Yang XL, Lo D, Xia X *et al.* What security questions do developers ask? A large-scale study of stack overflow posts. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 31(5): 910–924 Sept. 2016. DOI 10.1007/s11390-016-1672-0

What Security Questions Do Developers Ask? A Large-Scale Study of Stack Overflow Posts

Xin-Li Yang [1], David Lo [2], *Member, ACM, IEEE*, Xin Xia [1,*], *Member, CCF, ACM, IEEE*, Zhi-Yuan Wan [1], and Jian-Ling Sun [1], *Member, CCF, ACM*

[1] *College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*
[2] *School of Information Systems, Singapore Management University, Singapore, Singapore*

E-mail: zdyxl@zju.edu.cn; davidlo@smu.edu.sg; {xxkidd, wanzhiyuan, sunjl}@zju.edu.cn

Received March 21, 2016; revised August 14, 2016.

Abstract Security has always been a popular and critical topic. With the rapid development of information technology, it is always attracting people's attention. However, since security has a long history, it covers a wide range of topics which change a lot, from classic cryptography to recently popular mobile security. There is a need to investigate security-related topics and trends, which can be a guide for security researchers, security educators and security practitioners. To address the above-mentioned need, in this paper, we conduct a large-scale study on security-related questions on Stack Overflow. Stack Overflow is a popular on-line question and answer site for software developers to communicate, collaborate, and share information with one another. There are many different topics among the numerous questions posted on Stack Overflow and security-related questions occupy a large proportion and have an important and significant position. We first use two heuristics to extract from the dataset the questions that are related to security based on the tags of the posts. And then we use an advanced topic model, Latent Dirichlet Allocation (LDA) tuned using Genetic Algorithm (GA), to cluster different security-related questions based on their texts. After obtaining the different topics of security-related questions, we use their metadata to make various analyses. We summarize all the topics into five main categories, and investigate the popularity and difficulty of different topics as well. Based on the results of our study, we conclude several implications for researchers, educators and practitioners.

Keywords security, Stack Overflow, empirical study, topic model

1 Introduction

With the rapid development of information technology, security is always attracting people's attention. Users care about security to prevent their personal information from being leaked. Developers care about security to protect the software they develop. Security experts care about security to fight against hackers who can often find security holes to exploit. Security is very critical since it can cause financial loss, privacy leakage and confidentiality leakage.

There are many different security-related topics and the hot topic is always changing (e.g., from classic cryptography to recently popular mobile security). Therefore, we think there is a need to make a comprehensive study to investigate security-related topics and analyze the trend of security-related technologies.

For this purpose, we take Stack Overflow① as a dataset source. Stack Overflow is one of the most popular software information sites where people ask and answer technical questions about software development and maintenance. Stack Overflow contains millions of posts which cover a wide range of topics includ-

Regular Paper

Special Section on Software Systems 2016

This work is supported by the National Natural Science Foundation of China under Grant No. 61572426 and the National Key Technology Research and Development Program of the Ministry of Science and Technology of China under Grant No. 2015BAH17F01.

*Corresponding Author

①http://www.stackoverflow.com/, Aug. 2016.

ing programming-related, mobile-related and security-related topics.

There are several recent studies based on Stack Overflow. Barua *et al.* conducted a large empirical study on Stack Overflow to analyze the topics and trends of what developers talk about[1]. Rosen and Shihab studied questions mobile-developers ask on Stack Overflow[2]. Both of them reported some interesting and valuable conclusions.

Extending prior work that investigates Stack Overflow topics, we conduct a large-scale study on topics covered by security-related questions on Stack Overflow. We use a dataset named "posts.xml", which is publicly available on Stack Exchange Data Dump②. The dataset contains more than 21 million posts, each of which has a text (e.g., body) describing a question or an answer together with some metadata (e.g., creation date and view count). We first use two heuristics to extract questions that are related to security based on the tags of the posts. And then we use an advanced topic model, LDA (Latent Drichlet Allocation) tuned using GA (Genetic Algorithm), to cluster different security-related questions based on their texts. After we obtain different topics of security-related questions, we use their metadata to make various analyses.

Our empirical study investigates the following three research questions.

RQ-1. *What topics are covered by security-related questions asked on Stack Overflow?*

We use a topic model to investigate the security-related topics. Security-related questions on Stack Overflow cover a wide range of topics. These topics mainly belong to five main categories, i.e., web security, mobile security, cryptography, software security, and system security. And among them most questions are about web security.

RQ-2. *Which topics are the most popular among security-related questions?*

We measure the popularity of security-related topics by one major metric (i.e., the average number of views), and three minor metrics (i.e., the average number of comments, the average number of favourites, and the average score). The top four most popular security topics are "Password", "Hash", "Signature" and "SQL Injection", among which "Hash" and "SQL Injection" are the most valuable since they receive the largest number of comments and favourites, and the highest scores.

RQ-3. *Which security-related topics are the most difficult to answer?*

We measure the difficulty of security-related topics by two metrics (i.e., the average time needed for an accepted answer and the proportion of the average number of answers to the average number of views). The top eight most difficult security-related topics are "Java Security", "Asymetric Encryption", "Bug", "Browser Security", "Windows Authority", "Signature", "ASP.NET" and "Password". When considering both popularity and difficulty, "Signature" and "Password" are the two topics that deserve the most attentions since they are both popular and difficult.

The main contributions of this paper are as follows.

1) We conduct an empirical study on Stack Overflow to investigate and cluster security-related questions. To the best of our knowledge, it is the first large-scale study to investigate security-related topics and trends on Stack Overflow.

2) We report several interesting and valuable conclusions. We investigate the popularity and difficulty of security-related topics which provide some implications to researchers, educators and practitioners.

The remainder of this paper is organized as follows. We elaborate the motivation of our work and introduce Latent Dirichlet Allocation (LDA) in Section 2. The data collection step and our experimental approach are described in Section 3. Our experimental results are presented and discussed in Sections 4 and 5 respectively. Related work is introduced in Section 6 and Section 7 concludes the paper.

2 Preliminaries

In this section, we first introduce Stack Overflow and the security-related posts on Stack Overflow in Subsections 2.1 and 2.2, respectively. We then briefly introduce Latent Dirichlet Allocation (LDA), which is a classic topic model which we use to group different topics of security question posts in this paper.

2.1 Stack Overflow

Stack Overflow is one of the most popular question and answer websites. A significant fraction of the participants on Stack Overflow have deep expertise in a variety of areas. Developers ask questions about a wide range of topics and seek advice about the technical challenges they meet. The questions and answers

②https://archive.org/details/stackexchange, Aug. 2016.

are saved on the site and can be searched via search engines. Stack Overflow acts as a knowledge repository for various needs of developers. Analyzing and understanding the knowledge repository could lead to deep insights about the topics of interest to the developers.

There have been a number of studies on Stack Overflow. Some studies categorize the questions on Stack Overflow[3], identify design features[4], and analyze the textual contents of posts[1]. Some studies recommend tags for the questions on Stack Overflow[5-6], and some analyze the topics of Stack Overflow posts[2,7-8].

2.2 Security-Related Posts

Stack Overflow contains millions of posts which cover a wide range of topics, such as programming-related, mobile-related and security-related topics. Due to the importance of security, there is a significant proportion of security-related posts.

Fig.1 presents a top-rated security question on Stack Overflow. The title of the post is "How can I prevent SQL-injection in PHP?" The tags of the post are "php", "mysql", "sql", "security" and "sql-injection". Between the title and the tags is the body of the post, describing the question in detail. Also, there are several metadata in the margin of the post, such as the number of comments, the edit date.

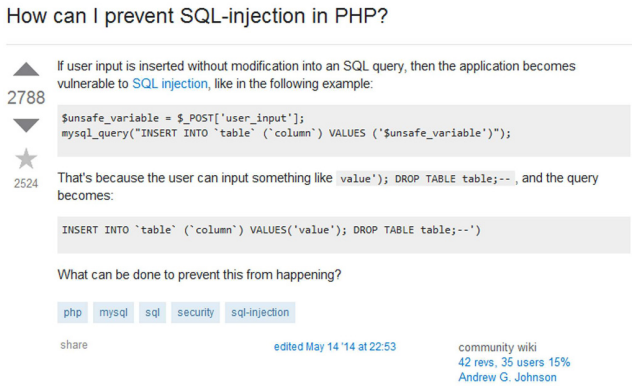


Fig.1. Security-related post on Stack Overflow.

Note that the tags of the above security-related post contain "security". However, not all security-related questions contain this tag. For example, Fig.2 shows a security-related post whose tags do not contain "security". Therefore we cannot determine security-related posts by simply checking whether the posts contain the tag of "security", since the extracted posts will not be sufficient and satisfactory. To address this limitation, in this paper, we first design two heuristics (which are elaborated in Section 3) to extract security-related tags, and then extract security-related posts according to the extracted tags.

Fig.2. Security-related post whose tags do not contain "security".

2.3 Latent Dirichlet Allocation

A topic model views a document to be a probability distribution of topics, while a topic is a probability distribution of words. In our setting, a document is the text in a post (i.e., body and title), and a topic is a higher-level concept corresponding to a distribution of words in the text. For example, we can have a topic "SQL Injection" when the text contains a distribution of words such as "sql", "inject", "query", "statement", "mysql".

Latent Dirichlet Allocation (LDA)[9] is a well-known topic model used for various tasks of software engineering research[10-12]. In theory, LDA is a generative probabilistic model, which assumes that the data (a collection of documents) is generated based on a certain statistical process. Specifically, LDA contains three steps.

1) *Step* 1: LDA generates a topic distribution vector ***theta*** and a term distribution vector ***phi*** based on two Dirichlet distributions[13], respectively.

2) *Step* 2: LDA generates a topic assignment vector $\boldsymbol{z}$ to assign each term in a document a specific topic according to the topic distribution vector of the document ***theta***.

3) *Step* 3: LDA generates each term in a document with the topic distribution vector ***phi*** and the topic assignment vector $\boldsymbol{z}$.

By repeating step 1 K times, K topics are generated. By repeating step 2 and step 3 N times, a document having N terms is generated. By repeating steps 1∼3 D times, a collection of D documents is generated.

In practice, LDA takes a document-by-term $(D \times N)$ matrix $\boldsymbol{a}$ as input, and outputs two matrices $\boldsymbol{b}$ and $\boldsymbol{c}$,

i.e., document-by-topic ($D \times K$) matrix and topic-by-term ($K \times N$) matrix. The document-by-term matrix $\boldsymbol{a}$ can be a term frequency matrix, in which a_{ij} represents the number of times that the j-th term appears in the i-th document. In the document-by-topic matrix $\boldsymbol{b}$, b_{ij} represents the probability of the i-th document belonging to the j-th topic. Generally, a document is regarded as belonging to the topic with the highest probability. In the topic-by-term matrix $\boldsymbol{c}$, c_{ij} represents the probability that the j-th term belongs to the i-th topic. Likewise, we assign a term to the topic with the highest probability and then we can conclude what a topic is about by looking up the terms it contains.

To some extent, LDA can be seen as a clustering algorithm. By assigning a specific topic for each document using the document-by-topic matrix, a clustering of documents can be completed. Specifically, documents assigned to the same topic are grouped together.

There are several off-the-shelf LDA implementations. In our work, we use a python package named lda[③], which is an implementation based on collapsed Gibbs sampling. In addition, we follow the default setting for all the parameters in the package.

3 Case Study Setup

In this section, we describe the details of the setup of our empirical experiments. We first present the details of data collection in Subsection 3.1, and then we elaborate our experimental approach in Subsection 3.2.

3.1 Data Collection

To conduct a comprehensive empirical study, we use a whole Stack Overflow dataset which is publicly available on Stack Exchange Data Dump. Our Stack Overflow dataset contains a total of 21 674 904 posts, spanning from July 2008 to September 2014. In the dataset, there are 7 990 787 (37%) question posts and 13 684 117 (63%) answer posts. For each post, it includes body and several metadata. The detailed information is shown in Table 1.

Table 1. Detailed Information of a Post

Name	Description
Id	ID of the post
PostTypeId	Type of the post: 1 represents a question post, and 2 represents an answer post
AcceptedAnswerId	ID of the corresponding accepted answer post for the question post (optional, and appears only when PostTypeId==1)
ParentID	ID of the corresponding question post for the answer post (optional, and appears only when PostTypeId==2)
CreationDate	Creation date of the post
Score	Average score by the viewers for the post
ViewCount	Total number of views for the post (optional, and appears only when PostTypeId==1)
Body	Body of the post
OwnerUserId	ID of the post owner (optional)
OwnerDisplayName	Username of the post owner (optional)
LastEditorUserId	ID of the person who last edited the post
LastEditorDisplayName	Username of the person who last edited the post
LastEditDate	Date when the post is last edited
LastActivityDate	Date when the status of the post is last changed
Title	Title of the post (optional, and appears only when PostTypeId==1)
Tags	Tags of the post (optional, and appears only when PostTypeId==1)
AnswerCount	Number of answers for the post (optional, and appears only when PostTypeId==1)
CommentCount	Number of comments for the post
FavoriteCount	Number of people who like the post (optional, and appears only when PostTypeId==1)
ClosedDate	Date when the post is closed (optional)

Notice the posts in the dataset can be related to any topics, and in this paper, we focus on security-related posts. To do so, we first extract the security-related question posts from Stack Overflow by leveraging the tags of the questions. And for the answer posts, we extract them according to the extracted question posts. Our method mainly contains following three steps.

First, we traverse the dataset to find the question

[③] http://pythonhosted.org/lda/, Aug. 2016.

posts whose tags contain the term "security". In total, we extract 28 476 posts. However, these posts are far from sufficient. Actually, some posts may not have a tag "security" even if they are about "security", since "security" is a very general term and the tags of a post may be in a much finer granularity. Therefore, we need to carefully extract some other tags which are related to "security".

In the second step, we extract the tags from all of the 28 476 posts extracted in the first step, and we refer to these tags as candidate tags. Next, we extract more tags related to "security" from the candidate tags. For each candidate tag t, we count three values, i.e., a, b and c. Specifically, a denotes the number of question posts whose tags contain t among all the posts extracted in the first step (i.e., the number of all question posts whose tags contain both t and "security"). b denotes the number of question posts whose tags contain t among all the posts in the original dataset (i.e., the number of all question posts whose tags contain t). Based on a and b, we can calculate the first value of $H1 = a/b$, which can indicate to what extent the tag t is exclusively related to "security". The value of $H1$ ranges from 0 to 1. The larger the value of $H1$ is, the more exclusive relation tag t has to "security". If the value of $H1$ equals 1, it means that tag t only appears together with "security" in the tags of the posts (i.e., the most exclusive). We can filter the tags by setting a threshold $Thre1$. For example, given $Thre1$ set to 0.1, a tag that appears together with "security" in less than 10% of all the questions whose tags contain the tag will be removed.

However, only using the above heuristic $H1$ to extract the tags would cause another problem. Suppose that a tag only appears in one post of the whole dataset and the post happens to be related to "security". In this case, the tag is so specific to a problem that it is not representative to "security", although its value of $H1$ equals 1. Therefore, we also want to filter this kind of tags. We denote c as the total number of question posts extracted in the first step (i.e., the number of all question posts whose tags contain "security"). Based on a and c, we can calculate the second value of $H2 = a/c$, which well solves the above problem if we set a second threshold $Thre2$ to filter the tags once more. For example, given $Thre2$ set to 0.01, a tag that appears in less than 1% of all the questions whose tags contain "security" will be removed.

By setting $Thre1$ and $Thre2$ well, we can obtain several tags, which are both exclusive and representative to "security". Table 2 shows different tag sets related to "security" we extract with different threshold configurations. In the following text, the threshold configuration (0.1, 0.01) is the default configuration, and the corresponding result is the default tag set we use (i.e., the first row in Table 2) to retrieve the security-related posts.

In the final step, we traverse the dataset again to find the question posts whose tags contain at least one of the tags in the tag set. The total number of such posts is 30 054 when we use the default tag set. And we mainly use these question posts and their corresponding answer posts to make analysis.

3.2 Data Analysis

We now detail our experimental approach. The approach mainly contains two phases, i.e., feature extraction phase (cf. Subsection 3.2.1) and topic modeling phase (cf. Subsection 3.2.2).

In the feature extraction phase, we first extract a number of features from the posts. These features are selected as representative terms that are useful in building a good topic model. In our work, we use the term frequency as features. Next, we build a topic model with the extracted features using LDA tuned using Genetic Algorithm (GA). GA is used to determine the optimal number of topics. And LDA clusters different security posts into different topics according to their corresponding topics.

Table 2. Different Tag Sets Related to "Security" Extracted by Different Threshold Configurations

Threshold Configuration ($Thre1$, $Thre2$)	Tag Set	Number of Tags
(0.1, 0.01)	security, sql-injection, passwords, encryption, xss, cryptography, web-security	7
(0.1, 0.005)	security, sql-injection, passwords, encryption, xss, cryptography, csrf, password-protection, web-security	9
(0.15, 0.01)	security, sql-injection, passwords, xss, web-security	5
(0.15, 0.005)	security, sql-injection, passwords, xss, csrf, password-protection, web-security	7
(0.2/0.25, 0.01/0.005)	security, xss, web-security	3
($\geqslant$ 0.3, 0.01/0.005)	security	1

3.2.1 Feature Extraction

As mentioned in Subsection 3.1, a question post includes title, body and several metadata. To cluster the posts, we need to build a corpus in which each row is a text for a post. For each post, we combine both title and body to form the final text. And we preprocess the texts in four main steps.

1) Step 1: we remove all the code snippets (which are enclosed in $< code >$ tag) in the text, since Barua *et al.* showed that code snippets do not help topic models[1].

2) Step 2: we remove all the HTML tags such as $< p >$ and $< /p >$ since they do not have useful information for the topic model.

3) Step 3: we remove the stop words, numbers, punctuation marks and other non-alphabetic characters since they add little value to the topic.

4) Step 4: we use the Snowball stemmers[14] to transform the remaining terms to their root forms (e.g., "reading" and "reads" are reduced to "read") in order to reduce the feature dimensions and unify similar words into a common representation.

After the above four steps, we compute the frequency of appearance in all the posts for each stemmed term. To further reduce the noise, we sort all the stemmed terms according to their total term frequency and discard the terms which appear less than 10 times.

The remaining 4 333 different stemmed terms are the final features we extract. We count the times of appearance for each term in each post and form a term frequency matrix $\boldsymbol{m}$. Specifically, w_{ij} denotes the number of times the j-th term appears in the i-th post.

3.2.2 LDA Tuned Using Genetic Algorithm

As mentioned in Section 2, we use LDA to group posts into different topics. In LDA, the number of topics K is an undetermined but important parameter. An overly large or overly small value of K may influence the performance of our approach severely. Therefore, we use an advanced LDA technique, tuned using GA, to search for an optimized value of K.

Genetic algorithms simulate evolutions by natural selection[15]. In GA, the parameters waiting to be searched are coded as an individual "chromosome" and a so-called fitness function is pre-defined. The fitness function is used to evaluate different parameter configurations by generating different fitness values. As a start, a population of p randomly-generated chromosomes are initiated, where each of them contains a random parameter configuration. Then, the population will evolve n generations to search for an optimal parameter configuration. For each generation, the population goes through three phrases: selection, crossover and mutation. In the selection phrase, different chromosomes are selected by a selection probability, which is transformed from their corresponding fitness values. The higher the fitness value is, the higher the selection probability is. In the crossover phrase, the selected chromosomes are paired in a random way and each pair of chromosomes are crossed over to generate a new pair of chromosomes by a crossover function. In the mutation phrase, the new generated chromosomes are mutated randomly by a mutation function and a mutation probability. Because of that, the whole population is updated and becomes a new generation after the aforementioned three phrases. With the generations evolving, better and better individuals (with higher fitness values) will emerge. Actually, GA has many configurable parameters, such as population size, the number of generations, mutation function and so on. For simplicity, we will not detail them in this paper. Interested readers can obtain more information from [15].

Algorithm 1 presents the GA process in adaptive LDA. In our work, our LDA-GA approach is implemented on top of Pyevolve④, an evolutionary computation framework. We set p to 20 and n to 50 since we have tried several configurations and empirically found that this setting can achieve a better fitness value and keep the result stable. For the search range, we set it to be integers in [2, 50] since there are at least 2 topics and 50 is likely to be sufficient for the maximum number of topics. We use one-dimensional integer list (G1DList) to represent the chromosomes and use the function "G1DListInitializatorInteger", which generates random integers in the search range, to initialize them. For the selection phase, we use the default function "GRankSelector", which is a rank selector. For the crossover phase, we use the function "G1DListCrossoverUniform", which performs crossover uniformly, and use the default crossover rate (0.9). For the mutation phase, we use the default function "G1DListMutatorSwap", which is a swap mutator for G1DList, and use the default mutation rate (0.02).

For the fitness function, we use the Silhouette coefficient as the fitness value. The Silhouette coefficient is a common evaluation metric for clustering[12,16-18].

④http://pyevolve.sourceforge.net/, Aug. 2016.

Algorithm 1 . GA Process in Adaptive LDA

1: **Input:** population size, p;
number of generations, n;
search scope of the number of topics, $[min, max]$;
2: **Output:** number of topics, k_best;
3: Pick p random values from the range $[min, max]$, using an initialization function and denote them as k;
4: For each value k_i, compute the Silhouette coefficient as its fitness value;
5: According to the fitness value, use a selection function to select better values in k;
6: Cross over selected values using a crossover function to generate new values and denote them as k_new;
7: Mutate some values in k_new using a mutation function, and replace k with these values;
8: Repeat step 4 to 7 for n times and output the value k_best with the best fitness value;

The computation of the Silhouette coefficient consists of three steps.

1) *Step* 1. For a document d_i, we calculate the maximum distance from d_i to the other documents in the same cluster, which is denoted as $a(d_i)$. And we calculate the minimum distance from d_i to the centroids of the other clusters (i.e., the clusters that do not contain d_i), which is denoted as $b(d_i)$.

2) *Step* 2. Given $a(d_i)$ and $b(d_i)$, we can calculate the Silhouette coefficient $S(d_i)$ for document d_i according to the following formula:

$$S(d_i) = \frac{b(d_i) - a(d_i)}{\max(a(d_i), b(d_i))}.$$

3) *Step* 3. We compute the mean value of all $S(d_i)$ as the overall Silhouette coefficient.

The scope of the Silhouette coefficient is $[-1, 1]$. A bigger value of the Silhouette coefficient indicates a better clustering. When a value of the number of topics achieves a high Silhouette coefficient, it means that the value leads to a good result for LDA. The higher Silhouette coefficient a value achieves, the more likely it is to be kept in the evolution process. Therefore, with LDA tuned using GA, we can find a proper number of topics for all the security question posts and assign each post to its corresponding topic.

4 Case Study

In this section, we present the experimental results of our study on the three research questions we are interested in (cf. Section 1).

All experiments for answering the three research questions are conducted on an Intel® Core™ T6570 with 2.10 GHz CPU and 4 GB RAM PC running Windows 7 (64-bit).

4.1 Results of RQ-1

RQ-1. *What topics are covered by security-related questions asked on Stack Overflow?*

Motivation. With the first research question, we aim to showcase the topics covered by security-related questions asked on Stack Overflow. This research question can give developers a deeper insight into the security-related questions and make them aware of different topics about security.

Approach. As mentioned in Section 3, we use an advanced topic model, LDA tuned using GA, to cluster the security-related questions. Generally, LDA needs a pre-defined number of topics K and it has different optimal values of K for different problems. Our approach can automatically determine a (near) optimal value of K according to the characteristic of a specific problem so that LDA can achieve a better result.

Results. By using our approach, we group the security-related questions into 30 topics. Table 3 presents the 30 topics including the topic names and the related top 10 key terms.

From Table 3, we find that the topics of the security-related questions cover a wide range. Some topics have a finer granularity while some have a coarser one. For example, "SSL" is a technique while "Memory" includes several techniques. Fig.3 presents the number of questions belonging to each topic. From the figure, we can explicitly see the distribution of questions in different topics. The topic that has the most questions is "Encryption/Decryption", and the topic that has the least questions is "Link".

In addition, we find that all the topics mainly belong to five main categories, i.e., web security, mobile security, cryptography, software security, and system security. Table 3 also assigns each topic to these categories in the brackets in the second column. Fig.4

Table 3. Topic Names and Related Top 10 Key Terms for Top 30 Topics

No.	Topic Name	Key Terms (After Stemming)
1	SQL injection (web security)	sql inject queri statement mysql string code prevent escap paramet
2	Timing attack (web security)	time attack forc question way veri second solut brute problem
3	IP address (web security)	ip address site access email websit host assembl server port
4	App access (mobile security)	app api android use applic devic access secur user mobil
5	Encryption/decryption (cryptography)	encrypt decrypt key data byte block cipher pad bit algorithm
6	PHP authentication (web security)	page login user php secur redirect button click submit check
7	Flash (web security)	secur softwar game flash card product develop tool custom report
8	Javascript (web security)	request post token ajax data server javascript json send jqueri header
9	Java security (software security)	java secur spring xml web applic configur class jar tomcat
10	Pseudo random (cryptography)	number generat random bit algorithm implement uniqu valu cryptograph time
11	Software development (software security)	chang set object field method class valu properti updat creat
12	Asymetric encryption (cryptography)	key public privat encrypt rsa generat decrypt secret data store
13	XSS (web security)	html xss javascript script tag attack prevent code site page
14	Access control (web security)	user access role admin secur permiss control group onli author
15	Database (system security)	databas data store tabl mysql server sql db connect user
16	File transmission (web security)	file imag upload download applet pdf read open save content
17	Bug (software security)	error tri work code follow test problem fail issu line
18	Browser security (web security)	site page browser secur url domain https web load websit
19	Encoding/decoding (cryptography)	string encod byte valu array base charact convert decod tri
20	Directory traversal (system security)	php file script access server folder directori secur apach root
21	Windows authority (system security)	window run machin applic access local server instal user process
22	Hash (cryptography)	hash function sha python algorithm md php implement differ bit
23	SSL (web security)	server client ssl send connect secur messag data https communic
24	Signature (web security)	certif sign signatur java verifi creat code openssl pkcs provid
25	Library (system security)	code secur librari net class sourc project doe dll framework
26	Link (web security)	com http www url exampl html domain org https link
27	ASP.NET (web security)	net servic asp web secur authent mvc wcf config applic
28	Session management (web security)	user session cooki id log authent login secur token set
29	Password (web/system security)	password user usernam login credenti enter store way chang authent
30	Memory (system security)	code program test write function return read buffer memori stack

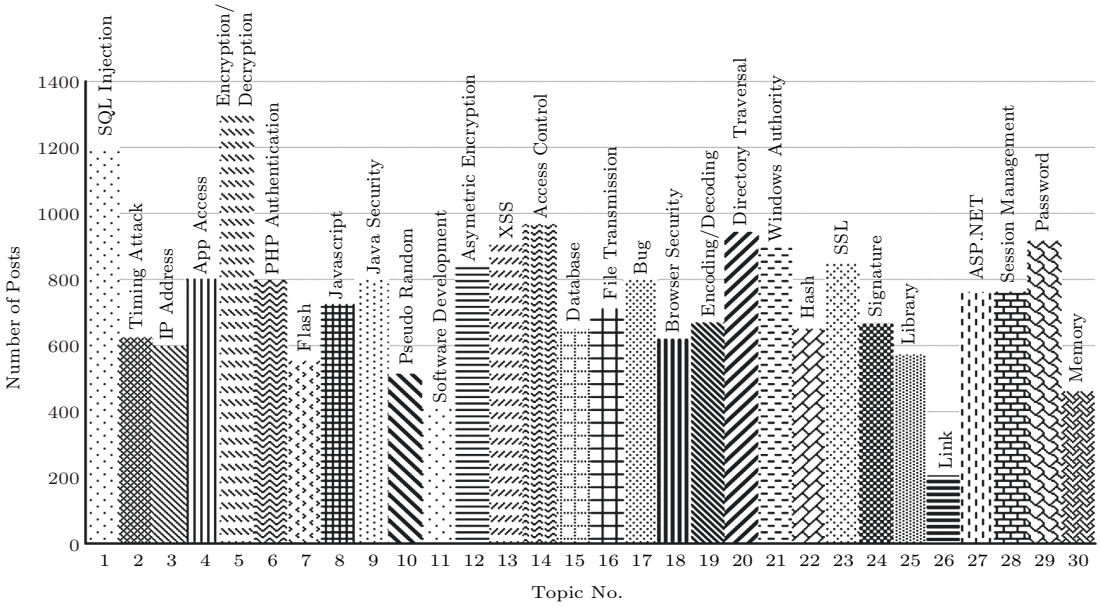


Fig.3. Statistics for each topic of questions.

presents the number of questions belonging to each category. From the figure, we notice that web security covers over half of all the security-related questions. It indicates that web security is very popular among Stack Overflow users.

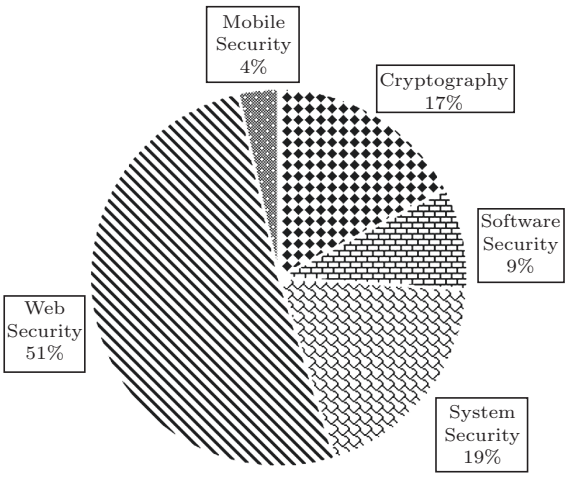


Fig.4. Statistics for each category of questions.

Summary. On Stack Overflow, security-related questions cover a wide range of topics. These topics mainly belong to five main categories, i.e., web security, mobile security, cryptography, software security, and system security. And among them, most questions are belong to web security category.

4.2 Results of RQ-2

RQ-2. *Which topics are the most popular among security-related questions?*

Motivation. Since we have known the topics of security-related questions asked on Stack Overflow, we want to go further by investigating which security-related topics are the most popular. Answer to this research question could help developers understand a general trend about the security-related questions.

Approach. To measure the popularity of a topic, we first collect all the questions related to this topic, and then we use four evaluation metrics based on the metadata of these questions, i.e., the average number of views of these questions, the average number of comments of these questions, the average number of favourites of these questions, and the average score of these questions. In the Stack Overflow Data Dump, the number of views of a question can be directly obtained from the attribute named "ViewCount". The number of comments a question has can be directly obtained from the attribute named "CommentCount" of the question post. The number of favourites that a question receives can be directly obtained from the attribute named "FavouriteCount" of the question post. The score of a question can be directly obtained from the attribute named "Score" of the question post.

Among the above four evaluation metrics, by default we use the average number of views as the main popularity evaluation metric, since it measures the average number of developers viewing the questions related to a topic. Intuitively, a popular question would attract more developers to view. Still, the other metrics also have some reference values to estimate the popularity of the topics.

Results. Table 4 presents the four evaluation metrics indicating the popularity of the topics. We notice that "Password", "Hash", "Signature" and "SQL Injection" are the top four most popular topics. "Password" is a classic topic in security and has been used in a number of applications, and "Signature" is a recently hot topic and is used in more and more applications. "Hash" is a related concept to "Password" and "Signature" and many passwords and signatures are created based on hash-related techniques. "SQL Injection" is a common problem that many developers may encounter. In addition, we can see that the average numbers of comments and favourites and the average scores of the questions belonging to the topics "Hash" and "SQL Injection" are all ranked in top, which further indicates that the two topics are very popular based on these metrics. On the contrary, "IP Address" and "Memory" seem to receive little attention. However, for the topic "Memory", it has a relatively high average number of comments (2.35) and average score (3.17).

From Table 4, on average, each security-related question receives 1 696 views, which indicates that people indeed value the security area.

Summary. On Stack Overflow, there are many popular security-related topics that people are talking about. The top four most popular topics in the security area are "Password", "Hash", "Signature" and "SQL Injection".

4.3 Results of RQ-3

RQ-3. *Which security-related topics are the most difficult to answer?*

Motivation. In the third research question, we want to investigate which security-related topics are the most

Table 4. Popularity of Each Topic

Topic Name (Corresponding Category)	Avg. ViewCount	Avg. CommentCount	Avg. FavouriteCount	Avg. Score
Password (web/system security)	2 731	1.65	1.09	3.04
Hash (cryptography)	2 691	2.35	2.39	5.44
Signature (web security)	2 494	1.30	0.96	2.41
SQL injection (web security)	2 194	2.64	2.95	6.11
Windows authority (system security)	2 170	1.28	0.94	2.36
Asymetric encryption (cryptography)	2 157	1.60	1.22	2.78
Encryption/decryption (cryptography)	2 145	1.99	0.88	2.26
Browser security (web security)	2 112	1.53	1.01	3.00
Java security (software security)	2 056	1.07	0.97	2.21
ASP.NET (web security)	1 857	0.88	1.48	2.81
Bug (software security)	1 743	1.95	0.55	2.12
Session management (web security)	1 725	1.61	1.88	3.38
Database (system security)	1 663	1.50	0.78	2.17
Access control (web security)	1 594	0.86	1.23	2.36
Flash (web security)	1 584	1.49	1.90	3.26
File transmission (web security)	1 555	1.82	0.74	2.02
Encoding/decoding (cryptography)	1 509	2.57	0.48	1.59
XSS (web security)	1 468	1.69	1.31	3.21
Timing attack (web security)	1 449	2.50	2.30	5.01
Javascript (web security)	1 399	1.58	1.06	3.17
Pseudo random (cryptography)	1 391	2.59	1.16	3.62
PHP authentication (web security)	1 354	1.81	0.92	1.91
Link (web security)	1 344	1.31	0.80	2.20
Directory traversal (system security)	1 326	1.85	0.68	1.87
Software development (software security)	1 321	1.44	0.92	2.89
App access (mobile security)	1 317	1.09	1.59	3.15
Library (system security)	1 303	1.70	1.22	3.21
SSL (web security)	1 251	1.35	0.72	2.08
Memory (system security)	1 091	2.35	1.52	3.17
IP address (web security)	887	1.28	0.79	2.23
Average value	1 696	1.69	1.21	2.90

difficult to answer. The answer to this research question can help developers value the difficult questions so that they can spend more time solving those questions. Particularly, if a topic is both popular and difficult, the topic should receive much more attention.

Approach. To measure the difficulty of a topic, we first collect all the questions related to this topic, and then we use two metrics based on the metadata of these questions, i.e., the average time span of these questions which get accepted answers, and the ratio of the average number of answers these questions receive to the average number of views these questions receive.

To compute the time span of a question, we take the creation date of a question as the starting time and the creation date of the corresponding accepted answer as the ending time. The two creation dates can be obtained from the attribute named "CreationDate" of a question and its accepted answer post respectively, where the accepted answer post can be indexed by the attribute named "AcceptedAnswerId" of the question post. Intuitively, the more time taken for an answer of a question to be accepted, the more difficult the question likely to be.

For the second metric (i.e., PD), we first obtain the number of answers and the number of views from the attributes named "AnswerCount" and "ViewCount" of each question in a topic, respectively. And then we compute their average values (i.e., $Avg.Answer$ and $Avg.View$). Finally, PD of a topic is computed as:

$$PD = \frac{Avg.Answer}{Avg.View} \times 100\%.$$

In general, if a question has a large number of views but a small number of answers, it means that only a few people can answer the question among the many viewers. Thus, we use PD to measure the difficulty of a topic, and the smaller the score is, the more difficult a question in this topic is.

Results. Table 5 presents the average time span, the average answer count, and PD score for the 30

topics. We find that different topics vary a lot in the average time span needed for an accepted answer. In particular, the problems about "Windows Authority", "ASP.NET" and "Browser Security" are the top three most difficult questions, which on average need over half a month for an answer to be submitted and accepted (18.03, 17.57 and 16.15 days respectively). On the contrary, the problems about "SQL Injection" and "Directory Traversal" are relatively easy to answer, which on average need only 2∼3 days for an accepted answer.

Table 5. Difficulty of Each Topic

Topic Name (Corresponding Category)	Average Time Span (Days)	PD (%)
Windows authority (system security)	18.03	0.09
ASP.NET (web security)	17.57	0.10
Browser security (web security)	16.15	0.10
Link (web security)	13.51	0.14
Password (web/system security)	13.07	0.09
Java security (software security)	12.56	0.09
App access (mobile security)	12.54	0.13
Signature (web security)	12.24	0.07
Bug (software security)	11.95	0.09
Access control (web security)	11.89	0.12
Library (system security)	11.31	0.18
Asymetric encryption (cryptography)	10.91	0.09
File transmission (web security)	10.35	0.14
Memory (system security)	9.40	0.22
Pseudo random (cryptography)	9.39	0.19
Javascript (web security)	8.62	0.16
Session management (web security)	8.34	0.13
Hash (cryptography)	8.28	0.09
Flash (web security)	7.93	0.17
Encryption/decryption (cryptography)	7.83	0.09
XSS (web security)	7.78	0.16
Software development (software security)	7.43	0.15
IP address (web security)	7.40	0.26
Database (system security)	5.63	0.14
SSL (web security)	5.61	0.16
Encoding/decoding (cryptography)	5.15	0.14
Timing attack (web security)	4.87	0.22
PHP authentication (web security)	4.25	0.17
Directory traversal (system security)	2.77	0.18
SQL injection (web security)	2.40	0.13

When considering the PD score, we notice the PD scores of "Windows Authority", "ASP.NET" and "Browser Security" are 0.09, 0.10 and 0.10 respectively, which are small and consistent with the first metric. Moreover, we find that "Signature", with its time span being over 10 days, has the smallest PD score of 0.07, which indicates that it is also very difficult to answer.

Finally, we pick out topics whose time span is more than 10 days and PD is less than 0.10 as the difficult topics, and in total, we find eight topics, i.e., "Java Security", "Asymetric Encryption", "Bug", "Browser Security", "Windows Authority", "Signature", "ASP.NET" and "Password".

Summary. The top eight difficult security-related topics are "Java Security", "Asymetric Encryption", "Bug", "Browser Security", "Windows Authority", "Signature", "ASP.NET" and "Password".

5 Discussion

5.1 Implications

5.1.1 For Researchers

Our large-scale empirical study provides an overall view about security-related topics that developers care about. We also highlight the most popular and difficult topics. The top eight most difficult security-related topics are "Java Security", "Asymetric Encryption", "Bug", "Browser Security", "Windows Authority", "Signature", "ASP.NET" and "Password". It would be interesting for researchers to develop techniques to help developers answer these questions. Domain-specific automated question answering is an interesting research topic that we encourage future research to focus on.

5.1.2 For Educators

We can see that web security is the category that has the most questions, which accounts for over half of all the security-related questions on Stack Overflow, which indicates that developers heavily need solutions and answers to the questions about web security. Therefore, educators could pay more attention to web security. In addition, they could pay more attention to the popular topics in security. They can publish more books and elaborate them in detail using more chapters to make learners have deeper understanding about them, and they can also write blogs to answer questions about them.

5.1.3 For Practitioners

The top eight difficult security-related topics are "Java Security", "Asymetric Encryption", "Bug", "Browser Security", "Windows Authority", "Signature", "ASP.NET" and "Password". Therefore, project managers may want to assign tasks of different levels of difficulty to different developers. For example, they can assign a task about "Java Security" to a senior developer and assign a task about "XSS" to a junior devel-

oper. The proper assignment can help the development process.

5.2 Threats to Validity

There are several threats that may potentially affect the validity of our study. Threats to internal validity relate to errors in our experiments. We have double checked our implementations and all the experimental results. The question extraction phase may miss several security-related questions due to our tag-based extraction approach.

Threats to external validity relate to the generalizability of our results. We have conducted the empirical study on 30 054 security-related questions on Stack Overflow. In the future, we plan to reduce the threats further by investigating more question and answer websites and ensure the generalizability of our conclusions.

6 Related Work

In this section, we briefly review related studies. We first review some previous studies based on Stack Overflow. Next, we describe some security-related studies in software engineering. Finally, we introduce several studies which leverage topic models.

6.1 Study on Stack Overflow

There are some recent empirical studies on Stack Overflow[1-2,7-8,19]. Barua *et al.* conducted a large empirical study on all the posts on Stack Overflow[1]. They used LDA to analyze the topics and trends of what developers talk about. Rosen and Shihab narrowed down the research scale by specifically studying mobile-related questions on Stack Overflow[2]. They also applied LDA to the dataset to investigate the topics mobile developers are interested in. Linares-Vásquez *et al.* performed an exploratory analysis of mobile-development issues using Stack Overflow[8]. They employed topic model to extract the main discussion topics from more than 400K mobile-development related questions. Beyer and Pinzger manually analyzed 450 Android-related posts on Stack Overflow and found that the most common question types are "How" and "What"[7]. They also found the dependencies between question types and problem categories. Nadi *et al.* performed an empirical investigation into the obstacles Java developers face with cryptography APIs, through triangulating data including top 100 Java cryptography related questions on Stack Overflow[19]. They identified nine main topics related to cryptography and the results suggest that developers do face difficulties using cryptography.

There are many other studies which leverage data on Stack Overflow[5-6,20-28]. Nie *et al.* proposed a novel technique, which leverages crowd knowledge from Stack Overflow, to improve the performance of code search algorithms[22]. Jiang *et al.* proposed a more accurate model, which also leverages crowd knowledge from Stack Overflow, to find the exact tutorial fragments explaining APIs[23]. Xia *et al.* proposed a tool called TagCombine that automatically recommends tags for question and answer sites, such as Stack Overflow[5]. They found TagCombine has better performance than the state-of-the-art technique TagRec. In later work, Wang *et al.* proposed another tag recommendation tool named EnTagRec which leverages historical tag assignments[6]. They found EnTagRec improves the performance of TagCombine further. Li *et al.* conducted an empirical study with 24 developers to investigate the needs and challenges developers face when performing development tasks[20]. They found that developers often refer to question and answer sites such as Stack Overflow, in which they search for useful information to solve various development problems. Bajaj *et al.* conducted a study on web development related posts on Stack Overflow[21]. They concluded several points about common challenges and misconceptions among web developers.

Our work is related to but different from the above studies. In this paper, we perform an empirical study on security-related posts on Stack Overflow.

6.2 Security in Software Engineering

In mobile security, malware detection received a lot of attention[29-32]. Avdiienko *et al.* presented MUDFLOW that uses machine learning techniques to mine Android application collections for malicious data flow patterns[29]. Malware can be identified by their malicious data flow patterns. Gorla *et al.* clustered Android apps by description topics and identified outliers by API usage within each cluster[30]. The outliers are the applications whose actual behavior would be unexpected given their description. Huang *et al.* used the text associated with GUI elements to detect whether the claimed behavior matches the actual behavior of the application[31]. Kirat and Vigna proposed an automatic technique MALGENE for extracting analysis

evasion signatures[32]. They leveraged a combination of data mining and data flow analysis techniques to automatically identify evasive behavior in the call events, as more and more malware can now be aware of the presence of the analysis environment (in order to evade detection).

In web security, researchers proposed several approaches to prevent security attacks[33−34]. Parameshwaran *et al.* proposed a technique to generate secure patches that replace unsafe string interpolation with safer code[33]. Fazzini *et al.* proposed an automated technique named AutoCSP to retrofit CSP to web applications[34].

6.3 Studies Leveraging Topic Model

The most related studies to ours are the recent study by Panichella *et al.*[12]. Panichella *et al.* introduced a novel solution named LDA-GA to solve software engineering tasks more effectively[12]. They used GA to search for a near optimal configuration for LDA, which leads to better performances on different software engineering tasks. In our paper, we use their algorithm as a sub-step to determine a proper number of categories for all the security-related questions.

There are also a large number of software engineering studies that have leveraged topic model[35-37] to achieve their functionality. For example, Nguyen *et al.* proposed an automated approach called BugScout to help developers reduce buggy code locating efforts by narrowing the search space of buggy files[35]. They developed a specialized topic model to correlate bug reports and the corresponding buggy files via their shared topics. In a later work, Nguyen *et al.* introduced a novel approach called DBTM, which again leverages topic model, to detect duplicate bug reports[36]. Their approach that combines both information retrieval and topic modeling techniques has taken the advantages of both IR-based features and topic-based features. Lukins *et al.* presented a static LDA-based technique for automatic bug localization[37]. Their study shows that the performance of the LDA-based technique is affected neither by the size of the software system nor by the stability of the source code base.

7 Conclusions

In this paper, we conducted a large-scale study by specifically investigating security-related questions on Stack Overflow. We first used two heuristics to extract security-related posts from Stack Overflow. And then we used an advanced topic model, LDA tuned using GA, to cluster different security-related questions based on their texts. After obtaining different topics of security-related questions, we used their metadata to make various analysis. We found that security-related questions on Stack Overflow cover a wide range of topics. These topics mainly belong to five main categories, i.e., web security, mobile security, cryptography, software security, and system security. And among them most questions are about web security. In addition, we found that the top four most popular topics in the security area are "Password", "Hash", "Signature" and "SQL Injection", and the top eight most difficulty security-related topics are "JAVA Security", "Asymetric Encryption", "Bug", "Browser Security", "Windows Authority", "Signature", "ASP.NET" and "Password", which need more attentions.

In the future, we plan to investigate security-related questions by combing more information of their corresponding answers on Stack Overflow. Also, we plan to investigate security-related posts on other software Q&A websites.

References

[1] Barua A, Thomas S W, Hassan A E. What are developers talking about? An analysis of topics and trends in stack overflow. *Empirical Software Engineering*, 2014, 19(3): 619-654.

[2] Rosen C, Shihab E. What are mobile developers asking about? A large scale study using stack overflow. *Empirical Software Engineering*, 2016, 21(3): 1192-1223.

[3] Treude C, Barzilay O, Storey M A. How do programmers ask and answer questions on the web? NIER track. In *Proc. the 33rd International Conference on Software Engineering* (*ICSE*), May 2011, pp.804-807.

[4] Mamykina L, Manoim B, Mittal M, Hripcsak G, Hartmann B. Design lessons from the fastest Q&A site in the west. In *Proc. the 29th SIGCHI Conference on Human Factors in Computing Systems*, May 2011, pp.2857-2866.

[5] Xia X, Lo D, Wang X Y, Zhou B. Tag recommendation in software information sites. In *Proc. the 10th Working Conference on Mining Software Repositories*, May 2013, pp.287-296.

[6] Wang S W, Lo D, Vasilescu B, Serebrenik A. EnTagRec: An enhanced tag recommendation system for software information sites. In *Proc. the 30th International Conference on Software Maintenance and Evolution* (*ICSME*), September 2014, pp.291-300.

[7] Beyer S, Pinzger M. A manual categorization of Android app development issues on stack overflow. In *Proc. the 30th International Conference on Software Maintenance and Evolution* (*ICSME*), September 2014, pp.531-535.

[8] Linares-Vásquez M, Dit B, Poshyvanyk D. An exploratory analysis of mobile development issues using Stack Overflow. In *Proc. the 10th Working Conference on Mining Software Repositories*, May 2013, pp.93-96.

[9] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 2003, 3: 993-1022.

[10] Asuncion H U, Asuncion A U, Taylor R N. Software traceability with topic modeling. In *Proc. the 32nd ACM/IEEE International Conference on Software Engineering* (*ICSE*), May 2010, pp.95-104.

[11] Thomas S W. Mining software repositories using topic models. In *Proc. the 33rd International Conference on Software Engineering*, May 2011, pp.1138-1139.

[12] Panichella A, Dit B, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A. How to effectively use topic models for software engineering tasks? An approach based on genetic algorithms. In *Proc. the 35th International Conference on Software Engineering*, May 2013, pp.522-531.

[13] Heinrich G. Parameter estimation for text analysis. Technical Report, vsonix GmbH + University of Leipzi, 2008. http://www.arbylon.net/publications/text-est.pdf, Aug. 2016.

[14] Porter M F. Snowball: A language for stemming algorithms. http://snowball.tartarus.org/texts/introduction.html, Aug. 2016.

[15] Goldberg D E, Holland J H. Genetic algorithms and machine learning. *Machine Learning*, 1988, 3(2/3): 95-99.

[16] Rousseeuw P J, Kaufman L. Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, 1990.

[17] Sander J, Ester M, Kriegel H P, Xu X W. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 1998, 2(2): 169-194.

[18] Hotho A, Maedche A, Staab S. Ontology-based text document clustering. *KI*, 2002, 16(4): 48-54.

[19] Nadi S, Krüger S, Mezini M, Bodden E. "Jumping through hoops": Why do Java developers struggle with cryptography APIs? In *Proc. the 38th International Conference on Software Engineering*, May 2016, pp.935-946.

[20] Li H W, Xing Z C, Peng X, Zhao W Y. What help do developers seek, when and how? In *Proc. the 20th Working Conference on Reverse Engineering* (*WCRE*), October 2013, pp.142-151.

[21] Bajaj K, Pattabiraman K, Mesbah A. Mining questions asked by web developers. In *Proc. the 11th Working Conference on Mining Software Repositories*, May 2014, pp.112-121.

[22] Nie L M, Jiang H, Ren Z L, Sun Z Y, Li X C. Query expansion based on crowd knowledge for code search. *IEEE Transactions on Services Computing*, 2016, PrePrints, doi:10.1109/TSC.2016.2560165.

[23] Jiang H, Zhang J X, Li X C, Ren Z L, Lo D. A more accurate model for finding tutorial segments explaining APIs. In *Proc. the 23rd International Conference on Software Analysis, Evolution, and Reengineering* (*SANER*), March 2016, pp.157-167.

[24] Zhang Y, Lo D, Xia X, Sun J L. Multi-factor duplicate question detection in Stack Overflow. *Journal of Computer Science and Technology*, 2015, 30(5): 981-997.

[25] Xia X, Lo D, Correa D, Sureka A, Shihab E. It takes two to tango: Deleted stack overflow question prediction with text and meta features. In *Proc. the 40th Annual International Computers, Software & Applications Conference* (*COMPSAC*), June 2016.

[26] Wang X Y, Xia X, Lo D. TagCombine: Recommending tags to contents in software information sites. *Journal of Computer Science and Technology*, 2015, 30(5): 1017-1035.

[27] Xu B W, Xing Z C, Xia X, Lo D, Wang Q Y, Li S P. Domain-specific cross-language relevant question retrieval. In *Proc. the 13th International Conference on Mining Software Repositories*, May 2016, pp.413-424.

[28] Xu B W, Ye D C, Xing Z C, Xia X, Chen G B, Li S P. Predicting semantically linkable knowledge in developer online forums via convolutional neural network. In *Proc. the 31st IEEE/ACM International Conference on Automated Software Engineering* (*ASE*), September 2016.

[29] Avdiienko V, Kuznetsov K, Gorla A, Zeller A, Arzt S, Rasthofer S, Bodden E. Mining apps for abnormal usage of sensitive data. In *Proc. the 37th IEEE International Conference on Software Engineering* (*ICSE*), May 2015, pp.426-436.

[30] Gorla A, Tavecchia I, Gross F, Zeller A. Checking app behavior against app descriptions. In *Proc. the 36th International Conference on Software Engineering*, May 2014, pp.1025-1035.

[31] Huang J J, Zhang X Y, Tan L, Wang P, Liang B. AsDroid: Detecting stealthy behaviors in Android applications by user interface and program behavior contradiction. In *Proc. the 36th International Conference on Software Engineering*, May 2014, pp.1036-1046.

[32] Kirat D, Vigna G. MalGene: Automatic extraction of malware analysis evasion signature. In *Proc. the 22nd ACM SIGSAC Conference on Computer and Communications Security*, October 2015, pp.769-780.

[33] Parameshwaran I, Budianto E, Shinde S, Dang H, Sadhu A, Saxena P. Auto-patching DOM-based XSS at scale. In *Proc. the 10th Joint Meeting on Foundations of Software Engineering*, March 2015, pp.272-283.

[34] Fazzini M, Saxena P, Orso A. AutoCSP: Automatically retrofitting CSP to web applications. In *Proc. the 37th International Conference on Software Engineering*, May 2015, pp.336-346.

[35] Nguyen A T, Nguyen T T, Al-Kofahi J, Nguyen H V, Nguyen T N. A topic-based approach for narrowing the search space of buggy files from a bug report. In *Proc. the 26th IEEE/ACM International Conference on Automated Software Engineering*, November 2011, pp.263-272.

[36] Nguyen A T, Nguyen T T, Nguyen T N, Lo D, Sun C N. Duplicate bug report detection with a combination of information retrieval and topic modeling. In *Proc. the 27th IEEE/ACM International Conference on Automated Software Engineering*, September 2012, pp.70-79.

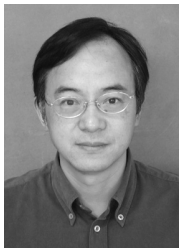
[37] Lukins S K, Kraft N A, Etzkorn L H. Bug localization using latent Dirichlet allocation. *Information and Software Technology*, 2010, 52(9): 972-990.

Xin-Li Yang is a Ph.D. candidate in the College of Computer Science and Technology, Zhejiang University, Hangzhou. His research interests include mining software repository and empirical study.

David Lo received his Ph.D. degree in computer science from the School of Computing, National University of Singapore, Singapore, in 2008. He is currently an assistant professor in the School of Information Systems, Singapore Management University, Singapore. He has close to 10 years of experience in software engineering and data mining research and has more than 130 publications in these areas. He received the Lee Foundation Fellow for Research Excellence from the Singapore Management University in 2009. He has won a number of research awards including an ACM Distinguished Paper Award for his work on bug report management. He has published in many top international conferences in software engineering, programming languages, data mining and databases, including ICSE, FSE, ASE, PLDI, KDD, WSDM, TKDE, ICDE, and VLDB. He has also served on the program committees of ICSE, ASE, KDD, VLDB, and many others. He is a steering committee member of the IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER) which is a merger of the two major conferences in software engineering, namely CSMR and WCRE. He also serves as the general chair of ASE 2016. He is a leading researcher in the emerging field of software analytics and has been invited to give keynote speeches and lectures on the topic in many venues, such as the 2010 Workshop on Mining Unstructured Data, the 2013 Génie Logiciel Empirique Workshop, the 2014 International Summer School on Leading Edge Software Engineering, and the 2014 Estonian Summer School in Computer and Systems Science.

Xin Xia received his Ph.D. degree in computer science from the College of Computer Science and Technology, Zhejiang University, Hangzhou, in 2014. He is currently a research assistant professor in the College of Computer Science and Technology at Zhejiang University. His research interests include software analytic, empirical study, and mining software repository.

Zhi-Yuan Wan is a postdoctoral research associate at Zhejiang University, Hangzhou. She obtained her Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, in 2014. Her research interests include software security, programming language and software engineering.



Jian-Ling Sun received his Ph.D. degree in computer science from the College of Computer Science and Technology, Zhejiang University, Hangzhou, in 1992. He is currently a professor in the College of Computer Science and Technology, Zhejiang University, Hangzhou. His research interests include database and web technology.