

Mining Duplicate Questions in Stack Overflow

Muhammad
Ahasanuzzaman
University of Dhaka
ahsan.du2010@gmail.com

Muhammad
Asaduzzaman
University of Saskatchewan
md.asad@usask.ca

Chanchal K. Roy
University of Saskatchewan
chanchal.roy@usask.ca

Kevin A. Schneider
University of Saskatchewan
kevin.schneider@usask.ca

ABSTRACT

Stack Overflow is a popular question answering site that is focused on programming problems. Despite efforts to prevent asking questions that have already been answered, the site contains duplicate questions. This may cause developers to unnecessarily wait for a question to be answered when it has already been asked and answered. The site currently depends on its moderators and users with high reputation to manually mark those questions as duplicates, which not only results in delayed responses but also requires additional efforts. In this paper, we first perform a manual investigation to understand why users submit duplicate questions in Stack Overflow. Based on our manual investigation we propose a classification technique that uses a number of carefully chosen features to identify duplicate questions. Evaluation using a large number of questions shows that our technique can detect duplicate questions with reasonable accuracy. We also compare our technique with DupPredictor, a state-of-the-art technique for detecting duplicate questions, and we found that our proposed technique has a better recall-rate than that technique.

CCS Concepts

•Information systems → *Social networking sites*;

Keywords

Stack Overflow; duplicate questions; discriminative classifier

1. INTRODUCTION

Community-based question answering sites (CQA) are becoming popular due to the presence of a large volume of information that are collected through active participation of its users. While many question answering sites facilitate discussion of a wide range of topics (such as Yahoo Answers¹),

¹<https://answers.yahoo.com/>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MSR'16, May 14-15, 2016, Austin, TX, USA

© 2016 ACM. ISBN 978-1-4503-4186-8/16/05...\$15.00

DOI: <http://dx.doi.org/10.1145/2901739.2901770>

Stack Overflow differs in that it is dedicated to software developers. Since its inception in 2008, Stack Overflow has become a central hub both for asking and answering questions related to programming problems. Users can vote on questions and answers, and the vote count serves as a rough measure for the quality of posts. Stack Overflow users can earn reputation and badges for their valued contributions, and these elements were introduced, in part, to encourage user participation. Reputation also unlocks a number of privileges to Stack Overflow users including access to site moderation tools. This is an indication that the site is governed not only by user generated content but also by its active users. As of January 2016, Stack Overflow had over 4.7 million registered users, more than 11 million questions, and 17 million answers.

Stack Overflow recommends that users search previous posts before asking a new question.² This is to avoid asking a question that already has been asked and that may already have been answered. Stack Overflow also suggests links to questions whose title matches the new question. Despite these efforts the site constantly faces duplicate questions; questions that are asked to solve the same problem. When two questions are duplicates of each other, one of them will be marked as a duplicate and go through the closing process. The other question will be marked as the master. Usually the recent question will be closed as a duplicate of the older question because the older question typically contains the best answer. During our manual investigation we found that although exact duplicates are very rare, many duplicate questions exist that are asked in different ways.

Currently Stack Overflow depends on the moderators and users with high reputation to manually analyze and mark duplicate questions, which is both time consuming and tedious work. As a result many duplicate questions remain unidentified or are mistakenly marked as duplicates [22]. An automatic duplicate question detection system can alleviate these problems by recommending possible duplicates of a question. Moderators can then focus their attention on a smaller set of questions. Such a system can also suggest likely duplicates as a user types a question. This may help users to find previous posts with answers that address the same problem and reduce waiting time to receive answers. Duplicate detection systems can also be useful in other applications. For example, results of such a system can help find related questions or diversify search results [32].

Although a number of studies have been performed on

²<http://stackoverflow.com/help/asking>

Stack Overflow [7, 10, 11, 14, 15, 16, 18, 21], there has been very little research on the problem of duplicate questions. Zhang et al. [22] recently developed a tool, called *DupPredictor*, that can identify potential duplicates of a given question by considering title, description, topic and tag similarity. To the best of our knowledge, this is the only work that addresses the problem of duplicate questions in Stack Overflow. Despite the contribution of *DupPredictor*, we see a gap between their work and what we can do regarding duplicate questions. In this paper, we build on their work to characterize and improve detection of duplicate questions.

Towards the goal of developing an automated technique for detecting duplicate questions, we first conduct experiments to understand their characteristics. We then perform a manual investigation to determine why developers create duplicate questions. Based on our manual investigation, we use the BM25 scoring function [31] and select a number of similarity features between pairs of questions to create a discriminative classifier. Together they help us suggesting potential duplicates of a given question. In general, we answer the following three research questions in this paper:

- (1) Why do developers duplicate questions in Stack Overflow?
A proactive approach to manage duplicate questions is to educate developers to ask questions effectively. To shape the process we also need to understand why developers duplicate questions. This can enable the moderators to take the necessary actions to control the spread of duplications.
- (2) Can we develop an automated technique for detecting duplicate questions from a machine learning perspective?
- (3) How does the technique perform in comparison with other duplicate question detection techniques?

The remainder of the paper is organized as follows. Section 2 briefly describes previous work related to our study. Section 3 characterizes duplicate questions. Section 4 explains why developers duplicate questions in Stack Overflow. We describe our proposed technique in Section 5. Section 6 presents evaluation results. We discuss the key issues related to our study in Section 7. Section 8 summarizes the threats to validity and Section 9 concludes the paper.

2. RELATED WORK

In this section we briefly describe previous work related to our study.

2.1 Characterization and modelling of Stack Overflow

A number of studies have been performed to characterize different aspects of Stack Overflow. This includes question kinds [6], patterns of user interaction [7], women engagement [11], analysis of code examples [9], topic distribution [12], web related discussion [15], personality traits of users [14], and distribution of developers in asking and answering questions [13]. A number of recommendation systems and predictive models are developed using Stack Overflow data to answer a wide range of questions. For example, Bacchelli et al. [10] integrate the crowd knowledge stored in the IDE by creating an Eclipse plugin, called *Sehawk*. Ponzelli et al. [16] develop an Eclipse plugin, called

Prompter, that given a context in the IDE automatically retrieve related discussions from Stack Overflow. Ponzelli et al. [19] leverages simple textual features, readability metrics and community related aspects to improve detection of low quality posts by reducing the size of the review queue. Correa and Sureka [21] study the characteristics of closed questions in Stack Overflow. In another study, they characterize the deleted questions and develop a predictive model to detect deleted questions at the time of question creation [18]. Chang and Pal [20] develop a recommendation model by considering user availability, compatibility and expertise to route questions to a groups of users, who will be more willing to participate and can provide better answer. However, none focus on the characterization and detection of duplicate questions in Stack Overflow.

2.2 Duplicate document detection

A number of techniques have been developed for detecting near-duplicate documents. For example, Broder et al. [4] uses shingles and Manku et al. [3] uses a finger printing technique developed by Charikar [5], also known as SimHash, to detect near duplicate web pages. Hajishirzi et al. [2] propose a technique that represents each document as sparse k-gram vector where weights are learned to optimize a similarity function. This improved similarity measure can be used to detect near duplicate documents. These techniques assume that two near duplicate documents differ in a small portion. However, this assumption is not correct for duplicate questions in Stack Overflow. Tao et al. [32] develop a framework that leverages syntactical characteristics, semantic similarity and contextual information to detect duplicate tweets. Both their data set and their approach differ from ours. To the best of our knowledge, the most relevant work to our study is that of Zhang et al. [22]. They propose a technique, called *DupPredictor*, which can identify possible duplicates of a new question by considering its title, textual content of the body, topic and tag similarity. Our work differs from theirs in a number of ways. First, we use a discriminative classification model to detect duplicate questions whereas *DupPredictor* uses a combine similarity score for duplicate detection. Second, we use a large number of features compared to their technique. Finally, we study the characteristics of duplicate questions and also investigate why they occur. None of these approaches are present in their work. Our data set is also considerably larger than what they used in their study.

2.3 Duplicate bug reports

Stack Overflow questions are similar to bug reports in that both contain unstructured data and both suffer from duplicates. A number of techniques have been developed to detect duplicate bug reports. They use vector space models [23], execution trace information [26], probabilistic retrieval models [27], domain knowledge and context information [25] to improve detection of duplicate bug reports. The characteristics of a bug report is different than a question in Stack Overflow. Bug reports contain a number of fields that provide both textual and non-textual information, but those fields are not present in Stack Overflow questions. As well, a bug report does not contain tags like those that are present in Stack Overflow questions. Due to such differences it is worthwhile to further investigate duplicate question detection in Stack Overflow.

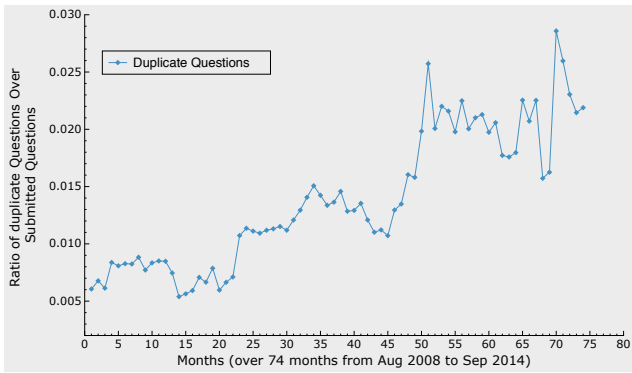


Figure 1: Monthly ratio of duplicate questions to total number of questions (Aug 2008-Sep 2014)

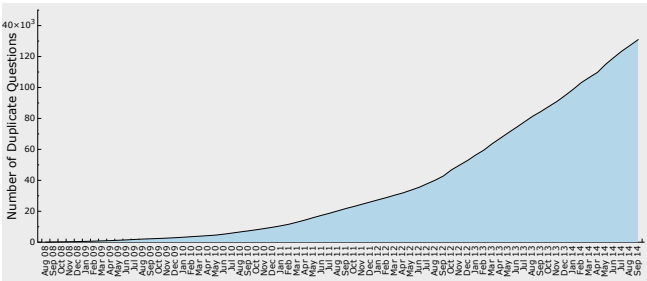


Figure 2: Cumulative distribution of duplicate questions (Aug 2008-Sep 2014). There is a sharp rise in duplicate questions after Sep 2012

3. CHARACTERIZATION OF DUPLICATE QUESTIONS

This section presents different characteristics of duplicate questions in Stack Overflow.

3.1 Data Set

We collected the Stack Overflow data dump that had been used for the mining challenge in MSR 2015 [29]. This data includes the publicly available history of question and answer posts, tags, votes on the posts, and the reputation of the users from August 2008 to September 2014. We downloaded the available eight database files which were more than 60GB in total size.

Duplicate questions in Stack Overflow are marked with a special marker. To collect these questions we parsed the *posts* database file. We extracted those questions that were closed as duplicate and appended '[Duplicate]' to their titles. Using this approach we identified 130,888 (0.13M) duplicate questions for the period August 2008 to September 2014. Next, we identified the questions to which these duplicate questions were linked. We called these questions the masters of the duplicated ones. We parsed the *postlinks* and *posts* database files in order to collect the master questions. We identified a total of 90,245 master questions.

3.2 Duplicate questions over time

We performed temporal trend analysis of duplicate questions on Stack Overflow. Figure 1 shows the ratio of duplicate questions to total number of questions in each month

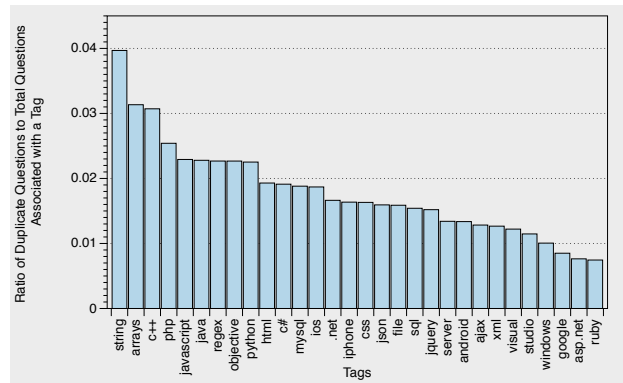


Figure 3: Distribution of duplicate questions for the most popular tags

over a 74-month period between August 2008 and September 2014. The figure shows that the number of duplicate questions is increasing over time. We observe that in Stack Overflow on average 4-5% questions are duplicated. This number can be higher as many duplicate questions are likely to be unidentified. We also notice an abrupt increase in the number of duplicate questions after September 2012. The reason behind this is that the popularity and activity of Stack Overflow increased at that time and more questions were posted. Figure 2 shows the cumulative distribution area chart for duplicate questions over a 74-month period between August 2008 and September 2014. The chart confirms that there is a sharp increase in duplicate questions after September 2012. Although Stack Overflow recommends that its users search and read the questions before posting to decrease accidental duplicate questions, we observe a continuous flow of duplicate questions to the site. We investigated possible reasons for this and discuss our results in Section 4.

3.3 Tag based duplicate question analysis

Stack Overflow users can attach tags to questions (short labels not more than a few words long) effectively linking them and creating a topic-related structure. We found a total of 38,205 tags in our data set. Among these tags *java*, *python*, *csharp*, *jquery* and *html* are among the very popular. We extracted the tags used in duplicate questions to identify which tags are most related to duplicate questions. Of these we selected a total of thirty of the most popular tags. Figure 3 shows the ratio of duplicate questions to total questions for these tags. Among the various tags, the *java* tag has the highest number of duplicate questions (17%); however, the ratio of duplicate questions to total number of questions is the highest for the *string* tag.

3.4 Time takes to close duplicate questions

Community users with at least a 15 reputation can flag a question as a duplicate that needs to be reviewed by the moderator. As well, users with a reputation of more than 3000 can close a duplicate question. We collect the time duration between a question being created and then marked as closed as a duplicate. To obtain the information, we parsed the *postlink* database. Whenever, a post is marked as a duplicate, this event is included in the *postlink* database and the field called *linkTypeId* is assigned the value 3 to indicate a duplicate link. Figure 4 shows the distribution of duplicate

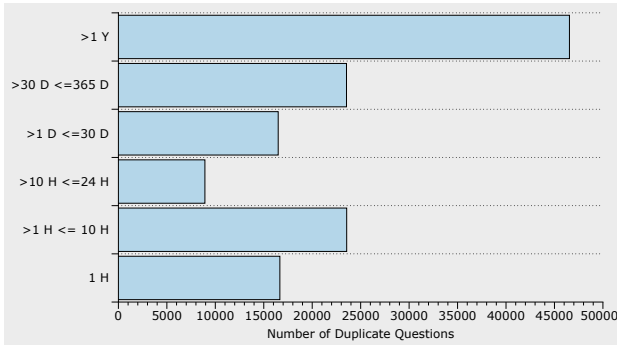


Figure 4: Time required to close duplicate questions

Table 1: Duplicate questions by user reputation.

Reputation	Users	Questions	Dupl.	Avg.
<100	43,929	320,430	50,384	6.36
100-1,000	29,469	1,258,626	42,170	29.84
1,000-10,000	15,362	2,617,606	33,653	77.78
>10,000	2,241	1,785,347	6,602	270.42

questions by closing time. From this time duration analysis, we found that almost 65% of duplicate questions took more than one day to be closed and only 29% of questions are marked closed within ten hours. Whenever a developer asks a question to get help in Stack Overflow, he may be on a tight schedule and every single second is crucial to him. If he knows that his question has already been answered then he could save time. An automatic duplicate question detection system can help in this regard.

3.5 Reputation of those who asked duplicate questions

We analyzed the reputation of those who posted duplicate questions on the Stack Overflow site. Table 1 shows how many duplicate questions are asked by users with different reputation categories. The last column of the table shows on average how many questions are asked before posting a duplicate question. Users who have the least experience (less than 100 reputation) post the most duplicate questions on the site. These are the users who are more interested in getting answer without spending much time on searching and reviewing the site. Then comes the users with minimal experience (reputation ranges from 100 to 1,000). However, the mid reputation users (1,000 to 10,000) post only 25% of the duplicate questions. Users with high reputation are very well informed and experienced enough that they post only a few duplicate questions.

4. WHY DUPLICATE QUESTIONS ARE SUBMITTED?

We were interested in finding out why duplicate questions are asked by users. Are they doing it accidentally? Are they creating duplicates because existing answers are not helpful? Do they search for duplicates or related questions prior to posting a question? We plan a manual analysis of both questions and comments to find reasons for duplication. Since it is not feasible to manually analyze all duplicate questions, we randomly sampled a total of 600 questions. To avoid

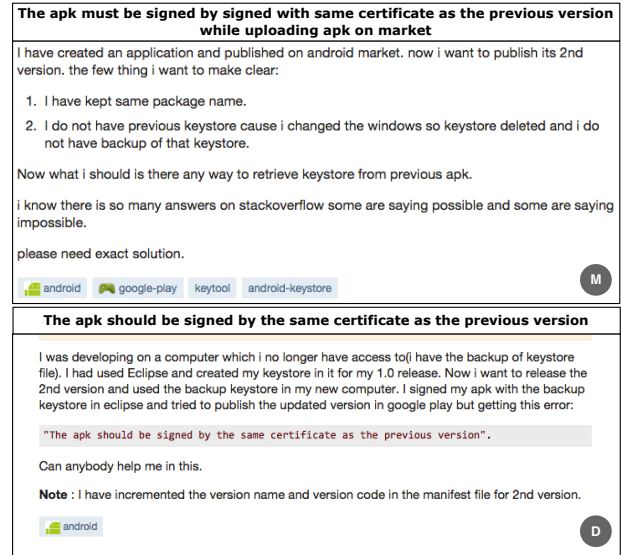


Figure 5: Not searching Stack Overflow first

bias we randomly select 100 questions from each year between 2009 and 2014, inclusive. The first two authors of this paper manually analyzed these questions and compared them with their master questions separately to find why duplicate questions were asked by users. We resolved conflicts through discussion. Our qualitative analysis identified the following possible reasons for question duplication. We include a number of examples of duplicate questions to support our discussion. In all figures, “M” denotes the master question and “D” represents its duplicate.

4.1 Not searching Stack Overflow first

Many new users to Stack Overflow do not know the single most important rule in this community – before asking a question they should search first. When a user types a question, Stack Overflow also offers a list of similar questions. It is also suggested to carefully review those questions before asking a question. Inexperienced new users or those posting a question for the first time want to quickly obtain an answer to their question without spending much time searching. Sometimes experienced users also neglect to search for related questions. Not searching first can lead to duplications. For example, Fig. 5 shows an example of a duplicate question. It is likely that the user did not search for the question first, because a question with a similar title already existed with answers.

4.2 Titles do not match

Sometimes the title of a duplicate question does not match the master question. Thus, when a user searches for a similar question, they should also review the question body and answers to identify questions that already have answers to the same problem. Many users asked duplicate questions, since the titles did not show a clear relation between them. In Fig. 6, the asker of the duplicate question (‘D’) overlooked the master question (‘M’), which could well satisfy their information needs, since the titles do not clearly indicate a relation between them. Therefore, they asked the question again.

Why cast null to object

I found a spot in some code I'm working on where `null` is cast to `Object` as it is passed to a method.

Why would this be done?

I am aware of [this question](#) which deals with overloaded methods, and using the cast to determine which version of the method to call.

But if the cast were not performed, wouldn't an overloaded method with a parameter typed as `Object` be chosen over any other matching version of the method if the method is called with a `null` argument? So what else does the cast accomplish?

M

What is the difference between "(object)null" and "null" in Java?

Take a look at the following example:

```
class nul
{
public static void main (String[] args)
{
System.out.println (String.valueOf((Object)null));
System.out.println (String.valueOf(null));
}
}
```

The first `println` writes `null` but the second throws a `NullPointerException`.

Why is only the second line worth an exception? And what is the difference between the two `null`s? Is there a *real* `null` and a *fake* `null` in Java?

D

Figure 6: Title dissimilarity leads to duplication

How to set a timer in Java?

How to set a Timer say for 2 minutes to try to connect to a database and then exception out if there is any issue in connecting

M

Animation using a timer?

I'm trying to understand how to use one some kind of basic animation. I can use threads, but I was told by multiple people not to multi-thread in java. I think I could do something like:

```
Timer t = new Timer(10, something);
t.start();
x++;
t.end();
```

That's my basic understanding of it. Can someone link a tutorial or explain how to make a time do something every 10 seconds or longer?

D

Figure 7: Task similar but domain different resulting in a duplicate question

4.3 Domain difference despite task similarity

One of the reasons for posting duplicate questions is there may be a difference in domain or application category. Users may ask a question about a task that matches a previous question but the question is for a different application or domain category. As a result, users may overlook those questions.

Fig. 7 shows an example. Here, question “D” is the duplicate of “M”. In both cases the objective is to create a timer. In the master question the user wants to create a timer that timeouts automatically after two minutes when exceptions occur while establishing a connection to a database. In the duplicate question the user wants to use a timer to create an animation. Although the title partially matches, it appears that the master question addresses the problem for a different domain unless the answers are explored. As a result, it is difficult for a user to find answers to their question.

4.4 Descriptive and difficult to comprehend

Questions having long description with or without source code examples are difficult to read and require more time

How to disable specific control in audio player HTML5

I just want to show the audio controls but prohibits the user to drag the button because I don't want the user to skip the song. How to make this work?


M

How to hide position slider in HTML5 audio tag? Hide "fake" input element inside audio element

I have built an internet radio at [radio.meteor.com](#) (code at [GitHub](#)). I'm using an HTML5 audio tag.

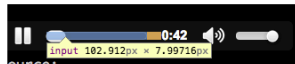
```
<audio preload id="player" controls
<source src="{{srcUrl}}" />
</audio>
```

[link to source](#)



Since it's only playing streams there's no point in the slider showing up. Is there a good way to hide the slide? Ideally not by putting another div on top to make the entire area black.

In Chrome it even seems like there's an input element inside the audio element. Isn't there a way to hide this input element? Would be great even if it only works inside Chrome.



D

(Yes, this question is somewhat duplicate of [How to disable specific control in audio player HTML5](#) - but that question is a lot less precise and this question is more useful to both people and more likely to be found.)

Figure 8: Concise question can cause duplication

to comprehend. In addition, long descriptions of a procedure or a system and code example may confuse a reader unless reviewed carefully. They lose interest in that particular question, which could have had answers to solve the problem.

For example, the question with id 6060998 (asked in 2010) has already an accepted answer. However, this question is very descriptive (73 lines) with lots of code examples. Therefore, the asker of another question (id 11757587, asked in 2012) overlooked the question and created a question for a similar problem. This question is identified as a duplicate after six hours. Although the answer could have been found within a few seconds, the user waited almost 6 hours because of the long description and difficulty in comprehending the question.

4.5 Too concise to properly understand

Sometimes questioners give very concise description of their problem without giving any supporting material or example. Despite the lack of description or supporting material these questions may receive answers. When other developers face the same problem they also search for questions with similar problem in Stack Overflow. But it may be difficult for them to connect his problem with the previous question because those questions did not describe the problem clearly. It may also be the case that the answers are not clear, perhaps lacking a proper explanation and so lead to the user asking the question again. Figure 8 shows an example of this phenomenon. Here the user that asked a duplicate question found the original question to be very short and unclear. Thus, they asked the question again with a better explanation, including providing some screen shots of their problem. As the user states:

Yes, this question is somewhat duplicate of [How to disable specific control in audio player HTML5](#) - but that question is a lot less precise and this question is more useful to both people and more likely to be found.

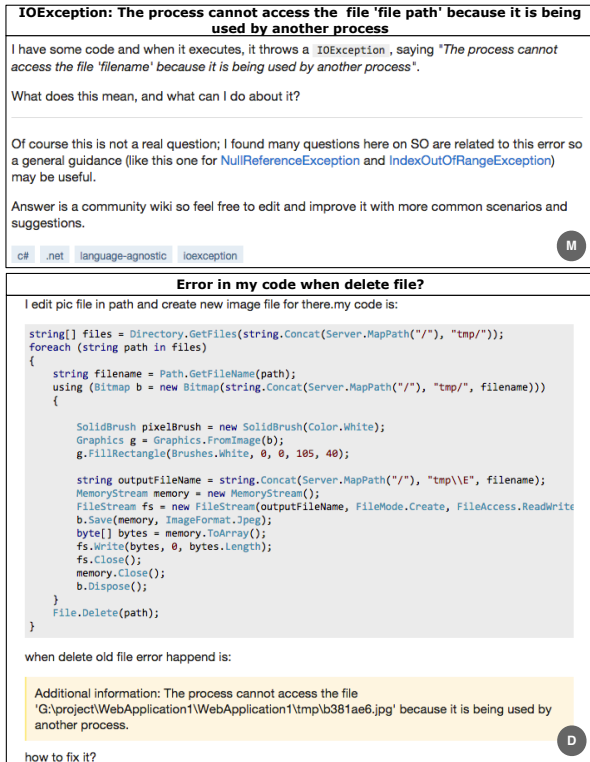


Figure 9: Lack of knowledge about a problem can lead to duplicate questions

4.6 Lack of knowledge about the problem

Some questions follow the pattern: “I am using ... and an error occurred.” It may be the case that the problem has already been asked and answered but the user could not find the relevant question because they lack the knowledge of what caused the problem. Fig. 9 provides an example. Although the question has already been answered, the user asking question “D” could not find the solution because they did not know what caused the problem in their code. Therefore, they asked the question again. The question was edited later for clarification and marked as a duplicate.

4.7 Lack knowledge of terminology/buzzwords

Users who are either new learners or have a lack of knowledge about a programming language or an API sometimes ask questions using improper terminology. Although they may read other questions before posting theirs, they misjudge many similar questions because they lack knowledge of the terminology of that particular topic. Hence, they post duplicate questions. For example, a questioner wants to save objects in a file so that they can be read again, which is known as serialization in the Java programming language. The question has already been answered. Because of the lack of knowledge about serialization they could not recognize the post that contained the answer to their problem.

We analyzed the distribution of the 600 randomly selected duplicate questions and categorized each of them. Although, a number of them may fall into different categories, we select a single category for each question. The table 2 shows the

Table 2: Reasons behind duplicate questions

Reasons	Percent
Not searching Stack Overflow first	35.5
Lack of knowledge about the problem	19.0
Titles do not match	17.0
Too concise to properly understand	12.0
Domain difference despite task similarity	7.0
Lack knowledge of terminology or buzzwords	5.5
Descriptive and difficult to comprehend	4.0

percentage of the questions in each category. In most cases new users of Stack Overflow ask questions without searching or reviewing related questions. Thus, a number of questions have fallen in that category. Reading only the title or even reading questions is not enough to find posts that may contain related answers. It is also required to review answers to the posts. Automatically suggesting duplicate questions will save the time of both question askers and moderators, and can be a useful feature of Stack Overflow.

5. PROPOSED TECHNIQUE

In this section we describe our proposed technique, called *Dupe*, for duplicate question detection. The problem of duplicate question detection can be viewed as a binary classification task. In a classical classification problem a model is generated by considering a collection of features of duplicate questions and their masters. Given a pair of questions and their corresponding feature values, the model can decide whether the pair of questions are duplicates or not. The model can also tell us the level of duplication. When a user asks a new question, we need to pair it with all of its possible duplicate candidate questions in Stack Overflow to decide which pair of questions are actually duplicates of each other. Since a large number of question pairs can be classified as duplicates, we cannot depend only on a binary classifier. We need to sort those questions using the level of duplication, which is a probabilistic value that can tell us how likely the new question is a duplicate of another question in Stack Overflow. We can then present the ranked result to a user.

Figure 10 summarizes the phases of *Dupe*. The technique consists of three different phases. In the first phase we preprocess questions and prepare them for feature extraction. In the second phase, we collect different features for each pair of questions and generate the binary classification model. The third phase is responsible for duplicate question detection using the previously trained classifier and presents a ranked list of results. We briefly describe each phase in the following section.

5.1 Preprocessing

After extracting each question from our data set we applied a preprocessing method consisting of the following steps:

- (1) For each question, we remove stop words from both title and body using a comprehensive list of stop words. We then separately collect the title, body content and tags.
- (2) Users can ask question using different forms of a word, such as organize, organizes and organizing. Moreover, there are families of derivationally related words with

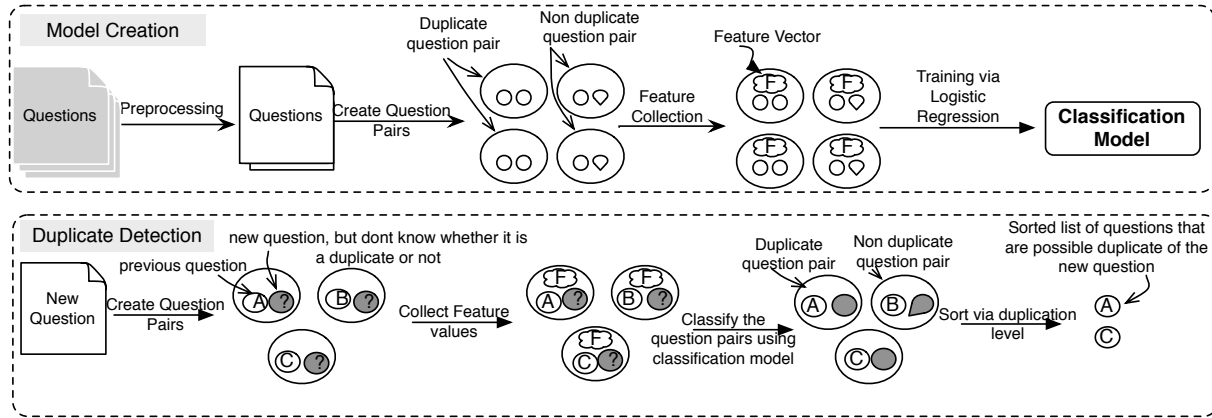


Figure 10: Classification model creation and duplicate detection via the model

similar meaning. To avoid unnecessary mismatches we perform stemming on textual features (title and body content) using Porter’s algorithm. Stemming removes the common morphological and inflexional endings from words.

- (3) In Stack Overflow tags are organized in a master-synonym relationship. A tag synonym is a tag that has exactly the same meaning as some other tag. A tag can be a subset of another tag that is also considered synonymous. We replace every occurrence of a synonym tag with its master.
- (4) The next step is to identify the duplicate questions and link them to their master. For each question we check the *LinkTypeId* of the *postlinks* table to determine whether the question is a duplicate. If the question is identified as a duplicate of another question we check the *RelatedPostId* to identify the master question. Similar to the duplicate bug report detection techniques [24], we also use the bucket data structure to store questions. Each bucket consists of a master and all of its duplicate questions.

5.2 Generate a discriminative classifier

5.2.1 Feature collection

In this section we describe briefly the set of features we collected for each of questions for the task of duplicate question detection. These features are selected based on our manual analysis of duplicate questions. We first describe five different features that we collect for the title of a pair of questions. However, we also collect the same features three more times considering title-body (title of the first question and body of the second question), body-title and body-body. It may be the case that titles do not match but the terms appear in one question of a title might appear in the body of another question or vice-versa. For tag and source code, we only determine the cosine similarity value between a pair of questions. The five selected features are as follows.

- (1) **Cosine Similarity Value:** The cosine similarity is a measure that calculates the cosine of the angle between two documents on the vector space model. This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space because we are considering only the

angle between two documents represented as vectors. The smaller the angle between the documents is, the higher they are similar.

- (2) **Term Overlap:** This feature determines the number of words common between a pair of titles and has been calculated by using the the text similarity function. If the first title contains t_1 words and the second title contains t_2 words then we calculate word overlapping in proportion to the size of t_1 and t_2 as follows: $\frac{2|t_1 \cap t_2|}{|t_1 \cup t_2|}$. This value is also normalized and is in the range of $[0, 1]$.
- (3) **Entity Overlap:** Named Entity Recognition (NER) locates and classifies elements in text into predefined categories such as the organization name, expression of times, person name, etc. If the two titles are similar they should contain the same entities. We use the Stanford Named Entity Recognizer³ to collect the entities and then determine the overlapping using the Jaccard coefficient.
- (4) **Entity Type Overlap:** In this case, we collect the types of entities and determine their overlapping.
- (5) **WordNet Similarity:** In WordNet⁴ nouns, verbs, adjectives and adverbs are grouped into cognitive synonyms, also known as synsets, each expressing a different concept. These synsets are interlinked with conceptual semantic and lexical relations. A number of techniques are available to determine the semantic relatedness over a set of terms leveraging synsets. We use the algorithm described by Hrist and Stonge [30] to determine semantic relatedness. Instead of implementing the algorithm we use WS4J, a Java library for calculating semantic relatedness or similarity and it contains an implementation of that algorithm.

5.2.2 Train the classifier

To train the classifier we need to generate a training data set. For each pair of questions we collect all the feature values as described in the previous section. These are the predictors or independent variables of our classification model. The target or response variable has two classes. It tells

³<http://nlp.stanford.edu/software/CRF-NER.shtml>

⁴<https://wordnet.princeton.edu/>

whether the pair of questions are duplicates of each other (positive examples) or not (negative examples). To generate the positive examples we take a pair of questions in such a way that one is the master and the other question is its duplicate. To generate the negative examples we randomly select pairs of questions in such a way that are not duplicates of each other. The number of negative examples can be much higher than the positive examples. To avoid bias, we use the same number of positive and negative examples to train our classification model. We use logistic regression to derive our classification model. Logistic regression is a discriminative classification model that operates on the real valued vector input. It is also a probabilistic classifier that given a test case generates the class predictive probability which is the likelihood of the test case to belong to that class. In our case for a positively predicted test case the class probability tells the likelihood of being duplicates of each other. We are more interested in knowing probability values instead of the result of the binary classification.

5.3 Duplicate detection using the model

To detect whether a question, q is a duplicate or not we need to compare q with all other questions in Stack Overflow. To do that we follow the following procedure.

- For each other question (x) in Stack Overflow we create a pair of questions, (q, x) . If q is a duplicate question, x would be the master question.
- Next, we determine the feature values of that question pair.
- The previously trained model is now used to classify the question pair and we collect the class probability value that tells the duplication level.
- After collecting the probability value for all pairs of questions, we sort them in descending order of probability values. We then recommend the top- k questions as the possible duplicates of q .

Since there are a large number of questions in Stack Overflow and not all questions are related to each other, it would be a naive approach to compare q with all other questions. For example, a question that discusses a problem or feature of the Python programming language cannot be a duplicate of a question related to Java. We found that tags are a good starting point to find related posts to a question. Given a question q that uses t tags, we collect all those questions that have one or more tags common with q . Questions that do not contain any answers cannot be a master question in Stack Overflow. Therefore we also filter those questions that do not have any answers. The remaining set of questions are used to create question pairs with q . We call this set the possible duplicate candidates of q . Computing feature values for each pair of questions is a time consuming task. To make the process faster we use an intermediate step using the BM25 algorithm. This also helps us to filter irrelevant questions.

BM25 is a ranking function used by many search engines to rank documents based on their similarity to the search query. In our case the query is created by concatenating title, body and tags of a question. Our goal is to apply BM25 to select a small set of duplicate candidates for q and then apply the discriminative classification technique on that set.

During our manual analysis we found that not all terms in a document are equally important. When searching for the duplicates for a given question q , we need to emphasize those terms that appear in q , but that do not frequently appear in other questions in Stack Overflow. Terms of q that frequently appear in other questions do not help us to distinguish a few questions from a large collection of questions. BM25 is a bag-of-words retrieval function that gives more weight to the infrequent query terms over those that are very frequent or common in the data set. Given a question q containing terms $w_1, w_2, w_3, \dots, w_n$, BM25 calculates the score for another question d using the following equation:

$$f(q, d) = \sum_{w \in q \cap d} c(w, q) \frac{(k+1)c(w, d)}{c(w, d) + k(1 - b + b \frac{|d|}{avgdl})} \log \frac{M+1}{df(w)} \quad (1)$$

Here, $c(w, q)$ denotes the term frequency of w in q (the number of times w appears in d). $df(w)$ refers to the question frequency of term w (The number of questions word w appear). M is the total number of questions in our data set. K, b are free parameters. $avgdl$ is the average length of questions comparing with q and finally, $|d|$ denotes the length of the other question.

Here, the parameter k controls how quickly an increase in term frequency results in term-frequency saturation. The default value is 1.2. Lower values result in quicker saturation, and higher values result in slower saturation. The parameter b controls how much effect field-length normalization should have. A value of 0.0 disables normalization completely, and a value of 1.0 normalizes fully. The default is 0.75.

We investigate the effect of the BM25 transformation model to our data set by varying the ranges of k and b . We find out that setting the value $k = 0.05$ and $b = 0.03$ gives the best performance. For a given question q , we apply BM25 to select n questions that are possible duplicates of q . We select the number of n in such a way so that the selected questions always contain the duplicate question of q . After careful investigation we found that $n = 10,000$ is a good value to work with.

6. EVALUATION

This section describes the evaluation of our proposed technique (*Dupe*) with the only available duplicate question detector for Stack Overflow, called *DupPredictor*.

6.1 Evaluation Metric

To evaluate the performance of the compared techniques, we use the notion of recall rate which has been used in a number of previous studies [22]. It can be defined as follows:

$$recall-rate_k = \frac{N_{dk}}{N_t}$$

Here, $recall-rate_k$ refers to the recall of a technique within the top- k recommendations. The term N_t refers to the total number duplicate question in our test data. Finally, N_{dk} denotes the total number of detected duplicate questions within the top- k recommendations.

6.2 Experimental Setup

For evaluation we consider six groups of questions involving six different popular programming languages in Stack Overflow. To create these question groups we collect all the questions in our data set that are tagged with *java*, *c++*,

Table 3: Evaluation Results

Question Group	Technique	Recall (%)		
		Top-5	Top-10	Top-20
Java	BM25	28.33	31.14	35.12
	Dupe	38.25	44.55	53.02
	DupPredictor	30	35	41
	SO Search	19.03	26.91	-
C++	BM25	23.32	26.36	31.09
	Dupe	37.14	40.12	49.93
	DupPredictor	26	28	35
	SO Search	18.81	22.40	-
Python	BM25	26.10	30.18	36.09
	Dupe	37.20	45.41	53.22
	DupPredictor	28.15	32.28	38.74
	SO Search	17.21	25.41	-
Ruby	BM25	35.56	40.84	47.53
	Dupe	51.16	59.56	66.11
	DupPredictor	33	37	39
	SO Search	15.54	22.81	-
Html	BM25	23.94	27.11	31.22
	Dupe	37.14	39.45	50.23
	DupPredictor	25	28	34
	SO Search	16.83	26.04	-
Objective-c	BM25	26.81	31.61	37.11
	Dupe	40.61	47.88	56.35
	DupPredictor	26	28	31
	SO Search	17.73	25.52	-

python, *ruby*, *html* and *objective-c*. For each group of questions we separate the duplicates from the others. To train our proposed technique we need to create both positive and negative examples. We use 80% of the duplicate questions to create the positive examples in the training data. To avoid bias we also incorporate the same number of negative examples using randomly selected non duplicate questions. Next, We randomly select 20% of the remaining duplicate questions to test our technique.

6.3 Evaluation Results

Table 3 shows the results of our evaluation for each question group. We present the results for recall-rate@5, recall-rate@10 and recall-rate@20. From the table we can see that our proposed technique (i.e., *Dupe*) performs better than *DupPredictor* for all six different question groups. For example, for the Java questions *DupPredictor* achieves a recall-rate@20 of 41% whereas *Dupe* achieves recall-rate@20 of 53.02%, an increase of 12.02% in recall value. We also observe an increase of 9.55% at recall-rate@10 for *Dupe* over *DupPredictor*. For the other question groups and also for different recall-rates, *Dupe* consistently achieves better results than *DupPredictor*. For example, for the C++ questions, *Dupe* achieves 11.14%, 12.12% and 14.93% higher recall values at recall-rate@5, recall-rate@10 and recall-rate@20 respectively. We also observe a similar result for the other question groups. We also include results for BM25 and the results show that applying BM25 alone does not give a good recall-rate. Given a question, the Stack Overflow search engine recommends ten relevant questions. We were interested in how useful the technique is in detecting duplicate questions. However, we observe that the technique performs the worst. This may be due to the fact that the target of the technique is different from ours.

7. DISCUSSION

In this section we discuss questions related to our study.

7.1 Which features are most important?

We consider a number of features to build our discriminative classification model. However, all of those features may not contribute equally to detect duplicate questions. To determine the impact of different features we run experiments using questions that are tagged with *Ruby*. Table 4 shows the recall-rate@5, recall-rate@10 and recall-rate@20 when we build our discriminative classification model using different sets of features. All other settings for our technique remain the same. From the table we can see that *Dupe* achieves 24.20%, 28.90% and 33.40% accuracy at recall-rate@5, recall-rate@10 and recall-rate@20 respectively by considering only the question titles and when similarity is calculated using cosine similarity. When we combine the above feature with the term overlap between question titles, the recall-rate drops instead of improving. Therefore, we avoid using term overlap and use cosine similarity for the remaining experiments. When we only consider similarity between question bodies we do not observe much difference in the recall-rate. Considering the tag similarity feature alone, gives us poor performance and the recall-rate@20 drops to 26%. However, we observe improvement of recall-rate when we combine the previous three features (title, body and tag similarity). The recall-rate improves when we add the title_body, body_title, and title_tag features with the previous feature set. When we combine features from steps 5 and 6, recall-rate improves by 5%, 7%, and 8% at recall-rate@5, recall-rate@10 and recall-rate@20 respectively. The recall-rate@20 reaches 66.10% when add the code feature. We also observe an increase for recall-rate@5 and recall-rate@10. Surprisingly, adding entity overlap, entity type overlap and WordNet similarity changes the recall-rate very insignificantly. This is most likely due to that fact the questions in Stack Overflow are very domain oriented. These feature values are computationally expensive and we do not recommend using them.

7.2 Why Dupe Performs better than DupPredictor

Our proposed technique outperforms *DupPredictor* in detecting duplicate questions in top-k rank. This section explains the reasons considering question with id 23687504 (duplicate) and 375427 (master) as an example.

First, the scoring function in *DupPredictor* suffers from noise and false positives. Before calculating similarity scores, the technique gives a weight to each term common between a pair of questions. Terms that appear frequently in questions get a higher weight, and as such, those terms that are not important in the current context can get a higher weight and dominate when calculating the similarity score. *DupPredictor* fails to consider that terms appearing in a question are the most discriminating when they infrequently occur in the data set, whereas the BM25 scoring function takes this into account.

Second, the performance of *DupPredictor* decreases as the data set size increases. The reason for this is that increasing the questions or data set, also increases the possibility of false matching. On the other hand, we select the top 10000 candidate questions using the BM25 model and then search for the duplicate question using the discriminative classifier.

Table 4: Impact of different features on the recall-rate

No.	Different Source of Information	Top 5 (%)	Top 10 (%)	Top 20 (%)
1	Title (Cosine Similarity)	24.20	28.90	33.40
2	Title (Term Overlap) + Title (Cosine Similarity)	17.24	20.13	23.41
3	Body (Cosine Similarity)	24.50	27.50	32.80
4	Tag (Cosine Similarity)	18.00	24.00	26.00
5	Title + body + tag	34.50	40.10	47.20
6	title_body+body_title + title_tag	40.80	47.10	55.50
7	Step-5 + Step-6	45.80	54.10	63.50
8	Step-7 + code (<i>Dupe</i>)	51.20	59.40	66.10
9	step 8 +(Entity Overlap + Entity Type Overlap + WordNet Similarity)	51.40	59.63	66.35

The first step reduces the number of questions to consider and eliminates the chance of false matching.

Third, we use the source code as a separate feature to train the classifier. Results from our study shows that adding source code information improves the recall-rate. For example, considering the source code, we can identify PIPE, subprocess and readline terms that can help us to match the duplicate question with the master question. This will not be possible if we do not consider the source code information.

Finally, considering only title, body, or tag similarity between a pair of questions may not be helpful to detect duplicates since the terms that are common between those questions can spread in different structural regions. An important term in the title of a duplicate question may not be present in the title of the master, but rather may reside in the body of that question. As a result, when we consider the cross information the recall-rate improves. For example, the term *readline* is present in the title of 23687504. The term is missing in the title of 37542 but it is present in the body.

7.3 Others

When we apply *Dupe* on the Stack Overflow data set and manually analyze the returned result, we found that *Dupe* is able to detect duplicate questions that have not yet been identified. For example, consider the following duplicate questions that have not yet been identified as duplicates: (a) Adding two ruby arrays⁵ and (b) Add two arrays into another array.⁶ Here, questions (a) and (b) are asking about the same concept: how one can sum the result of two arrays in Ruby. Question (a) is the duplicate of question (b), and it has not yet been discovered until our proposed model identified them as duplicates. If they were identified as duplicates early then askers would not have to wait for the answer. It shows the importance or need of our proposed model. We also found that given a question, the results returned by *Dupe* contains very relevant questions. Thus, our technique is not only useful to find duplicate question but can also be used to suggest questions that are related to each other.

7.4 Runtime Performance

There are two typical ways our technique can be used. First, the technique can be activated when a user submits a new question to Stack Overflow or the moderators can apply the technique periodically to detect duplicate questions.

⁵<http://stackoverflow.com/questions/12584585>

⁶<http://stackoverflow.com/questions/13078593>

The time required to detect duplicate questions varies. Depending on the number of previous questions that need to be searched, the value can range from 10 seconds to 30 seconds on a single node machine for a single question.

8. THREATS TO VALIDITY

There are a number of threats to this study. First, to answer why duplicate questions are submitted in Stack Overflow, we manually analyze a small number of questions. Manual investigation is time consuming and it is challenging to analyze a large sample of questions. To avoid bias in question selection we randomly select questions from different years and each question was analyzed by two different authors of the paper. Second, we consider questions covering six different programming languages in our evaluation. However, these programming languages are very popular in Stack Overflow and a large number of questions are associated with them. For example, in our data set we found 709,994 questions that are tagged with *java*.

Third, we re-implement the *DupPredictor* since both the data and implementation of the technique are not available. Although we cannot guarantee that our replication of the technique does not contain any error, we have spent a considerable amount of time implementing and testing the technique to minimize the possibility of introducing errors.

9. CONCLUSION

In this paper we investigated duplicate questions in Stack Overflow. We start with investigating the characteristics of duplicate questions. We found that despite the continuous effort to avoid duplicate questions, either by providing instructions of how to ask questions or suggesting related/duplicate questions at the time of writing a new question, the number of duplicate questions is increasing steadily. We also manually investigated questions to determine why duplicate questions are submitted. We found that not searching Stack Overflow first contributes the most to question duplication. Finally, we used a discriminative model classifier together with BM25 scoring function for detecting duplicate questions. We evaluated our technique with a large number of questions involving different programming languages. We found that our proposed technique, *Dupe* outperforms *DupPredictor*, the only available duplicate question detector for Stack Overflow, at recall-rate@5, recall-rate@10 and recall-rate@20. In the future, we plan to evaluate our technique with more questions covering various other programming languages. We are currently working on implementing the technique as a web service.

10. REFERENCES

- [1] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms", In Proc. of SIGIR, 2006, pp. 284-291.
- [2] H. Hajishirzi, W. Yih, and A. Kolcz, "Adaptive near-duplicate detection via similarity learning". In Proc. of SIGIR, 2010, pp. 419-426.
- [3] G. S. Manku, A. Jain, and A. D. Sarma, "Detecting near-duplicates for web crawling", In Proc. of WWW, 2007, pp. 141-150.
- [4] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, "Syntactic Clustering of the Web", In Proc. of WWW, 1997, pp. 1157-1166.
- [5] M. S. Charikar, "Similarity Estimation Techniques from Rounding Algorithms", In Proc. of STOC, 2002, pp. 380-388.
- [6] C. Treude, O. Barzilay, and M. Storey, "How do programmers ask and answer questions on the web?", In Proc. of ICSE, 2011, pp. 804-807.
- [7] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest Q&A site in the west", In Proc. of CHI, 2011, pp. 2857-2866.
- [8] A. Pal, R. Farzan, J. A. Konstan and R. E. Kraut, "Early detection of potential experts in question answering communities", In Proc. of UMAP, 2011, pp. 231-242.
- [9] S. M. Nasehi, J. Sillito, F. Maurer and C. Burns, "What Makes a Good Code Example? A Study of Programming Q&A in Stack Overflow", In Proc. of ICSM, 2012, pp. 25-34.
- [10] A. Bacchelli, L. Ponzanelli and M. Lanza, "Harnessing Stack Overflow for the IDE", In Proc. of RSSE, 2012, pp. 26-30.
- [11] B. Vasilescu, A. Capiluppi and A. Serebrenik, "Gender, representation and online participation: A quantitative study of StackOverflow", In Proc. of SocialInformatics, 2012, pp. 332-338.
- [12] A. Barua, S. W. Thomas and A. E. Hassan, "What are developers talking about? An analysis of topics and trends in Stack Overflow", Journal of Empirical Software Engineering, 2014, pp. 619-654.
- [13] S. Wang, D. Lo and L. Jiang, "An Empirical Study on Developer Interactions in StackOverflow", In Proc. of SAC, 2013, pp. 1019-1024.
- [14] B. Bazelli, A. Hindle, E. Stroulia, "On the Personality Traits of StackOverřřCow Users", In Proc. of ICSM, 2013, pp. 460-463.
- [15] K. Bajaj, K. Pattabiraman, and Ali Mesbah, "Mining Questions Asked by Web Developers", In Proc. of MSR, 2014, pp. 112-121.
- [16] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, M. Lanza, "Mining StackOverflow to Turn the IDE into a Self-confident Programming Prompter", In Proc. of MSR, 2014, pp. 102-111.
- [17] F. Calefato, F. Lanubile, M. C. Marasciulo, N. Novielli, "Mining Successful Answers in Stack Overflow", In Proc. of MSR, 2015, pp. 430-433.
- [18] D. Correa and A. Sureka, "Fit or Unfit: Analysis and Prediction of 'Closed Questions' on Stack Overflow", In Proc. of COSN, 2013, pp. 201-212.
- [19] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, "Improving Low Quality Stack Overflow Post Detection", In Proc. of ICSME, 2014, pp. 541-544.
- [20] S. Chang and A. Pal, "Routing questions for collaborative answering in community question answering", In Proc. of ASONAM, 2013, pp. 494-501.
- [21] D. Correa and A. Sureka, "Chaff from the Wheat: Characterization and Modeling of Deleted Questions on Stack Overflow", In Proc. of WWW, 2014, pp. 631-642.
- [22] Y. Zhang, D. Lo, X. Xia, and J. Sun, "Multi-Factor Duplicate Question Detection in Stack Overflow", Journal of Computer Science and Technology, 2015, pp. 981-997.
- [23] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of duplicate defect reports using natural language processing", in Proc. of ICSE, 2007, pp. 499-510.
- [24] C. Sun, D. Lo, X. Wang, J. Jiang, and S. C. Khoo, "A discriminative model approach for accurate duplicate bug report retrieval", in Proc. of ICSE, 2010, pp. 45-56.
- [25] A. Alipour, A. Hindle, and E. Stroulia, "A contextual approach towards more accurate duplicate bug report detection", In Proc. of MSR, 2013, pp. 183-192.
- [26] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information", In Proc. of ICSE, 2008, pp. 461-470.
- [27] C. Sun, D. Lo, S. Khoo, and J. Jiang, "Towards more accurate retrieval of duplicate bug reports", In Proc. of ASE, 2011, pp. 253-262.
- [28] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful ...really?", In Proc. of ICSM, 2008, pp. 337-345.
- [29] A. T. T. Ying, "Mining challenge 2015: Comparing and combining different information sources on the stack overflow data set", In Proc. of MSR, 2015.
- [30] G. Hirst and D. St-Onge, "Lexical Chains as representation of context for the detection and correction malapropisms", In WordNet: An Electronic Lexical Database (Language, Speech, and Communication), 1998.
- [31] "Introduction to Information Retrieval", <http://www-nlp.stanford.edu/IR-book/>
- [32] K. Tao, F. Abel, C. Hauff, G. Houben, and U. Gadiraju, "Groundhog Day: Near-Duplicate Detection on Twitter", In Proc. of WWW, 2013, pp. 1273-1283.