania participation is lower than Asia but has more contribution compared to the combined participation of South America and Africa.

de Souza *et al.* (2014b) developed a method that utilizes the present "crowd knowledge" on SO to propose any valuable knowledge that can help developers in development activities. The suggested technique proposes an ordered list of Q&As from SO depending upon the searched terms. They proposed a classifier aim to tackle only "how-to" posts on SO. The output of the study revealed that for 77.14 percent of the evaluated activities, at least one proposed pair turned out to be appropriately related to target software development issue. Besides, for almost all activities, at least one proposed pair possessed a source code snippet deemed reproducible. Later on, Vasilescu *et al.* examined the involvement of participants over the passage of time, since SE came into practice by using a case study of R-data analysis tool. The output revealed that activities to help users were diverted more toward SE than r-help. The users active on both channels are more active compared to those who concentrate specifically on only one of them. The users give quick answers on SE compared to r-help, signifying us they are inspired by the gamified environment (Vasilescu *et al.*, 2014, Vasilescu, 2014).

Ye *et al.* (2014) developed an approach called interrogative guided re-ranking method, aimed for question-oriented software text retrieval. They generated numerous software document classifiers, which learn from a various number of Q&A sets present on SO. Afterwards, they applied the document classifiers to the document corpus and presented a re-scoring method, which integrates the classification score and the text retrieval score thereby enhancing the precision of retrieval. Later on, Zou, Ye, Lu, Mylopoulos and Zhang (2015) extended the work of Ye *et al.* (2014) through investigating the "answer style" of software questions posted with different interrogatives. They generated classifiers in the software text corpus and suggested a re-ranking method to improve the search outcomes. The evaluation revealed text retrieval enhancement in comparison to the baseline. All these evaluations outputs depict that the proposed method is capable of identifying answers to FAQs with more accuracy.

Wang, Yin, Wang, Yang and Zou (2015) developed an automatic method through combining the semantic similarity with temporal locality among the available Android issues and SO posts. They examined the suitability of three retrieval approaches, namely, vector space model (VSM), latent semantic indexing (LSI), LDA, and was revealed that VSM outperforms the rest. The output revealed that the method achieved a precision of 54.82 percent for top ten recommendations when suggesting SO posts to Android issues and 66.83 percent vice versa which is considerably better than the available methods.

Rekha and Venkatapathy's (2015) work investigated the usage of SO via conducting a survey. The output depicted that SO probably do not provide the new learners with their learning necessities, which thus do not encourage many users to contribute and frequently interact on SO. Similarly, Yang *et al.* (2015) focused on investigating thoroughly whether the knowledge creation process can be boosted through knowing and utilizing the integrated effects of the interests and the level of expertise of the users along with motivations, i.e. both intrinsic and extrinsic. The output revealed that how they dispense and associate throughout the users of the forum and topics. In addition, the results also revealed that how topic-specific mixtures of the users' motivations and expertise can assist in increasing the process of knowledge creation.

Singh *et al.* (2015) developed an approach to supplement the learning stuff automatically through real-world questions about the concept learned. They utilized Q&A from SO to supplement the interface of e-textbook, thus relating the concepts taught to the knowledge seekers. Similarly, Caalefato *et al.* (2015) examined the factors of Q&A that assist in creating and sharing knowledge. They made an empirical model based on existing literature of the factors that forecast the probability of receiving an appropriate answer when posting a new question. The factors in the model are grouped into three classes of features; presentation quality, affect and time. They used a multivariate logistic regression framework aimed to guess the predictability of accomplishment of a question dependent upon the author's group of predictors, i.e., the metrics that make into action the time, affect, and presentation quality.

Chen and Xingg (2016) investigated whether the questions developers post on Q&A sites are associated with the information programmers seek by using different search engines. They examined the association of 185 common famous technical terms programmers seek on Google and post on SO. The findings depicted that the queried/posted technical terms have a strong association with the evolution of time. The querying/posing of new technical specific terms possess stronger association in comparison with general technical and older terms.

Ye, Xing and Kapre (2016) aimed to develop an approach to investigate the URL distribution on SO. They utilized open coding method for assessment and proposed some quantitative analysis methods to explore the structural and dynamic characteristics of the evolving network of knowledge present in SO. With this study, they gained an in-depth know-how of knowledge sharing the process on SO and depicted the suggestions of URL sharing attitude for Q&A website design and developers who use the multi-facet rich knowledge on SO. Similarly, Chen, Gao and Xing (2016) developed a novel method to suggest similar libraries dependent upon the knowledge base of similar libraries extracted from the tags of millions of SO posted questions. The uniqueness of the developed method is to resolve analogical libraries questions through integrating modern word embedding method and the existing domain-specific relational and categorical knowledge extracted from SO. The output of the research revealed that the proposed method is capable of making a definite recommendation of analogical libraries.

Zagalsky *et al.* (2016) empirically investigated how R programming community creates and curates knowledge linked with the Q&A in its communication channels. On SO, the knowledge shared and curated is mainly crowd-sourced and has topic restrictions. On the other hand, the R-mailing list knowledge shared is participatory and has no topic restrictions. The work of Joorabchi *et al.* (2016) applied a heuristic research method together with a suitable text mining method to examine topics in the posts discussed on CQA site SO. The output revealed that most frequently discussed topics are about computer science especially software development, and hence, lead the authors to categorize the issues faced by different learners in the said area.

Ponzanelli, Bavota, Mocci, di Penta, Oliveto, Hasan, Russo, Haiduc and Lanza (2016) and Ponzanelli, Bavota, Mocci, di Penta, Oliveto, Russo, Haiduc and Lanza (2016) focused on developing a novel method called CodeTube, aimed to retrieve related pieces of information from the video tutorials on software development. The developed method retrieves video fragments through integrating the present code information retrieved within video frames along with the speech data given by audio transcripts. The developed method also supplements software development video fragments automatically with associated SO discussions on the topics. The output of survey and interviews revealed that CodeTube has much more potential compared to the existing video tutorials providers on software development.

Summary and future work: in this section, a comprehensive overview of research done on all aspects of knowledge sharing, learning and searching on SO is presented. The research works here primarily focused on different issues related to knowledge sharing, learning and searching on SO and proposed several methods for knowledge recommendations, integrating SO search in programming IDE, and improving knowledge creation on SO. However, there are still some areas which need further research and are as follows:

- The methods for integrating SO search in programming IDE needs to be improved since they currently support only Eclipse IDE. Thus, in future developing it as a web service application programming interface (API), so that more developers can use it in real time and also enable them to use the API in their different IDEs.

- Another area of research worth investigating is developing a tool which could mine analogical APIs across different libraries or programming languages in SO or other similar software repositories, e.g., GitHub.
- The majority of current research focused on domain-specific search in Q&A websites, thus in future more efficient techniques are needed which could support entity-centric or semantic search on SO and other similar Q&A communities.
- The existing methods developed for question-oriented text retrieval lack performance, thus in future investigating applying fuzzy learning or other efficient methods for text classification will improve the classification precision.

4.1.4 Exploration of gender and experts or expertise. The studies included in this section are comprised of identifying experts and genders present on Q&A websites. The papers discussed several topics like identification of right experts/expertise for a newly question posted, participation level of genders, knowledge of expertise in various fields, association of knowledge of professionals with their ages, contribution level of expertise, behavior of experts/expertise on Q&A websites, and so on, as summarized chronologically in Table VI. We can observe that these research papers studied mainly SO and partially other data sources, i.e., Turbo Tax and GitHub, for exploration of gender and experts/expertise. Besides, the majority of these research papers examined semi-automatically or automatically massive amount of data (minimum 105,533 posts and 694 users, maximum 20 M SO posts and 1.7 M users) for their works.

Pal and Konstan (2010) investigated question selection bias to know the users' behavior on CQA sites. They depicted that users give preference to answering only those questions where they believe to contribute substantially. The results of bias evolution depicted no considerable variations with the passage of time, signifying that they primarily come from users' inherent properties. Similarly, Pal, Harper and Konstan (2012) developed a mathematical model to specify the questions selection preferences (QSP) procedure of the CQA members. They specified QSP depending upon the existing value of previous answers to a question. The outcome signified that QSP is very useful in identifying the current and future potential experts. Thus, they supplemented the previous research work by developing a statistical model capable of identifying existing and future potential experts in the common CQA sites. In another study, Pal, Chang and Konstan (2012) presented a chronological study of the existing experts in CQA websites and assessed the variations in their behavioral forms with the evolution of time. The results depicted that the models depending upon the evolution data of the users can turn out to be more efficient at identifying experts compared to the models which neglect support for evolution factor.

Chang and Pal (2013) developed a routing framework for questions that takes into account the suitability, accessibility, and expertise of the community users. In their previous work (Pal, Harper and Konstan, 2012) on the same issue, they focused on directing a question to the most suitable expert. However, in this work, they aimed to direct questions to the group of experts who is keen to contribute and give considerably better answers to the questions posted on SO. The output of the empirical examination revealed that the more answers and comments a question has, the more worth a question-answer thread has. Besides, the output of the experiments over a huge data set of SO revealed that the proposed recommendation model is very efficient and performs well compared to other existing baseline models. Similarly, Riahi *et al.* (2012) proposed an approach for identifying most

Paper	Investigation	Technique/tool/algorithm	Data source	No. of posts/users
Pal and Konstan (2010)	Semi-automatic	Ridge and LR, Gaussian distribution	SO TurboTax	TurboTax 1,321,502 Data
Riahi *et al.* (2012)	Semi-automatic	Language model, TF-IDF, LDA, Segmented topic model (STM)	SO	118,510
Vasilescu *et al.* (2012)	Semi-automatic	Python Tool, Mann-Whitney tests	SO	1,078,708 users
Posnett *et al.* (2012)	Automatic	Stickiness coefficient	SE	261,317
Hanrahan *et al.* (2012)	Automatic	Pearson's product momentum/rank correlation coefficient	SO	6.1 M Posts 640 K users
Pal, Harper and Konstan (2012)	Semi-automatic	Z-score model, Support vector machines (SVM), DT, Gaussian discriminant analysis	SO TurboTax	3,272,523 SO Data TurboTax 1,321,502 Data
3272523 SO, TurboTax 1321502	Semi-automatic	Z-score model, GMM based clustering algorithm, Bayesian information criteria, SVM	SO TurboTax	
Venkataramani *et al.* (2013)	Manual	Mining	SO GitHub	SO-5,000 GitHub – 20 Java
Chang and Pal (2013)	Semi-automatic	Spectral clustering, LDA, RF, SVM	SO	649,702
Morrison and Murphy-Hill (2013)	Semi-automatic	LR	SO	1,694,981 users
Yang *et al.* (2013)	Automatic	Gaussian mixture model (GMM), Gibbs sampling (GS), PageRank	SO	105,533
Yang, Tao, Bozzon and Houben (2014)	Automatic	Expertise metrics, Mean expertise contribution	SO	1,544,610
Meng, Gandon and Faron-Zucker (2014)	Automatic	LDA	SO	15,327,727
Guimaraes *et al.* (2015)	Automatic	Minimum description length, Continuous wavelet transform peak finding, Levenberg-Marquardt (LM)	SO	19.8 M posts, 1.7 M Users
Lin and Serebrenik (2016)	Semi-automatic	strawman, genderComputer, Gender guesser, Google+, etc.	SO	694 users
Kumar and Pedanekar (2016)	Automatic	Graph partitioning, Link based methods, pageRank	SE	157,000 users
Murgia *et al.* (2016)	Semi-automatic	User study	SO&Git	20 M SO, 50 errors

Table VI.
Chronological summary of an exploration of gender and experts or expertise-related SO literature showing the paper, investigation, technique/tool/algorithm, data source, and the number of posts/users used in the study

suitable experts whenever a new question is posted on SO. They examined the suitability of two statistical topic models, namely, LDA and segmented topic model (STM), for solving this problem. They compare these methods with more traditional approaches like TF-IDF and Language Model. The output revealed that LDA and STM performed best compared to other methods in retrieving a suitable set of best experts for a newly posted question. More specifically, the results depict that STM performed consistently better compared to LDA.

The work of Vasilescu *et al.* (2012) aimed to investigate the level of participation of women on SO quantitatively. They made a comparison through evaluating their level and duration of participation on SO in contrast to men. The output of the study depicted that men's participation is dominant on SO, i.e., more participation, produce more reputation, and involve more in the "game". Similarly, Posnett *et al.* (2012) investigated empirically the duration of expert users who post and the quality of answers in SE. The output depicted that the score of answer reduces and the duration of people with the community is not associated with the quality of their answers. It was interesting to see that the answering ability, which is not like the reputation, is evident from the initial time and stays until during one's duration with the community. On the contrary, users giving low-rated answers probably have performed so right from the initial stage.

Hanrahan *et al.* (2012) reported about the design of hybrid intelligence systems via investigating SO. They aimed to identify and make indicators for the problematic issues and experts users. They also aimed to investigate how such difficult problems are tackled and circulated across the community amongst several experts.

Morrison and Murphy-Hill (2013) investigated the tight association between the age of programmers and their programming knowledge. They investigated the degree to which older age programmers obtain knowledge about available newest technologies. The output depicted that reputation score of programmers gets higher well into the age of 50s, and at the age of 30s leans toward exploring less new areas compared to those who are younger or older from them.

Venkataramani developed a model to order the expertise of programmers in a specific area by mining their activities in various open-source development projects. To depict the applicability of the model, they developed a recommendation system for SO, which utilized the data extracted from open-source code repository GitHub (Venkataramani *et al.*, 2013). Similarly, Yang *et al.* (2013) presented a new probabilistic generative model with Gaussian mixture model (GMM) hybrid known as the Topic Expertise Model (TEM), aimed to model the topics and expertise by combining the textual content model and the link structure analysis. The output of TEM leads to build the CQARank, aimed to quantify the user expertise and interests score associated with diverse topics. The proposed method is capable of identifying the relevant experts not only with their similar topical choices but also the relevant expertise in the specific topic by utilizing the history of the posts based on numerous community reviews and voting.

Later on, Yang, Tao, Bozzon and Houben (2014) proposed a new metric for identifying experts which ultimately helps in identifying the competencies of users by concentrating on the worth of their contributions on SO. The output revealed two categories of users: sparrows (more active users) and owls (more expert/knowledgeable users). Besides, they discussed the contribution of owls and sparrows regarding their engagement, knowledge creation, and sequential growth in the online community SO. Similarly, Meng, Gandon and Faron-Zucker (2014) proposed an approach known as Question & Answer Social Media (QASM) system built upon the social network analysis to tackle two main resources in CQA sites: users and contents. They first developed QASM vocabulary which is utilized to formalize not only the interests levels but also the expertise of users on the topics. They then discussed the proposed method aimed to elicit this required knowledge from CQA sites. Afterwards, they depicted how this knowledge is utilized not only to identify appropriate experts for a question but also to find similar questions on SO.

Guimaraes *et al.* (2015) investigated the users' associations to topic-based Q&A community SO and to know how these associations characterize enduring community dynamics. The output revealed exciting findings, i.e., the expert users mostly change and engage in several communities, the revisiting expert users play vital role in the sustainability of the community and communities are strongly dependent on each other.

Murgia *et al.* (2016) investigated the possibility of human-bot interaction on SO. They produced a bot imitating a regular expert user responsible to answer the questions related to resolving 50 Git error messages extracted from SO posts. In the first evaluation, the bot imitated to be a human and in the second time the bot exposed its system identity. The typical reactions from the users on SO elicited by these two different bot variants (though functionally similar) were considerably dissimilar.

Kumar and Pedanekar (2016) presented the theme of mining forms of user expertise in a unique online social Q&A developer's community where expert users often answer the questions asked by other expert users. They reported their research findings on investigating the SE sub-community called Super User community by mining data of 157,000 expert users. The research output revealed that expert users in social Q&A community platforms are mostly engaged in diverse area subjects instead of sticking to contribute in one area.

Lin and Serebrenik (2016) assessed the applicability of 16 different gender identification approaches on several data sets derived from SO, namely, IWC, FLOSS, and Diversity comprised of 694 user profiles in total. They applied three metrics to evaluate the data sets, namely, Adjusted Rand Index, wam and cub. The output depicted that the approaches integrating different data sources perform considerably better.

Summary and future work: in this section, the research works focus remained on identifying the potential experts/expertise and their role on Q&A websites. The issues addressed in the selected researched papers were quite diverse and interesting, for instance, the role of identifying right experts/expertise on SO for a newly posted question, investigating the level of participation across different genders, the expert knowledge of professionals across various fields, investigating the association of knowledge of professionals with their ages, assessing the contribution level of expertise on SO, and the distinctive behavior of experts on Q&A websites, etc. Some the areas worth investigating in future are as follows:

- The majority of the current works did not apply some advanced methods like deep learning and knowledge graph methods to understand the evolution of experts/expertise and Q&A communities as well, so in future, the application and suitability of these methods need investigation too.
- There is a need of research on different recommendation strategies for routing questions to a relevant expert or group of expertise.
- It will be interesting to investigate why the quality of content shared on Q&A decreases as the community evolves? What measures need to be adopted by Q&A communities to tackle this issue?
- There is a need for research on investigating how does gender orientation affect individual participation/contribution in Q&A communities?

4.2 SO content applications

In this section, we included all those studies where SO platform content was mined and utilized for different software development activities. Specifically, SO posts were utilized for several activities of SDLC, e.g., automatic comment generation, extracting code fragments for fixing bugs, extracting software features/topics, and so on. This will enable practitioners/researchers to get an overview of which stages of SDLC are mostly supported/utilized in SO and identifying which stages of SDLC are under-utilized for improving the quality of software.

4.2.1 Extracting, analyzing and improving source code. This section entails all those papers that worked on the methods for extracting source code fragments in posts/informal documentation, analyzing those source code fragments for fixing issues/bugs, and finally improving the quality of extracted source code from SO posts for their context. We can observe in Table VII that these research papers study and use mainly SO and GitHub for

Paper	Investigation	Technique/tool/algorithm	Data source	No. of post/code snippets/issues
Rigby and Robillard (2013)	Semi-automatic	Vector space model (VSM), Latent semantic indexing (LSI), RecoDoc	SO	188
Subramanian and Holmes (2013)	Semi-automatic	Snippet analysis framework, JDT	SO	21,250 code
Correa and Sureka (2013b)	Semi-automatic	Experiments (Textual similarly analysis and contextual data analysis), surveys	SO, Android, Chrome	9.9 M posts, 36,286 Android and 142,175 Chrome Issues
Subramanian *et al.* (2014)	Semi-automatic	AST, ESPRIMA Parser	SO, GitHub	15 M SO, 5,000 snippets
Diamantopoulos and Symeonidis (2015)	Automatic	Performed certain preprocessing steps, VSM, TF-IDF	SO	300 K
Arwan *et al.* (2015)	Semi-automatic	Preprocessing steps, LDA	SO	159
Vinayakarao *et al.* (2015)	Semi-automatic	Abstract syntax tree (AST), TF-IDF	SO	19 M
Sanchez and Whitehead (2015)	Semi-automatic	User study	SO	25 M SO
Gao *et al.* (2015)	Semi-automatic	AST, GumTree, Linear Optimal Algorithm	SO and Github	10 Pages, 73,868 issues
Yang *et al.* (2016)	Automatic	Roslyn, JDT, SpiderMonkey, Python, Google search API	SO	3 M code snippets
Tavakoli *et al.* (2016)	Automatic	Simian tool, Wala tool, Survey	SO	40

Table VII. Chronological summary of extracting, analyzing and improving source code related SO literature showing the paper, investigation method, Technique/Tool/Algorithm, data source, the number of posts/sentences, number of code snippets and no of issues used/evaluated in the study

extracting, analyzing, and improving source code. Furthermore, the majority of these research papers examined semi-automatically or automatically the vast amount of data (minimum 40 posts, 5,000 code snippets, and 36,286 issues, whereas maximum 25 M SO posts, 3 M code snippets, and 142,175 issues) evaluated, respectively for their works.

Rigby and Robillard (2013) developed a traceability retrieval method known as automatic code element extractor (ACE), aimed to retrieve the code components present in several documents. In comparison to the earlier works, this new method does not need an index of code element to discover links. Thus, makes it mainly suitable for the exploration of informal documentation. Furthermore, the developed three-feature decision tree (DT) classifier output reveals precision of 0.65-0.74 and recall value of 0.30-0.65 dependent on the theme of the document. Similarly, Subramanian and Holmes (2013) analyzed code snippets to extract valuable information from the plain-text pieces in SO posts. The output depicted the identification of 253,137 method calls and type references from the available SO code snippets. The output revealed that detecting these useful structural associations from code snippets do better than lexical search in practice. Later on, Subramanian *et al.* (2014). extended the work of Subramanian and Holmes (2013) in the same line by developing a method called Baker. They use the constraint-based method to uniquely detect fine-grained type references, method calls, and field references present in source code fragments with high accuracy. The method was assessed through showing its capability in accurately linking source code to documentation.

Diamantopoulos and Symeonidis (2015) addressed the issue of recommending matching questions in SO by evaluating approximately 300 K posts. They proposed a matching scheme which utilizes not only questions' title, tags, and body, but also makes use of the available source code snippets. The output depicted that the presence of code snippets are a vital source of information for identifying links or for deciding to know whether some user has already posted a question in SO. Besides, it is capable of conducting effective searching on SO even with less articulated information provided in the questions posted. Similarly, Arwan *et al.* (2015) proposed a method for discovering code on SO by making use of concept location in the data preprocessing step. They used LDA for creating inference concept location from the source code. In the proposed system, software developers will search for concepts and then the system will in return propose source code snippets grounded on the relevant concepts, Ye, Lu, Mylopoulos and Zhang.

Vinayakarao *et al.* (2015) proposed an automatic method aimed to create a corpus of structurally different but functionally alike source code samples from the unstructured sources. They utilized information retrieval and partial program analysis methods to reach a corpus of such code examples. The output of the method depicted that it retrieves structurally different snippets by accomplishing a precision of 83 percent. The work of Sanchez and Whitehead (2015) introduced the concept of source code curation to do some actions comprised of determining certain source code of concern, refine it, and finally giving it in a significant and structured way. They developed an approach called Vesperin system, aimed to curate the source code targeted in the direction of curating Java code samples present on 50 K SO posts.

Correa and Sureka (2013b) empirically investigated the role and effect of SO in resolving issues by conducting several experiments on data from two issue trackers of Google Chromium and Android. They conducted experiments based on textual similarly analysis and contextual data analysis to suggest SO posts for the subsequent bug report. The output revealed the existence of linkage to SO in detailed discussions and depicted association among a lower mean time to repair (in one data set) with the existence of SO links. Besides, the results also revealed that the aggregate number of comments is less given in response in a situation when bug reports containing SO links are present but increases in situations when no SO links are present.

Later on, Gao *et al.* (2015) developed an automatic method to fix the numerous repeating crash bugs through thoroughly analyzing SO. They mined the queries from crash traces and got a list of Q&A pages through making use of a search engine. Then, assessed the code present in every page, applied edit scripts to the source code, and finally filtered out the wrong patches. The output revealed that the method is very exact in correcting real-world crash bugs and can supplement the available bug-fixing techniques having support for tackling different bugs' class.

Tavakoli developed an Eclipse plugin called Example Recommender (ExRec) to assist developers in refining the quality level of code snippets present in SO posts. With the help of ExRec, the developers can quickly shift a piece of code present in an answer to a chosen question in SO to its code, apply changes to that piece of code, and then post the better-quality code snippet back to SO as a response to that specific question. Thus, this tool enables programmers not only to improve the code snippet but also the quality of answers re-posted on SO. The output of user study with 13 skilled developers participants revealed that ExRec is capable of efficiently enhancing the quality level of the code snippets present on SO (Tavakoli *et al.*, 2016).

The focus of Yang *et al.*'s (2016) work remained on investigating the usability of code snippets present in SO posts. The intent behind performing the usability analysis is to know the degree to which user-written code snippets in four languages could be utilized as chunks for automatic program generation. Python and JavaScript turned out to be the languages for which the major of the code snippets are usable. On the contrary, Java and C# have the lowest usability rate. Besides, the qualitative analysis performed on usable Python snippets depicted the features of the answers that solve the posted question. Finally, the authors utilized Google search to examine the orientation of usability and the natural language annotations present near the code snippets and discover how such snippets in SO can provide a base to be used for automatic program generation.

Summary and future work: in this section, the majority of the research studies have focused on proposing methods for efficiently searching and extracting source code snippets from SO posts and analyzing those extracted snippets for several purposes: to improve the quality of code snippets for efficient reuse in their own context, reposting the improved quality code snippets back to SO for others reuse and utilizing it for fixing issues/bugs. It is one of the most critical areas for developers, but less research work has been done so far. Some of the areas that need further investigations in future are as follows:

- There is a need for research on developing new methods for improving the quality of extracted code snippets since the existing proposed methods lack in recommendation precision and performance.
- It will be worth to investigate on improving existing methods for fixing issues/bugs with extracted code from SO posts since the current methods lack scalability.
- Moreover, research on improving current methods capability of code extraction via applying deep learning, since the existing methods lack accuracy due to applying basic topic modeling techniques.

4.2.2 Automatic comment generation. This section discusses all those papers that mainly focused on methods for generating comments from SO extracted posts. In most of the cases, the questions posted by developers on SO usually possess code fragments solutions along with descriptions which are usable and helpful for other developers. Thus, the papers included here have discussed how to retrieve and map such posts and use it in their context for generating comments. As can be observed in Table VIII, all of the papers extracted and used data from such posts from SO investigated/assessed semi-automatically (minimum 1.5 M, maximum 10.1 M) posts for their research studies.

Wong *et al.* (2013) proposed a novel approach for generating comments automatically through mining comments from 5,509,302 SO posts by analyzing Java and Android tagged questions to retrieve 132,767 code-description mappings. They mainly focused on mining SO posts for retrieving such mappings and consequently utilized them for automatically generating description comments for identical code segments that are matched in the open-source projects. The output revealed automatically generating 102 comments for 23 Java and Android projects through AutoComment. Later on, Movshovitz-Attias and Cohen (2013) applied statistical language models for predicting comments from 9 Java open-source

Paper	Investigation	Technique/tool/algorithm	Data source	No. of posts
Wong *et al.* (2013)	Semi-automatic	Preprocessing, User Study, S CoreNLP	SO	5,509,302
Movshovitz-Attias and Cohen (2013)	Semi-automatic	LDA, n-grams	SO, 9 Java Projects	10.1 M SO
Vassallo *et al.* (2014)	Semi-automatic	Java Reflection API, VSM, Searches URL	SO	1.5 M
Guerrouj, Bourque and Rigby (2015)	Semi-automatic	ACE, n-gram, TF-IDF, K-means clustering	SO	2.4 M
Ponzanelli *et al.* (2015b)	Semi-automatic	Island Parsing, LexRank	SO	

Table VIII. Chronological summary of automatic comment generation-related SO literature showing the paper, investigation, Technique/Tool/ Algorithm, data source, and the number of posts/sentence used in the study

projects and SO posts containing Java tags. The results revealed that n-grams models are very efficient and time saving in comment completion.

Similarly, Vassallo *et al.* (2014) developed an IDE plugin known as CODES (mining sourCe cOde descriptions from developErs diScussions) by adopting a "social" approach for software re-documentation. The presented CODES approach first accesses/searches SO posts, then retrieves the matched method documentation from software developers' posts, and finally generates Javadoc descriptions from it. The outcome revealed that CODES is capable of retrieving descriptions for approximately 20 and 28 percent of the Lucene and Hibernate methods.

Later on, Guerrouj, Bourque and Rigby (2015) developed a better technique capable of automatically generating the identifier summaries by making use of identifier context within the informal documentation. The uniqueness of their approach is to make use of informal documentation, as well as source code textual information present in it to give precise summaries. They used SO posts with a focus on Android projects and evaluated arbitrarily selected sample of 100 Android identifiers. The output revealed an R-precision of 54 percent concerning the summaries created by the two selected human annotators. Similarly, Ponzanelli *et al.* (2015b) proposed a new technique for summarizing heterogeneous (e.g. code, text, XML) software artifacts. They used and extended LexRank to incorporate various features of distinct-type information enclosed in them. Consequently, the evaluation gave remarkable output and they argued how the improved output advocate that unrelated information present in software artifacts is substantial to explore to progress of the current state-of-the-art summarization techniques.

Summary and future work: this section primarily focused on investigating automatically generating and predicting code comments from SO discussions, and generation of summaries for library identifiers. Program comprehension is an important activity and first task in the maintenance of any software. This area is one of the most critical areas in program comprehension but unfortunately has been under-researched with only five studies so far. Some of the topics worth investigating in future are as follows:

- Investigating on how to improve the existing methods concerning the correctness of the comments generated for methods, since the generated comments have redundant information.
- Developing methods to enable the automatic generation of comments for classes and packages extracted from the posts Q&A website SO.
- Besides, code elements summaries from developers' discussions and bug reports are also worth exploring in future.

4.2.3 API usage. This section summarizes all those papers that have wholly or partially discussed and analyzed about API usage in SO posts for various purposes: augmenting API documentation, detecting API usage obstacles in iOS and Android applications, API-related design problems, API issues and so on, as chronologically summarized in Table IX. We can observe that these research papers used a wide range of data sources, namely, Android, F-droid, Maven, and posts from SO. Besides, the majority of these research papers examined semi-automatically huge amount of data (minimum 1,574 posts and 154 apps, maximum 25 M SO posts and 109,273 apps) for their works.

Paper	Investigation	Technique/tool/algorithm	Data source	No. of post/apps
Parnin *et al.* (2012)	Semi-automatic	Google code search, Spearman's rank correlation coefficient, jSoup	SO	7 M
Stevens *et al.* (2013)	Manual	Stowaway	SO and Android	10.1 M SO, 10,000 APPs
Wang and Godfrey (2013)	Semi-automatic	Latent dirichlet allocation (LDA)	SO, iOS, Android	305 APPs
Kavaler *et al.* (2013)	Semi-automatic	Google code search, Spearman's rank correlation coefficient, jSoup, Javadoc custom Doclet, APK Tool	SO and Android	4.2 M SO, 109,273 APPs
de Souza *et al.* (2014a)	Semi-automatic	LDA	SO	15.1 M SO
Wang, Keivanloo and Zou (2014)	Semi-automatic	SQL scripts, Wild-card search, Mann-Whitney and Cliff's Delta tests,	SO and Programmableweb	50,397
Linares-Vásquez *et al.* (2014)	Semi-automatic	Google code search, Spearman's rank correlation coefficient, jSoup	SO	213,836
Sushine *et al.* (2015)	Manual	Think aloud protocol (six participants), Manual analysis	SO	5,039
Guerrouj, Azad and Rigby (2015)	Semi-automatic	Automated code element (ACE), Mann-Whitney and Cliff's Delta tests	SO and android	591,555 154 APPs
Wang, Malik and Godfrey (2015)	Semi-automatic	Control chart, LDA	SO	141,324
Mastrangelo *et al.* (2015)	Semi-automatic	Island grammar, Heterogeneous abstract Syntax tree (H-AST),	SO Maven	25 M SO, Maven 86,479
Treude and Robillard (2016)	Manual	LexRank, Maximal marginal relevance, Knowledge patterns	SO	1,574

Table IX. Chronological summary of API usage SO literature showing the papers, investigation method, technique/tool/algorithm, data source, and the number of posts/sentences/Apps used in the study

Parnin *et al.* (2012) empirically examined how SO enables crowd documentation. They investigated crowd documentation for three popular APIs, namely, Android, GWT, and Java. They gathered the usage data by making use of Google Code Search, assessed their quality level, and determined things that are covered in SO documentation for these three APIs. The results revealed that the community is capable of producing a useful source of content possessing code samples and discussion that is keenly viewed and utilized by various software developers. Later on, Treude and Robillard (2016) developed an approach capable of supplementing API documentation automatically with "insight sentences" extracted from SO posts that are linked to a specific API type by offering intuition missing in the API documentation of that particular category. Afterwards, they came up with a new machine learning-based approach called supervised insight sentence extractor that makes use of features of the sentences themselves, their configuration, their Q&A, and their creators as well as part-of-speech tags and the resemblance of a sentence to the respective API documentation.

Stevens *et al.* (2013) examined approximately 10,000 free Android apps and established a robust sub-linear association between the reputation of permission and how often it is misused. The outcome revealed that reputation of permission is closely linked with its misuse, whereas influence and interference had a slight impact. Besides, the data analysis of SO revealed that more often used permissions are discussed mostly in questions posted and receive answers as well.

Wang and Godfrey (2013) assessed API-related posts on SO about iOS and Android apps development. They considered API-related posts about API usage hurdles; however, they still manage to identify various iOS and Android API classes that turned out to challenge the mobile app developers. Similarly, Kavaler *et al.* (2013) investigated the association between the usage of Android API classes in approximately 109,273 Android apps and all the posts that specifically discuss these classes on SO. The output revealed that volume of questions expands with usage. However, there is a presence of a non-linear saturation effect, proposing that knowledge of the most popular classes apparently be quite easily found, the number of questions are independent of the available documentation for a class; however, the volume of classes do have an impact, hinting that more difficult classes increase the demand for information, and the circumstances are inverted for answers.

In the same line of research, Linares-Vásquez *et al.* (2014) investigated how do variations in Android APIs increase Q&A in SO and assess whether it is valid or not for certain types of modification. The results revealed that trend of asking questions of Android app developers changes, and thus they ask more questions whenever there is a slight modification in the behavior of APIs or deleting public methods from APIs. Similarly, Wang, Keivanloo and Zou (2014) analyzed modifications among the following versions of APIs and categorized the recognized changes to know how the representational state transfer web APIs progresses over the passage of time. The results revealed that supplementing new methods in the new version leads to an increase in questions and views from software developers.

In addition, Guerrouj, Azad and Rigby (2015) investigated the effect of App churn on the success of different kinds of apps via thoroughly analyzing approximately 154 free Android apps. They proposed a new method aimed to extract Android API elements that are used by apps that programmers modify between different releases. The results revealed that the more app churns are, the less are user ratings. The output also suggests that the more classes and methods are discussed in SO posts, the more the app developers modify them.

de Souza *et al.* (2014a) proposed an automatic methodology to categorize the data available on SO aimed to make cookbooks (recipe-oriented books) for the APIs used in mobile apps development. To assess the applicability of their approach, they produced cookbooks for three famous APIs, namely, SWT, STL, and LINQ. Besides, all the required features these cookbooks must possess were recognized and afterward were empirically assessed with the aim to know up to what level the generated cookbook fulfilled those features.

Wang, Malik and Godfrey (2015) proposed an approach aimed to suggest SO posts that are focused on API design-related problems. For setting up a comparison baseline, they introduced two further recommendation approaches, namely, reputation-based and random recommenders. The output revealed that the proposed technique outperformed the other two baseline methods by accomplishing an accuracy of 93 percent and proved to be much stable when applied to SO posts of Android and iOS.

Mastrangelo *et al.* (2015) examined approximately 86,479 Java archives to know how Java unsafe API abilities are utilized in practice in several libraries and up to what extent the third-party usage of unsafe API can influence the current Java code and different applications. They investigated the questions and discussions software developers do about unsafe API on SO and as a result detected certain usage patterns.

Sushine *et al.* (2015) investigated object-oriented APIs protocols by conducting two different types of qualitative studies. The output revealed that developers utilize more time mainly on four kinds of protocol state space. These outputs propose that the protocol targeted tools, languages, and verification methods will have more efficacies if they make software developers capable of doing a state search proficiently.

Summary and future work: this section primarily focused on investigating the approaches and issues of API usage from SO posts. Software documentation in general and API documentation in particular is created by few and used by many. Thus, often have a poor quality and lacks detailed instructions or examples. All of the included API studies thus far have identified this issue by analyzing SO posts. Most of the software developers face API problems when changes to API classes or methods are made, and they are unaware of it (i.e. due to the lack of proper documentation), thus, ultimately leads to an increase in SO posts discussion about particular API. Some of the research areas worth exploring in future are as follows:

- The future research should focus on investigating and suggesting best practices to the API providers for quality documentation.
- Need of developing automatic notification or recommendation systems to inform software developers about any changes made to a particular API (specifically when it modifies the behavior of methods/delete pubic methods) timely.
- Investigating the suitability of applying deep learning methods to analyze the API-related posts on SO for augmenting existing API documentation.
- There is a dire need of comparison of API-related posts on other CQA websites with SO to get explicit findings on the causes of API documentation demands and to analyze other platforms, e.g., Windows app store, Blackberry, API posts on SO.

4.2.4 Topics or issues on SO. This section summarizes research in the area of identifying topics or issues on SO for software development. It is also one of the significant areas closely linked to software development on SO, and thus the number of articles published is also significant due to its characteristics and pivotal role in software development. The research done in this area primarily investigated the following: categorizing development issues, developers' interactions, identifying topics/overlapping communities, ways of posing Q&A on SO and so on about tools/methods, as chronologically summarized in Table X. The majority of these research papers examined automatically or semi-automatically except a fewer manually a massive amount of SO data (minimum 120 posts, maximum 15.1 M) for identifying the topics or issues on SO.

Treude *et al.* (2011) investigated the role of Q&A site SO in software development. Analyzing SO data using qualitative coding depicted that Q&A websites are mainly proficient at code reviews, describing different conceptual issues and answering the questions of newcomers. Besides, mostly SO is used to seek for how-to questions, and the most famous programming languages questions are posted about C#, Java, PHP, and JavaScript.

Zolaktaf developed a statistical topical model called question answering topic model to model archives of Q&As. The proposed model captures the associations among the posted Q&A through modeling their topical dependencies. The results depicted that the model accomplishes better performance in capturing precise answer for a searched question compared to the results achieved by LDA baseline model. The proposed model is also capable of being utilized for tagging Q&As automatically, since it is helpful in giving the abilities of topical browsing for legacy Q&A collections (Zolaktaf *et al.*, 2012).

Similarly, Linares-Vásquez *et al.* (2013) explored SO to identify the issues related to mobile application development. The outcome of their research revealed that most of the questions posted on SO include topics related to general questions and compatibility issues of mobile development. Specifically, the less frequent topics were found about crash reports

Paper	Investigation	Technique/tool/algorithm	Data source	No. of posts
Treude *et al.* (2011)	Manual	Qualitative coding	SO	38,419
Zolaktaf *et al.* (2012)	Semi-automatic	LDA, TopN, Mean average precision	SO	4,128,352
Wang *et al.* (2013)	Automatic	LDA, Performed preprocessing steps	SO	63,863
Allamanis and Sutton (2013)	Automatic	LDA, Performed preprocessing steps	SO	9.6 M
Campbell *et al.* (2013)	Semi-automatic	LDA, Performed preprocessing steps	SO	15.1 M
Pinto and Kamei (2013)	Semi-automatic	Parsing	SO	1,439
Saxe *et al.* (2013)	Automatic	Own algorithm, Process monitor system call instrumentation tool	SO	6.47 M
Linares-Vásquez *et al.* (2013)	Automatic	LDA, Performed preprocessing steps	SO	400 K
Pinto *et al.* (2014)	Manual	Thematic analysis	SO	325
Barua *et al.* (2014)	Automatic	LDA, Performed preprocessing steps	SO	3,474,987
San Pedro and Karatzoglou (2014)	Automatic	LR, Linear SVM, GS, Stochastic EM, Popularity Ranking, TF-IDF, LSI	SE-Cross Validate	50,248
Bajaj *et al.* (2014)	Automatic	LDA, Performed preprocessing steps	SO	500 K
Beyer and Pinzger (2014)	Manual	Card Sorting, k-NN	SO	450
Campos and de Almeida Maia (2014)	Manual	NB, Multilayer perceptron, SVM, K-NN, J4.8 DT & RF	SO	120
Meng, Gandon, Zucker and Song (2014)	Automatic	Graph based, Clustering based, LDA	SO	1.1 M
Zou, Xu, Guo, Yan, Yang and Zhang (2015)	Automatic	LDA, Performed preprocessing steps	SO	921 K
Nagy and Cleve (2015)	Automatic	AST Based clone detection	SO	271,117
Chowdhury and Hindle (2015)	Automatic	MNB, SVM, Performed certain preprocessing steps	SO	200 K
Meng, Gandon and Zucker (2015)	Automatic	Graph based, Clustering based, LDA	SO	1.1 M
Meng, Gandon, Faron-Zucker and Song (2015)	Automatic	Graph based, Clustering based, LDA	SO	1.1 M
Pinto *et al.* (2015)	Manual	Thematic analysis	SO	250
Ma *et al.* (2015)	Semi-automatic	LDA, TEM, nDCG	SO	367,689
Ponzanelli *et al.* (2015a)	Semi-automatic	Island parsing	SO	708 K
Malik *et al.* (2015)	Manual	Thematic analysis, Three fold filtering approach, relational database	SO	5,009
Xu *et al.* (2016)	Automatic	Fudan NLP and IctclasNLP Algorithm, User study, etc.	SO	714,599
Chen and Xing (2016)	Semi-automatic	ARM, Community detection, Louvain method, Google trends	SO	7.8 M Posts 39948 Tags
Chen, Xing and Han (2016)	Semi-automatic	TagWiki, ARM, Community detection, Louvain method, google dtrens	SO	8,978,719 Posts 39,948 Tags
Rosen and Shihab, (2016)	Automatic	LDA, Performed preprocessing steps	SO	13,232,821

Table X. Chronological summary of topics or issues on stack overflow-related literature showing the paper, investigation, Technique/Tool/Algorithm, data source, and the number of posts/sentences used in the study

and database connections. Later on, Beyer and Pinzger (2014) manually investigated 450 Android-related posts on SO, aimed to get insights into the issues of Android app development faced by app developers. They presented the classification of SO posts automatically using Apache Lucene's KNN algorithm, which significantly outperformed the baseline. The study revealed that developers are primarily facing problems with proper usage of API components, i.e., ask about issues faced due to version changes, errors linked to network, database, and fragments.

The work of Wang *et al.* (2013) focused on investigating how developers interact with one another on SO. They analyze the distribution of developers who post Q&As. They also identified if there is a separation of SO community into questioners and answerers. For analyzing the vast content of millions of Q&A, they used the LDA topic modeling approach. They found the nature of developers whether most of them are primarily questioners or answerers.

Saxe *et al.* (2013) detected symbols in a repository of malicious executable files, e.g., registry keys, file names, and API calls names are also present in SO data. They revealed that through assessing function call symbol co-occurrence in SO discussions as well as the semantic tags related to these discussions, one can make functions relationship graphs over the symbols which depict aiding to detect malware software capabilities.

Pinto and Kamei (2013) classified questions about refactoring tools on SO. The outcome of the research leads to a broad categorization of defects and required features in refactoring tools. Besides, most often developers do not depend on refactoring tools. However, they wish to have new unimplemented features quite often in the refactoring tools they use. Similarly, Pinto *et al.* (2015) investigated popular questions posted about concurrent programming on SO. It became evident that despite the fact some questions are associated with real practical issues, e.g., "how to fix this concurrency bug," most of them are about some basic concepts, e.g., "what is a mutex?", posted by the well-experienced SO users. Surprisingly, they did not find any post about how to efficiently use concurrent programming techniques to increase the application performance?

Allamanis presented a thorough topic modeling analysis that unites question concepts, types, and code. They used LDA to link software development concepts and constructs/identifiers with specific categories of questions, e.g., "how to perform encoding" (Allamanis and Sutton, 2013). Campbell *et al.* (2013) investigated the existence of any topics which are not sufficiently covered or left by documentation of the project. They integrated questions from SO and considered documentation of PHP and Python. The results revealed that certain amount of topics in SO do not intersect with the documentation of the projects. Thus, it enabled them to identify topics having insufficient project documentation.

Campos and de Almeida Maia (2014) carried out a thorough evaluation of different classification algorithms to identify the best one which can classify questions efficiently from SO. The classification algorithms used are as follows: Naive Bayes (NB), Multilayer Perceptron, SVM, K-Nearest Neighbors (K-NN), J4.8 DT, and Random Forests. In the experimental study, the posts were split into three domain categories: How-to-do-it, Need-to-know, and Seeking-something. They found that NB classifier has an overall success rate of 84.16 and 92.5 percent on How-to-do-it questions category. Similarly, Meng, Gandon, Faron-Zucker and Song (2015), Meng, Gandon and Zucker (2015), Meng, Gandon, Zucker and Song (2014) suggested a proficient approach for collecting topics from Q&A to identify the communities of interest. They thoroughly compared and applied three detection methods: graph based, clustering based, and LDA based on SO. The outcome revealed that method based on topic modeling and user membership assignment not only to be efficient but also preserved their detection-level ability.

Pinto *et al.* (2014) investigated different perceptions of developers on the issues of software energy consumption on SO. The study revealed that questions on software energy consumption are deemed to be more exciting and challenging compared to other average SO questions. Besides, all questions posted that have more emphasis on source code modifications get more popularity and answers from users. Finally, they listed state-of-the-art recommendations on how to overcome these energy consumption issues. Similarly, Malik *et al.* (2015) empirically investigated the salient features of questions about energy, different issues faced by developers, and the most frequently discussed APIs on SO. The outcome revealed that developers are mostly interested in all those energy-related problems that primarily focus on improper implementations, sensor, and radio utilization. They also identified mostly used android APIs that frequently comes in SO.

In addition, Rosen and Shihab (2016) investigated what most of the mobile developers talk and ask about SO by using LDA. The outcome revealed that programmers are posting questions mainly related to app distribution, mobile APIs, data management, sensors, mobile tools, and UI development. They also successfully identified in details which issues in mobile app related development are hard, the issues specific to the platform, and examined different kinds of questions mobile app developers post on SO (e.g. what, how, or why).

Bajaj *et al.* (2014) analyzed web-related posts on SO to recognize the issues and misconceptions among developers. They thoroughly analyzed entire Q&A which is linked to web development to excerpt the hottest topics of threads using LDA. The output revealed that posts related to cross-browser are now less important, posts related to document object model APIs and event handling issues are deemed as the significant source of confusion for web development, rise in popularity of HTML5 specifically in (mobile) web applications, web-related topics are gaining fame specifically in mobile development, and posts on SO show that expert software developers have troubles in understanding new features added to HTML5 and JavaScript.

Barua *et al.* (2014) suggested a technique to thoroughly investigate and assess rich textual discussions made on SO by using LDA. The output revealed useful insights; most frequently discussed topics among programmers' ranges widely from jobs to version control systems to C# syntax. The questions posted in some topics become the source of discussions in other topics, and topics in the area of web development, mobile applications, Git, and MySQL trend are increasing with the passage of time and need further research.

San Pedro developed a method based on a supervised Bayesian method to recommend question and to model the expertise in CQA. Thus, ultimately cut down the waiting period for users to wait for the responses and evading question shortage. They proposed a new algorithm known as rank Supervised Latent Dirichlet allocation (RankSLDA), which is an extension of supervised LDA model by taking into account a learning-to-rank model. They made a comparison of RankSLDA with various other methods having data from the SE-Cross Validate. The proposed algorithm RankSLDA considerably performed well compared to other existing methods (San Pedro and Karatzoglou, 2014).

Ponzanelli *et al.* (2015a) developed a full island grammar which can model posts discussed about any topic on SO through making heterogeneous abstract syntax tree (H-AST). They focused on posts talking about Java which are approximately 708 K discussion posts on SO. The resultant data set then models every SO thread of discussion, creating a complete H-AST for every single kind of structured fragment comprised of JSON, XML, Java, Stack traces, and supplementing this information by using a set of simple meta-information. The proposed data set enables the users to conduct an integrated evaluation of SO via accessing the H-AST of any thread of discussions on SO.

Nagy developed a mining method for SO, aimed to find the error-prone patterns in SQL queries. Thus, finding such patterns can aid programmers to neglect the use of error-prone constructs or whenever they make use of such kind of constructs the SO posts will aid them to apply the language adequately. The primary goal of this work is thus to suggest the preliminary steps toward making a recommendation system that can help software programmers in creating accurate SQL queries (Nagy and Cleve, 2015).

Chowdhury and Hindle (2015) assessed the working performance of two machine learning algorithms, namely, multinomial naïve Bayes (MNB) and SVM, to identify off-topic posts in programming-related internet relay chat (IRC) channels. Thus, both of the proposed classifiers could aid to detect on-topic posts for an IRC user. The posts on SO and video comments on YouTube for classifying IRC discussions can be easily shared to make topic-specific classifiers without the need of any manual annotation.

Zou, Xu, Guo, Yan, Yang and Zhang (2015) used LDA to identify the topics of SO discussions and wordlists to detect associations between the discussions and non-functional requirements (NFRs). The aim remained on detecting most wanted and unsolved NFRs, the growth and trends of NFRs. Consequently, it was revealed that usability and reliability are most frequently discussed topics, while a little focus is placed on maintainability and efficiency issues.

Ma *et al.* (2015) developed a method called tri-role topic model (TRTM), aimed to model three specific roles of users: askers, answerers, and voters. These three roles are responsible for authoring a question, choosing a question to answer, participating and voting the given answers. The output revealed that TRTM is capable of assisting users to get suitable rankings of answers, specifically for the newly posted and less famous questions. The model was assessed on normalized discounted cumulative gain (nDCG), which revealed that TRTM performed well compared to existing methods TEM and LDA.

Xu *et al.* (2016) proposed an automated cross-language relevant question retrieval (CLRQR) system to extract relevant English questions on SO for non-native English speakers. The CLRQR system collects necessary information from the title and the detail description of the Chinese input question. Afterwards, carryout domain-specific translation of necessary Chinese information into the English language, and articulates a query with highest scored English words for extracting appropriate questions from SO. They arbitrarily selected 80 Java Qs from Chinese Q&A websites as the input Chinese Qs. Each of the four proposed approaches retrieves the top ten closest match questions for a given Chinese question. The outcome of the performed experiments proved that CLRQR system outperforms the four baseline approaches and the improvements are statistically significant.

Chen and Xing (2016) investigated the technology landscape in SO through applying certain association rule mining and community detection techniques on the tags to mine the technology landscape discussion threads. The output of the assessment revealed that mined technology landscape is capable of providing a summarized view of different practiced technologies, the multi-faceted associations between these practiced technologies and the evolving trend of these technologies. Later on, Chen, Xing and Han (2016) extended their prior work of Chen and Xing (2016) by developing a system called TechLand to help technology landscape inquiries with topics, social and trending knowledge of technologies summarized from millions of SO questions. They practically implemented TechLand system and assessed its effectiveness in contrast to developers forum answers to hundred technology landscape questions on SO. The output of the assessment revealed that TechLand system is capable of helping developers in technology landscape explorations by giving them straight, objective, and accumulated information about the existing practiced technologies, the associations between the technologies and the evolving trends in technologies.

Summary and future work: this section briefly presented the different works focused on identifying software development issues specifically mobile application development, ways of posting questions on different issues, investigating the nature of developers interactions, investigating different topics specifically the software feature requests or bugs and so on discussed in SO posts. However, the majority of them employed basic topic modeling techniques which is a shortcoming of their research. Some of the related future works are as follows:

- Most of the researchers employed basic topic modeling for identifying topics or issues in SO posts. Thus, in future, effort should be spent on applying more advanced methods, i.e., deep learning methods, for identifying topics or issues in SO posts.

- There is a need for comprehensive comparative studies between SO and other Q&A sites with the aim to investigate and gain a deeper understanding of the issues or topics discussed and to know similarities or dissimilarities among them with the evolution of time.
- There is scarce research on cross-language questions retrieval and translation methods especially on software developers Q&A websites, thus in future effort should be spent on developing such methods.
- Investigating on how to improve the existing methods for efficiently identifying and extracting requirements (functional and non-functional) present in SO posts and devising appropriate strategies for prioritizing the extracted software requirements.

5. Validity threats

The validity threats (Wright *et al.*, 2010) of our literature survey are discussed below.

5.1 Internal validity

There are some internal validity threats related to biases in searching, selecting, data extraction process, and categorization of the studies included. We mitigated the threat of missing papers by conducting general searches in Google scholar. However, the threat of missing relevant papers is a threat to the validity of any literature survey paper, including ours. We followed the agreed criteria for selecting, extracting data and categorizing papers, but still there are chances of errors. To reduce this risk to a minimum, we performed inter-rater reliability test, which helped to minimize the researcher's study selection bias.

5.2 Construct validity

The research studies included in our survey may not cover the whole aspects of SO community. However, to reduce this threat, we explicitly limited the scope of our survey to the research studies only on software development in SO. Besides, the data extraction from the included research studies was done with the author's agreement by focusing on several facets, i.e., aim/purpose, investigation, and tools/techniques, used in the research paper.

5.3 External validity

We performed a literature survey on studies published during the year 2008 till June 2016 on software development in SO. This fact indicates that we may have missed some relevant studies of equal importance. Thus, we cannot generalize our conclusions for whole SO community. However, the outcomes of our literature survey allow us to draw insights to guide further investigations.

6. Conclusion and future work

6.1 Conclusion

The rapid increase in popularity, usage, and content of CQA platform SO demands for a systematic investigation and analysis via mining for software development. We presented how the user-generated content shared on SO is useful and supportive in all stages of software development for academics/practitioners. For instance, investigating the type of knowledge present in SO posts useful for software development, characteristics of writing quality posts, type of software development discussions made on SO, which stages of software development lifecycle are supported by in large, and which methods/tools were used to mine the user-generated content on SO and so on. Our paper gathered, overviewed, summarized, and systematically categorized 166 core papers into two main categories, namely, "SO design and usage" and "SO content applications" having further eight topics/themes done on mining SO fully or partially for software development. We found approximately 70 percent of surveyed papers used millions of SO posts as can be seen in the statistics of Figure 5. We believe that extracting and constructing a data set of SO posts in millions scale is necessary for solid and valuable research, however, with some caution needed. We found that most of the studies just focused on a limited number of software development tasks and the majority of the research done is quantitative. Besides, it is also observed that more automatic methods/techniques are needed to enable researchers to investigate the whole SO content for software development efficiently and effectively. Furthermore, Q&A community providers can also get insights for better designing such forums, since our survey highlighted the important design characteristic and usage features which have an impact on user contribution to the Q&A community. Our survey also recommends practitioners and researchers to utilize such forums for the identified under-utilized tasks of software development. Moreover, our study serves as a starting point for future researchers/practitioners to investigate the whole SO for software development and can thus help them to start further research work on the gaps identified.

6.2 Future work

We propose the following avenues for future research in explicitly mining SO and in general mining other software repositories to improve software development as given below.

There is a need of thorough systematic comparison studies between several software repositories (i.e. mailing lists, Q&A forums, users reviews/comments in mobile application stores, bug reports, software documentation, and comments/reviews) in terms of nature of the data (structured and unstructured) having text or multi-media content and semantics of the text. Investigating the suitability of tools/techniques used to mine/analyze the software repositories under investigation and to determine/learn the similarities/differences of trends between the cross mining software repositories research studies.

Besides, our survey just focused on overviewing, summarizing, and systematically classifying the research studies done on SO about software development; however, we neglected to compare the approaches/methods/tools used in those studies to extract information from SO for software development. Therefore, in future, the focus should be put on trade-off studies with specific emphasis on investigating, analyzing, and comparing the methods/tools used in those studies to mine SO for software development. Additionally, we also encourage future researchers to investigate each of the identified categories separately in SO thoroughly for software development and perform comprehensive comparative studies with other similar Q&A sites by evaluating the methods/tools used. The emphasis of our paper is on how academics/practitioners can get benefit from the valuable user-generated content shared on various online social networks, specifically from Q&A community SO for SDLC which is listed as the "vertical" axis. Thus, we encourage researchers to focus on the "horizontal" axis in future as our current comprehensive SO survey will serve as a stepping stone to enter in this area by providing them a brief overview of what has been done so far.

The work done on question-oriented text retrieval (Zou, Ye, Lu, Mylopoulos and Zhang, 2015; Ye *et al.*, 2014) and quality of code (Tavakoli *et al.*, 2016) in SO is almost negligible. Thus, demands more investigation regarding enhancing the existing classification methods for retrieving text from Q&A forums like SO and developing efficient methods to improve the quality of retrieved code snippets.

There is a need to develop efficient and reliable multi-lingual search, retrieval, and translating systems that could enable software developers to get an advantage and utilize diverse available knowledge on CQA sites like SO in their local languages.

Mining research has been abundant on SO quantitatively which is a good thing, however, to know the exact reasoning like "why," "how," there is a dire need to have qualitative studies sidewise supported by the available mining software repositories techniques.

There are plenty of research opportunities in applying state-of-the-art deep learning methods in mining/analyzing several software repositories (SO, mailing lists, and software documentation), since most of the research studies included here have just applied basic machine learning methods.

The focus of majority of surveyed papers and in general mining software repositories, i.e., CQA forums SO, mailing lists, and comments/reviews, have remained mainly on mining the textual content in these repositories with fewer to have mined the multi-media content like video tutorials (Ponzanelli, Bavota, Mocci, di Penta, Oliveto, Russo, Haiduc and Lanza, 2016; Ponzanelli, Bavota, Mocci, di Penta, Oliveto, Hasan, Russo, Haiduc and Lanza, 2016) and pictures. Thus, in future, the research should focus more on mining software repositories having multi-media content and developing effective techniques/methods if the support of existing techniques/methods is found limited.

There are fewer studies which have explicitly addressed the area of requirements engineering in SO fully. SO has a diverse pool of professional crowd, i.e., software developers and users. There are several research opportunities because the knowledge and opinions shared/discussed by them have software requirements/features requests/bugs about software tools/platforms. Thus, specific methods are needed to mine, i.e., identify or elicit those software requirements/features request from posts of SO, specify/model those software requirements and finally validate them. The research on identifying/eliciting software requirements from SO posts/Q&A is still at its infancy stage; so many opportunities are yet to be explored.

There is a need to investigate the possibility of utilizing existing tools/techniques used in mining, i.e., identifying or eliciting software features/requirements from reviews of mobile application stores for mining, i.e., identifying or eliciting software features from CQA forums like SO. However, some adherent perils should be taken care of like mobile application store reviews are syntactically and semantically different than the posts/Q&A on CQA forums like SO, the vocabulary used is different, length of the text is different, and so on. In a nutshell, a cross-cutting and inter-domain analysis study is needed to explore further the possibility of utilizing existing techniques/methods and proposing new ones for mining, i.e., identifying or eliciting software requirements from this two domain (SO posts and user reviews in mobile application stores) and other software repositories.

Notes

1. https://git-scm.com/
2. http://subversion.apache.org/
3. www.nongnu.org/cvs/
4. www.youtube.com/
5. www.slideshare.net/
6. www.quora.com
7. www.stackexchange.com
8. www.reddit.com
9. www.github.com
10. https://github.com/about

11. http://stackexchange.com/about
12. www.youtube.com/yt/press/statistics.htm
13. http://stackoverflow.com/company/about
14. http://stackoverflow.com/questions/23610425
15. http://stackoverflow.com/questions/17061187
16. http://stackoverflow.com/questions/40680259

References

Ahasanuzzaman, M., Asaduzzaman, M., Roy, C.K. and Schneider, K.A. (2016), "Mining duplicate questions in stack overflow", *Proceedings of the 13th International Workshop on Mining Software Repositories, ACM*, pp. 402-412.

Ahmad, A. and Khan, H. (2008), "The importance of knowledge management practices in overcoming the global software engineering challenges in requirements understanding", master thesis research, Blekinge Institute of Technology.

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

Alnuem, M.A., Ahmad, A. and Khan, H. (2012), "Requirements understanding: a challenge in global software development, industrial surveys in Kingdom of Saudi Arabia", *IEEE 36th Annual Computer Software and Applications Conference (COMPSAC)*, *Izmir*, pp. 297-306.

Anand, D. and Vahab, F.A. (2015), "Predicting post importance in question answer forums based on topic-wise user expertise", *International Conference on Distributed Computing and Internet Technology*, Springer, pp. 365-376.

Anderson, A., Huttenlocher, D., Kleinberg, J. and Leskovec, J. (2012), "Discovering value from community activity on focused question answering sites: a case study of stack overflow", *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 850-858.

Anderson, A., Huttenlocher, D., Kleinberg, J. and Leskovec, J. (2013), "Steering user behavior with badges", *Proceedings of the 22nd International Conference on World Wide Web, ACM*, pp. 95-106.

Arora, P., Ganguly, D. and Jones, G.J. (2015), "The good, the bad and their kins: identifying questions with negative scores in StackOverflow", *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 1232-1239.

Arwan, A., Rochimah, S. and Akbar, R.J. (2015), "Source code retrieval on StackOverflow using LDA", *3rd International Conference on Information and Communication Technology (ICoICT), IEEE*, pp. 295-299.

Asaduzzaman, M., Mashiyat, A.S., Roy, C.K. and Schneider, K.A. (2013), "Answering questions about unanswered questions of stack overflow", *Proceedings of the 10th Working Conference on Mining Software Repositories*, IEEE Press, pp. 97-100.

Bacchelli, A., Ponzanelli, L. and Lanza, M. (2012), "Harnessing stack overflow for the IDE", *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, IEEE Press, pp. 26-30.

Bajaj, K., Pattabiraman, K. and Mesbah, A. (2014), "Mining questions asked by web developers", *Proceedings of the 11th Working Conference on Mining Software Repositories, ACM*, pp. 112-121.

Baltadzhieva, A. and Chrupała, G. (2015a), "Predicting the quality of questions on Stackoverflow", *Recent Advances in Natural Language Processing (RANLP)*, p. 32.

Baltadzhieva, A. and Chrupała, G. (2015b), "Question quality in community question answering forums: a survey", *ACM SIGKDD Explorations Newsletter*, Vol. 17, pp. 8-13.

Barua, A., Thomas, S.W. and Hassan, A.E. (2014), "What are developers talking about? An analysis of topics and trends in stack overflow", *Empirical Software Engineering*, Vol. 19 No. 3, pp. 619-654.

Bavota, G. (2016), "Mining unstructured data in software repositories: current and future trends", *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 1-12.

Bazelli, B., Hindle, A. and Stroulia, E. (2013), "On the personality traits of stackoverflow users", *ICSM*, pp. 460-463.

Beyer, S. and Pinzger, M. (2014), "A manual categorization of android app development issues on stack overflow", *ICSME*, pp. 531-535.

Beyer, S. and Pinzger, M. (2015), "Synonym suggestion for tags on stack overflow", *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension*, *IEEE Press*, pp. 94-103.

Beyer, S. and Pinzger, M. (2016), "Grouping android tag synonyms on stack overflow", *Proceedings of the 13th International Workshop on Mining Software Repositories, ACM*, pp. 430-440.

Bhat, V., Gokhale, A., Jadhav, R., Pudipeddi, J. and Akoglu, L. (2014), "Min (e) d your tags: analysis of question response time in stackoverflow", *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 328-335.

Bhat, V., Gokhale, A., Jadhav, R., pudipeddi, J. and Akoglu, L. (2015), "Effects of tag usage on question response time analysis and prediction in StackOverflow", *Social Network Analysis and Mining*, Vol. 5 No. 24, pp. 1-13.

Blumberg, R. and Atre, S. (2003), "The problem with unstructured data", 13.

Borjigen, C. (2015), "Mass collaborative knowledge management towards the next generation of knowledge management studies", *Program: Electronic Library and Information Systems*, Vol. 49 No. 3, pp. 325-342.

Bosu, A., Corley, C.S., Heaton, D., Chatterji, D., Carver, J.C. and kraft, N.A. (2013), "Building reputation in stackoverflow: an empirical investigation", *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 89-92.

Boudaer, G. and Loeckx, J. (2016), "Enriching topic modelling with users' histories for improving tag prediction in Q&A systems", *Proceedings of the 25th International Conference Companion on World Wide Web, International World Wide Web Conferences Steering Committee*, pp. 669-672.

Brereton, P., Kitchenham, B.A., Budgen, D., Turner, M. and Khalil, M. (2007), "Lessons from applying the systematic literature review process within the software engineering domain", *The Journal of Systems and Software*, Vol. 80 No. 4, pp. 571-583.

Caalefato, F., Lanubile, F., Raffaella, M. and Merolla, N.N. (2015), "Success factors for effective knowledge sharing in community-based question-answering", International Forum on Knowledge Asset Dynamics (IFKAD 2015).

Calefato, F., Lanubile, F., Marasciulo, M.C. and Novielli, N. (2015), "Mining successful answers in stack overflow", *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 430-433.

Campbell, J.C., Zhang, C., Xu, Z., Hindle, A. and Miller, J. (2013), "Deficient documentation detection: a methodology to locate deficient project documentation using topic analysis", *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 57-60.

Campos, E.C. and de Almeida Maia, M. (2014), "Automatic categorization of questions from Q&A sites", *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pp. 641-643.

Carreno, L.V.G. and Winbladh, K. (2013), "Analysis of user comments: an approach for software requirements evolution", *IEEE International Conference on Software Engineering (ICSE)*.

Cavusoglu, H., Li, Z. and Huang, K.-W. (2015), "Can gamification motivate voluntary contributions? The case of StackOverflow Q&A community", *Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing*, pp. 171-174.

cG Galina, E.L. and Kuznetsov, A.M. (2013), "Predict closed questions on StackOverflow", *Proceedings of the Ninth Spring Researcher's Colloquium on Database and Information Systems*.

Chang, S. and Pal, A. (2013), "Routing questions for collaborative answering in community question answering", *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 494-501.

Chen, C. and Xing, Z. (2016), "Mining technology landscape from stack overflow", *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Vol. 14.

Chen, C. and Xingg, Z. (2016), "Towards correlating search on Google and asking on stack overflow", *IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, *Atlanta, GA*.

Chen, C., Gao, S. and Xing, Z. (2016), "Mining analogical libraries in Q&A discussions – incorporating relational and categorical knowledge into word embedding", *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 338-348.

Chen, C., Xing, Z. and Han, L. (2016), "Techland: assisting technology landscape inquiries with insights from stack overflow", *32nd ICSME, IEEE*.

Chen, N., Liny, J., Hoi, S.C.H., Xiao, X. and Zhang, B. (2014), "AR-miner: mining informative reviews for developers from mobile app marketplace", *IEEE International Conference on Software Engineering (ICSE),*, *Hyderabad*, *May 31-June 7*, pp. 767-778.

Chen, T.-H., Thomas, S.W. and Hassan, A.E. (2015), "A survey on the use of topic models when mining software repositories", *Empirical Software Engineering*, Vol. 21 No. 5, pp. 1843-1919.

Chowdhury, S.A. and Hindle, A. (2015), "Mining StackOverflow to filter out off-topic IRC discussion", *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 422-425.

Correa, D. and Sureka, A. (2013a), "Fit or unfit: analysis and prediction of 'closed questions' on stack overflow", *Proceedings of the First ACM Conference on Online Social Networks*, pp. 201-212.

Correa, D. and Sureka, A. (2013b), "Integrating issue tracking systems with community-based question and answering websites", *22nd Australian Software Engineering Conference, IEEE*, pp. 88-96.

Correa, D. and Sureka, A. (2014), "Chaff from the wheat: characterization and modeling of deleted questions on stack overflow", *Proceedings of the 23rd International Conference on World Wide Web, ACM*, pp. 631-642.

de F Farias, M.A., Novais, R., Júnior, M.C., da Silva Carvalho, L.P., Mendonça, M. and Spínola, R.O. (2016), "A systematic mapping study on mining software repositories", *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 1472-1479.

de Souza, C.R.B., Filho, F.F., Miranda, M., Ferreira, R.P., Treude, C. and Singer, L. (2016), "The social side of software platform ecosystems", *Proceedings of the CHI Conference on Human Factors in Computing Systems, ACM*, *May 7-12,*, pp. 3204-3214.

de Souza, L.B., Campos, E.C. and Maia, M.D.A. (2014a), "On the extraction of cookbooks for APIs from the Crowd Knowledge", *Brazilian Symposium on Software Engineering (SBES), IEEE*, pp. 21-30.

de Souza, L.B., Campos, E.C. and Maia, M.D.A. (2014b), "Ranking crowd knowledge to assist software development", *ACM Proceedings of the 22nd International Conference on Program Comprehension*, pp. 72-82.

Diamantopoulos, T. and Symeonidis, A.L. (2015), "Employing source code information to improve question-answering in stack overflow", *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 454-457.

Duijn, M., Kučera, A. and Bacchelli, A. (2015), "Quality questions need quality code: classifying code fragments on stack overflow", *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 410-413.

Ercan, S., Stokkink, Q. and Bacchelli, A. (2015), "Automatic assessments of code explanations: predicting answering times on stack overflow", *IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 442-445.

Eye, A.V. and Mun, E.Y. (2006), *Analyzing Rater Agreement Manifest Variable Methods*, Taylor & Francis Group.

Ganguly, D. and Jones, G.J. (2015), “Partially labeled supervised topic models for RetrievingSimilar questions in CQA forums”, *Proceedings of the 2015 International Conference on The Theory of Information Retrieval, ACM*, pp. 161-170.

Gantayat, N., Dhoolia, P., Padhye, R., Mani, S. and Sinha, V.S. (2015), “The synergy between voting and acceptance of answers on stackoverflow, or the lack thereof”, *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 406-409.

Gao, Q., Zhang, H., Wang, J., Xiong, Y., Zhang, L. and Mei, H. (2015), “Fixing recurring crash bugs via analyzing Q&A sites”, *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 307-318.

Ginsca, A.L. and Popescu, A. (2013), “User profiling for answer quality assessment in Q&A communities”, *Proceedings of the Workshop on Data-Driven user Behavioral Modelling and Mining from Social Media, ACM*, pp. 25-28.

Gkotsis, G., Stepanyan, K., Pedrinaci, C., Domingue, J. and Liakata, M. (2014), “It’s all in the content: state of the art best answer prediction based on discretisation of shallow linguistic features”, *Proceedings of the ACM Conference on Web Science*, pp. 202-210.

Godfrey, M.W., Hassan, A.E., Herbsleb, J.D., Murphy, G.C., Robillard, M., Devanbu, P., Mockus, A., Perry, D.E. and Notkin, D. (2008), “Future of mining software archives: a roundtable”, *IEEE Software*, Vol. 26 No. 1, pp. 67-70.

Gómez, C., Cleary, B. and Singer, L. (2013), “A study of innovation diffusion through link sharing on stack overflow”, *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 81-84.

Grant, S. and Betts, B. (2013), “Encouraging user behaviour with achievements: an empirical study”, *10th IEEE Working Conference on Mining Software Repositories (MSR)*, pp. 65-68.

Guerrouj, L., Azad, S. and Rigby, P.C. (2015), “The influence of app churn on app success and StackOverflow discussions”, *IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 321-330.

Guerrouj, L., Bourque, D. and Rigby, P.C. (2015), “Leveraging informal documentation to summarize classes and methods in context”, *IEEE/ACM 37th IEEE International Conference on Software Engineering,* pp. 639-642.

Guimaraes, A., Da Silva, A.P.C. and Almeida, J.M. (2015), “On the dynamics of topic-based communities in online knowledge-sharing networks”, *Second European Network Intelligence Conference (ENIC), IEEE*, pp. 1-8.

Gupta, R. and Reddy, P.K. (2016), “Learning from gurus: analysis and modeling of reopened questions on stack overflow”, *Proceedings of the 3rd IKDD Conference on Data Science, ACM*, Vol. 13.

Gyongyi, Z., Koutrika, G., Pedersen, J. and Garcia-Molina, H. (2007), “Questioning Yahoo! Answers”.

Halavais, A., Kwon, K.H., Havener, S. and Striker, J. (2014), “Badges of friendship: social influence and badge acquisition on Stack Overflow”, *47th Hawaii International Conference on System Sciences*, pp. 1607-1615.

Hannah, S. (2005), “Sorting out card sorting: comparing methods for information architects, usability specialists, and other practitioners”, MS Thesis, University of Oregon, Portland, OR.

Hanrahan, B.V., Convertino, G. and Nelson, L. (2012), “Modeling problem difficulty and expertise in stackoverflow”, *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion*, pp. 91-94.

Hart, K. and Sarma, A. (2014), “Perceptions of answer quality in an online technical question and answer forum”, *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, pp. 103-106.

Hassan, A.E. (2008), “The road ahead for mining software repositories”, *Frontiers of Software Maintenance.*

Iacob, C. and Harrison, R. (2013), “Retrieving and analyzing mobile apps feature requests from online reviews”, *IEEE Mining Software Repositories (MSR)*, *San Francisco, CA*, pp. 41-44.

Jafari, M., Hesamamiri, R., Sadjadi, J. and Bourouni, A. (2012), "Assessing the dynamic behavior of online Q&A knowledge markets a system dynamics approach", *Program: Electronic Library and Information Systems*, Vol. 46 No. 3, pp. 341-360.

Jiarpakdee, J., Ihara, A. and Matsumoto, K.-I. (2016), "Understanding question quality through affective aspect in Q&A site", *Proceedings of the 1st International Workshop on Emotion Awareness in Software Engineering, ACM*, pp. 12-17.

Jin, Y., Yang, X., Kula, R.G., Choi, E., Inoue, K. and Iida, H. (2015), "Quick trigger on stack overflow: a study of gamification-influenced member tendencies", *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 434-437.

Joorabchi, A., English, M. and Mahdi, A.E. (2015), "Automatic mapping of user tags to Wikipedia concepts: the case of a Q&A website-stackoverflow", *Journal of Information Science*, 0165551515586669.

Joorabchi, A., English, M. and Mahdi, A.E. (2016), "Text mining stackoverflow: an insight into challenges and subject-related difficulties faced by computer science learners", *Journal of Enterprise Information Management*, Vol. 29 No. 2, pp. 255-275.

Kavaler, D., Posnett, D., Gibler, C., Chen, H., Devanbu, P. and Filkov, V. (2013), "Using and asking: APIs used in the Android market and asked about in stackoverflow", *International Conference on Social Informatics*, *Springer*, pp. 405-418.

Khan, H., Ahmad, A. and Alnuem, M.A. (2012), "Knowledge management: a solution to requirements understanding in global software engineering", *Research Journal of Applied Sciences, Engineering and Technology*.

Khan, H., Ahmad, A., Johansson, C. and Alnuem, M.A. (2011), "Requirements understanding in global software engineering industrial surveys", *International Conference on Computer and Software Modeling (IPCSIT)*, *IACSIT Press*, *Singapore*, pp. 167-173.

Khusro, S., Alam, A. and Khalid, S. (2017), "Social question and answer sites: the story so far", *Program: Electronic Library and Information Systems*, Vol. 51 No. 2, pp. 170-192.

Kitchenham, B. (2004), "Procedures for performing systematic reviews", No. 33, Keele University, Keele, pp. 1-26.

Kitchenham, B.A. and Charters, S. (2007), "Guidelines for performing systematic literature reviews in software engineering", Software Engineering Group, School of Computer Science and Mathematics, Keele University, Keele.

Kumar, V. and Pedanekar, N. (2016), "Mining shapes of expertise in online social Q&A communities", *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, pp. 317-320.

Lal, S., Correa, D. and Sureka, A. (2014), "MiQs: characterization and prediction of migrated questions on StackExchange", Indraprastha Institute of Information Technology.

Latorre, N., Minelli, R., Mocci, A., Ponzanelli, L. and Lanza, M. (2015), "SODA: the stack overflow data set almanac", *2015 IEEE 5th Workshop on Mining Unstructured Data (MUD)*, pp. 1-5.

Li, G., Zhu, H., Lu, T., Ding, X. and Gu, N. (2015), "Is it good to be like Wikipedia? Exploring the trade-offs of introducing collaborative editing model to Q&A sites", *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, pp. 1080-1091.

Li, Z., Huang, K.-W. and Cavusoglu, H. (2012), "Quantifying the impact of badges on user engagement in online Q&A communities".

Lin, B. and Serebrenik, A. (2016), "Recognizing gender of stack overflow users", *Proceedings of the 13th International Workshop on Mining Software Repositories*, pp. 425-429.

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

Linares-Vásquez, M., Bavota, G., DI Penta, M., Oliveto, R. and Poshyvanyk, D. (2014), “How do API changes trigger stack overflow discussions? A study on the Android SDK”, *Proceedings of the 22nd International Conference on Program Comprehension, ACM*, pp. 83-94.

Lotufo, R., Passos, L. and Czarnecki, K. (2012), “Towards improving bug tracking systems with game mechanisms”, *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 2-11.

MacLeod, L., Storey, M.-A. and Bergen, A. (2015), “Code, camera, action: how software developers document and share program knowledge using YouTube”, *IEEE 23rd International Conference on Program Comprehension (ICPC)*, *Florence*, *May 16-24*, pp. 104-114.

Ma, Z., Sun, A., Yuan, Q. and Cong, G. (2015), “A tri-role topic model for domain-specific question answering”, *AAAI*, pp. 224-230.

Malik, H., Zhao, P. and Godfrey, M. (2015), “Going green: An exploratory analysis of energy-related questions”, *IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 418-421.

Mamykina, L., Manoim, B., Mittal, M., Hripcsak, G. and hartmann, B. (2011), “Design lessons from the fastest Q&A site in the west”, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2857-2866.

Marder, A. (2015), “Stack overflow badges and user behavior: an econometric approach”, *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 450-453.

Mastrangelo, L., Ponzanelli, L., Mocci, A., Lanza, M., Hauswirth, M. and Nystrom, N. (2015), “Use at your own risk: the Java unsafe API in the wild”, *ACM Sigplan Notices*, Vol. 50 No. 10, pp. 695-710.

Meng, Z., Gandon, F. and Faron-Zucker, C. (2014), “QASM: a Q&A social media system based on social semantic”, *Proceedings of the International Conference on Posters & Demonstrations Track-Volume 1272, CEUR-WS, org*, pp. 333-336.

Meng, Z., Gandon, F. and Zucker, C.F. (2015), “Simplified detection and labeling of overlapping communities of interest in question-and-answer sites”, *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pp. 107-114.

Meng, Z., Gandon, F., Faron-Zucker, C. and Song, G. (2015), “Detecting topics and overlapping communities in question and answer sites”, *Social Network Analysis and Mining*, Vol. 5 No. 27, pp. 1-17.

Meng, Z., Gandon, F., Zucker, C.F. and Song, G. (2014), “Empirical study on overlapping community detection in question and answer sites”, *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 344-348.

Mo, W., Shen, B., Chen, Y. and Zhu, J. (2015), “Tbil: a tagging-based approach to identity linkage across software communities”, *Asia-Pacific Software Engineering Conference (APSEC), IEEE*, pp. 56-63.

Morrison, P. and Murphy-Hill, E. (2013), “Is programming knowledge related to age? An exploration of stack overflow”, *10th IEEE Working Conference on Mining Software Repositories (MSR)*, pp. 69-72.

Movshovitz-Attias, D. and Cohen, W.W. (2013), “Natural language models for predicting programming comments”, *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Sofia*, pp. 35-40.

Movshovitz-Attias, D., Movshovitz-Attias, Y., Steenkiste, P. and Faloutsos, C. (2013), “Analysis of the reputation system and user contributions on a question answering website: stackoverflow”, *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 886-893.

Murgia, A., Janssens, D., Demeyer, S. and Vasilescu, B. (2016), “Among the machines: human-bot interaction on social Q&A websites”, *Proceedings of the CHI Conference Extended Abstracts on Human Factors in Computing Systems, ACM*, pp. 1272-1279.

Nagy, C. and Cleve, A. (2015), “Mining stack overflow for discovering error patterns in SQL queries”, *IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 516-520.

Nasehi, S.M., Sillito, J., Maurer, F. and Burns, C. (2012), "What makes a good code example? A study of programming Q&A in StackOverflow", *28th IEEE International Conference on Software Maintenance (ICSM), IEEE*, pp. 25-34.

Niazi, M., Mahmood, S., Alshayeb, M., Riaz, M.R., Faisal, K., Cerpa, N., Khan, S.U. and Richardson, I. (2016), "Challenges of project management in global software development: a client-vendor analysis", *Information and Software Technology*, Vol. 80, December, pp. 1-19.

Novielli, N., Calefato, F. and Lanubile, F. (2014), "Towards discovering the role of emotions in stack overflow", *Proceedings of the 6th International Workshop on Social Software Engineering, ACM*, pp. 33-36.

Novielli, N., Calefato, F. and Lanubile, F. (2015), "The challenges of sentiment detection in the social programmer ecosystem", *Proceedings of the 7th International Workshop on Social Software Engineering, ACM*, pp. 33-40.

Pagano, D. and Maalej, W. (2011), "How do developers blog? An exploratory study", *8th IEEE Working Conference on Mining Software Repositories (MSR), ACM, Honolulu, HI, May 21-22*, pp. 123-132.

Pal, A. and Konstan, J.A. (2010), "Expert identification in community question answering: exploring question selection bias", *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ACM*, pp. 1505-1508.

Pal, A., Chang, S. and Konstan, J.A. (2012), "Evolution of experts in question answering communities", ICWSM.

Pal, A., Harper, F.M. and Konstan, J.A. (2012), "Exploring question selection bias to identify experts and potential experts in community question answering", *ACM Transactions on Information Systems (TOIS)*, Vol. 30 No. 2, p. 10.

Parnin, C., Treude, C. and Storey, M.-A. (2013), "Blogging developer knowledge: motivations, challenges, and future directions", *21st International Conference on Program Comprehension (ICPC). IEEE, San Francisco, CA*, pp. 211-214.

Parnin, C., Treude, C., Grammel, L. and Storey, M.-A. (2012), "Crowd documentation: exploring the coverage and the dynamics of API discussions on stack overflow", technical report, Georgia Institute of Technology.

Pinto, G., Castor, F. and Liu, Y.D. (2014), "Mining questions about software energy consumption", *Proceedings of the 11th Working Conference on Mining Software Repositories, ACM*, pp. 22-31.

Pinto, G., Torres, W. and Castor, F. (2015), "A study on the most popular questions about concurrent programming", *Proceedings of the 6th Workshop on Evaluation and Usability of Programming Languages and Tools, ACM*, pp. 39-46.

Pinto, G.H. and Kamei, F. (2013), "What programmers say about refactoring tools? An empirical investigation of Stack Overflow", *Proceedings of the 2013 ACM Workshop on Workshop on Refactoring Tools, ACM*, pp. 33-36.

Ponzanelli, L., Bacchelli, A. and Lanza, M. (2013a), "Leveraging crowd knowledge for software comprehension and development ", *17th European Conference on Software Maintenance and Reengineering (CSMR), IEEE*, pp. 57-66.

Ponzanelli, L., Bacchelli, A. and Lanza, M. (2013b), "Seahawk: stack overflow in the IDE", *Proceedings of the 2013 International Conference on Software Engineering*, *IEEE Press*, pp. 1295-1298.

Ponzanelli, L., Mocci, A. and Lanza, M. (2015a), "StORMeD: Stack Overflow ready made data", *IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 474-477.

Ponzanelli, L., Mocci, A. and Lanza, M. (2015b), "Summarizing complex development artifacts by mining heterogeneous data", *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 401-405.

Ponzanelli, L., Mocci, A., Bacchelli, A. and Lanza, M. (2014), "Understanding and classifying the quality of technical forum questions", *14th International Conference on Quality Software, IEEE*, pp. 343-352.

Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R. and Lanza, M. (2014a), "Mining StackOverflow to turn the IDE into a self-confident programming prompter", *Proceedings of the 11th Working Conference on Mining Software Repositories, ACM*, pp. 102-111.

Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R. and Lanza, M. (2014b), "Prompter: a self-confident recommender system", *ICSME*, pp. 577-580.

Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R. and Lanza, M. (2015), "Prompter turning the IDE into a self-confident programming assistant", *Empirical Software Engineering*, pp. 1-42.

Ponzanelli, L., Mocci, A., Bacchelli, A., Lanza, M. and Fullerton, D. (2014), "Improving low quality stack overflow post detection", *ICSME*, pp. 541-544.

Ponzanelli, L., Bavota, G., Mocci, A., di Penta, M., Oliveto, R., Russo, B., Haiduc, S. and Lanza, M. (2016), "CodeTube: extracting relevant fragments from software development video tutorials", *Proceedings of the 38th International Conference on Software Engineering Companion, ACM*, pp. 645-648.

Ponzanelli, L., Bavota, G., Mocci, A., di Penta, M., Oliveto, R., Hasan, M., Russo, B., Haiduc, S. and Lanza, M. (2016), "Too long; didn't watch!: extracting relevant fragments from software development video tutorials", *Proceedings of the 38th International Conference on Software Engineering, ACM*, pp. 261-272.

Posnett, D., Warburg, E., Devanbu, P. and Filkov, V. (2012), "Mining stack exchange: expertise is evident from initial contributions", *International Conference on Social Informatics (SocialInformatics), IEEE*, pp. 199-204.

Rahman, M.M. and Roy, C.K. (2015), "An insight into the unresolved questions at stack overflow", *IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 426-429.

Rahman, M.M., Yeasmin, S. and Roy, C.K. (2013), "An IDE-based context-aware meta search engine", *WCRE*, pp. 467-471.

Rahman, M.M., Yeasmin, S. and Roy, C.K. (2014), "Towards a context-aware IDE-based meta search engine for recommendation about programming errors and exceptions", *Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE)*, pp. 194-203.

Rekha, V.S. and Venkatapathy, S. (2015), "Understanding the usage of online forums as learning platforms", *Procedia Computer Science*, Vol. 46, pp. 499-506.

Rekha, V.S., Divya, N. and Bagavathi, P.S. (2014), "A hybrid auto-tagging system for stackoverflow forum questions", *Proceedings of the International Conference on Interdisciplinary Advances in Applied Computing, ACM*, Vol. 56.

Riahi, F., Zolaktaf, Z., Shafiei, M. and Milios, E. (2012), "Finding expert users in community question answering", *Proceedings of the 21st International Conference on World Wide Web, ACM*, pp. 791-798.

Rigby, P.C. and Robillard, M.P. (2013), "Discovering essential code elements in informal documentation", *Proceedings of the 2013 International Conference on Software Engineering, IEEE Press*, pp. 832-841.

Romano, D. and Pinzger, M. (2013), "Towards a weighted voting system for Q&A sites", *IEEE International Conference on Software Maintenance, IEEE.*

Romero, J.J.F., Guerrero, M.G. and Calder, F. (2015), "Multi-class multi-tag classifier system for StackOverflow questions", *IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, pp. 1-6.

Rosen, C. and Shihab, E. (2016), "What are mobile developers asking about? A large scale study using stack overflow", *Empirical Software Engineering*, Vol. 21 No. 3, pp. 1192-1223.

Saha, A.K., Saha, R.K. and Schneider, K.A. (2013), "A discriminative model approach for suggesting tags automatically for stack overflow questions", *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 73-76.

Saha, R.K., Saha, A.K. and Perry, D.E. (2013), "Toward understanding the causes of unanswered questions in software information sites: a case study of stack overflow", *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM*, pp. 663-666.

San Pedro, J. and Karatzoglou, A. (2014), "Question recommendation for collaborative question answering systems with RankSLDA", *Proceedings of the 8th ACM Conference on Recommender Systems*, pp. 193-200.

Sanchez, H. and Whitehead, J. (2015), "Source code curation on StackOverflow: the vesperin system", *IEEE/ACM 37th IEEE International Conference on Software Engineering*, pp. 661-664.

Saxe, J., Mentis, D. and Greamo, C. (2013), "Mining web technical discussions to identify malware capabilities", *IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pp. 1-5.

Schenk, D. and Lungu, M. (2013), "Geo-locating the knowledge transfer in StackOverflow", *Proceedings of the International Workshop on Social Software Engineering, ACM*, pp. 21-24.

Shah, C. and Pomerantz, J. (2010), "Evaluating and predicting answer quality in community QA", *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 411-418.

Singh, G.K., Kumar, V., bhat, S. and Pedanekar, N. (2015), "Automatically augmenting learning material with practical questions to increase its relevance", *Frontiers in Education Conference (FIE), IEEE*, pp. 1-7.

Singh, P. and Simperl, E. (2016), "Using Semantics to Search Answers for Unanswered Questions in Q&A Forums", *Proceedings of the 25th International Conference Companion on World Wide Web, International World Wide Web Conferences Steering Committee*, pp. 699-706.

Sinha, T., Wei, W. and Carley, K. (2015), "Modeling similarity in incentivized interaction: a longitudinal case study of StackOverFlow", *29th Annual International Conference on Neural Information and Processing Systems*.

Sinha, V.S., Mani, S. and Gupta, M. (2013), "Exploring activeness of users in QA forums", *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 77-80.

Slag, R., De Waard, M. and Bacchelli, A. (2015), "One-day flies on StackOverflow: why the vast majority of stackoverflow users only posts once", *Proceedings of the 12th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 458-461.

Sommerville, I. (2010), *Software Engineering*, Pearson Education.

Squire, M. (2015), " 'Should we move to stack overflow?' Measuring the utility of social media for developer support", *IEEE/ACM 37th IEEE International Conference on Software Engineering,*, pp. 219-228.

Squire, M. and Funkhouser, C. (2014), "'A bit of code': how the Stack Overflow community creates quality postings", *47th Hawaii International Conference on System Sciences, IEEE*, pp. 1425-1434.

Stanley, C. and Byrne, M.D. (2013), "Predicting tags for stackoverflow posts", Proceedings of ICCM.

Stevens, R., Ganz, J., Filkov, V., Devanbu, P. and Chen, H. (2013), "Asking for (and about) permissions used by android apps", *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 31-40.

Storey, M.-A. (2015), "Selecting research methods for studying a participatory culture in software development: keynote", *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering (EASE), ACM, April 27-29*.

Storey, M.-A., Singer, L., Cleary, B., Filho, F.F. and Zagalsky, A. (2014), "The (R)Evolution of social media in software engineering", ACM Future of Software Engineering (FOSE) pp. 100-116.

Subramanian, S. and Holmes, R. (2013), "Making sense of online code snippets", *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 85-88.

Subramanian, S., Inozemtseva, L. and Holmes, R. (2014), "Live API documentation", *Proceedings of the 36th International Conference on Software Engineering, ACM*, pp. 643-652.

Sushine, J., Herbsleb, J.D. and Aldrich, J. (2015), "Searching the state space: a qualitative study of API protocol usability", *IEEE 23rd International Conference on Program Comprehension, IEEE*, pp. 82-93.

Tavakoli, M., Heydarnoori, A. and Ghafari, M. (2016), "Improving the quality of code snippets in stack overflow", *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, pp. 1492-1497.

Tian, Y., Lo, D. and Lawall, J. (2014), "Automated construction of a software-specific word similarity database", *Software Evolution Week-IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE)*, pp. 44-53.

Tian, Y., Achananuparp, P., Lubis, I.N., Lo, D. and Lim, E.-P. (2012), "What does software engineering community microblog about?", *9th IEEE Working Conference on Mining Software Repositories (MSR), Zurich*, pp. 247-250.

Tian, Y., Kochhar, P.S., Lim, E.-P., Zhu, F. and Lo, D. (2013), "Predicting best answerers for new questions: an approach leveraging topic modeling and collaborative voting", *Workshops at the International Conference on Social Informatics*, *Springer*, pp. 55-68.

Tichy, W. (2010), RE: An interview with Prof. Andreas Zeller: mining your way to software reliability.

Treude, C. and Robillard, M.P. (2016), "Augmenting API documentation with insights from Stack Overflow", *Proceedings of the 38th International Conference on Software Engineering, ACM*, pp. 392-403.

Treude, C., Barzilay, O. and Storey, M.-A. (2011), "How do programmers ask and answer questions on the web? Nier track", *33rd International Conference on Software Engineering (ICSE), IEEE*, pp. 804-807.

Udanor, C., Aneke, S. and Ogbuokiri, O. (2016), "Determining social media impact on the politics of developing countries using social network analytics", *Program: Electronic Library and Information Systems*, Vol. 50 No. 4, pp. 481-507.

Vasilescu, B. (2014), "Software developers are humans, too!", *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work and social Computing, ACM*, pp. 97-100.

Vasilescu, B., Capiluppi, A. and Serebrenik, A. (2012), "Gender, representation and online participation: a quantitative study of StackOverflow", *International Conference on Social Informatics (SocialInformatics), IEEE*, pp. 332-338.

Vasilescu, B., Filkov, V. and Serebrenik, A. (2013), "StackOverflow and GitHub: associations between software development and crowdsourced knowledge", *International Conference on Social Computing (SocialCom), IEEE*, pp. 188-195.

Vasilescu, B., Serebrenik, A., Devanbu, P. and Filkov, V. (2014), "How social Q&A sites are changing knowledge sharing in open source software communities", *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing*, pp. 342-354.

Vassallo, C., Panichella, S., Di Penta, M. and Canfora, G. (2014), "Codes: mining source code descriptions from developers discussions", *Proceedings of the 22nd International Conference on Program Comprehension, ACM*, pp. 106-109.

Venkataramani, R., Gupta, A., Asadullah, A., Muddu, B. and Bhat, V. (2013), "Discovery of technical expertise from open source code repositories", *Proceedings of the 22nd International Conference on World Wide Web, ACM*, pp. 97-98.

Vinayakarao, V., Purandare, R. and Nori, A.V. (2015), "Structurally heterogeneous source code examples from unstructured knowledge sources", *Proceedings of the Workshop on Partial Evaluation and Program Manipulation, ACM*, pp. 21-26.

Wang, H.-C., Yang, C.-T. and Yen, Y.-H. (2017), "Answer selection and expert finding in community question answering services a question answering promoter", *Program: Electronic Library and Information Systems*, Vol. 51 No. 1, pp. 17-34.

Wang, S., Keivanloo, I. and Zou, Y. (2014), "How do developers react to RESTful API evolution?", *International Conference on Service-Oriented Computing*, Springer, pp. 245-259.

Wang, S., Lo, D. and Jiang, L. (2013), "An empirical study on developer interactions in stackoverflow", *Proceedings of the 28th Annual ACM Symposium on Applied Computing, ACM*, pp. 1019-1024.

Wang, S., Lo, D., Vasilescu, B. and Serebrenik, A. (2014), "EnTagRec: an enhanced tag recommendation system for software information sites", *ICSME*, pp. 291-300.

Wang, T., Yin, G., Wang, H., Yang, C. and Zou, P. (2015), "Automatic knowledge sharing across communities: a case study on android issue tracker and stack overflow", IEEE Symposium on Service-Oriented System Engineering (SOSE), IEEE, pp. 107-116.

Wang, W. and Godfrey, M.W. (2013), "Detecting API usage obstacles: A study of iOS and Android developer questions", *10th IEEE Working Conference on Mining Software Repositories (MSR)*, pp. 61-64.

Wang, W., Malik, H. and Godfrey, M.W. (2015), "Recommending posts concerning API issues in developer Q&A sites", *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 224-234.

Wang, X.-Y., Xia, X. and Lo, D. (2015), "TagCombine: recommending tags to contents in software information sites", *Journal of Computer Science and Technology*, Vol. 30 No. 5, pp. 1017-1035.

Wong, E., Yang, J. and Tan, L. (2013), "Autocomment: mining question and answer sites for automatic comment generation", *IEEE/ACM 28th International Conference on Automated Software Engineering (ASE), IEEE*, pp. 562-567.

Wright, H.K., Kim, M. and Perry, D.E. (2010), "Validity concerns in software engineering research", *FoSER*, ACM, Santa Fe, NM.

Xia, X., Lo, D., Wang, X. and Zhou, B. (2013), "Tag recommendation in software information sites", *Proceedings of the 10th Working Conference on Mining Software Repositories*, *IEEE Press*, pp. 287-296.

Xia, X., Lo, D., Correa, D., Sureka, A. and Shihab, E. (2016), "It Takes two to tango: deleted stack overflow question prediction with text and meta features", *40th Annual International Computers, Software & Applications Conference (COMPSAC), IEEE, June*.

Xu, B., Xing, Z., Xia, X., Lo, D., Wang, Q. and Li, S. (2016), "Domain-specific cross-language relevant question retrieval", *Proceedings of the 13th International Workshop on Mining Software Repositories, ACM*, pp. 413-424.

Yang, D., Hussain, A. and Lopes, C.V. (2016), "From query to usable code: an analysis of stack overflow code snippets", *Proceedings of the 13th International Workshop on Mining Software Repositories, ACM*, pp. 391-402.

Yang, J., Bozzon, A. and Houben, G.-J. (2015), "Harnessing engagement for knowledge creation acceleration in collaborative Q&A systems", *International Conference on User Modeling, Adaptation, and Personalization*, *Springer*, pp. 315-327.

Yang, J., Hauff, C., Bozzon, A. and Houben, G.-J. (2014), "Asking the right question in collaborative Q&A systems", *Proceedings of the 25th ACM Conference on Hypertext and Social Media, ACM*, pp. 179-189.

Yang, J., Tao, K., Bozzon, A. and Houben, G.-J. (2014), "Sparrows and owls: characterisation of expert behaviour in stackoverflow", *International Conference on User Modeling, Adaptation, and Personalization*, *Springer*, pp. 266-277.

Yang, L., Qiu, M., Gottipati, S., Zhu, F., Jiang, J., Sun, H. and Chen, Z. (2013), "CQArank: jointly model topics and expertise in community question answering", *Proceedings of the 22nd ACM international conference on Information and Knowledge Management, ACM*, pp. 99-108.

Yao, Y., Tong, H., Xie, T., Akoglu, L., Xu, F. and Lu, J. (2013), "Want a good answer? ask a good question first!", arXiv preprint arXiv:1311.6876.

Ye, D., Xing, Z. and Kapre, N. (2016), "The structure and dynamics of knowledge network in domain-specific Q&A sites: a case study of stack overflow", *Empirical Software Engineering*, Vol. 22 No. 1, pp. 375-406.

Ye, D., Xing, Z., Li, J. and Kapre, N. (2016), "Software-specific part-of-speech tagging: an experimental study on stack overflow", *Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM*, pp. 1378-1385.

Ye, D., Xing, Z., Foo, C.Y., Ang, Z.Q., Li, J. and Kapre, N. (2016), "Software-specific named entity recognition in software engineering social content", *IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, pp. 90-101.

Ye, T., Xie, B., Zou, Y. and Chen, X. (2014), "Interrogative-guided re-ranking for question-oriented software text retrieval", *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, ACM*, pp. 115-120.

Zagalsky, A., Teshima, C.G., German, D.M., Storey, M.-A. and Poo-Caamaño, G. (2016), "How the R community creates and curates knowledge: a comparative study of stack overflow and mailing lists", *Proceedings of the 13th International Workshop on Mining Software Repositories, ACM*, pp. 441-451.

Zhang, Y., Lo, D., Xia, X. and Sun, J.-L. (2015), "Multi-factor duplicate question detection in stack overflow", *Journal of Computer Science and Technology*, Vol. 30 No. 5, pp. 981-997.

Zhu, J., Shen, B., Cai, X. and Wang, H. (2015), "Building a large-scale software programming taxonomy from stackoverflow", *27th International Conference on Software Engineering and Knowledge Engineering (SEKE'2015)*, pp. 391-396.

Zolaktaf, Z., Riahi, F., Shafiei, M. and Milios, E. (2012), "Modeling Community Question-Answering Archives", Second Workshop on Computational Social Science and the Wisdom of Crowds (NIPS 2011).

Zou, J., Xu, L., Guo, W., Yan, M., Yang, D. and Zhang, X. (2015), "Which non-functional requirements do developers focus on? An empirical study on stack overflow using topic analysis", *IEEE/ACM 12th Working Conference on Mining Software Repositories*, pp. 446-449.

Zou, Y., Ye, T., Lu, Y., Mylopoulos, J. and Zhang, L. (2015), "Learning to rank for question-oriented software text retrieval (T)", *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 1-11.

About the authors

Arshad Ahmad has received an MS Degree in Software Engineering from the Blekinge Institute of Technology, Sweden in 2008. He is currently a PhD Student at the School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include requirements engineering, text mining, sentiment analysis, machine learning, and data mining.

Chong Feng received his PhD Degree in Computer Science from the University of Science and Technology of China, Hefei, in 2005. He is now working as an Associate Professor of Computer Science and Technology at the Beijing Institute of Technology, Beijing. His current research interests focus on social media processing, information extraction and machine translation. Chong Feng is the corresponding author and can be contacted at: fengchong@bit.edu.cn

Shi Ge has received his MS Degree in Computer Science from the Beijing Institute of Technology, China in 2016 and is currently a PhD Student at School of Computer Science and Technology, Beijing Institute of Technology, China. His research interests include knowledge representation, entity linking, and relation extraction.

Abdallah Yousif has received his MPhil Degree in Computer Science from the University of Gezira, Sudan in 2010. He is currently a PhD Student at School of Computer Science and Technology, Beijing Institute of Technology, China. He also works as a Lecturer in the Sudan Technological University, Sudan. His research interests include sentiment analysis, machine learning, citation analysis, and data mining.