Empirical Software Engineering
https://doi.org/10.1007/s10664-019-09758-x

What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories

Stefanie Beyer[1] · Christian Macho[1] · Massimiliano Di Penta[2] · Martin Pinzger[1]

Published online: 28 August 2019

Abstract

On question and answer sites, such as Stack Overflow (SO), developers use tags to label the content of a post and to support developers in question searching and browsing. However, these tags mainly refer to technological aspects instead of the purpose of the question. Tagging questions with their *purpose* can add a new dimension to the identification of discussed topics in posts on SO. In this paper, we aim at automating the classification of SO question posts into seven question categories. As a first step, we harmonized existing taxonomies of question categories and then, we manually classified 1,000 SO questions according to our new taxonomy. Additionally to the question category, we marked the phrases that indicate a question category for each of the posts. We then use this data set to automate the classification of posts using two approaches. For the first approach, we manually analyzed the phrases to find patterns. Based on regular expressions, we implemented a classifier, for each of the categories, that determines whether a post belongs to a category. These regular expressions are derived by analyzing patterns in the phrases. In the second approach, we use the curated data set to train classification models of supervised machine learning algorithms (Random Forest and Support Vector Machines). For the machine learning algorithms, we experimented with 1,312 different configurations regarding the preprocessing of the text and the representation of the input data. Then, we compared the performance of the regex approach with the performance of the best configuration that uses machine learning algorithms on a validation set of 110 posts. The results show that using the regular expression approach, we can classify posts into the correct question category with an average precision and recall of 0.90, and an MCC of 0.68. Additionally, we applied the regex approach on all questions of SO that deal with Android app development and investigated the co-occurrence of question categories in posts. We found that the categories API USAGE, CONCEPTUAL, and DISCREPANCY are the most frequently assigned question categories and that they also occur together frequently. Our approach can be used to support developers in browsing SO discussions or researchers in building recommender systems based on SO.

Keywords Question categories · Machine learning · Stack Overflow · Android

Communicated by: Chanchal Roy, Janet Siegmund, and David Lo

✉ Stefanie Beyer
stefanie.beyer@aau.at

Extended author information available on the last page of the article.

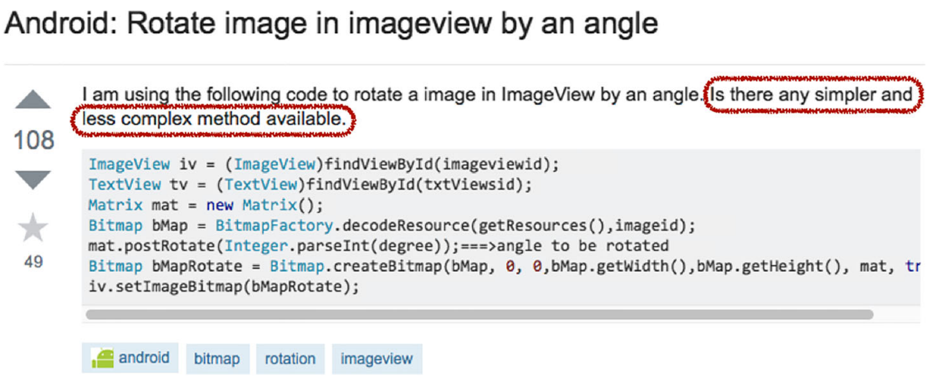
1 Introduction

Many developers use question and answer forums, such as Stack Overflow (SO), to discuss and solve their development issues. As a consequence, there are more than 16,000,000 diverse questions on SO that deal with developers' problems. For these questions, there exist more than 27,000,000 answer posts. In the following, we refer to the *question posts* as posts. If not explicitly mentioned, the answer posts are not included. On the one hand side this is good, since it enables developers to find solutions for their problems, on the other hand it is challenging to find the right solution in such a large amount of posts. Furthermore, developers ask for a better data organization of Q&A forums to increase the search efficiency and limit the time to find adequate solutions (Wu et al. 2018).

To refine the search and describe the questions briefly, each question post on SO is labeled with 1 to 5 tags, as shown in Fig. 1, that describes the problem of the post. These tags are often used by researchers as a starting point for the investigation of the topics that are discussed on SO (Barua et al. 2012; Treude et al. 2011) The tags mainly aim at classifying posts based on their technological content, *e.g.*, whether a post is related to Android, Java, Hadoop, etc. Hence, tags fail to classify questions based on their purpose *e.g.*, discussing a possible defect, demonstrating proper API usage, providing opinions about a given technology, or — some more general — conceptual suggestions.

However, as shown by recent research, it is not sufficient to analyze only the topics that are discussed. When investigating the issues of developers, the reasons of developers to ask questions should be considered as well (Beyer et al. 2017). These reasons are diverse and categorizing the questions based on the reasons *why* they are asked is needed to determine the role that SO plays for software developers (Rosen and Shihab 2015). Furthermore, as found by Allamanis and Sutton (2013), the investigation of such reasons can provide more insights into the most difficult aspects of software development and the usage of APIs. Knowing question categories of posts can help developers to find answers on SO easier and it can support SO-based recommender systems integrated into the IDE, such as Seahawk and Prompter by Ponzanelli et al. (2013, 2014).

Existing studies already aim at extracting the problem and question categories of posts on SO by applying *manual* categorizations (Rosen and Shihab 2015; Treude et al. 2011), topic modeling (Allamanis and Sutton 2013), or k-nearest-neighbor (k-NN) clustering (Beyer and Pinzger 2014). However, the manual approaches do not scale to larger sets of unlabeled



Android: Rotate image in imageview by an angle

I am using the following code to rotate a image in ImageView by an angle. Is there any simpler and less complex method available.

```
ImageView iv = (ImageView)findViewById(imageviewid);
TextView tv = (TextView)findViewById(txtViewsid);
Matrix mat = new Matrix();
Bitmap bMap = BitmapFactory.decodeResource(getResources(),imageid);
mat.postRotate(Integer.parseInt(degree));===>angle to be rotated
Bitmap bMapRotate = Bitmap.createBitmap(bMap, 0, 0,bMap.getWidth(),bMap.getHeight(), mat, tr
iv.setImageBitmap(bMapRotate);
```

Fig. 1 Question 8981845 from SO with the phrase marked in red that is indicating the question category Review

questions. The unsupervised topic modeling approach cannot directly be used to evaluate the performance of the classification of questions. against a baseline, and the k-NN algorithm shows a precision of only 41.33%. Furthermore, existing approaches use different but similar taxonomies of question categories.

In this paper, we address these gaps and provide a common taxonomy for classifying posts into question categories and investigate how, and to what extent we can classify SO posts into such categories using different approaches, based on regular expressions (regex) and machine learning algorithms. To get further insights into the reasons for discussion, we apply the best performing approach to all Android related posts and investigate how the question categories are distributed over the posts and which question categories often occur together.

Regarding the *question categories*, we start from the definition provided by Allamanis and Sutton (2013):

> *"By question types we mean the set of reasons questions are asked and what the users are trying to accomplish. Question types represent the kind of information requested in a way that is orthogonal to any particular technology. For example, some questions are about build issues, whereas others request references for learning a particular programming language."*

In contrast, *problem categories* — which can be expressed by SO tags — refer to the topics or technologies that are discussed, such as SQL, CSS, user interface, Java, Python, or Android. The problem categories do not reveal the reason *why* a developer asks a question.

In this paper, we focus on SO posts related to Android to investigate the *question categories* in the posts. Then, we aim at an automated classification of SO posts into these categories. We decided to focus on a specific domain to show whether our approach works there. If the approach works well, the generalization to a broader range of domains is easier than to investigate why an approach that is set for various domains at once might not work. We use Android as a case study since Android is one of the topics with the most increasing popularity on SO (Barua et al. 2012; Wen et al. 2016) and several previous studies (Beyer and Pinzger 2014; Rosen and Shihab 2015) also used Android to build their taxonomies.

Using the SO posts related to Android, we investigate how developers ask questions on SO and address our first research question:

- *RQ-1:* What are the most frequently used question categories of Android posts on SO?

We answer this question by analyzing the question categories and reasons for questions found in the existing studies (Allamanis and Sutton 2013; Beyer et al. 2017, 2014; Rosen and Shihab 2015; Treude et al. 2011), and by harmonizing them in one taxonomy. As a result, we obtain the 7 question categories: API CHANGE, API USAGE, CONCEPTUAL, DISCREPANCY, LEARNING, ERRORS, and REVIEW.

We then manually label 1,000 Android related posts of SO and record each phrase, *i.e.*, a sentence, part of a sentence, or paragraph of the text, that indicates a question category.

The set of posts and phrases is then used to automate the classification of posts into 7 question categories. With this, we aim to answer our second research question:

- *RQ-2:* To what extent can we automatically classify Stack Overflow posts into the 7 question categories?

We implemented two automated approaches to answer our research question: The first one uses regular expressions to classify a post into a question category. The second approach trains models of supervised machine learning algorithms that automate the classification of posts. To investigate each approach separately, we split RQ-2 into two the following two subquestions:

- *RQ-2.1:* What is the performance of our regex approach for classifying Stack Overflow posts into the 7 question categories?
- *RQ-2.2:* What is the performance of our best supervised machine learning model to classify Stack Overflow posts into the 7 question categories?

The first approach uses regular expressions that are based on the patterns found in phrases that indicate a question category.

The second approach uses the set of posts and phrases to train models that automate the classification of posts using the supervised machine learning algorithms Random Forest (RF) (Breiman 2001) and Support Vector Machines (SVM) (Cortes and Vapnik 1995). We trained the models with 1,312 configurations of the input data and achieved the best performance by using RF with phrases. We rerun the experiment with the best configuration 100 times to reduce the bias of the selection of the training and test set.

Then, we evaluated the performance of our classifiers on an independent test set of 110 SO posts that were neither used to extract patterns for the regular expressions nor to train and test the models before. The results show that the regex approach outperforms the machine learning algorithms with an average precision, recall, and MCC (Matthews Correlation Coefficient) of 0.90, 0.90, and 0.68, respectively. Furthermore, this approach is much faster and easier to adapt. We used the regex approach and applied it on all Android related posts of the SO dump from September 2017 to answer our third research question:

- *RQ-3:* How are the question categories distributed across all Android-related posts and to how many categories are posts assigned?

First of all, the application of the regex approach to all 1,052,568 Android questions confirmed our findings of RQ-1 that API USAGE, DISCREPANCY, and CONCEPTUAL are the most frequently used question categories. Furthermore, the results show that the majority of the posts is classified in one to three categories and that the categories are mostly not overlapping and the differentiation of the categories is clear.

Our results have several implications for developers and researchers. By integrating the proposed classifier into SO, the search efficiency could be improved. The question categories of posts could work as tags and hence, developers can search by question categories. For example, developers can use our approach to find API specific challenges by question category. Also, the classification can be leveraged by researchers to build better SO-based recommender systems. Furthermore, our results showed that machine learning algorithms are not always the better choice to build a classifier, in particular, if the borders between the classes are clear.

In summary, the main contributions of this paper are:

- A taxonomy of 7 question categories that harmonizes the taxonomies of prior studies.
- A manually labeled data set that maps 2,192 phrases of 1,000 posts to 7 question categories.
- An approach to automatically classify posts into the 7 question categories using regular expressions.
- An approach to automatically classify posts into the 7 question categories using machine learning algorithms
- An evaluation of the performance of the classifiers on an independent data set.

- An investigation of the question categories in all Android-related questions of SO that were asked until December 2017 using our best performing approach.

Furthermore, we provide all supplementary material that allows the replication and extension of our approach (Beyer et al. 2019).

This paper extends our ICPC 2018 paper 'Automatically Classifying Posts into Question Categories on Stack overflow' (Beyer et al. 2018). The field of program comprehension, as defined by the ICPC committee, is broad and ranges from the comprehension of source code, to software artifacts and the software lifecycle. With this paper, we contribute to the comprehension of software artifacts (posts of Stack Overflow) by providing an approach to automatically label posts with question categories. This in turn enables a more comprehensive understanding of the problems that Android app developers face.

Compared to the original paper, we provide an entirely new automated approach to classify posts into question categories using regular expressions. Furthermore, we extended our feature model for the classification with supervised machine learning algorithms by considering the length of the posts, the readability and sentiment score, and whether they contain code snippets.

We performed an additional experiment using the best performing approach *i.e.*, the regex approach, to study the question categories of all of the 1,052,568 Android-related questions on SO.

The remainder of this paper is organized as follows: In Section 2, we describe how we harmonized the existing question categories from prior studies. In Section 3, we describe the manual analysis of the posts and present the answer to *RQ-1*. In Section 4, we describe the setup of the automated classification, consisting of the general settings for both classifiers, as well as the specific settings for the regex approach and the experiments with the machine learning algorithms. The results of our experiments of the classification are presented in Section 5. Furthermore, we evaluate and compare the performance of our approaches in Section 6. We applied the best performing approach to 1,052,568 Android-related posts and present the results in Section 7. In Section 8, we discuss the implications of our results, as well as the threats to validity in Section 8.3. In Section 9, we present related work. Finally, in Section 10, we draw conclusions and discuss future work.

2 A taxonomy of Question Categories

In this section, we present our taxonomy of seven question categories that we derived from five taxonomies presented in previous studies. We selected the papers based on their content, whether they deal with question categories in the context of Android app development. To the best of our knowledge, the selected papers were the only ones that fit to these criteria when we started with our study. Analyzing the prior studies of Allamanis and Sutton (2013), Rosen and Shihab (2015), Treude et al. (2011), Beyer and Pinzger (2014), and Beyer et al. (2017) that investigate the posts according to their *question categories*, we found 5 different taxonomies. We decided to use these taxonomies rather than creating a new taxonomy, for instance through card sorting, since they are already validated and suitable to this context.

To harmonize the taxonomies, we compared the definitions of each category and merged similar categories. We removed categories, such as hardware, device, environment, external libraries, or novice, as well as categories dealing with different dimensions of the problems, such as questions asked by newbies, non-functional questions, and noise, because we found that they represent *problem categories* and not *question categories*. The final categorization was discussed with and validated by two additional researchers of our department who are familiar with analyzing SO posts.

Finally, we came up with 7 question categories merged from the prior studies:

API usage This category subsumes questions of the types *How to implement something* and *Way of using something* (Allamanis and Sutton 2013), as well as the category *How-to* (Beyer and Pinzger 2014; Treude et al. 2011), and the *Interaction of API classes* (Beyer et al. 2017). The posts falling into this category contain questions asking for suggestions on how to implement some functionality or how to use an API. The questioner is asking for concrete instructions.

Discrepancy This question category contains the categories *Do not work* (Allamanis and Sutton 2013), *Discrepancy* (Treude et al. 2011), *What is the Problem...?* Beyer and Pinzger (2014), as well as *Why*.[1] The posts of this category contain questions about problems and unexpected behavior of code snippets whereas the questioner has no clue how to solve it.

Errors This question category is equivalent to the category *Error* and *Exception Handling* from Beyer and Pinzger (2014) and Treude et al. (2011). Furthermore, it overlaps with the category *Why* (Rosen and Shihab 2015).[1] Similar to the previous category, posts of this category deal with problems of exceptions and errors. Often, the questioner posts an exception and the stack trace and asks for help in fixing an error or understanding what the exception means.

Review This category merges the categories *Decision Help* and *Review* (Treude et al. 2011), the category *Better Solution* (Beyer and Pinzger 2014), and *What* (Rosen and Shihab 2015),[2] as well as *How/Why something works* (Allamanis and Sutton 2013).[3] Questioners of these posts ask for better solutions or reviewing of their code snippets. Often, they also ask for best practice approaches or ask for help to make decisions, for instance, which API to select.

Conceptual This category is equivalent to the category *Conceptual* (Treude et al. 2011) and subsumes the categories *Why...?* and *Is it possible...?* Beyer and Pinzger (2014). Furthermore, it merges the categories *What* (Rosen and Shihab 2015)[2] and *How/Why something works*[3] (Allamanis and Sutton 2013). The posts of this category consist of questions about the limitations of an API and API behavior, as well as about understanding concepts, such as design patterns or architectural styles, and background information about some API functionality.

API change This question category is equivalent to the categories *Version* (Beyer and Pinzger 2014) and *API Changes* (Beyer et al. 2017). These posts contain questions that arise due to the changes in an API or due to compatibility issues between different versions of an API.

[1] The category *Why* from Rosen and Shihab (2015) dealing with questions about non working code, errors, or unexpected behavior is split into DISCREPANCY and ERRORS.

[2] Rosen and Shihab (2015) merge abstract questions, questions about concepts, as well as asking for help to make a decision into the question category *What*.

[3] Allamanis and Sutton (2013) merge questions about understanding, reading, explaining and checking into the category *How/Why something works*.

Learning This category merges the categories *Learning a Language/Technology* (Allamanis and Sutton 2013) and *Tutorials/Documentation* (Beyer et al. 2017). In these posts, the questioners ask for documentation or tutorials to learn a tool or language. In contrast to the first category, they do not aim at asking for a solution or instructions on how to do something. Instead, they aim at asking for support to learn on their own.

Table 1 shows an overview of the categories taken from prior studies and how we merged or split them. Categories in the same row match each other, categories that stretch over multiple rows are split or merged.

3 Manual Classification

In this section, we present our manual classification of 1,000 Android-related SO posts into the seven question categories. First, we describe the approach to obtain the 500 posts that were used in our previous work (Beyer et al. 2018). Then, we describe how we increased the number of labeled posts 500 to 1,000. Based on the results of the obtained classification, we answer RQ-1.

3.1 Experimental Setup

We used the posts' data dump of SO from September 2017. Since our goal is to analyze posts that are related to Android app development, we selected posts that are tagged with `android`. From the resulting 1,052,568 posts, we randomly selected 1,000 posts from SO.

These posts were then manually labeled by two researchers of our department as follows: Each person got a set of 1,000 posts and marked each phrase that indicates a question category. A phrase can be a paragraph, a sentence, or a part of a sentence. Hence, a post can have more than one category, as well as several times the same category.

The first set of 50 posts was jointly labeled by both investigators to agree on a common categorization strategy. The remaining posts were labeled by each investigator separately. We calculated the *Fleiss-Kappa* inter-rater agreement (Fleiss 1971) and obtained a $\kappa = 0.49$, meaning moderate agreement. However, we compared our results and found that the main differences were because of overlooked phrases of the investigators. We also discussed the posts in which the assigned question categories differed until we agreed on the labels of the posts. The main discussion was about whether a phrase refers to the question category CONCEPTUAL or REVIEW.

Figure 1 shows an example of labeling the post with the id 8981845. The phrase indicating that the post belongs to the question category REVIEW, is marked in red.

In the set of 500 posts, we found only 10 posts with the category API CHANGE and 15 posts with the category LEARNING. We decided to increase the number of posts for each of these two question categories to 30, to obtain more reliable classification models. For both question categories, we randomly selected 100 additional posts that contain at least one phrase indicating the category. Then, we manually assigned the question categories to the posts until we got 20 additional posts with the category API CHANGE and 15 additional posts with the category LEARNING. We end up with a set of 500 labeled posts and 1,147 phrases.

To increase the dataset we created in our previous work (Beyer et al. 2018), we randomly selected 550 posts that were not used for testing, training, or validation in the previous study. We intentionally selected more than 500 posts, since the labeling of the previous training and validation set showed that there are posts without any phrases that indicate a category.

Table 1 Our 7 question categories harmonized from the five prior approaches (Allamanis and Sutton 2013; Beyer et al. 2017; Beyer and Pinzger 2014; Rosen and Shihab 2015; Treude et al. 2011)

?	Rosen and Shihab (2015)	Allamanis and Sutton (2013)	Treude et al. (2011)	Beyer and Pinzger (2014)	Beyer et al. (2017)
API USAGE	*How:* A how type of questions asks for ways to achieve a goal. These questions can ask for instructions on how to do something programmatically to how to setup an environment. A sample how question asks: How can I disable landscape mode for some of the views in my Android app?	*How to implement something:* create, to create, is creating, call, can create, add, want to create	*How-to:* Questions that ask for instructions, e.g. ”How to crop image by 160 degrees from center in asp.net”.	*How-to:* the questioner does not know how to implement it. The questioner often asks how to integrate a given solution into her own code or asks for examples.	
		Way of using something: to use, can use, to do, want to use, to get, can do, instead of			*Interaction of API Classes:* Furthermore, several posts discuss the interaction of API classes, such as Activity, AsyncTask, and Intents.
DISCREPANCY	*Why:* why type of questions are used to ask the reason, cause, or purpose for something. They typically involve questions clarifying why an error has happened or why their code is not doing what they expect. An example why question is: I don’t understand why it randomly occurs?	*Do not work:* doesn’t work, work, try, didn’t, won’t, isn’t, wrong, run, happen, cause, occur, fail, work, check, to see, fine, due	*Discrepancy:* Some unexpected behavior that the person asking the question wants to be explained, e.g. “iphone - Core-motion acceleration always zero”.	*What is the Problem:* problems where the questioner has an idea how to solve it, but was not able to implement it correctly. The posts often contain How to...? questions, for which there is no working solution.	

Table 1 (continued)

?	Rosen and Shihab (2015)	Allamanis and Sutton (2013)	Treude et al. (2011)	Beyer and Pinzger (2014)	Beyer et al. (2017)
ERRORS			*Error:* Questions that include a specific error message, e.g. C# Obscure error: file ”could not be refactored”	*Error:* describe the occurrence of errors, exceptions, crashes or even compiler errors. All posts in this category contain a stack trace, error message, or warning.	*Exception Handling:* 17 posts discuss problems with handling exceptions.
REVIEW	*What:* A what type of question asks for information about something. They can be more abstract and conceptual in nature, ask for help in making a decision, or ask about non-functional requirements. For example questions about specific information about a programming concept: Explain to me what is a setter and getter. What are setters and getters? couldn't find it on wikipedia and in other places.		*Decision Help:* Asking for an opinion, *e.g.*, Should a business object know about its corresponding contract object.	*Better Solution:* contain questions for better solutions or best practice solutions. Typically, the questioner already has an unsatisfactory solution for the problem.	
		How/Why something works: hope, make, understand, give, to make, work, read, explain, check	*Review:* Questions that are either implicitly or explicitly asking for a code review, e.g. ”Simple file download via HTTP - is this sufficient?”.		

Table 1 (continued)

?	Rosen and Shihab (2015)	Allamanis and Sutton (2013)	Treude et al. (2011)	Beyer and Pinzger (2014)	Beyer et al. (2017)
CONCEPTUAL			*Conceptual:* Questions that are abstract and do not have a concrete use case, e.g. ”Concept of xml sitemaps”.	*Why:* focus on obtaining background information on a component or life cycle. The questioner asks for explanation and understanding. *Is it possible:* contain questions to get more information about the possibilities and limitations of Android apps or several APIs.	
API CHANGE				*Version:* deal with problems that occur when changing the API level. Furthermore, this category contains posts that deal with the compatibility of API versions.	*API Changes:* Further 3 posts discuss how to implement features for newer or older versions of the API. In 2 of the 100 posts the problem relates to deprecated methods in the API classes. 3 posts discuss bugs in the Android API and restrictions of Android versions to access Micro SD cards.
			Learning a Language/ Technology: learn, to learn, start, read, understand, recommend, find, good		*Tutorials/Docu:* In 10 posts, the developers mention tutorials and documentation should cover parts of the Android API in more detail.

From the 550 posts, we selected the first 500 where we could find a question category. For the labeling, we followed the same approach as before except the jointly labeling of the posts. We calculated again the *Fleiss-Kappa* inter-rater agreement and achieved a $\kappa = 0.45$ for the new set of labeled posts. As for the first set of 500 posts, we discussed the deviant labels until we reached an agreement.

3.2 Results

In total, we manually analyzed 1052 posts and for 1,000 posts, we could identify 2,192 phrases leading to a question category.

For 52 posts, we could not find any phrase that indicates one of our seven question categories.

The post 17485804[4] represents an example of such a post that we could not assign to any of the seven question categories. Reading the question, it was unclear to both investigators if the questioner asks for help on the implementation or if she asks for hints on how to use the app.

Using the set of 1,000 posts, we then analyzed how often each question category and each phrase occurs. The results are presented in Table 2, showing the number of posts and the number of phrases for each question category, as well as the most common phrases (including their count) found in the posts for each category.

The results show that API USAGE is the most frequently used question category assigned to 388 out of the 1,000 posts (38.8%) and 537 phrases. 247 times the question category was identified by the phrase ”*how to*”. The second most frequently assigned question category is DISCREPANCY with 313 posts (31.3% of the posts) and 434 phrases. The phrase ”*i try/tried to*” is the most frequently occurring phrase, namely 125 times, to identify this question category. Interestingly, the question category with the second highest number of phrases, namely 457, is ERRORS contained by 225 posts (22,5%). Furthermore, 49 posts (4.9%) were assigned to API CHANGE and 38 posts (3.8%) were assigned to the question category LEARNING.

Note that the post counts sum up to more than 1,000 because a post can be assigned to more than one question category. Based on these results, we can answer the first research question *”What are the most frequently used question categories of Android posts on SO?”* with: Most posts, namely 388 out of 1000 (38.8%), fall into the question category API USAGE followed by the categories DISCREPANCY with 313 posts (31.3%) and CONCEPTUAL with 268 posts (26.8%).

Our findings confirm the results of the prior studies presented in Beyer and Pinzger (2014), Rosen and Shihab (2015), and Treude et al. (2011) showing that API USAGE is the most frequently used question category. Similarly to these studies, the categories CONCEPTUAL, DISCREPANCY, and ERRORS showed to be among the top 2 to 4 most frequently used categories.

4 Setup of the Automated Classification

In this section, we first describe the general settings that hold for both the automated classification based on regular expressions and the automated classification with machine learning

[4] https://stackoverflow.com/questions/17485804/showing-overlay-help-in-android-app

Table 2 Number of posts per question category and most frequently used phrases to identify each question category

Category	# of posts	# of phrases	Most frequently used phrases (count)
API USAGE	388	537	how to (247), how can/could I (140), how do I/does (58)
CONCEPTUAL	268	379	is there a/any way to (68), what is the difference between/the use of/the purpose of (45), can I use (12), is it possible to (55)
DISCREPANCY	313	434	i try/tried to (125), do/does not work (81), what is/am i doing wrong (52), solve/fix/I have the problem (72)
ERRORS	225	457	(fatal/uncaught/throwing) exception (189), get/getting/got (an) error(s) (75)
REVIEW	172	229	is there a better/best/proper/correct/more efficient/simpler way to (51), (what) should I use/switch/do (26), is this/my understandings right/wrong (19)
API CHANGE	49	80	before/after (the) update/upgrade (to API/version/level) (16), work above/below/with API level/android/version $x.x$ (but) (9)
LEARNING	38	46	suggest/give me/find (links to) tutorial(s)/material (29)

algorithms. Then, we describe the specific experimental setup for the regex approach and the machine learning algorithms.

4.1 General Settings

In the following, we describe our settings regarding the classification, the data set, and the metrics to evaluate the performance of the classifier.

Binary Classifier The manual classification of the posts showed that a post may belong to more than one question category. Hence, we have a multi-label classification problem. For this reason, we do not rely on a single (multi-category) classifier, classifying each post into one of the seven categories. Instead, using the *binary relevance method* (Read et al. 2011), we transform the multi-label classification into a binary classification: We implemented a classifier/ a model for each question category to determine if a post falls into that category.

Since a post can have multiple labels, we selected for each post only the positive instances, the others are excluded. For example, consider the following three posts p, q, and r: p contains one phrase of the category API USAGE, q one phrase of the category REVIEW, and r one phrase of both categories. To train a model that classifies whether a post belongs to the API USAGE category, we select the posts p and r because they contain phrases that belong to API USAGE and use them as TRUE instances. For the FALSE instances, we only include post q. Post r is excluded from the FALSE instances.

Data set For the refinement of the classifiers for the regex approach and for the training and testing of the models, we used the set of 1,000 posts resulting from our manual classification before. From each post, we extracted the title and the body and concatenated them. Furthermore, we removed HTML tags, as well as code snippets which are enclosed by the tags `<code>` and `</code>`, and contain more than one word between the tags.

Performance To measure and compare the performance of the classifier, we computed the accuracy, precision, recall, f-score, AUC, and Mathews correlation coefficient (MCC) (Chicco 2017; Powers 2011) metrics. Note that we report metrics for both sides of the classification: whether a post was classified correctly as belonging to a question category (class$_T$) and whether a post was classified correctly as *not* belonging to a question category (class$_F$).

- *Accuracy (acc)* is the ratio of correctly classified posts into class$_T$ and class$_F$ with respect to all classified posts. Values range from 0 (low accuracy) to 1 (high accuracy).
- *Precision (prec)* is the ratio of correctly classified posts with respect to all posts classified into the question category. Values range from 0 (low precision) to 1 (high precision). The weighted average precision is calculated as the mean of prec$_T$ and prec$_F$ with respect to the number of posts predicted for each class.
- *Recall (rec)* is the ratio of correctly classified posts with respect to the posts that are actually observed as true instances. Values range from 0 (low recall) to 1 (high recall). The weighted average recall is calculated as the mean of rec$_T$ and rec$_F$ with respect to the number of posts labeled with each class.
- *F-score (f)* denotes the harmonic mean of precision and recall. The values range from 0 (low F-score) to 1 (high F-score). The weighted average F-score is calculated as the mean of f$_T$ and f$_F$ with respect to the number of posts labeled with each class.
- *Area under ROC-Curve (AUC)* measures the ability to classify posts correctly into a question category using various discrimination thresholds. An AUC value of 1 denotes the best performance, and 0.5 indicates that the performance equals a random classifier (*i.e.*, guessing).
- *Matthews Correlation Coefficient (MCC)* measures the performance of binary classifiers by considering the correctly classified posts, true positives (TP) and true negatives(TN), and the misclassified posts false positives (FP) and false negatives (FN). It is determined by calculating the ratio of the difference of the product of the correctly classified instances (TP · TN) and the product of the misclassified instances (FP · FN) to the root of the product of the sum of each combination of TP, TN, FP, and FN: $mcc = \frac{(TP \cdot TN)-(FP \cdot FN)}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}}$
  The values range from -1 (total disagreement between labeled posts and the classifier) to 1 (perfect classifier) and a value of 0 means that the classifier is as good as any random prediction.

4.2 Experimental Setup of the Regex Approach

In the following, we describe our approach to automate the classification of posts into the 7 question categories using regular expressions.

During the manual classification of posts into question categories, we additionally marked the phrases that indicate a certain question category. We analyzed these phrases and extracted patterns that we composed to regular expressions. Based on the regular expressions, we implemented an automated classifier for each of the categories to decide whether a post belongs to this question category. To obtain the regular expressions, we used the following steps:

1. *Recurrent phrases:* As shown in Table 2, there are phrases that recurrently point to a certain category for each question category. For instance, the phrase `how to` was marked 247 times to indicate the question category API USAGE. As a first step, we identified these recurrent phrases for each category and used them in our classifier. For the category API CHANGE, there are no recurrent phrases. In this case, we skipped this step.
2. *Patterns in phrases:* We searched for patterns in the phrases that occur with different verbs or personal pronouns and formed regular expressions that fit these patterns. For instance, for the question category API USAGE, we found the pattern `how <verb> <personal pronoun> do` which match the phrases such as `how can I do`, `how can you do`, and `how can one do`. These phrases are implemented in the regular expression `'how can (I|you|one) do'`. This is done similarly for the other question categories.
3. *Combinations:* The patterns that we obtained in the preceding two steps do not cover all possible phrases that indicate a question category. To expand our set of regular expressions to a broader variety of phrases, we combined patterns if the combination would lead to a meaningful phrase. Hence, we cover a variety of phrases that even do not occur in the set of posts that we manually investigated. Exemplary, for API CHANGE, we combined the phrases `before upgrading` and `after updating` to the pattern `(before|after)( [^\s]+){0,5} (upgrad(\w{1,4})|updat(\w{1,4})| downgrad(\w{0,4}))`. By stemming the verbs, we are able to catch also different tenses.
4. *Anti-patterns:* While analyzing the recurrent phrases for each category, we found that sometimes the phrases of different categories are similar but not identical since they appear most often together with unambiguous phrases or words that indicate the question category clearly. Hence, we decided to implement anti-patterns that use these unambiguous phrases to indicate that a post does not belong to a category. Examples for anti-patterns of the question category API USAGE are `how (((can|do|does|would) (I|you|one))|to) solve` and `understand ( [^\s]+){1,5} how`.
5. *Refinements:* In an iterative refinement process, we revised the set of regular expressions to improve the performance of the classifiers. We substituted too generous phrases, such as `how to` for the question category API USAGE with more specific ones. To obtain more specific phrases, we manually investigated the context of the phrases in the posts and extended the phrases. For instance, the phrase `how to` is substituted by the regular expression `how to (use|do|achieve|get|implement)`.

To decide whether a post belongs to a question category, we sum up the count how often each regular expression for pattern (p) and anti-pattern (ap) matches. Then, we subtract the number of matched anti-patterns from the number of matched patterns. If the result is positive, the post is classified into the category. A positive result means that the post contains more phrases that indicate a question category than phrases that indicate that a post does not belong to this category. In contrast, if there are more phrases that indicate that a post does not belong to a category, or the number of matched phrases for the patterns and antipatterns is equal, we do not assign the post to this category.

$$category = \begin{cases} TRUE, & \text{for } p - ap > 0 \\ FALSE, & \text{for } p - ap \leq 0 \end{cases}$$

Please note that the regular expressions including the patterns and antipatterns for all question categories can be found in our replication package$^{??}$.

4.3 Experimental Setup Using Machine Learning Algorithms

Previous research on the efficiency of machine learning algorithms in text classification tasks shows that classical, supervised machine learning algorithms, such as (RF) or (SVM), can perform equally well or even better than deep learning techniques (Fu and Menzies 2017). Furthermore, deep learning techniques usually are more complex, slower, and tend to over-fit the models when a small data set is used.

Therefore, we selected the supervised machine learning algorithms RF (Breiman 2001) and SVM (Cortes and Vapnik 1995) for our experiments to find models that can automate the classification of SO posts into the seven question categories. We ran the experiments using the default parameters provided by the respective implementation of R: $ntree$(number of trees) $= 500$ for RF, and $gamma = 0.1$, $epsilon = 0.1$, and $cost = 1$ for SVM.

Furthermore, we investigated whether part-of-speech patterns indicate question categories, following a similar approach as Chaparro et al. (2017) for bug reports. To get the part-of-speech tags, we used spaCy,[5] a Python-based part-of-speech tagger that has been shown to work best for SO data compared to other NLP libraries (Omran and Treude 2017). Using spaCy, we created the part-of-speech tags for the title, the body, and the phrases of a post. While Chaparro et al. also used NLP patterns, we opted for a simple, effective, and pretty consolidated approach to classify text, such as the one successfully used by Villarroel et al. (2016) and Scalabrino et al. (2017), when classifying app reviews.

We divide our data set into a training set and a testing set, consisting of 90% and 10% of the data, respectively. We apply random stratified sampling to ensure that 10% or at least three posts of each category are contained in the test set. We used random sampling instead of a n-fold cross-validation because it shows better results than n-fold cross-validation (Kohavi 1995).

To determine the configuration that yields the best results, we conducted our experiments using various configurations concerning the input type, the removal of stop words, the analysis of the text in n-grams, pruning of frequently used tokens, and using re-sampling of the input data. Additionally, we optionally consider the readability of the posts, the sentiments expressed in the posts, the number of words a post consists of, and whether a post contains a code snippet or not. Note, not all possible combinations make sense and are applicable. Pruning n-grams of the size 3 does not work, since too many tokens would be removed. Therefore, we excluded all runs that combine n-grams of size 3 and pruning. Furthermore, we did not perform stop word removal for POS tags.

In the following, we detail these configuration options:

Input type: We selected either the text (TXT), or part-of-speech tags (POS), or both representations (COMBI) of the data. When using the TXT or COMBI representation of

[5] https://spacy.io

the posts, we lowercased and stemmed the text using *R*'s implementation of Porter's stemming algorithm (Porter 1997).

Stop words (sw). We applied *stop word removal*, using a modified version of the default list of English stop words provided by *R*. We removed the words "*but*, *no*, *not*, *there*", and "*to*" from the list of stop words, because they are often used in our phrases and can indicate differences between the seven categories. For instance, in the sentence "*How to iterate an array in Java*" the phrase "*How to*" indicates the question category API USAGE while in the sentence "*How could this be fixed?*" the whole phrase indicates the category DISCREPANCY. The stop-word "*to*" helps to differentiate between the two question categories, hence, we kept it in the list.

N-grams. We computed the *n-gram* tokens for n=1, n=2, and n=3. When using the COMBI representation of the data, a separate n is given for the TXT and the POS representation of the data. We refer to them as n_{txt} and n_{pos}, respectively.

Pruning. When pruning was used, tokens that occur in more than 80% of all posts were removed because they do not add information for the classification. We also experimented with pruning tokens using other thresholds, such as 50% of the posts, which was stated in the examples of R. Hence, we run a limited set of experiments with different thresholds for pruning the tokens and obtained lower results for our performance metrics than with 80%. Furthermore, we aim at avoiding a combinatorial explosion of experiments when running all experiments with thresholds varying between 50% and 100%, and decided to run the experiments only with and without pruning and set the pruning threshold to 80%.

Re-balancing. Considering the distribution of the question categories presented in Table 2 in Section 3, we noticed that our data set is unbalanced. For instance, the most frequently found question category API USAGE is found 537 times in 388 posts, and the least frequently found question categories API CHANGE and LEARNING are found 80 and 46 times in 80 and 46 posts, respectively. To deal with the unbalanced dataset, we re-balanced our training set using SMOTE (Chawla et al. 2002). SMOTE is an algorithm that creates artificial examples of the minority category, based on the features of the k nearest neighbors of instances of the minority category. We used the default setting of the R implementation of SMOTE with k=5 (Torgo 2016). If the re-balancing option is selected, SMOTE creates artificial instances of the minority category. Since we did not use a single, multi-label classifiers for the 7 categories, but, rather, multiple binary classifiers, we applied SMOTE to re-balance the training set of each binary classifier, so that the minority class (i.e., posts belonging to that category) and the majority class (other posts) were balanced.

Word Count (wc). We counted the number of words considering the number of words from the title and the body of the posts.

Code-Snippets (code). During the preprocessing of the posts, we check whether a post contains a code snippet, meaning a part of the text that is enclosed by the tags `<code>` and `</code>` and contains more than one word.

Readability (read). To obtain the readability of the posts, we computed various readability metrics, such as the Flesch-Kincaid readability, the Automated Readability Index, and the SMOG Index. The Flesch-Kincaid readability estimates the complexity of texts, the Automated Readability Index indicates the age of the audience that would understand the text, and the SMOG Index estimates the years of education that are needed to understand the text (Mc Laughlin 1969; Kincaid et al. 1975).

Sentiment (senti). We counted the words in the title and body of the posts that refer to very negative, negative, neutral, positive, and very positive sentiments using the Natural Language ToolKit's package for sentiment analysis (Loper and Bird 2002).

Overall, we obtained 1,312 different configurations of our input data: 320 when TXT is used, 160 when POS is used, and 832 different configurations when COMBI is used. We used each configuration to compute a model for each of the 7 question categories.

5 Results

In this section, we first describe the results of the classification using the regex approach. Second, we report the results of the best performing classification models of RF and SVM.

5.1 Results Using the Regex Approach

Table 3 shows the results of our regex approach to classify posts into question categories. With this results, we can answer our research question RQ-2.1 *What is the performance of our regex approach for classifying Stack Overflow posts into the 7 question categories?* as follows: With the regex approach, we can classify a post into the correct question category with an average precision, recall, and MCC of 0.91, 0.91, and 0.68, respectively. We favor a high precision over a high recall, since we aim at labeling posts and we argue that we better do not assign a label to a post than label many posts wrongly. Hence, the recall of the TRUE category is low with an average of 0.69 across all question categories.

5.2 Results of the Automated Classification with Machine Learning Algorithms

As described in Section 4.3, we experimented with 1,312 various configurations to classify posts into the 7 question categories using the machine learning algorithms RF and SVM. However, we focus on the presentation of the best configurations using RF and SVM with the full text and the phrases as input setting. Hence, we first describe how we determined the best configuration and then, we present the results using the best configurations and answer research question RQ-2.2.

5.2.1 Determining the Best Configuration

To determine the best configuration for classifying posts into our seven question categories, we used the following approach:

We computed the models for each question category and each configuration with both machine learning algorithms (RF and SVM), first, using the *full text* and, second, using the *phrases* of the posts as input for training the models. For testing, we always used the full text of the posts, since the goal is to classify a post and not the single phrases of it. Overall, we performed 7 (categories) × 1,312 (configurations) × 2 (RF or SVM) × 2 (full text or phrases) = 183,680 experiments. Also, we ran each of these experiments 20 times using the stratified sampling described before. We limited the number of runs to 20, because such a large number of experiments took several weeks to compute on machines with 128 GB RAM and 48 cores or 755 GB Ram and 80 cores.

For each experiment, we computed the performance metrics accuracy, precision, recall, f-score, AUC, and MCC averaged over the 20 runs.

Table 3 Performance of the regex approach for each question category

Category	acc	AUC	MCC	prec$_{avg}$	rec$_{avg}$	f$_{avg}$	prec$_T$	rec$_T$	f$_T$	prec$_F$	rec$_F$	f$_F$
API CHANGE	0.97	0.80	0.66	0.97	0.97	0.97	0.75	0.61	0.67	0.97	0.97	0.97
API USAGE	0.89	0.88	0.76	0.89	0.89	0.89	0.85	0.86	0.85	0.89	0.89	0.89
CONCEPTUAL	0.86	0.83	0.65	0.86	0.86	0.86	0.74	0.75	0.74	0.86	0.86	0.86
DISCREPANCY	0.82	0.77	0.56	0.81	0.82	0.81	0.75	0.63	0.68	0.81	0.82	0.81
LEARNING	0.98	0.81	0.73	0.98	0.98	0.98	0.86	0.63	0.73	0.98	0.98	0.98
ERRORS	0.94	0.89	0.81	0.93	0.94	0.93	0.90	0.80	0.85	0.93	0.94	0.93
REVIEW	0.90	0.75	0.60	0.89	0.90	0.89	0.82	0.52	0.64	0.89	0.90	0.89
above average	0.91	0.82	0.68	0.91	0.91	0.90	0.81	0.69	0.74	0.91	0.91	0.90

To determine the *best* performing configuration out of the 1,312 configurations of input type (TXT, POS, COMBI), stop words (T, F), pruning (T, F), n-grams (n_{txt}, n_{pos}), resampling (T, F), readability (T, F), sentiments (T, F), code-snippets (T, F), and word count (T, F), we used the MCC as trade-off between precision and recall for both sides of the classification. Although the AUC is often recommended for assessing the performance of a (binary) classifier, it does not always work well for unbalanced datasets. Instead, the MCC is more stable for unbalanced datasets since it considers the amount of positive and negative instances. Furthermore, in contrast to the f-score, it shows only high scores if the classification for both, the positive and negative instances, show good results (Chicco 2017).

Then, we compared the results obtained by using the full text and the phrases as input for RF and SVM and selected the configuration that shows the best performance.

5.2.2 Results Using the Full Text

In the first experiment, we used the full text of the posts and computed the models with RF and SVM for each of the seven question categories. Table 4 shows the configurations and performance values for each question category with the highest weighted average MCC on 20 runs obtained with RF. Table 5 shows the results obtained with SVM.

The results show that RF uses different inputs and configurations for obtaining the classification models with the best performance.

In contrast, the configurations to obtain the best models with SVM do not vary that much. For instance, the best models obtained with SVM all use COMBI as input type with resampling of the data. Furthermore, 6 out of 7 classifiers don't consider the information about code snippets and don't remove stopwords.

Comparing the values for the MCC, the best models obtained with both, RF and SVM, show an overall MCC of 0.39 and 0.42, respectively. This is also shown by the results per question category, since SVM outperforms RF for the categories API CHANGE, DISCREPANCY, LEARNING, and REVIEW in terms of MCC. Although RF shows on average better scores for precision (+0.03), recall (+0.02), and AUC (+0.02), we consider SVM slightly RF, showing a higher score for MCC (+0.03) which is considered as more stable than the other metrics concerning the classification of positive and negative instances (Chicco 2017).

5.2.3 Results Using the Phrases

In the second experiment, we used the phrases of the posts to train the classification models. As for the previous experiment, we used the full text of the posts for testing the classifier because our goal is to classify a post based on its full text and not on its phrases. Tables 6 and 7 show the configurations of the best performing models and the results obtained with RF and SVM averaged over the 20 runs.

For RF, the configurations that lead to the highest MCC and differs per question category. For instance, RF obtains the best performance for the question categories API CHANGE using the COMBI input type. For the other categories RF obtains the best performance using the TXT as input. The POS input does not lead to the highest MCC for any category. In contrast, the models of SVM show the highest MCC when the dataset is resampled but not pruned and the sentiments aren't considered. Furthermore, the models of SVM show a better MCC when the COMBI input type is used with n_{txt}=1 and n_{pos}=3 for 5 out of 7 categories.

Comparing the performance of the models computed with RF and SVM, the average MCC of the RF models over all categories is 0.59 and higher than the average MCC of the SVM models, which is 0.33. Also the values of the other performance metrics obtained by

Table 4 Best configuration and performance over 20 runs using **RF** with the **full text** as input

Category	Type	n-grams	sw	prune	re-sample	read	senti	code	wc	acc	AUC	MCC	prec$_{avg}$	rec$_{avg}$	f$_{avg}$
API CHANGE	pos	1	F	F	T	T	F	F	T	0.95	0.80	0.24	0.93	0.95	0.94
API USAGE	combi	$n_{txt} = 1$, $n_{pos} = 3$	F	F	T	F	F	F	T	0.86	0.95	0.71	0.87	0.86	0.86
CONCEPTUAL	txt	2	F	T	F	F	F	F	F	0.81	0.87	0.48	0.84	0.81	0.78
DISCREPANCY	txt	2	F	F	T	F	T	F	F	0.73	0.81	0.31	0.79	0.73	0.65
LEARNING	pos	1	F	F	T	F	F	F	F	0.96	0.65	0.21	0.94	0.96	0.95
ERRORS	combi	$n_{txt} = 1$, $n_{pos} = 1$	T	T	F	F	F	F	F	0.88	0.95	0.64	0.89	0.88	0.87
REVIEW	combi	$n_{txt} = 1$, $n_{pos} = 3$	T	T	F	T	F	F	T	0.83	0.71	0.15	0.79	0.83	0.76
above average										0.86	0.82	0.39	0.86	0.86	0.83

Table 5 Best configuration and performance over 20 runs using **SVM** with the **full text** as input

Category	Type	n-grams	sw	prune	re-sample	read	senti	code	wc	acc	AUC	MCC	prec_{avg}	rec_{avg}	f_{avg}
API CHANGE	combi	$n_{txt} = 1$, $n_{pos} = 2$	F	T	T	F	F	F	T	0.97	0.92	0.58	0.96	0.97	0.96
API USAGE	combi	$n_{txt} = 1$, $n_{pos} = 1$	F	T	T	F	F	F	T	0.77	0.87	0.52	0.79	0.77	0.76
CONCEPTUAL	combi	$n_{txt} = 1$, $n_{pos} = 2$	F	F	T	F	T	F	T	0.77	0.75	0.34	0.75	0.77	0.75
DISCREPANCY	combi	$n_{txt} = 1$, $n_{pos} = 2$	F	F	T	T	T	F	T	0.75	0.77	0.40	0.74	0.75	0.74
LEARNING	combi	$n_{txt} = 1$, $n_{pos} = 2$	F	F	T	F	F	T	F	0.95	0.72	0.29	0.95	0.95	0.95
ERRORS	combi	$n_{txt} = 1$, $n_{pos} = 1$	F	T	T	T	F	F	F	0.85	0.86	0.55	0.85	0.85	0.84
REVIEW	combi	$n_{txt} = 1$, $n_{pos} = 3$	T	F	T	T	T	F	F	0.80	0.72	0.27	0.79	0.80	0.79
above average										0.84	0.80	0.42	0.83	0.84	0.83

Table 6 Best configuration and performance over 20 runs using **RF** with the **phrases** as input

Category	Type	n-grams	sw	prune	re-sample	read	senti	code	wc	acc	AUC	MCC	prec$_{avg}$	rec$_{avg}$	f$_{avg}$
API CHANGE	combi	n$_{txt}$=1, n$_{pos}$=3	T	F	T	T	T	T	F	0.96	0.92	0.67	0.97	0.96	0.97
API USAGE	txt	2	F	T	F	F	T	F	T	0.86	0.93	0.71	0.86	0.86	0.86
CONCEPTUAL	txt	2	F	T	T	T	T	T	T	0.85	0.86	0.58	0.84	0.85	0.84
DISCREPANCY	txt	1	F	F	F	F	F	T	F	0.76	0.86	0.52	0.80	0.76	0.77
LEARNING	txt	1	T	T	T	F	F	T	T	0.96	0.91	0.56	0.97	0.96	0.96
ERRORS	txt	1	T	T	F	T	T	F	T	0.90	0.94	0.69	0.90	0.90	0.89
REVIEW	combi	n$_{txt}$=1, n$_{pos}$=3	F	F	F	T	F	T	T	0.80	0.82	0.43	0.84	0.80	0.81
above average										0.87	0.89	0.59	0.88	0.87	0.87

Table 7 Best configuration and performance over 20 runs using **SVM** with the **phrases** as input

Category	Type	n-grams	sw	prune	re-sample	read	senti	code	wc	acc	AUC	MCC	prec_{avg}	rec_{avg}	f_{avg}
API CHANGE	combi	$\text{n}_{txt} = 2$, $\text{n}_{pos} = 3$	F	F	T	T	F	F	T	0.85	0.81	0.22	0.93	0.85	0.89
API USAGE	combi	$\text{n}_{txt} = 1$, $\text{n}_{pos} = 3$	F	F	T	F	F	F	F	0.75	0.82	0.48	0.80	0.75	0.71
CONCEPTUAL	combi	$\text{n}_{txt} = 1$, $\text{n}_{pos} = 3$	F	F	T	F	F	F	T	0.68	0.78	0.38	0.77	0.68	0.70
DISCREPANCY	combi	$\text{n}_{txt} = 1$, $\text{n}_{pos} = 3$	T	F	T	F	F	F	T	0.59	0.78	0.35	0.76	0.59	0.59
LEARNING	combi	$\text{n}_{txt} = 1$, $\text{n}_{pos} = 3$	F	F	T	T	F	F	T	0.83	0.82	0.29	0.95	0.83	0.87
ERRORS	combi	$\text{n}_{txt} = 1$, $\text{n}_{pos} = 3$	T	F	T	F	F	T	T	0.81	0.78	0.34	0.83	0.81	0.75
REVIEW	txt	n = 3	T	F	T	T	F	F	T	0.83	0.63	0.26	0.82	0.83	0.79
above average										0.76	0.77	0.33	0.84	0.76	0.76

the RF models are higher than the values of the SVM models. Comparing the MCC per question category, the RF models outperform the SVM models for each category. This is also true for all the other performance metrics, except for the accuracy and recall of the models for the question category REVIEW where the models of SVM show a slightly better performance (+0.03 each). In sum, training the models using the phrases of the posts as input, the models trained with RF outperform the models trained with SVM.

5.2.4 Results with the Best Performing Configuration

To determine the best configuration for classifying posts into the seven question categories, we compare the best performing models obtained with RF and SVM based on their performance metrics. With an overall average precision of 0.88, recall of 0.87, and MCC of 0.59, the models trained with RF using the phrases as input text clearly stand out.

This finding also holds for all MCC for each question category with one exception: the best models trained with RF and the full text and RF and the phrases perform equally in classifying posts into the question category API CHANGE (see Tables 4 and 6).

Based on these results, the configurations shown in Table 6 are considered as the *best configurations* to classify posts into the seven question categories.

To reduce the bias that might have been introduced by selecting the training and test data using the stratified sampling approach, we recomputed the classification models with the best configurations obtained with the RF and phrases of the posts from before 100 times instead of 20 times and answer research question RQ-2.2 *What is the performance of our best supervised machine learning model to classify Stack Overflow posts into the 7 question categories?*: Using RF with phrases as input, we can classify posts correctly into the seven question categories with an average precision, recall, and MCC of 0.87, 0.87, and 0.54, respectively.

Table 8 reports the performance values of the classification models averaged on 100 runs, including detailed performance values for class$_T$ and class$_F$.

6 Evaluation of the Classifiers

In this section, we compare the performance of the regex approach, the performance of the RF and phrases, and the Zero-R classification. Finally, we evaluate the regex approach and the RF and phrases model on an independent data set and present the answer to RQ-2.

6.1 Comparison of the Regex Approach and RF to Zero-R

The Zero-R classifier simply assigns each post to the majority class. Therefore, it is often used as a baseline for comparing the performance of different machine learning algorithms. We applied the 1,000 posts to the Zero-R classifier and report the results in Table 9. For the comparison with the regex approach and RF, we consider the results presented in Tables 3 and 8.

When comparing the averaged values of all three approaches we found that both the RF and the regex approaches clearly outperform the Zero-R classifier.

The MCC for the Zero-R classifier for all categories is 0, hence RF and regex clearly outperform this classifier with values of 0.54 and 0.68, respectively. Also the other performance metrics show, that Zero-R is outcut by the other classifiers. The RF shows a higher overall average accuracy (acc) of +0.06, AUC of +0.38, average precision (prec$_{avg}$) of +0.22,

Table 8 Results per question category rerunning the experiment with RF and phrases as input 100 times

Category	acc	AUC	MCC	prec$_{avg}$	rec$_{avg}$	f$_{avg}$	prec$_T$	rec$_T$	f$_T$	prec$_F$	rec$_F$	f$_F$
API CHANGE	0.95	0.90	0.54	0.96	0.95	0.95	0.52	0.64	0.55	0.98	0.96	0.97
API USAGE	0.85	0.93	0.68	0.85	0.85	0.85	0.83	0.77	0.80	0.86	0.90	0.88
CONCEPTUAL	0.84	0.85	0.56	0.84	0.84	0.83	0.79	0.54	0.64	0.85	0.95	0.90
DISCREPANCY	0.75	0.84	0.49	0.79	0.75	0.75	0.58	0.79	0.66	0.88	0.73	0.80
LEARNING	0.96	0.91	0.49	0.96	0.96	0.96	0.53	0.52	0.50	0.98	0.98	0.98
ERRORS	0.88	0.93	0.64	0.88	0.88	0.87	0.88	0.56	0.68	0.88	0.98	0.93
REVIEW	0.80	0.79	0.39	0.82	0.80	0.80	0.45	0.58	0.50	0.90	0.84	0.87
above average	0.86	0.88	0.54	0.87	0.86	0.86	0.66	0.63	0.62	0.91	0.91	0.90

Table 9 The performance of the classification of posts using **Zero-R** for each question category

Category	acc	AUC	MCC	prec$_{avg}$	rec$_{avg}$	f$_{avg}$	prec$_T$	rec$_T$	f$_T$	prec$_F$	rec$_F$	f$_F$
API CHANGE	0.95	0.50	0.00	0.90	0.95	0.93	0.00	0.00	0.00	0.95	1.00	0.97
API USAGE	0.63	0.50	0.00	0.39	0.63	0.48	0.00	0.00	0.00	0.63	1.00	0.77
CONCEPTUAL	0.74	0.50	0.00	0.55	0.74	0.63	0.00	0.00	0.00	0.74	1.00	0.85
DISCREPANCY	0.70	0.50	0.00	0.49	0.70	0.58	0.00	0.00	0.00	0.70	1.00	0.82
LEARNING	0.96	0.50	0.00	0.93	0.96	0.95	0.00	0.00	0.00	0.96	1.00	0.98
ERRORS	0.78	0.50	0.00	0.61	0.78	0.69	0.00	0.00	0.00	0.78	1.00	0.88
REVIEW	0.83	0.50	0.00	0.70	0.83	0.76	0.00	0.00	0.00	0.83	1.00	0.91
above average	0.80	0.50	0.00	0.65	0.80	0.72	0.00	0.00	0.00	0.80	1.00	0.88

average recall (rec$_{avg}$) of +0.06, and average f-score (f$_{avg}$) +0.14. The regex approach shows higher values for the accuracy (acc) of +0.11, AUC of +0.32, average precision (prec$_{avg}$) of +0.26, average recall (rec$_{avg}$) of +0.11, and average f-score (f$_{avg}$) +0.18. For all categories, Zero-R classifies all posts into class$_F$ considering the distribution of the labels shown in Table 2. As a consequence, precision, recall, as well as f-score for class$_T$ are 0 and, regarding this class, both of our approaches outperform the Zero-R classifier for each category.

For the class$_F$, the recall of the Zero-R models is, as expected, 1.0 for all question categories and regarding this metric Zero-R outperforms both of our approaches. However, the RF models with the best configuration as well as the regex approach perform better in terms of precision for each of the seven question categories.

Summarizing the results, the Zero-R classifier is clearly outperformed by our approaches using regular expressions and RF with phrases. Furthermore, the regex approach shows a better performance than the RF classifier in average for precision, recall, and MCC with values of 0.91, 0.91, and 0.68, respectively.

6.2 Evaluation with an Independent Sample-Set

We evaluated the performance of the regex approach and our best performing models with an independent sample set of 110 posts that has neither been used for the refinement of the regex approach nor for training and testing the models.

We labeled the posts following the same approach as described in Section 3.1. We aimed at having at least 100 posts for our evaluation. Since the previous study showed that not each post contains phrases leading to a category, we randomly sampled 120 posts related to Android from the SO data dump. For 110 out of 120 posts we could identify at least one phrase that indicates a question category. Hence, we used this set of 110 posts for our evaluation.

The distribution of question categories in this data set is similar to the set of 1,000 posts used before and described in Table 2. 49 posts were assigned to the question category API USAGE, 37 to the category DISCREPANCY, 34 posts were assigned to the question category ERRORS, 26 to the category CONCEPTUAL, 12 to the category REVIEW, 6 to the category LEARNING, and 2 to the category API CHANGE.

We applied the RF model with phrases as input and the best configuration 100 times to the 100 posts and obtained the results listed in Table 10. Additionally, we rerun the regex with the set of unseen posts and report the results in Table 11

The results show that using the validation set, the models with RF and phrases correctly classify posts with an average precision, recall, and MCC of 0.86, 0.86, and 0.47, respectively. This confirms the results shown by the 100 runs with the initial set of 1,000 posts, since the validation showed almost the same performance on average in terms of acc, AUC, rec$_{avg}$f-score f$_{avg}$. For the average scores for rec$_F$ and f$_F$ we observe increasing values with +0.03 and +0.01, respectively. For the other performance metrics, we observe a decrease in the performance, at most for MCC with -0.07 and rec$_T$ with -0.06. We assume that the decrease in the performance stems from the selection of the data in the test set. The independent set for testing stays the same over 100 runs. In contrast, the set of 1,000 posts is split 100 times using stratified sampling into a test and a validation set.

The results, when using the regex approach to classify the new set of posts, show the same value for f$_{avg}$ and f$_F$ show a slight decrease for prec$_{avg}$, rec$_{avg}$, prec$_F$ and rec$_F$ (-0.01 each), and a bigger decrease for MCC (-0.05), prec$_T$ (-0.06), rec$_T$ (-0.04), and f$_T$ (-0.05). Regarding the question categories, we observed the biggest decrease for the category API CHANGE and REVIEW. We manually inspected the posts of the test set that are labeled with

Table 10 The performance of the classification on the test set of 110 SO posts using RF and phrases as input text

Category	acc	AUC	MCC	prec$_{avg}$	rec$_{avg}$	f$_{avg}$	prec$_T$	rec$_T$	f$_T$	prec$_F$	rec$_F$	f$_F$
API CHANGE	0.94	0.95	0.26	0.98	0.94	0.95	0.15	0.53	0.23	0.99	0.95	0.97
API USAGE	0.83	0.90	0.65	0.83	0.83	0.83	0.83	0.77	0.80	0.83	0.87	0.85
CONCEPTUAL	0.82	0.85	0.43	0.80	0.82	0.80	0.70	0.40	0.51	0.84	0.95	0.89
DISCREPANCY	0.78	0.82	0.50	0.78	0.78	0.77	0.76	0.53	0.62	0.79	0.92	0.85
LEARNING	0.97	0.80	0.62	0.96	0.97	0.96	0.79	0.53	0.63	0.97	0.99	0.98
ERRORS	0.79	0.94	0.48	0.79	0.79	0.77	0.78	0.46	0.58	0.80	0.94	0.86
REVIEW	0.90	0.87	0.36	0.88	0.90	0.88	0.55	0.31	0.40	0.92	0.97	0.94
above average	0.86	0.88	0.47	0.86	0.86	0.85	0.65	0.50	0.54	0.88	0.94	0.91

Table 11 The performance of the classification on the test set of 110 using the regex

Category	acc	AUC	MCC	prec$_{avg}$	rec$_{avg}$	f$_{avg}$	prec$_T$	rec$_T$	f$_T$	prec$_F$	rec$_F$	f$_F$
API CHANGE	0.98	0.75	0.49	0.98	0.98	0.98	0.50	0.50	0.50	0.98	0.98	0.98
API USAGE	0.86	0.86	0.72	0.86	0.86	0.86	0.85	0.84	0.85	0.86	0.86	0.86
CONCEPTUAL	0.82	0.75	0.50	0.82	0.82	0.82	0.62	0.62	0.62	0.82	0.82	0.82
DISCREPANCY	0.85	0.82	0.67	0.85	0.85	0.85	0.84	0.70	0.76	0.85	0.85	0.85
LEARNING	0.98	0.91	0.82	0.98	0.98	0.98	0.83	0.83	0.83	0.98	0.98	0.98
ERRORS	0.87	0.81	0.69	0.88	0.87	0.87	0.92	0.65	0.76	0.88	0.87	0.87
REVIEW	0.92	0.70	0.51	0.91	0.92	0.91	0.71	0.42	0.53	0.91	0.92	0.91
above average	0.90	0.80	0.63	0.90	0.90	0.90	0.75	0.65	0.69	0.90	0.90	0.90

API CHANGE and observed a very unusual combination of phrases that is not covered by our regex approach yet. However, the results still confirm our findings of research question RQ-2.1.

Comparing the RF models with the regex approach, we observe that for MCC, f_{avg}, rec_T, and f_T the regex approach shows the better results for all question categories. For the accuracy (acc), $prec_{avg}$, and rec_{avg}, again the regex approach holds better scores over all question categories but the acc and $prec_{avg}$ of the category CONCEPTUAL, and the $prec_{avg}$ of API CHANGE. Furthermore, the regex approach outperforms the RF models also in terms of precision $prec_T$, except for the category CONCEPTUAL, whereas RF obtained higher values. Regarding $prec_F$, rec_F, and f_F, we observe that the RF slightly outperforms the regex approach, which is also shown in the average values for rec_F, and f_F. Furthermore, RF shows better values for the AUC. However, the AUC metric considers the "confidence" of the model that a post belongs to a category (*e.g.*, 68%). To that end, it happens that the AUC values of Table 10 are higher because the decisions were clearer (*e.g.*, with a higher confidence). However, this does not affect the number of correct classifications. In contrast, MCC only uses the decision (TRUE or FALSE) for the calculation and ignores the confidence. Hence, it can happen that a model with a higher MCC value can achieve a lower AUC value on the same data set. However, according to Chicco et al., the MCC is considered to be more stable performance metric in a binary classification task (Chicco 2017). Hence, we choose to make our decisions based on the MCC values and select the regex approach as the best performing classifier.

Based on these results, we present the answer to research question RQ-2 *"To what extent can we automatically classify Stack Overflow posts into the 7 question categories?"*: Applying an new data set to the regex approach that is not used for the validation of the classifier before, we can automatically classify posts with an average precision, recall, and MCC of 0.90, 0.90, and 0.68.

For further details about the evaluation, we refer the reader to our supplementary material (Beyer et al. 2019).

7 Question Categories of Android Related Questions on SO

In this section, we apply the best performing classification approach *i.e.*, the regex approach, to the whole data set of Android-related posts to answer our third research question. We use the SO data dump as of September 2017. This data dump contains a total of 1,052,568 posts that are tagged with `android`. We conducted this experiment to get insights into the distribution of the question types over all Android-related posts. We used the regex approach to classify each of the 1,052,568 posts. After this step, we investigated how many posts each post was assigned and to which categories each post was assigned.

Table 12 shows the number of posts that were assigned to the respective question category using the regex approach. All the columns sum up to the total number of studied posts of 1,052,568. However, the sum of the rows does not necessarily sum up to the number of total posts, because a post can be assigned to zero to seven question categories. We see that the category with the most assigned posts is API USAGE containing 376,294 posts (36%), directly followed by the DISCREPANCY category with 276,984 (26%), and the CONCEPTUAL category having 231,180 (22%) assigned posts. Comparing the relative values of the TRUE row to the relative values of the manual classification shown in Table 2 in Section 2, we see that the distribution of the classification with the regex approach is similar

Table 12 Number of posts classified into the respective question category

Category	API change	API usage	Conceptual	Discrepancy	Learning	Errors	Review
TRUE	23,873	376,294	276,984	211,304	21,129	200,218	70,805
	2%	36%	22%	26%	2%	19%	7%
TRUE manual	5%	39%	27%	31%	4%	23%	17%
FALSE	1,024,503	699,241	825,367	841,264	1,020,857	912,420	1,012,844
	97%	64%	78%	80%	97%	87%	96%

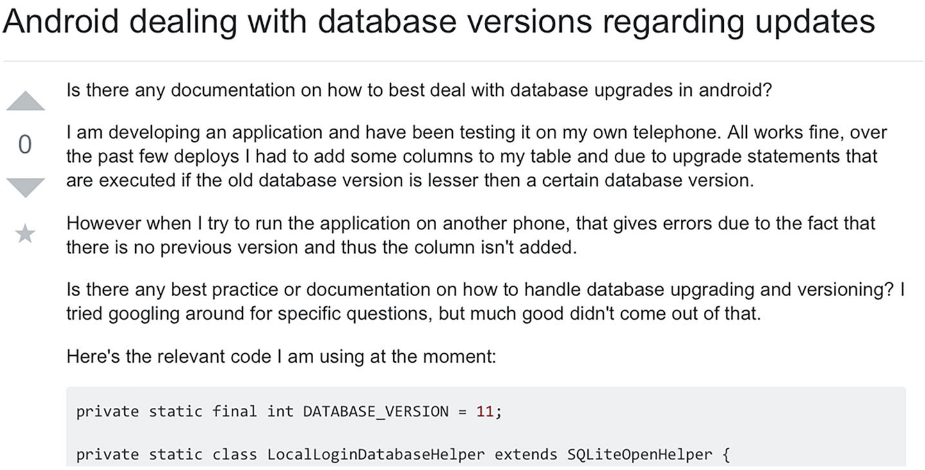
but more restrictive. Each of the categories has less assigned posts compared to the manual classification.

As one post can be assigned to more than one category, we also investigated the number of categories that a post was assigned. We find that 242,087 posts were not assigned to any of the seven question categories. Although one could argue that this is a flaw of the approach, we argue that this is expected because we only assign a category to a post if the regular expressions give enough indication. Furthermore, 501,278 posts were assigned to a single question category, 238,742 to two categories, and 60,903 to three. Only 8,804 posts were assigned to four categories, 728 to five, and 26 posts to six categories. No post was assigned to all of the categories. The majority (76%) of the posts is assigned to 1-3 question categories and the posts that were assigned to 4-6 categories make less than 1% of all posts.

In the following, we give examples of posts and their respective categorization. We start with the example in Fig. 2 that shows the post with the ID 13767705. This post was classified in all of the question categories except the category Discrepancy. This classification is reasonable because in the title, we find the phrase ”handle database upgrading and versioning” which indicates the category API change. Furthermore, we find in the text the phrase ”how to handle” which indicates API usage, the phrases ”Is there any documentation on” which indicate Learning, the phrase ”Is there any best practice” indicating Conceptual, the phrase ”that gives error” indicating Errors, and ”Here’s the relevant code I am using” indicating Review. We see that it is possible and reasonable that a post can be assigned to several question categories.

As a second example, we present the post with the ID 17485804 depicted in Fig. 3. This post has not been assigned to any of the categories. If we investigate the contents of the post, we recognize that the author apparently asks a very generic question (”Any help?”) and also lacks a detailed description of her problem or question. As described in the manual analysis Section 3, this post was excluded from the manually labeled data set because also manually no category could be assigned.

Additionally to the investigation of the classified categories, we also study the posts that were assigned to more than one category in more detail. Specifically, we investigate whether we can find patterns in the assignments *i.e.*, question categories that often occur together.



Android dealing with database versions regarding updates

Is there any documentation on how to best deal with database upgrades in android?

I am developing an application and have been testing it on my own telephone. All works fine, over the past few deploys I had to add some columns to my table and due to upgrade statements that are executed if the old database version is lesser then a certain database version.

However when I try to run the application on another phone, that gives errors due to the fact that there is no previous version and thus the column isn't added.

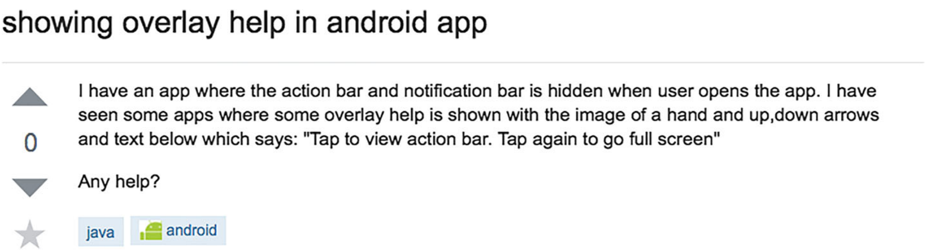
Is there any best practice or documentation on how to handle database upgrading and versioning? I tried googling around for specific questions, but much good didn't come out of that.

Here's the relevant code I am using at the moment:

```
private static final int DATABASE_VERSION = 11;

private static class LocalLoginDatabaseHelper extends SQLiteOpenHelper {
```

Fig. 2 Question 8910089 from SO. This post was classified in every question category but documentation



showing overlay help in android app

I have an app where the action bar and notification bar is hidden when user opens the app. I have seen some apps where some overlay help is shown with the image of a hand and up,down arrows and text below which says: "Tap to view action bar. Tap again to go full screen"

Any help?

Fig. 3 Question 17485804 from SO. This post could not be assigned to any question category

With this analysis, we aim at two targets. First, we can validate the taxonomy and find starting points for refinements of the taxonomy. For example, if two question categories occur together frequently, we need to check whether they are distinctively defined or if they overlap. Second, we aim at identifying posts that concern multiple targets. This can further be used to assist developers when asking questions on SO by indicating that they address more than one concern in their posts, for example.

We used the well-known *apriori* algorithm (Agrawal 1993, 1994) to perform the association rule analysis and we measure the performance of the found rules with the support and confidence metrics. Support expresses the ratio of which the rule can be found in the data set and confidence expresses the ratio in which the rule is correct. Recent research (Le and Lo 2015) suggests to include additional metrics for finding and evaluating association rules, such as lift and odds-ratio. However, we are not interested in a fine-grained analysis of the performance measures and rules, but we aim at verifying that there is no huge overlap in the question categories *i.e.*, very high support and confidence of a rule. Therefore, we argue that it is sufficient to use support and confidence for our purpose. We first investigate rules of size two (*i.e.*, $A => B$) and then validate the findings by investigating also rules of size three (*i.e.*, $A, B => C$).

In total, we found 21 rules, 18 binary rules and three tertiary rules. Table 13 depicts the binary rules with support > 0.05 and the three tertiary rules. First, we see that the support and confidence are not high which indicates that the rules are not very frequently occurring. This observation indicates that the question categories are not overlapping too much. Second, we see that there are pairs of rules that follow the pattern $A => B$ and $B => A$ which indicates that the two categories often occur together. Furthermore, we can see that the rules form transitive dependencies. For example, Discrepancy => API usage => Errors form a transitive relation between three question categories. This indicates that these three question categories often occur together. Indeed, we can find these rules in the tertiary rules depicted in Table 13. The support of 0.02 in each of the three rules indicates that in 2% of the rules of size three, we can find this relationship.

Based on these results, we present the answer RQ-3 *"How are the question categories distributed across all Android-related posts and to how many categories are posts assigned?"* with: The most frequently used question category is API usage with 376,294 posts, followed by Discrepancy (279,984), and Conceptual (231,180). This finding confirms also the results of RQ-2. Furthermore, we found that the majority of the posts is classified in one to three categories. We observed that the question categories are mostly not overlapping. The strongest relationship between the question categories is the co-occurrence of the three categories Discrepancy, API usage, and Errors.

Table 13 Frequently co-occurring question categories

ID	lhs		rhs	Support	Confidence
	Binary rules				
1	DISCREPANCY	=>	API USAGE	0.10	0.30
2	API USAGE	=>	DISCREPANCY	0.10	0.22
3	ERRORS	=>	DISCREPANCY	0.09	0.38
4	DISCREPANCY	=>	ERRORS	0.09	0.27
5	CONCEPTUAL	=>	API USAGE	0.09	0.31
6	API USAGE	=>	CONCEPTUAL	0.09	0.19
7	ERRORS	=>	API USAGE	0.06	0.23
8	API USAGE	=>	ERRORS	0.06	0.12
9	CONCEPTUAL	=>	DISCREPANCY	0.06	0.21
10	DISCREPANCY	=>	CONCEPTUAL	0.06	0.18
	tertiary rules				
11	DISCREPANCY,ERRORS	=>	API USAGE	0.02	0.26
12	API USAGE,ERRORS	=>	DISCREPANCY	0.02	0.44
13	API USAGE,DISCREPANCY	=>	ERRORS	0.02	0.24

8 Discussion

In this section, we first summarize and discuss our results, followed by the applications and implications of our results on the automated classification of posts into question categories. Then, we discuss the threats to the validity of our study.

8.1 Interpretation of Results

In this paper, we manually classify 1,000 posts into question categories and marked 2,192 phrases (words, parts of a sentence, or sentences) that indicate a question category. Based on this set of 1,000, we implement two approaches to automate the classification of posts into question categories. The first approach uses regular expressions based on patterns in the phrases. The second approach uses machine learning algorithms RF and SVM. Also, we experiment with 1,312 configurations of the input settings to classify posts. We validate the regex approach and the models of the best performing configuration on an independent validation set of 100 posts that was neither used for the implementation of the regex approach nor for training and testing of the models. The results showed that using the best performing approach, namely the regex approach, we can correctly classify posts into question categories with an average precision, recall, and f-score of 0.90, 0.90, and 0.90, respectively.

Before the extension to 1,000 posts, we performed the experiments with 500 posts. When comparing the results from the RF, we found a slight decrease in performance (-0.01 for AUC and -0.01 for f$_{avg}$) when running the experiments with 1,000 posts. This holds also for the validation with the independent data set (-0.04 for AUC and -0.03 for f$_{avg}$). We assume that with the new labeled data we added also more noise to the dataset and hence, the performance decreased. Regarding the regex approach, the results with 1,000 posts remained the same or showed a slight improvement (+0.01 for AUC and +0.00 for f$_{avg}$), whereas the results with the validation set were clearly better (+0.13 for AUC and +0.05). We argue that with the adaption of the regex classifier to the set of 1,000 posts, more patterns are covered and hence also the results with the validation set improved.

To gain more insights into the questions in the posts and to further evaluate the regex approach, we used this approach to classify the 1,052,568 questions on Stack Overflow that are related to Android app development. We found that the majority of the posts are classified into the question categories API USAGE, DISCREPANCY, and CONCEPTUAL. However, there are more than 23,000 posts that deal with problems due to changes in the API and more than 21,000 posts where developers ask for more tutorials and documentation, showing the need of developers for more support for learning APIs and how to deal with changes in the APIs. Furthermore, we found that the majority of posts are labeled with 1 to 3 question categories, however, there are many questions that are not classified into any question category and 26 posts that were labeled with 6 question categories. We investigated the posts that were assigned to more than one category in more detail and found that there is no particular pair of categories that occur frequently together. We conclude that the seven categories are well separated. Furthermore, a manual inspection of the posts showed that even a classification into 6 question categories may be reasonable, while a post without a question category is often very vague and imprecise, so that it is often not clear what the questioner wants to know. Hence, our classifier could recommend posts that should be revised since the intention of the person asking the question is not clear. Furthermore, the results showed that our categories have a little overlap, which indicates that the distinction between the categories is clear, while a large overlap would have shown the need to refine the taxonomy of question categories.

8.2 Applications and Implications

In the following, we discuss the applications and implications of our approaches and results for researchers and developers.

For researchers Researchers can benefit from our approach and results to classify posts into the seven question categories.

For instance, our approach could help to improve existing code recommender systems using SO, such as Seahawk and Prompter from Ponzanelli et al. (2013, 2014). Indeed, our approach could allow recommender systems to filter the posts according to the seven question categories, and thereby improve the accuracy of their recommendations. Exemplary, when a recommender system suggests posts based on the exceptions that a developer got, our approach can limit the set of recommended posts to posts with the question category ERRORS.

Furthermore, our approach can improve existing research on analyzing and identifying topics discussed on SO posts, such as presented in Barua et al. (2012), Beyer and Pinzger (2016), and Linares-Vásquez et al. (2013). While these approaches mainly focus on the technologies and topics that are discussed, with our question categories, an orthogonal view on the topics discussed on SO is provided, namely the reason why the question is asked. This enables researchers to investigate the relationships between topics and reasons and thereby study the *what* and *why* of discussions on SO. This means that researchers can analyze which questions are specific for which topics and hence, this enables them to address these problems more appropriately. In addition, our analysis of the frequently co-occurring categories showed that the categories are mostly well separated. However, the results of RQ3 indicate that further research to refine the categories should start with the categories that often occur together, *i.e.*, DISCREPANCY, API USAGE, and ERRORS.

For developers Furthermore, our approach can be integrated into SO helping software developers and API developers. SO could add a new type of tag, indicating the question category of a post. Using our approach, the posts can be tagged automatically with question categories. As stated by Wu et al. (2018), developers aim to better organize information in Q&A forums to increase the search efficiency. Hence, our approach can directly address this gap by supporting software developers searching for posts not only by topics but also by question categories.

Moreover, API developers could benefit from our approach when searching for starting points to improve their APIs and investigating the challenges of software developers that use the APIs. For instance, problems related to exception handling that often lead to issues in mobile apps (Coelho et al. 2015; Zhang and Elbaum 2014) can be found in posts of the category ERRORS. Discussions related to the change of APIs can be found by searching for posts of the category API CHANGE. Additionally, API developers can consider the posts tagged with the question category LEARNING as a starting point when improving and supplementing the documentation and tutorials on their APIs. Considering our results, developers can benefit from a feature on Stack Overflow that indicates that a post has not been assigned to a category. We argue that posts that do not belong to a category should be rephrased to increase the chance to receive a proper answer.

8.3 Threats to Validity

Threats to *construct validity* include the choice of spaCy of Omran and Treude (2017) to compute the part-of-speech tags. This threat is mitigated by the fact that spaCy is the approach with the highest accuracy, namely 90%, on data from SO. Another threat concerns the usage of binary classification instead of multi-label classification. However, Read et al. (2011) stated that binary classification is often overseen by researchers although it can lead to high performance. It also scales to large datasets and has less computation complexity.

Threats to *internal validity* concern the selection of the posts used for manual labeling. We randomly selected 1,000 posts and identified 2,192 phrases that indicate a question category. The initial dataset consisted of 10 posts of the API CHANGE question category and 15 posts of the LEARNING question category. In comparison to the other categories, this number of posts is low, therefore, we decided to label additional posts. Since we found only 10 posts in 500 posts for the API CHANGE category, there was no chance to label as many posts as we would need to get an equal size of posts per category. Hence, we decided to label posts until there were at least 30 posts for each category. This allowed us to perform the labeling in a reasonable time and to draw our conclusions for this categories with 80% confidence and 12% margin of error.

For each of the question categories API USAGE, CONCEPTUAL, DISCREPANCY and ERRORS, we found more than 385 phrases in the final dataset, which allows us to draw conclusions for these categories with 99% confidence and with 7% margin of error. For the category REVIEW, we found 229 phrases, which enables us to conclude that our results hold with 95% confidence and with 6.5% margin of error. For API CHANGE, we identified 80 phrases and for LEARNING 46 which allows us to draw conclusions for these categories with confidence level of 90% and 80%, respectively, and with a margin of error of 10%, each. Moreover, we are aware that the number of phrases to identify the categories API CHANGE and LEARNING could be enlarged.

To get an equally high number of phrases such as for REVIEW, we would need to investigate 5 times more posts, which we consider as a task for future work. Moreover, we argue that we focus on the large question categories, such as API usage, Conceptual, and Errors and hence, we consider the confidence level and margin of error which that we can make our conclusions over all categories is sufficient.

Regarding the independent sample set that we used for the evaluation of our approaches, the selected posts of the categories API usage, Discrepancy, and Errors are representative with 80% confidence and 10% margin of error, the posts for Conceptual with 80% and 11%, the category Review with 80% and 20%, and the API change and doc with 80% and 35%. We are aware that this limits the validation of our approach, however, we consider the evaluation of our approaches with a bigger data set for future work.

Furthermore, the manual categorization of the posts could be biased. To address this threat, we used the question categories obtained from prior studies and had two researchers to label the posts separately. Then, we computed the inter-rater agreement and let the two researchers discuss and converge on conflicting classifications.

Threats to *external validity* concern the generalizability of our results. While we used SO posts related to Android to perform our experiments, our seven question categories have been derived from several existing taxonomies that considered posts from various operating systems and other posts on SO. As a result, our question categories apply to other domains.

Another threat concerns the evaluation of our approaches to automate the categorization of posts. For the machine learning algorithms, we trained and tested the models with 1,000 posts from SO. We mitigated this threat, first, by applying randomized stratified sampling to divide the data set for training and testing and second, by testing the models with an independent sample set of manually labeled 110 posts. Regarding the regex approach, we used the set of 1,000 posts to obtain the regular expressions which could lead to an over-fitting of the classifier. We are also aware that the possibility of overseen patterns exists. However, to address this threat, we also evaluated this approach with the independent sample set and by manually investigating randomly selected posts. The evaluation of the regex approach with more independent will be addressed in future work. This supports that our classifiers are valid for the domain of Android posts. For other domains, the classification models might need to be retrained and the regex approach might need some adaption. However, this is subject to our future work.

9 Related Work

Several researchers have leveraged SO posts to investigate the nature of questions asked by software developers.

Treude et al. (2011) were the first ones investigating the question categories of posts of SO. In 385 manually analyzed posts, they found 10 question categories: *How-to*, *Discrepancy*, *Environment*, *Error*, *Decision Help*, *Conceptual*, *Review*, *Non-Functional*, *Novice*, and *Noise*. Similarly, Rosen and Shihab (2015) manually categorized 384 posts of SO for the mobile operating systems Android, Apple, and Windows each into three main question categories: *How*, *What*, and *Why*. Beyer and Pinzger (2014) applied card sorting to 450 Android related posts of SO and found 8 main question types: *How to...?, What is the Problem...?, Error...?, Is it possible...?, Why...?, Better Solution...?, Version...?,* and *Device...?* Based on the manually labeled dataset, they used Apache Lucene's k-NN algorithm to automate the classification and achieved a precision of 41.33%. Similarly, Zou et al. (2015) used Lucene to rank and classify posts into question categories by analyzing the style of the posts' answers.

Allamanis and Sutton (2013) used LDA, an unsupervised machine learning algorithm, to find question categories in posts of SO. They found 5 major question categories: *Do not work, How/Why something works, Implement something, Way of using,* and *Learning*. Also, they found that question categories do not vary across programming languages. In Beyer et al. (2017), Beyer et al. investigated 100 Android related posts of SO to evaluate if certain properties of the Android API classes lead to more references of these classes on SO. Besides some API properties, they found that the reasons for posting questions on SO concern problems with the interpretation of exceptions, asking for documentation or tutorials, problems due to changes in the API, problems with hardware components or external libraries, and questions of newbies.

There exist also other approaches not related to SO that aim at the identification of question categories asked by developers working in teams. Letovsky (1987) interviewed developers and identified 5 question types: why, how, what, whether, and discrepancy. Fritz and Murphy (2010) investigated the questions asked by developers within a project and provided a list of 78 that developers want to ask their co-workers. In LaToza and Myers (2010), Latoza et al. surveyed professional software developers to investigate hard-to-answer questions. They found 5 question categories: *Rationale*, *Intent and implementation*, *Debugging*, *Refactoring*, and *History*. Hou and Li (2011) analyzed newsgroup discussions about Java Swing and present a taxonomy of API obstacles.

There is also ongoing research in topic finding on SO. Linares Linares-Vásquez et al. (2013) as well as Barua et al. (2012) used LDA to obtain the topics of posts on SO. Linares Vasquez et al. investigated which questions are answered and which ones not whereby Barua et al. analyzed the evolution of topics over time. In Beyer and Pinzger (2016), Beyer et al. presented their approach to group tag synonym pairs of SO with community detection algorithms to identify topics in SO posts.

Several studies deal with analyzing domain-specific topics on SO. Joorabchi et al. (2013) identified the challenges of mobile app developers by interviewing senior developers. Studies from Kartik et al. (2014), Lee et al. (2018), Martinez and Lecomte (2017), Villanes et al. (2017), and Yang et al. (2016) as well as Mehrab et al. (Mehrab et al. 2018) investigate the topics related to web development, NoSQL, cross-platform issues, security-related questions, questions about Android testing, and questions about Django and Laravel, respectively, using LDA. Zhang and Hou (2013) extracted problematic API features from Java Swing related posts based on the sentences in the posts using the Stanford NLP library and part-of-speech tagging. Additionally, Zhang and Hou (2013) used SVM to categorize the content of posts related to the Java Swing API.

As pointed out by prior studies (Rosen and Shihab 2015; Beyer et al. 2017), the reasons *why* developers ask questions are diverse and need to be considered to get further insights into the problems developers face. Although existing studies (Allamanis and Sutton 2013; Beyer and Pinzger 2014; Rosen and Shihab 2015; Treude et al. 2011) already aimed at addressing this issue, they present diverse taxonomies of question categories that only partly overlap with each other. Among them, there are two approaches that propose an automated classification of posts into question categories. The approach presented by Allamanis and Sutton (2013) is based on LDA, an unsupervised machine learning approach. The precision of this approach cannot be evaluated. The approach by Beyer and Pinzger (2014) uses k-NN showing a low precision of only 41.33%.

Recent research from Wu et al. (2018) shows that there is a demand to improve the search efficiency of developers by optimizing information enhancement and management, as well as data organization. Li et al. (2018) present CnCxL2R, a recommender API documentation that uses the information of the official documentation of APIs and the posts of Stack Overflow to return a ranked list of API documentation.

In this paper, we analyze the existing taxonomies and harmonize them to one taxonomy. We argue that a post can belong to more than one question category and hence, we allow multi-labeling. Similarly to prior studies (Beyer and Pinzger 2014; Rosen and Shihab 2015; Treude et al. 2011), we start with a manual classification of the posts. However, to the best of our knowledge, we are the first ones that additionally mark the phrases (words, parts of sentences, or sentences) that indicate a question category and leverage them in the automated classifier. Also, our approach helps to structure the data of Stack Overflow which can consequently help to improve the search efficiency of developers on Stack Overflow.

10 Conclusions

In this paper, we investigate how Android app developers ask questions on SO, and to what extent we can automate the classification of posts into question categories. As a first step, we compared the taxonomies found by prior studies (Allamanis and Sutton 2013; Beyer et al. 2017, 2014; Rosen and Shihab 2015; Treude et al. 2011) and harmonized them into seven question categories. Then, we manually classified 1,000 posts into the question categories and found that most of the questions belong to the question categories API USAGE, CONCEPTUAL, and DISCREPANCY. Additionally, we marked 2,192 phrases (words, part of a sentence, or sentences) that indicate a question category.

Then, we used the manually created data set to automate the classification of posts into question categories with a binary classification model instead of a multi-label classification. Hence, we built a classifier for each category separately. We implemented two approaches to automatically classify posts.

In the first approach, we implemented a classifier based on regular expressions. We derived the regular expressions by analyzing the recurrent patterns in the phrases and combining them to regular expressions.

In the second approach, we build models of (RF) and (SVM) to classify posts. To obtain the best configuration for the models of RF and SVM, we computed the models for each category in 1,312 combinations varying the input data, input representation, as well as the preprocessing of the text in terms of stop word removal, pruning, using n-grams, and re-sampling of the data. Additionally, we take into account the length of the posts, the readability score, the sentiment score, and whether the posts contain code snippets. We compared the performance of the models and found, that across all categories, RF with phrases as input data showed the best classification performance.

We evaluated our approaches in two steps. First, we compared them with the Zero-R classifier, where a post is classified into the majority category. Second, we applied the regex approach and the RF on an independent data set of 110 posts that were neither used for deriving the regular expressions, nor for training and testing the models. The results showed that both classifiers clearly outperform the Zero-R classifier. Furthermore, the evaluation with the independent data set showed that the regex approach outperforms the RF with an average precision and recall of 0.90 and 0.90, respectively.

To further evaluate our best performing approach, we used the regex approach to label all Android-related questions that are contained in the SO data dump from September 2017. We investigated how the question categories are distributed over the 1,000 questions. We found that the majority of the posts are labeled with 1 to 3 categories and that the question categories are mostly not overlapping.

The implications of our findings on the classification of contributions into the seven question categories concern researchers and developers.

With the question categories, researchers can have an additional view on the problems of software developers, detached from the technical problems that are addressed by the tags of the posts. This may help researchers to improve SO based recommender systems or to investigate the topics of discussions from a different point of view.

Furthermore, integrated on SO, our approach could on the one hand help developers to find solutions easier and on the other hand enable API developers to better identify issues with their APIs.

For *future work*, we consider the extension of our approach to a multi-label classification and compare the results to the classification of Beyer and Pinzger (2014) directly. Also, we plan to improve the classifier by considering account additional features, such as the tags of the posts. Also, it could be possible to automatically learn regular expressions from a set of analyzed, manually-labeled posts. Furthermore, we plan to combine the question categories with approaches that extract the topics that are discussed on SO to investigate the *what* and *why* of discussions in more detail. Finally, we plan to evaluate our approach on other domains than Android. After that, we plan to collect feedback from the SO community in order to possibly integrate the approach in the SO querying features.

Acknowledgments Open access funding provided by University of Klagenfurt.

References

Agrawal R, Imieliński T, Swami A (1993) Mining association rules between sets of items in large databases. In: International conference on management of data. ACM, pp 207–216

Agrawal R, Srikant R et al (1994) Fast algorithms for mining association rules. In: Proceedings of the International Conference of Very Large Data Bases, vol 1215, pp 487–499

Allamanis M, Sutton C (2013) Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In: Proceedings of the Working Conference on Mining Software Repositories. IEEE, pp 53–56

Barua A, Thomas S, Hassan AE (2012) What are developers talking about? an analysis of topics and trends in Stack Overflow. Empir Softw Eng 19:1–36

Beyer S, Pinzger M (2014) A manual categorization of android app development issues on Stack Overflow. In: Proceedings of the International Conference on Software Maintenance and Evolution. IEEE, pp 531–535

Beyer S, Pinzger M (2016) Grouping android tag synonyms on Stack Overflow. In: Proceedings of the Working Conference on Mining Software Repositories. IEEE, pp 430–440

Beyer S, Macho C, Di Penta M, Pinzger M (2017) Analyzing the relationships between android api classes and their references on stack overflow. Technical report, University of Klagenfurt University of Sannio

Beyer S, Macho C, Pinzger M, Di Penta M (2018) Automatically classifying posts into question categories on stack overflow. In: Proceedings of the International Conference on Program Comprehension. ACM, pp 211–221

Beyer S, Macho C, Di Penta M, Pinzger M (2019) qc_replication_package.zip. https://doi.org/10.6084/m9.figshare.8870123.v1

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Chaparro O, Lu J, Zampetti F, Moreno L, Di Penta M, Marcus A, Bavota G, Ng V (2017) Detecting missing information in bug descriptions. In: Proceedings of the Joint Meeting on Foundations of Software Engineering. ACM, pp 396–407

Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357

Chicco D (2017) Ten quick tips for machine learning in computational biology. BioData Min 10(1):35

Coelho R, Almeida L, Gousios G, van Deursen A (2015) Unveiling exception handling bug hazards in android based on github and google code issues. In: Proceedings of the Working Conference of Mining Software Repositories. IEEE, pp 134–145

Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297

Fleiss JL (1971) Measuring nominal scale agreement among many raters. Psychol Bullet 76(5):378

Fritz T, Murphy GC (2010) Using information fragments to answer the questions developers ask. In: Proceedings of the International Conference on Software Engineering. ACM, pp 175–184

Fu W, Menzies T (2017) Easy over hard: a case study on deep learning. In: Proceedings of the Joint Meeting on Foundations of Software Engineering. ACM, pp 49–60

Hou D, Li L (2011) Obstacles in using frameworks and apis: an exploratory study of programmers' newsgroup discussions. In: Proceedings of the International Conference on Program Comprehension. IEEE, pp 91–100

Joorabchi ME, Mesbah A, Kruchten P (2013) Real challenges in mobile app development. In: Proceedings of the International Symposium on Empirical Software Engineering and Measurement. ACM/IEEE, pp 15–24

Kartik B, Karthik P, Ali M (2014) Mining questions asked by web developers. In: Proceedings of the Working Conference on Mining Software Repositories. ACM

Kincaid JP, Fishburne Jr RP, Rogers RL, Chissom BS (1975) Derivation of new readability formulas (automated readability index fog count and flesch reading ease formula) for navy enlisted personnel

Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai, Montreal, vol 14, 1137–1145

LaToza TD, Myers BA (2010) Hard-to-answer questions about code. In: Evaluation and usability of programming languages and tools. ACM, pp 8

Le TDB, Lo D (2015) Beyond support and confidence: Exploring interestingness measures for rule-based specification mining. In: Proceedings of the International Conference on Software Analysis, Evolution and Reengineering. IEEE, pp 331–340

Lee M, Jeon S, Song M (2018) Understanding user's interests in nosql databases in stack overflow. In: Proceedings of the International Conference on Emerging Databases. Springer, pp 128–137

Letovsky S (1987) Cognitive processes in program comprehension. J Syst Softw 7(4):325–339

Li J, Xing Z, Kabir A (2018), Leveraging official content and social context to recommend software documentation. IEEE Transactions on Services Computing

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

Loper E, Bird S (2002) Nltk: The natural language toolkit. In: Inproceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics

Martinez M, Lecomte S (2017) Discovering discussion topics about development of cross-platform mobile applications using a cross-compiler development framework. arXiv:1712.09569

Mc Laughlin GH (1969) Smog grading-a new readability formula. J Read 12(8):639–646

Mehrab Z, Bin Yousuf R, Tahmid IA, Rifat S (2018) Mining developer questions about major web frameworks. In: Proceedings of the International Conference on Web Information Systems and Technologies. SciTePress, pp 191–198

Omran FNAA, Treude C (2017) Choosing an nlp library for analyzing software documentation: a systematic literature review and a series of experiments. In: Proceedings of the International Conference on Mining Software Repositories, pp 187–197

Ponzanelli L, Bacchelli A, Lanza M (2013) Seahawk: stack overflow in the ID. In: Proceedings of the 2013 International Conference on Software Engineering. IEEE Press, pp 1295–1298

Ponzanelli L, Bavota G, Di Penta M, Oliveto R, Lanza M (2014) Mining stackoverflow to turn the ide into a self-confident programming prompter. In: Proceedings of the Working Conference on Mining Software Repositories. ACM, pp 102–111

Porter MF (1997) An algorithm for suffix stripping. In: Sparck Jones K, Willett P (eds) Readings in information retrieval. Morgan Kaufmann Publishers Inc, pp 313–316

Powers DM (2011) Evaluation: from precision, recall and f-measure to roc, informedness markedness and correlation

Read J, Pfahringer B, Holmes F, Frank E (2011) Classifier chains for multi-label classification. Mach Learn 85(3):333

Rosen C, Shihab E (2015) What are mobile developers asking about? a large scale study using stack overflow. Empir Softw Eng 21:1–32

Scalabrino S, Bavota G, Russo B, Oliveto R, Di Penta M (2017) Listening to the crowd for the release planning of mobile apps. IEEE Transactions on Software Engineering

Torgo L (2016) Data mining with r: learning with case studies. CRC Press, Boca Raton

Treude C, Barzilay O, Storey MA (2011) How do programmers ask and answer questions on the web? (NIER Track). In: Proceedings of the International Conference on Software Engineering. ACM, pp 804–807

Villanes IK, Ascate SM, Gomes J, Dias-Neto AC (2017) What are software engineers asking about android testing on stack overflow?. In: Proceedings of the Brazilian Symposium on Software Engineering. ACM, pp 104–113

Villarroel L, Bavota G, Russo B, Oliveto R, Di Penta M (2016) Release planning of mobile apps based on user reviews. In: Proceedings of the International Conference on Software Engineering. ACM, pp 14–24

Wen J, Sun G, Luo F (2016) Data driven development trend analysis of mainstream information technologies. In: Proceedings of the International Conference on Service Science. IEEE, pp 39–45

Wu Y, Wang S, Bezemer CP, Inoue K (2018) How do developers utilize source code from stack overflow? Empir Softw Eng 24:1–37

Yang X, Lo D, Xia X, Wan Z, Sun J (2016) What security questions do developers ask? a large-scale study of stack overflow posts. J Comput Sci Technol 31(5):910–924

Zhang Y, Hou D (2013) Extracting problematic api features from forum discussions. In: Proceedings of the International Conference on Program Comprehension. IEEE, pp 142–151

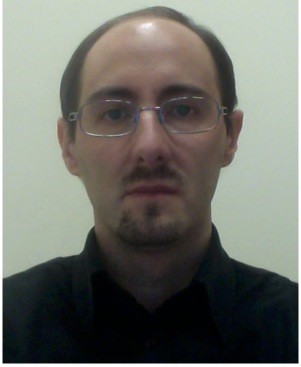
Zhang P, Elbaum S (2014) Amplifying tests to validate exception handling code: an extended study in the mobile application domain. ACM Trans Softw Eng Methodol 23(4):32

Zou Y, Ye T, Lu Y, Mylopoulos J, Zhang L (2015) Learning to rank for question-oriented software text retrieval. In: Proceedings of the International Conference on Automated Software Engineering. IEEE, pp 1–11

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Stefanie Beyer is a PostDoc assistant in the Software Engineering Research Group at the Alpen-Adria Universität Klagenfurt in Austria. In 2013, she received a MSc in Software Enginnering and Internet Computing from the Technical University of Vienna and in 2018, she received a PhD in Informatics from the Alpen-Adria Universität Klagenfurt, both with distinction. Her PhD thesis was supervised by Prof. Martin Pinzger. Her research interests include mining software repositories, program analysis, and software evolution.

Christian Macho is a PostDoc assistant in the Software Engineering Group (SERG) at the Alpen-Adria-Universität Klagenfurt. He received his MSc from the Technical University Vienna in March 2015 and his PhD from the Alpen-Adria-Universität Klagenfurt in 2019 both with distinction. His research interests include software evolution, mining software repositories, program analysis, build systems, continuous integration, automated repair, and empirical studies in software engineering.



Massimiliano Di Penta is an associate professor at the University of Sannio, Italy. His research interests include software maintenance and evolution, mining software repositories, empirical software engineering, search-based software engineering, and service-centric software engineering. He is an author of over 270 papers appeared in international journals, conferences, and workshops. He serves and has served in the organizing and program committees of more than 100 conferences, including ICSE, FSE, ASE, ICSME. He is in the editorial board of the Empirical Software Engineering Journal edited by Springer, ACM Transactions on Software Engineering and Methodology, and of the Journal of Software: Evolution and Processes edited by Wiley, and has served the editorial board of the IEEE Transactions on Software Engineering.

Martin Pinzger is a full professor at the University of Klagenfurt, Austria where he is heading the Software Engineering Research Group. His research interests are in software evolution, mining software repositories, program analysis, software visualization, and automating software engineering tasks. He is a member of ACM and a senior member of IEEE.

Affiliations

Stefanie Beyer[1] · **Christian Macho**[1] · **Massimiliano Di Penta**[2] · **Martin Pinzger**[1]

Christian Macho
christian.macho@aau.at

Massimiliano Di Penta
dipenta@unisannio.it

Martin Pinzger
martin.pinzger@aau.at

[1] University of Klagenfurt, Klagenfurt, Austria

[2] University of Sannio, Sannio, Italy