Automatically Classifying Posts into Question Categories on Stack Overflow

Stefanie Beyer, Christian Macho, Martin Pinzger
University of Klagenfurt
Klagenfurt, Austria
<firstname>.<lastname>@aau.at

Massimiliano Di Penta
University of Sannio
Sannio, Italy
dipenta@unisannio.it

ABSTRACT

Software developers frequently solve development issues with the help of question and answer web forums, such as Stack Overflow (SO). While tags exist to support question searching and browsing, they are more related to technological aspects than to the question purposes. Tagging questions with their *purpose* can add a new dimension to the investigation of topics discussed in posts on SO. In this paper, we aim to automate such a classification of SO posts into seven question categories. As a first step, we have manually created a curated data set of 500 SO posts, classified into the seven categories. Using this data set, we apply machine learning algorithms (Random Forest and Support Vector Machines) to build a classification model for SO questions. We then experiment with 82 different configurations regarding the preprocessing of the text and representation of the input data. The results of the best performing models show that our models can classify posts into the correct question category with an average precision and recall of 0.88 and 0.87 when using Random Forest and the phrases indicating a question category as input data for the training. The obtained model can be used to aid developers in browsing SO discussions or researchers in building recommenders based on SO.

ACM Reference Format:
Stefanie Beyer, Christian Macho, Martin Pinzger and Massimiliano Di Penta. 2018. Automatically Classifying Posts into Question Categories on Stack Overflow . In *ICPC '18: ICPC '18: 26th IEEE/ACM International Confernece on Program Comprehension , May 27–28, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3196321.3196333

1 INTRODUCTION

The popularity and importance of question and answer forums, such as Stack Overflow (SO), is high since they provide an important source for helping software developers in solving their development issues. The reasons of developers to ask questions on SO are diverse and recent research shows that it is not sufficient to investigate only the topics discussed on SO [4]. On the one side, developers leverage SO tags to support their search and browsing activities. On the other side, 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, API usage, providing some opinions about a given technology, or else some more general, conceptual suggestions. Therefore, the capability of categorizing questions based on the reasons *why* they are asked is needed to determine the role that SO plays for software developers [27]. Furthermore, as found by Allamanis *et al.* [1], 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 can support SO-based recommender systems integrated into the IDE, such as Seahawk and Prompter from Pozanelli *et al.* [23, 24].

Existing studies already aim at extracting the problem and question categories of posts on SO by applying *manual* categorizations [27, 31], topic modeling [1], or k-nearest-neighbor (k-NN) clustering [5]. However, the manual approaches do not scale to larger sets of unlabeled posts. The unsupervised topic modeling approach cannot directly be used to evaluate the performance of the classification of posts 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.

The goal of this paper is two-fold, *i.e.*, (i) to build a common taxonomy for classifying posts into question categories, and (ii) to investigate how, and to what extent we can classify SO posts into such categories. Regarding the *question categories*, we start from the definition provided by Allamanis *et al.* [1]:

> *"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 *question categories*, and then try to automatically classify SO posts into these categories. Android is one of the topics with the most increasing popularity on SO [3, 34] and several previous studies [5, 27] 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?

ICPC '18, May 27–28, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5714-2/18/05…$15.00
https://doi.org/10.1145/3196321.3196333

We answer this question by analyzing the question categories and reasons for questions found in the existing studies [1, 4, 5, 27, 31], 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 500 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.

This set of posts and phrases is then used for building models to automate the classification of posts using the supervised machine learning algorithms Random Forest (RF) [7] and Support Vector Machine (SVM) [11]. We study various configurations of the input data, which leads to our second research question:

RQ-2 What is the best configuration to automate the classification of SO posts into the 7 question categories?

We run four experiments using RF and SVM either with the text or with the phrases as input text for training the classification models for each question category. Furthermore, we run each experiment with 82 different configurations regarding the text representation, stop word removal, pruning, and re-sampling of the input data. We then compare the performance of the models measured in terms of precision, recall, f-score, Area Under the Receiver Operating Characteristics Curve (AUC), and accuracy to determine the best configuration. In our experiments, the best results are achieved when using RF with the phrases of the post as input.

Finally, we evaluate the performance of these models on an independent test set of 100 SO posts and by comparing it to the performance of the Zero-R classifier. This leads to our third research question:

RQ-3 What is the performance of our models to classify SO posts into the 7 question categories?

The results show that our models can classify SO posts into the seven question categories with an average precision of 0.88, recall of 0.87, and f-score of 0.87. The comparison with the Zero-R classifier shows that our models clearly outperform the Zero-R models for all question categories.

Our results have several implications for developers and researchers. Integrating our models into SO, developers can search by question category. For example, developers can use our models to find API specific challenges by question category. Also, the classification can be leveraged by researchers to build better SO-based recommenders. 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 1147 phrases of 500 posts to 7 question categories.
- An approach to automatically classify posts into the 7 question categories.
- An evaluation of the performance of RF and SVM for the classification of posts into each question category.

Furthermore, we provide all supplementary material that allows the replication and extension of our approach.[1]

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. Analyzing the prior studies of Allamanis *et al.* [1], Rosen *et al.* [27], Treude *et al.* [31], and Beyer *et al.* [4, 5] 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* [1], as well as the category *How-to* [5, 31], and the *Interaction of API classes* [4]. 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* [1], *Discrepancy* [31], *What is the Problem...?* [5], as well as *Why*.[2] 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 [5, 31]. Furthermore, it overlaps with the category *Why* [27].[2] 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* [31], the category *Better Solution* [5], and *What* [27],[3] as well as *How/Why something works* [1].[4] 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* [31] and subsumes the categories *Why...?* and *Is it possible...?* [5]. Furthermore, it merges the categories *What* [27][3] and *How/Why something works*[4] [1]. 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* [5] and *API Changes* [4]. 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] https://github.com/icpc18submission34/icpc18submission34

[2] The category *Why* from Rosen *et al.* [27] dealing with questions about non working code, errors, or unexpected behavior is split into DISCREPANCY and ERRORS.

[3] Rosen *et al.* [27] merge abstract questions, questions about concepts, as well as asking for help to make a decision into the question category *What*.

[4] Allamanis *et al.* [1] merge questions about understanding, reading, explaining and checking into the category *How/Why something works*.

Table 1: Our 7 question categories harmonized from the five prior approaches [1, 4, 5, 27, 31].

?	Rosen *et al.* [27]	Allamanis *et al.* [1]	Treude *et al.* [31]	Beyer *et al.* [5]	Beyer *et al.* [4]
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 explained, e.g. "iphone - Coremotion 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.	
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.	
CONCEPTUAL		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?".		
			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			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.

Learning. This category merges the categories *Learning a Language/Technology* [1] and *Tutorials/Documentation* [4]. 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 500 Android-related SO posts into the seven question categories. With the result, we answer the first research question "What are the most frequently used question categories of Android posts on SO?".

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 500 posts from SO. These posts were then manually labeled by two researchers of our department as follows: Each person got a set of 500 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 450 posts were labeled by each investigator separately. We calculated the *Fleiss-Kappa* inter-rater agreement [12] 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. 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.

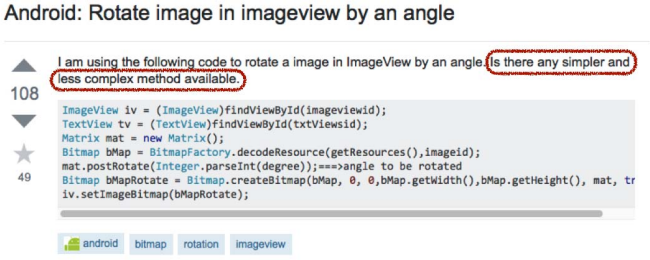

Figure 1: Question 8981845 from SO with the phrase marked in red that is indicating the question category Review.

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 additional 100 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.

3.2 Results

In total, we manually analyzed 535 posts. For 500 posts, we could identify 1147 phrases leading to a question category which allows us to draw our conclusions with 95% confidence and 5% margin of error. For 35 posts, we could not find any phrase that indicates one of our seven question categories. The post 17485804[5] 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 500 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 206 out of the 500 posts (41,2%) and 293 phrases. 145 times the question category was identified by the phrase "*how to*". The second most frequently assigned question category is Conceptual with 145 posts (29% of the posts) and 211 phrases. The phrase "*is there a/any way to*" is the most frequently occurring phrase, namely 36 times, to identify this question category. Interestingly, the question category with the second highest number of phrases, namely 246, is Errors contained by 93 posts (18,6%). As mentioned before, 30 posts (6%) each were assigned to the question categories API change and Learning. Note that the post counts sum up to more than 500 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 206 out of 500 (41,2%), fall into the question category API usage followed by the categories Conceptual with 145 posts (29%) and Discrepancy with 129 posts (25,8%).

Our findings confirm the results of the prior studies presented in [5, 27, 31] 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 AUTOMATED CLASSIFICATION

In this section, we first describe the setup of the experiments to automatically classify posts into question categories. Then, we present our approach and the results to determine the best configuration for the classification.

4.1 Experimental Setup

Previous research on the efficiency of machine learning algorithms in text classification tasks shows that classical, supervised machine learning algorithms, such as Random Forest (RF) or Support Vector Machine (SVM), can perform equally well or even better than deep

[5] 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	206	293	how to (145), how can/could I (75), how do I (28)
Conceptual	145	211	is there a/any way to (36), what is the difference between/the use of/the purpose of (26), can I use (25), is it possible to (21)
Discrepancy	129	206	i try/tried to (60), do/does not work (45), what is/am i doing wrong (26), solve/fix/I have the problem (24)
Errors	93	246	(fatal/uncaught/throwing) exception (130), get/getting/got (an) error(s) (34)
Review	79	101	is there a better/best/proper/correct/more efficient/simpler way to (32), (what) should I use/switch/do (13), is this/my understandings right/wrong (8)
API change	30	54	before/after (the) update/upgrade (to API/version/level) (14), work above/below/with API level/android/version $x.x$ (but) (6)
Learning	30	36	suggest/give me/find (links to) tutorial(s) (21)

learning techniques [14]. 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 [7] and SVM [11] 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.

A post can be classified into 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* [26], we transform the multi-label classification into a binary classification: We train 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.

For the training and testing of the models, we used the set of 500 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.

Furthermore, we investigated whether part-of-speech patterns indicate question categories, following a similar approach as Chaparro *et al.* [8] for bug reports. To get the part-of-speech tags, we used spaCy,[6] a Python-based part-of-speech tagger that has been shown to work best for SO data compared to other NLP libraries [22]. 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 already approved approach to classify text, such as the one successfully used by Villarroel *et al.* [33] and Scalabrino *et al.* [29], 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 when the data set is not large.

To determine the configuration that yields the best results, we ran 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. 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 the posts, we lowercased and stemmed the text using *R*'s implementation of Porter's stemming algorithm [25].

Stop words. 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 experimented also with pruning tokens

[6] https://spacy.io

occurring in more than 50% of the posts, as suggested in the default settings, but obtained the best results using 80% as threshold.

Re-sampling. Considering the distribution of the question categories presented in Table 2 in Section 2, we noticed that our data set is unbalanced. For instance, the most frequently found question category API USAGE is found 293 times in 206 posts, and the least frequently found question categories LEARNING and API CHANGE are found 36 and 54 times, respectively, in 30 posts. To deal with the unbalanced dataset, we re-balanced our training set using SMOTE [9]. 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 [30].

Overall, we obtained 82 different configurations of our input data: 20 when TXT is used, 10 when POS is used, and 52 different configurations when COMBI is used. We used each configuration to compute a model for each of the 7 question categories.

To measure and compare the performance of the models, we computed the accuracy, precision, recall, f-score, and auc metrics for each run. 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).

4.2 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) × 82 (configurations) × 2 (RF or SVM) × 2 (full text or phrases) = 2,296 experiments. Also, we ran each of these experiments 20 times using the stratified sampling described before. We limited the number of runs to 20 since the large number of experiments took 4 days to compute on a machine with 128 GB RAM and 48 cores.

For each experiment, we computed the performance metrics accuracy, precision, recall, f-score, and auc averaged over the 20 runs. To determine the *best* performing configuration out of the 82 configurations of input type (TXT, POS, COMBI), stop words (T, F), pruning (T, F), n-grams (n_{txt}, n_{pos}), and re-sampling (T, F), we used the weighted average f-score 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 precision and recall curve is more stable and gives more insights as found by Saito *et al.* [28]. 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.

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 3 shows the configurations and performance values for each question category with the highest weighted average f-score on 20 runs obtained with RF. Table 4 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. Comparing the values for the f-score, the best models obtained with both, RF and SVM, show an overall f-score (f_{avg}) of 0.81. Comparing the results per question category, the models computed with SVM slightly outperform the models computed with RF in five question categories. RF shows a higher f-score only for the model for classifying the question category API USAGE, with an f-score of 0.89 compared to 0.72 obtained by the SVM model. Regarding the question category DISCREPANCY, both classifiers perform equally well with an f-score of 0.72. In sum, SVM performs slightly better than RF when using the full text as input for the models.

Results using the phrases. In the second experiment, we used the phrases of the posts to train the classification models. Note, as mentioned before, we considered the full text of the post for testing. Table 5 and Table 6 show the configurations of the best performing models and the results obtained with RF and SVM averaged over the 20 runs.

Regarding the configurations, the models with the highest f-score obtained with both, RF and SVM, differ per question category. For instance, while RF obtains the best performance for the question categories API CHANGE, CONCEPTUAL, and DISCREPANCY using the COMBI input type, it obtains the best performance using the TXT input for the other categories. Also regarding the n-grams, both classifiers obtain the best models for the seven question categories with different configurations. While RF obtains the best models without removing stop-words (F) except for the category ERRORS, most of the best SVM models are obtained by removing stop-words (T).

Table 3: Best configuration and performance over 20 runs using RF with the full text as input.

category	type	n-grams	stop words	prune	re-sample	acc	$prec_{avg}$	rec_{avg}	f_{avg}	auc
API CHANGE	pos	n=1	T	F	T	0.93	0.92	0.93	0.92	0.83
API USAGE	combi	n_{txt}=1 n_{pos}=2	F	T	T	0.89	0.89	0.89	0.89	0.95
CONCEPTUAL	pos	n=1	F	F	T	0.66	0.64	0.66	0.64	0.61
DISCREPANCY	pos	n=1	T	F	T	0.74	0.72	0.74	0.72	0.68
LEARNING	pos	n=1	F	F	T	0.94	0.91	0.94	0.92	0.65
ERRORS	combi	n_{txt}=1 n_{pos}=1	T	T	F	0.84	0.85	0.84	0.81	0.96
REVIEW	pos	n=2	T	T	T	0.85	0.76	0.85	0.78	0.71
average	-	-	-	-	-	0.83	0.81	0.83	0.81	0.77

Table 4: Best configuration and performance over 20 runs using SVM with the full text as input.

category	type	n-grams	stop words	prune	re-sample	acc	$prec_{avg}$	rec_{avg}	f_{avg}	auc
API CHANGE	combi	n_{txt}=1 n_{pos}=2	T	T	T	0.97	0.96	0.97	0.96	0.94
API USAGE	combi	n_{txt}=1 n_{pos}=2	F	T	T	0.74	0.75	0.74	0.72	0.84
CONCEPTUAL	combi	n_{txt}=1 n_{pos}=2	F	F	T	0.73	0.70	0.73	0.69	0.74
DISCREPANCY	combi	n_{txt}=1 n_{pos}=1	F	F	T	0.72	0.72	0.72	0.72	0.73
LEARNING	combi	n_{txt}=1 n_{pos}=3	T	F	T	0.95	0.92	0.95	0.93	0.84
ERRORS	combi	n_{txt}=1 n_{pos}=2	F	T	T	0.84	0.84	0.84	0.84	0.80
REVIEW	combi	n_{txt}=1 n_{pos}=3	T	F	T	0.85	0.83	0.85	0.82	0.70
average	-	-	-	-	-	0.83	0.82	0.83	0.81	0.80

Table 5: Best configuration and performance over 20 runs using RF with the phrases as input.

category	type	n-grams	stop words	prune	re-sample	acc	$prec_{avg}$	rec_{avg}	f_{avg}	auc
API CHANGE	combi	n_{txt}=1 n_{pos}=1	F	F	T	0.95	0.97	0.95	0.96	0.98
API USAGE	txt	n=2	F	T	F	0.85	0.86	0.85	0.85	0.92
CONCEPTUAL	combi	n_{txt}=2 n_{pos}=1	F	T	F	0.82	0.82	0.82	0.82	0.84
DISCREPANCY	combi	n_{txt}=2 n_{pos}=1	F	T	F	0.79	0.79	0.79	0.79	0.81
LEARNING	txt	n=1	F	F	T	0.95	0.96	0.95	0.95	0.91
ERRORS	txt	n=1	T	T	F	0.90	0.90	0.90	0.89	0.95
REVIEW	txt	n=3	F	F	T	0.90	0.91	0.90	0.88	0.82
average	-	-	-	-	-	0.88	0.89	0.88	0.88	0.89

Table 6: Best configuration and performance over 20 runs using SVM with the phrases as input.

category	type	n-grams	stop words	prune	re-sample	acc	$prec_{avg}$	rec_{avg}	f_{avg}	auc
API CHANGE	txt	n=2	T	F	T	0.95	0.91	0.95	0.93	0.76
API USAGE	combi	n_{txt}=1 n_{pos}=3	F	F	T	0.76	0.80	0.76	0.74	0.86
CONCEPTUAL	txt	n=3	F	F	T	0.75	0.78	0.75	0.69	0.71
DISCREPANCY	txt	n=2	T	T	T	0.78	0.76	0.78	0.72	0.70
LEARNING	txt	n=3	T	F	T	0.94	0.90	0.94	0.92	0.55
ERRORS	combi	n_{txt}=1 n_{pos}=3	T	F	T	0.82	0.77	0.82	0.75	0.67
REVIEW	txt	n=3	T	F	T	0.86	0.85	0.86	0.82	0.62
average	-	-	-	-	-	0.84	0.82	0.84	0.80	0.70

When comparing the performance of the models computed with RF and SVM, the average f-score (f_{avg}) of the RF models over all categories is 0.88 and clearly higher than the average f-score of the SVM models, which is 0.80. Also the values of the other performance metrics obtained by the RF models are higher than the values of the SVM models. Comparing the f-scores per question category, the RF models outperform the SVM models in 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 API CHANGE in which RF and SVM tie in terms of average accuracy (0.95) and recall (0.95). In sum, training the models using the phrases of the posts as input, the models trained with RF outperform the models trained with SVM.

Comparing the results of full text and phrases. 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 accuracy of 0.88, precision of 0.89, recall of 0.88, F-score of 0.88, and auc of 0.89, the models trained with RF using the phrases as input text clearly stand out. This finding also holds for each question category with one exception: the best model trained with RF and the full text to classify the question category API USAGE (see Table 3) shows better performance than the best model trained with RF and the phrases as input (see Table 5).

Based on these results, we answer the second research question *"What is the best configuration to automate the classification of posts into the 7 question categories?"* with: The best configurations are obtained by using RF and the phrases of the posts as input to train the classification models. On the level of question categories, the configurations shown in Table 5 are considered as the *best configurations* to classify posts into the seven question categories.

5 PERFORMANCE OF THE BEST CONFIGURATION

In this section, we report further evaluations of the best classifier models among those compared in Section 4 through a cross-validation. In this section, we first compare the performance with the Zero-R classification and, second, we apply the models to a test set of 100 posts that have not been used for training the models.

5.1 Comparison of 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.

As preparation for the comparison with the Zero-R classifier, we performed two steps. First, 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. This was done to mitigate the bias that might have been introduced by selecting the training and test data using the stratified sampling approach. Second, we also analyzed the impact of parameter tuning on the performance of the classification models. Specifically, we used the `tune` function of *R* to vary the number of trees (ntrees) for RF for computing the models of each question category. As a result, we did not find any further improvement in the performance of our models, therefore, we kept the default setting of ntree=500.

Table 7 reports the performance values of the classification models averaged on 100 runs. The table also details the performance values for $class_T$ and $class_F$. The performance values of the models obtained with the Zero-R classifier are reported in Table 8.

Comparing the values, we can see that over all seven question categories, RF outperforms Zero-R showing a higher overall average accuracy (acc) of +0.07, average precision ($prec_{avg}$) of +0.23, average recall (rec_{avg}) of +0.07, and average f-score (f_{avg}) +0.16. They only tie in the accuracy and recall for the question category API CHANGE. Using Zero-R, for each category all posts are classified 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, our approach outperforms 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 the RF. However, the RF models with the best configuration perform better in terms of precision for each of the seven question categories. Regarding the f-score, the RF models outperform Zero-R in four out of the seven question categories, namely API USAGE, CONCEPTUAL, ERRORS, and REVIEW, and tie in the other three categories.

Summarizing the results, our approach clearly outperforms the Zero-R classifier with a weighted average precision, recall, and f-score of 0.88, 0.87, and 0.87, respectively.

5.2 Evaluation with an Independent Sample-Set

As a final step, we evaluated the performance of our best performing models with an independent sample set of 100 posts that has not been used for training and testing the models.

We labeled 100 more posts following the same approach as described in Section 3.1. 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. We selected the top-100 posts where a question category was identified for this evaluation. The distribution of question categories in this data set is similar to the set of 500 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.

Applying the best models 100 times to the 100 posts, we obtained the results listed in Table 9. The results show that using the validation test our approach performs on average over all categories with a precision, recall, and f-score of 0.85, 0.83, and 0.84, respectively. This confirms the results shown by the 100 runs with the initial set of 500 posts, since the validation showed the same performance for the question categories API CHANGE, CONCEPTUAL, LEARNING, and REVIEW. For the question categories API USAGE, DISCREPANCY, and ERRORS, we observe a decrease in the f-score f_{avg} with -0.04, -0.07, and -0.10, respectively. 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 500 posts is split 100 times using stratified sampling into a test and a validation set. Hence, we assume that the results obtained from the 100 runs using the set of 500 posts for training and testing are more stable and more reliable, and use them to answer the third research question *"What is the performance of our models to classify SO posts into the 7 question categories?"* with: Using RF with the phrases of the posts as input, models can be trained that classify posts into the seven question categories with an average accuracy of 0.87, precision of 0.88, recall of 0.87, f-score of 0.87, and auc of 0.88. For further details about the evaluation, we refer the reader to our supplementary material.[1]

6 THREATS TO VALIDITY

Threats to *construct validity* include the choice of spaCy of Omran *et al.* [22] 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.* [26] 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.

Table 7: Results per question category rerunning the experiment with the best configurations 100 times.

category	acc	auc	$prec_{avg}$	rec_{avg}	f_{avg}	$prec_T$	rec_T	f_T	$prec_F$	rec_F	f_F
API change	0.94	0.96	0.97	0.94	0.95	0.56	0.89	0.66	0.99	0.94	0.97
API usage	0.87	0.93	0.87	0.87	0.86	0.86	0.81	0.83	0.87	0.90	0.89
Conceptual	0.80	0.84	0.80	0.80	0.79	0.69	0.62	0.64	0.85	0.88	0.86
Discrepancy	0.77	0.79	0.77	0.77	0.77	0.57	0.52	0.53	0.84	0.86	0.85
Learning	0.95	0.90	0.95	0.95	0.95	0.64	0.53	0.54	0.97	0.98	0.97
Errors	0.90	0.95	0.90	0.90	0.89	0.85	0.59	0.68	0.91	0.97	0.94
Review	0.89	0.79	0.89	0.89	0.87	0.87	0.39	0.52	0.90	0.99	0.94
average	0.87	0.88	0.88	0.87	0.87	0.72	0.62	0.63	0.90	0.93	0.92

Table 8: The performance of the classification of posts using Zero-R for each question category.

category	acc	auc	$prec_{avg}$	rec_{avg}	f_{avg}	$prec_T$	rec_T	f_T	$prec_F$	rec_F	f_F
API change	0.94	0.50	0.88	0.94	0.91	0.00	0.00	0.00	0.94	1.00	0.97
API usage	0.59	0.50	0.35	0.59	0.44	0.00	0.00	0.00	0.59	1.00	0.74
Conceptual	0.71	0.50	0.50	0.71	0.59	0.00	0.00	0.00	0.71	1.00	0.83
Discrepancy	0.74	0.50	0.55	0.74	0.63	0.00	0.00	0.00	0.74	1.00	0.85
Learning	0.94	0.50	0.88	0.94	0.91	0.00	0.00	0.00	0.94	1.00	0.97
Errors	0.81	0.50	0.66	0.81	0.73	0.00	0.00	0.00	0.81	1.00	0.90
Review	0.84	0.50	0.71	0.84	0.77	0.00	0.00	0.00	0.84	1.00	0.91
average	0.80	0.50	0.65	0.80	0.71	0.00	0.00	0.00	0.80	1.00	0.88

Table 9: The performance of the classification on the test set of 100 SO posts using RF and phrases as input text.

category	acc	$prec_{avg}$	rec_{avg}	f_{avg}	auc
API change	0.92	0.97	0.92	0.94	0.90
API usage	0.82	0.82	0.82	0.82	0.88
Conceptual	0.79	0.78	0.79	0.78	0.84
Discrepancy	0.72	0.71	0.72	0.70	0.78
Learning	0.94	0.94	0.94	0.94	0.73
Errors	0.79	0.79	0.79	0.79	0.94
Review	0.92	0.93	0.92	0.90	0.66
average	0.84	0.85	0.83	0.84	0.82

Threats to *internal validity* concern the selection of the posts used for manual labeling. We randomly selected 500 posts which allows us to draw conclusions with 95% confidence and with 5% margin of error which we consider as sufficient. 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 models to automate the categorization of posts since we trained and tested the models with 500 posts from SO. We mitigated this threat, first, by performing random selection and, second, by testing the models with an independent sample set of manually labeled 100 posts. This supports that our classification models are valid for the domain of Android posts. For other domains, the classification models might need to be retrained which is subject to our future work.

7 RELATED WORK

In the last years, the posts on SO were often used to investigate the categories and topics of questions asked by software developers.

Treude *et al.* [31] 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 *et al.* [27] 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 *et al.* [5] 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.* [38] used Lucene to rank and classify posts into question categories by analyzing the style of the posts' answers.

Allamanis *et al.* [1] 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*. Furthermore, they found that question categories do not vary across programming languages. In [4], 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 *et al.* [19] interviewed developers and identified 5 question types: why, how, what, whether, and discrepancy. Furthermore, Fritz and Murphy [13] investigated the questions asked by developers within a project and provided a list of 78 that developers want to ask their co-workers. In [17], 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*.Furthermore, Hou *et al.* [15] analyzed newsgroup discussions about Java Swing and present a taxonomy of API obstacles.

There is also ongoing research in topic finding on SO. Linares Vasquez *et al.* [20] as well as Barua *et al.* [3] 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 evolvution of topics over time. In [6], Beyer *et al.* presented their approach to group tag synonym pairs of SO with community detection algorithms to identify topics in SO posts.

Furthermore, several studies deal with analyzing domain specific topics on SO. Joorbachi *et al.* [16] identified the challenges of mobile app developers by interviewing senior developers. Studies from Bajaj *et al.* [2], Lee *et al.* [18], Martinez *et al.* [21], Villanes *et al.* [32], as well as Yang *et al.* [35] investigate the topics related to web development, NoSQL, cross-platform issues, security related questions, and questions about Android testing, respectively, using LDA. Furthermore, Zhang *et al.* [37] 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 *et al.* [37] used SVM to categorize the content of posts related to the Java Swing API.

As pointed out by prior studies [4, 27], 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 [1, 5, 27, 31] 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 *et al.* [1] is based on LDA, an unsupervised machine learning approach. The precision of this approach can not be evaluated. The approach by Beyer *et al.* [5] uses k-NN showing a low precision of only 41.33%.

In this paper, we analyze the existing taxonomies and harmonize them to one taxonomy. Furthermore, we argue that a post can belong to more than one question category and hence, we allow multi-labeling. Similar to prior studies [5, 27, 31], 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 use them to train the classification models. Also, the results of our evaluation show that using the phrases helps to improve the performance of the models.

8 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 [1, 4, 5, 27, 31] and harmonized them into seven question categories. Then, we manually classified 500 posts into the question categories and marked in total 1147 phrases (words, part of a sentence, or sentences) indicating a question category. To investigate how Android app developers ask questions, we analyzed which phrases are used most frequently to identify each question category.

We automated the classification of posts into question categories and applied Random Forest (RF) and Support Vector Machine (SVM) on the data. Instead of a multi label classification model, we used a binary classification and train a model for each category separately. To obtain the best setting for the models, we computed the models for each category in 82 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. We found that RF with phrases as input data showed the best classification performance. Using this configuration, we can classify posts correctly into question categories with an average precision and recall of 0.88 and 0.87, respectively.

Both researchers and developers 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 Pozanelli *et al.* [23, 24]. Indeed, our approach could allow recommenders to filter the posts according to the seven question categories, and thereby improve the accuracy of their recommendations. Furthermore, our approach can improve existing research on analyzing and identifying topics discussed on SO posts, such as presented in [3, 6, 20]. With our question categories, an orthogonal view on the topics discussed on SO is provided. This enables researchers to investigate the relationships between topics and reasons and thereby study the *what* and *why* of discussions on SO.

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. These tags help software developers searching for posts not only by topics but also by question categories. Furthermore, 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 [10, 36] 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.

For *future work*, we consider the extension of our approach to a multi-label classification and compare the results to the classification of Beyer *et al.* [5] directly. Furthermore, we plan to compare our approach to a classification based on regular expressions.

REFERENCES

[1] M. Allamanis and C. Sutton. 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, 53–56.

[2] Kartik B., Karthik P., and Ali MM. 2014. Mining questions asked by web developers. In *Proceedings of the Working Conference on Mining Software Repositories*. ACM.

[3] A. Barua, S. Thomas, and A. E. Hassan. 2012. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Software Engineering* 19 (2012), 1–36.

[4] S. Beyer, C. Macho, M. Di Penta, and M. Pinzger. 2017. *Analyzing the Relationships between Android API Classes and their References on Stack Overflow*. Technical Report. University of Klagenfurt, University of Sannio.

[5] S. Beyer and M. Pinzger. 2014. A manual categorization of android app development issues on Stack Overflow. In *Proceedings of the International Conference on Software Maintenance and Evolution*. IEEE, 531–535.

[6] S. Beyer and M. Pinzger. 2016. Grouping android tag synonyms on Stack Overflow. In *Proceedings of the Working Conference on Mining Software Repositories*. IEEE, 430–440.

[7] L. Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[8] O. Chaparro, J. Lu, F. Zampetti, L. Moreno, M. Di Penta, A. Marcus, G. Bavota, and V. Ng. 2017. Detecting Missing Information in Bug Descriptions. In *Proceedings of the Joint Meeting on Foundations of Software Engineering*. ACM, 396–407.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[10] R. Coelho, L. Almeida, G. Gousios, and A. van Deursen. 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, 134–145.

[11] C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine Learning* 20, 3 (1995), 273–297.

[12] J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.

[13] T. Fritz and G. C. Murphy. 2010. Using information fragments to answer the questions developers ask. In *Proceedings of the International Conference on Software Engineering*. ACM, 175–184.

[14] W. Fu and T. Menzies. 2017. Easy over Hard: A Case Study on Deep Learning. In *Proceedings of the Joint Meeting on Foundations of Software Engineering*. ACM, 49–60.

[15] D. Hou and L. Li. 2011. Obstacles in using frameworks and APIs: An exploratory study of programmers' newsgroup discussions. In *Proceedings of the International Conference on Program Comprehension*. IEEE, 91–100.

[16] M. E. Joorabchi, A. Mesbah, and P. Kruchten. 2013. Real Challenges in Mobile App Development. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*. ACM/IEEE, 15–24.

[17] T. D. LaToza and B. A. Myers. 2010. Hard-to-answer questions about code. In *Evaluation and Usability of Programming Languages and Tools*. ACM, 8.

[18] M. Lee, S. Jeon, and M. Song. 2018. Understanding User's Interests in NoSQL Databases in Stack Overflow. In *Proceedings of the International Conference on Emerging Databases*. Springer, 128–137.

[19] S. Letovsky. 1987. Cognitive processes in program comprehension. *Journal of Systems and software* 7, 4 (1987), 325–339.

[20] M. Linares-Vásquez, B. Dit, and D. Poshyvanyk. 2013. An Exploratory Analysis of Mobile Development Issues Using Stack Overflow. In *Proceedings of the Working Conference on Mining Software Repositories*. IEEE Press, 93–96.

[21] M. Martinez and S. Lecomte. 2017. Discovering discussion topics about development of cross-platform mobile applications using a cross-compiler development framework. *arXiv preprint arXiv:1712.09569* (2017).

[22] F. N. A. Al Omran and C. Treude. 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*. 187–197.

[23] L. Ponzanelli, A. Bacchelli, and M. Lanza. 2013. Seahawk: stack overflow in the ID. In *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 1295–1298.

[24] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, and M. Lanza. 2014. Mining StackOverflow to turn the IDE into a self-confident programming prompter. In *Proceedings of the Working Conference on Mining Software Repositories*. ACM, 102–111.

[25] M. F. Porter. 1997. An Algorithm for Suffix Stripping. In *Readings in Information Retrieval*, K. Sparck Jones and P. Willett (Eds.). Morgan Kaufmann Publishers Inc., 313–316.

[26] J. Read, B. Pfahringer, F. Holmes, and E. Frank. 2011. Classifier chains for multi-label classification. *Machine Learning* 85, 3 (30 Jun 2011), 333.

[27] C. Rosen and E. Shihab. 2015. What are mobile developers asking about? A large scale study using stack overflow. *Empirical Software Engineering* 21 (2015), 1–32.

[28] T. Saito and M. Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one* 10, 3 (2015).

[29] S. Scalabrino, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta. 2017. Listening to the Crowd for the Release Planning of Mobile Apps. *IEEE Transactions on Software Engineering* (2017).

[30] L. Torgo. 2016. *Data mining with R: learning with case studies*. CRC press.

[31] C. Treude, O. Barzilay, and M. A. Storey. [n. d.]. How Do Programmers Ask and Answer Questions on the Web? (IER Track), Year = 2011. In *Proceedings of the International Conference on Software Engineering*. ACM, 804–807.

[32] I. K. Villanes, S. M. Ascate, J. Gomes, and A. C. Dias-Neto. 2017. What Are Software Engineers Asking About Android Testing on Stack Overflow?. In *Proceedings of the Brazilian Symposium on Software Engineering*. ACM, 104–113.

[33] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the International Conference on Software Engineering*. ACM, 14–24.

[34] J. Wen, G. Sun, and F. Luo. 2016. Data driven development trend analysis of mainstream information technologies. In *Proceedings of the International Conference on Service Science*. IEEE, 39–45.

[35] X. Yang, D. Lo, X. Xia, Z. Wan, and J. Sun. 2016. What security questions do developers ask? a large-scale study of stack overflow posts. *Journal of Computer Science and Technology* 31, 5 (2016), 910–924.

[36] P. Zhang and S. Elbaum. 2014. Amplifying tests to validate exception handling code: An extended study in the mobile application domain. *ACM Transactions on Software Engineering and Methodology* 23, 4 (2014), 32.

[37] Y. Zhang and D. Hou. 2013. Extracting problematic API features from forum discussions. In *Proceedings of the International Conference on Program Comprehension*. IEEE, 142–151.

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