The Journal of Systems & Software

A Study on Classifying Stack Overflow Questions based on Difficulty by Utilizing Contextual Features

--Manuscript Draft--

Manuscript Number:	JSSOFTWARE-D-23-00013
Article Type:	VSI:SEAA-2022 [MGE - Regina Hebig]
Keywords:	Stack Overflow; Question Difficulty; Text Analysis
Corresponding Author:	Zannatun Sristy BANGLADESH
First Author:	Maliha Noushin Raida
Order of Authors:	Maliha Noushin Raida
	Zannatun Naim Sristy
	Nawshin Ulfat
	Sheikh Moonwara Anjum Monisha
	Md. Jubair Ibna Mostafa
	Md. Nazmul Haque
Abstract:	Technical question-answering sites like Stack Overflow are gaining enormous attention from practitioners of specialized fields looking to exchange their programming knowledge. They ask questions on different topics with varying degrees of complexity and difficulty. All practitioners do not have the same level of expertise on those topics to respond to such questions. However, the current approach used by Stack Overflow mostly filters questions based on topics alone and does not take difficulty into account. For this reason, a large percentage of questions fail to attract the attention of appropriate users, resulting in questions having no answer or a significant delay in response time. To address these limitations, we incorporate three models, TF-IDF, LDA, and Doc2Vec, to extract semantic and context-dependent features that can measure the difficulty of questions. Each of these models is paired with different classifiers along with other features to classify the questions based on difficulty. Extensive experiments on three different datasets exhibit the effectiveness of our models and Doc2Vec outperforms the other models. We also identified that the contextual features are correlated with question difficulty, and one subset of features outperforms others. The proposed approach can be beneficial for building an automatic tagger based on question difficulty.
Suggested Reviewers:	Sk Adnan Hassan, MSc Software Engineer, Walmart Global Tech skadnan@vt.edu
	Dr. Kazi Muheymin-Us-Sakib, PhD Professor, University of Dhaka sakib@iit.du.ac.bd

Powered by Editorial Manager® and ProduXion Manager® from Aries Systems Corporation

3 January, 2023

To

Editor

Journal of Systems and Software

Subject: Submission of an original research paper for publication.

Dear Concern,

I have the pleasure to submit the manuscript entitled "A Study on Classifying Stack Overflow Questions based on Difficulty by Utilizing Contextual Features" authored by Maliha Noushin Raida, Zannatun Naim Sristy, Nawshin Ulfat, Sheikh Moonwara Anjum Monisha, Md. Jubair Ibna Mostafa, and Md. Nazmul Haque to consider for publication as a research article in your prestigious journal.

The paper aims to classify stack overflow questions according to the difficulty level of each question. To do this, we extracted three kinds of contextual features to train different models and compare them to highlight the strength of the extracted features. It shows that utilizing contextual features of a question to the Doc2Vec model performs better than other state-of-the-art models to classify questions. In this research, we curated data from Stack Overflow online archive and manually annotated 1245 questions with difficulty levels. We hope that the stack overflow community will be benefited from the research.

The paper contains original research and has not been submitted/published earlier in any journal, and is not being considered for publication elsewhere. All authors have seen and approved the manuscript and have contributed significantly to the paper.

I highly appreciate a prompt review on your part and will be happy to help you with any additional materials (if required) regarding the manuscript.

Corresponding Author

Zannatun Naim Sristy

Department of Computer Science and Engineering

Islamic University of Technology

E-mail: zannatunnaim@iut-dhaka.edu

 Click here to view linked References

A Study on Classifying Stack Overflow Questions based on Difficulty by Utilizing Contextual Features

Maliha Noushin Raida[a,b], Zannatun Naim Sristy[a,b,*], Nawshin Ulfat[a,b], Sheikh Moonwara Anjum Monisha[a,b], Md. Jubair Ibna Mostafa[a,b] and Md. Nazmul Haque[a,b]

[a] Software Engineering Lab (SELab), Islamic University of Technology (IUT), Gazipur, 1704, Bangladesh
[b] Department of Computer Science and Engineering, University of Technology (IUT), Gazipur, 1704, Bangladesh

ARTICLE INFO

Keywords:
Stack Overflow
Question Difficulty
Text Analysis

ABSTRACT

Technical question-answering sites like Stack Overflow are gaining enormous attention from practitioners of specialized fields looking to exchange their programming knowledge. They ask questions on different topics with varying degrees of complexity and difficulty. All practitioners do not have the same level of expertise on those topics to respond to such questions. However, the current approach used by Stack Overflow mostly filters questions based on topics alone and does not take difficulty into account. For this reason, a large percentage of questions fail to attract the attention of appropriate users, resulting in questions having no answer or a significant delay in response time. To address these limitations, we incorporate three models, TF-IDF, LDA, and Doc2Vec, to extract semantic and context-dependent features that can measure the difficulty of questions. Each of these models is paired with different classifiers along with other features to classify the questions based on difficulty. Extensive experiments on three different datasets exhibit the effectiveness of our models, and Doc2Vec outperforms the other models. We also identified that the contextual features are correlated with question difficulty, and one subset of features outperforms others. The proposed approach can be beneficial for building an automatic tagger based on question difficulty.

1. Introduction

Developers frequently use community-driven Question and Answering (Q&A) sites like Stack Overflow (SO) to solve inquires related to programming. Every day, over 6,000 new questions are posted on SO, and approximately 10 million users follow the site [1]. The users, from beginners to experts, participate in constructive knowledge exchanges on this site, forming a dynamic programming community. Anyone can ask questions about various topics to fix their issues, and other users can respond or offer their thoughts. To make this procedure more user-friendly, SO offers several filtering and preference choices such as Interesting[1], Bounties[2], Watched Tags[3], Ignore Tags[3] for suggesting appropriate ones. However, with quantitative analysis on the live server [4], we found that it takes around 16 days to get an answer while the standard deviation varies up to 113 days. Another major concern is the growth of knowledge shared in SO, as 30% of the total questions remain unanswered.

Researchers have addressed this issue from different aspects, such as: looking into the causes of unanswered questions and identifying various factors that lead to questions becoming unanswered. For example, Wang et al. [2] conducted an empirical study on four Stack Exchange websites to figure out the causes of the slow response times from the Q&A systems. But rather than the questions, they paid more attention to the user profile. Additionally, some reasons (e.g., frequent/non-frequent users) are hard to quantify in reality, and it is not stated how these factors might be used to estimate the unresolved questions. To solve these issues, Mondal et al. [3] explored the factors that contribute to unanswered questions and suggested four models that predict potential unanswered questions. However, they did not consider the textual information from the question itself, which can help in capturing the context. Besides, it may vary depending on how difficult the question is.

Considering the aforementioned issues, researchers are now looking at the problem from a different perspective by assessing the difficulty of a question based on various features related to the question. These features can be categorized into three: ***A Priori***, the features relate to any features that can be extracted from the post body only (includes text, code, and external links) which are available immediately following the sending of a post (question), ***Pre-Hoc***, the features related to the question and the questioner that are also available before receiving any response and ***Post-Hoc***, the features that can be retrieved at a later stage when questions may have views, comments, and answers. Based on these three types of features, several approaches [4, 5] have been proposed to understand the question difficulty. They considered user (questioner/answerer) profile (number of questions/answers, reputation), question features (views, upvote, downvote), and answer features (difference between the date a question was asked and the date the answer was given, comments). However, they did not consider the contents (textual information and code snippet) of the question as a

*Corresponding author
malihanoushin@iut-dhaka.edu (M.N. Raida); zannatunnaim@iut-dhaka.edu (Z.N. Sristy); nawshinulfat@iut-dhaka.edu (N. Ulfat); moonwaraanjum@iut-dhaka.edu (S.M.A. Monisha); jubair@iut-dhaka.edu (Md.J.I. Mostafa); nazmul.haque@iut-dhaka.edu (Md.N. Haque)

[1] https://stackoverflow.com/?tab=interesting
[2] https://stackoverflow.com/?tab=bounties
[3] https://stackoverflow.help/en/articles/5611335-watch-or-ignore-tags
[4] https://data.stackexchange.com/stackoverflow/queries

feature. Using the contents of the question, Neung and Twittie [6] proposed a "concept hierarchy" method to measure the question difficulty. They measured the question scope with the help of concepts representing keywords extracted from the question dataset and calculated the scores of the correlating features. They followed two previous approaches for their work and found that those worked better together. Although, they acknowledged their work would not be applicable as a generalized approach as they only considered difficulty identification related to vocabulary words.

To solve the limitation of the vocabulary and rule-based approach, Hassan et al. [7] used Term Frequency-Inverse Document Frequency (TF-IDF) based supervised learning technique considering various ***Pre-Hoc*** and ***Post-Hoc*** features and questions of three predefined topics from SO. TF-IDF based methods can not capture the contextual and semantic information, which is very important to understand the difficulty. Also, the code snippets presented in the question bodies were not considered, which might be a promising feature for difficulty measurement. Although a number of models were developed to measure a question's complexity, it is unknown which approach works the best. Overall, existing literature hardly discusses the performance of ***Pre-Hoc*** and ***Post-Hoc*** features to classify the question based on difficulty and to make some inferences.

In this research, we fill in the gaps indicated above and propose a variety of supervised models to estimate the difficulty of SO questions by using the semantic and contextual feature from the questions as well as the user's background. The main objective of these models is to extract prominent features and classify the difficulty level of questions. These models utilized the different levels of the limited information namely, ***A Priori***, ***Pre-Hoc*** features, and lastly, the ***Post-Hoc*** feature. All the collected features are most susceptible to the difficulty of SO questions, and we further extend our research to explore the relationship between the features and difficulty level.

To accomplish our objective of the proposed models, we first manually categorized the 738 randomly selected SO questions on Java that would serve as an addition to the 507 posts that already existed but were small and lacking topic-independent data. The labeling was done into three classes, basic, intermediate, and advanced, based on their difficulties. The questions were labeled by each of the annotators while mentioning the reason for categorizing the question in a particular class. Moreover, the major voting approach was followed for the final label. The whole validation of manual labeling was executed by the experts. Since the entire labeled dataset is independent of any particular topics, it can also be considered generic.

Hence, to inquire about the details of the evaluation, we articulated the research questions as follows:

RQ 1: Which model performs well to define question difficulty level?

We implemented three supervised learning models (Tf-Idf, Topic Modeling, and Doc2Vec) with different classifiers (Random Forest, Extreme Gradient Boosting (XG-Boost), Ada-Boost, and Support Vector Machine (SVM)) to find the most appropriate one for SO data and evaluate them with the performance metrics of accuracy, precision, recall, f1-score, and AUROC. After evaluation, it was observed that Doc2Vec with XG-Boost outperformed other question difficulty classification models with an accuracy of 0.657, F1-measure of 0.632, and AUROC of 0.746 with features available as new questions appeared.

Since ***Pre-Hoc*** is a superset of ***A Priori*** features that are only available after the question is posted, therefore how it performs with respect to ***Post-Hoc*** features which are retrieved at a later stage leads to the following **RQ 2**.

RQ 2: How do ***Pre-Hoc*** features perform in comparison to ***Post-Hoc*** features to classify questions based on difficulty?

We conducted a comparative analysis of classification models using ***Pre-Hoc*** and ***Post-Hoc*** features separately. It is found that the robustness of ***Pre-Hoc*** features is relatively better for estimating the question difficulty level by using any classification models. Hence, these features are correlated with the question's difficulty. However, the correlation between these features prompts the following research question. **RQ 3**.

RQ 3: How do different features correlate with the difficulty level of a question?

To determine the relationship between features and question difficulty, we carried both qualitative and quantitative analysis. We discovered that some features (e.g., question size, LOC, accept rate, URL+image count, first and accepted answer interval) are proportionally related to the question difficulty, and some features (e.g., number of views, answer count) are inversely proportional.

The main contributions of this paper are:

- We analyzed the contextual features of posts along with some related information the user's profile (questioner/answerer) and evaluated the efficacy of these features with three state-of-the-art supervised models and four classifiers. These models are evaluated and compared with one another to find Java generalized difficulty estimation models with different feature sets, limiting the details of the questions and users.

- For newly asked questions with unknown questioners, we identified a well-tuned ***A Priori*** feature set that can correctly classify the difficulty of the question.

- We expanded the existing ***Pre-Hoc*** and ***Post-Hoc*** features to capture contextual information and to make them more comprehensive.

- We discovered the relationship between the difficulty level of each question and different features i.e. question length, line of code, user reputation, badges, and different intervals.

This research paper is organized as follows: Section 2 discusses the related works to our objective. Section 3 describes the methodology of our study where at first, the Data generation process is presented part by part. Then, different models and their implementation details are presented. The result and discussion for each model are elaborated and compared in section 4. The overall implications of our study are projected in section 5. In section 6, we mentioned all the threats and validated each of them. Lastly, in section 7, we concluded our paper and mentioned potential future opportunities to enhance the work further.

2. Related Works

Over the years, SO has become a massive repository of problems and solutions associated with programming. As an obvious outcome, it drew the attention of many scholars to conduct studies on SO to identify insightful practices, new trends, and evolutionary behaviours of users and maintainers. These studies help to improve SO as a collaborative platform for knowledge sharing. In this section, numerous research has been summarized into two parts: Artifacts Evaluation and Trends Analysis, which is discussed in section 2.1, and Stack Overflow System Improvement in section 2.2.

2.1. Artifacts Evaluation and Trends Analysis

Stack Overflow, being a question answering community, has been producing a vast amount of data on various topics. Several pieces of research [8, 9] have been conducted to extract different topics as well as cluster the data. For example, Barua et al.[10] were curious about the topics of the problems discussed over SO and proposed an LDA(latent Dirichlet allocation) based methodology to analyze the textual contents to get insights into the main topics discussed on the site. Whereas, Allamanis et al.[8, 9] also applied LDA in their research where they used topic modeling on the SO questions to figure out the programming languages, coding, and IDE-related problems that the developers are facing. On the other hand, Treude et al.[11] were more interested in the API-related data provided by SO and proposed a machine learning-based approach called SISE (Supervised Insight Sentence Extractor) to find the relative insights from the SO data and integrate the insights into the corresponding API documentation.

All the aforementioned approaches tried to find trends, topics, and discussions among users to characterize collaborative knowledge sharing. Researchers also identified a hostile environment [12, 13] that can hinder the users' engagement. This reason leads to getting fewer answers to questions. Although these studies considered the textual information of questions, they did not consider difficulty-based question analysis.

2.2. Stack Overflow system improvement

As a question and answering site, the primary purpose of SO is to allow its users to ask questions and help to get suitable answers to their questions. However, as the number of questions asked increases every day, so does the number of unanswered questions, which decreases the effectiveness of a Q&A site. To provide a more effective service, the questions should reach potential answerers who can provide suitable answers. To understand the problem, Wang et al.[2] conducted an empirical investigation on four of the most popular stack exchange websites to determine the reasons behind slow responses from Community Question Answering systems. They listed 46 factors and four dimensions: question, asker, answer, and answerer in their research.

To increase user engagement to answer the question, researchers proposed numerous approaches for question ranking[14],[15], some of the research works consider feature selection [16] to rank the question. In this domain, researchers also worked on the user expertise assessment [17], [18], [19], [20] and question difficulty estimation [6] to filter out question that can increase the user engagement.

2.2.1. User Expertise Analysis

Liu et al.[17] suggested that users can be rated based on their answering profile, and new questions can be routed to the top-rated users based on their availability. Wang et al.[21] tried to make a personalized recommendation system for new questions on Community Question Answering sites using the Twitter-LDA model and NEWHITS algorithm, while Jinaki et al.[22] worked on question routing and question assignment to users with the suitable expertise, and Wang et al.[19] proposed an approach for answerer recommendation system. Diyanati et al.[20] also worked on determining user expertise by comment mining.

2.2.2. Question Difficulty Analysis

Lu Lin et al.[14] used a probability-based model to find the hard questions and proposed a question difficulty rank(KG-DRank) algorithm based on the knowledge gap. In another research work, Liu et al.[23] proposed a model for question difficulty estimation based on a competitive model while combining the concept of user expertise with the question difficulty. To mitigate the sparse data and cold start problem of [23], Wang et al.[24] combined the textual descriptions of the questions with the previous work[23] and introduced a novel approach named Regularized Competitive Model. Neung and Twitte[6] applied concept hierarchy to measure the question difficulty. Their work builds concept hierarchy from the words used. Unlike these works, Hassan et al.[7] proposed a supervised learning based difficulty-aware scoring system for Stack Overflow. On the other hand, D.Thukral et al.[25] first considered the temporal effect to estimate question difficulty; with that, they also generated a graph network for the questions of Stack Exchange and estimated the relative difficulty of the questions.

Among the research works mentioned above, a huge portion of work has been concerned with the resolved questions, overlooking unresolved questions. Even though several researchers worked with new questions [17], [21], [24], [2] and considered the problem of having new user [25],[24] they worked on questions of specific topics. However, none of

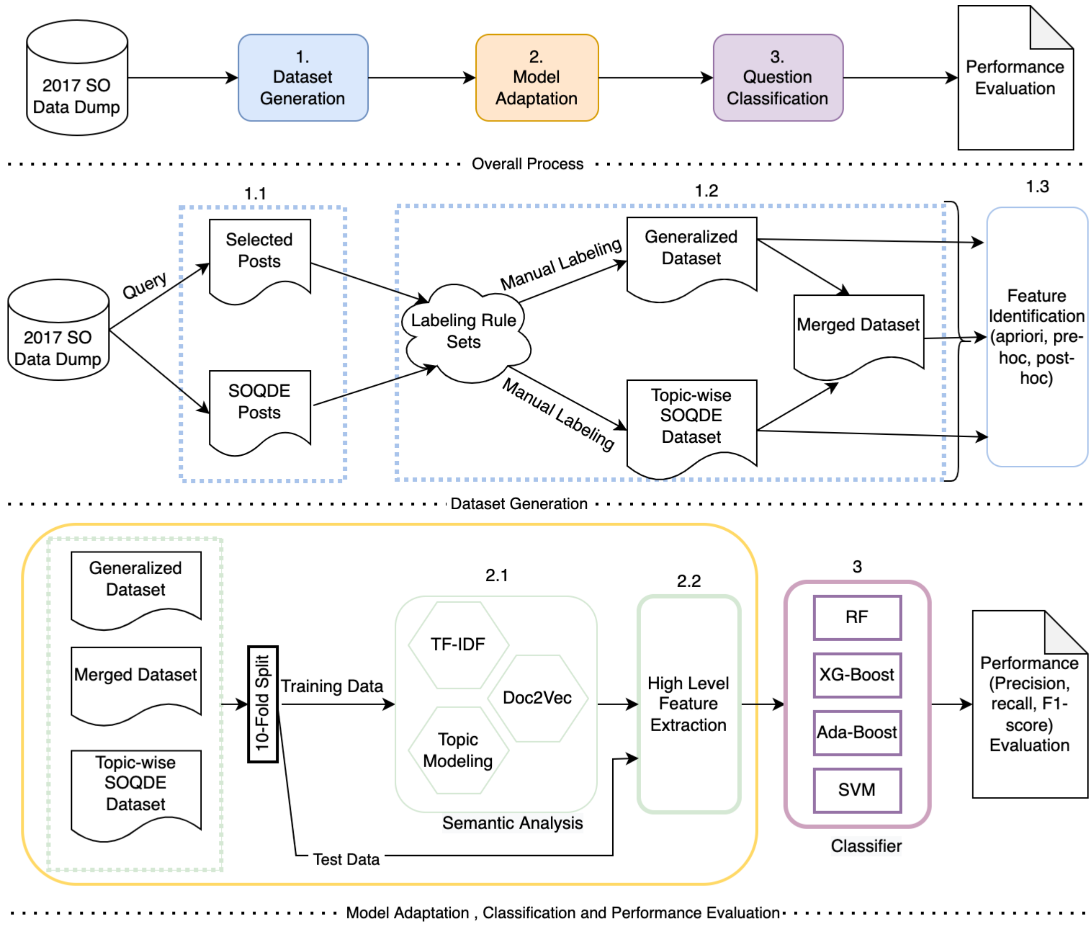


Figure 1: Overall process of difficulty based question classification, expanding the process of dataset generation, and model adaptation

those works provides a generalized approach to estimating the question difficulty.

3. Methodology

The overall approach consists of three parts, namely Dataset Generation, Model Adaption, and Question Classification, as shown in Figure 1. Dataset Generation step extracts and prepares SO questions, which will be used as input in the Model Adaptation step. This generation step engages Data Extraction, Data Labeling, and Feature Identification steps described in the subsections of 3.1. In section 3.2, Model Adaptation uses preprocessing step 3.2.1 with the Semantic Analysis and Feature Extraction to extract latent high level features, which are described in the subsection 3.2.2 and 3.2.3 respectively. In section 3.3, Classification uses various classifiers to classify questions by learning features of the Model Adaptation part. Lastly, the Performance Evaluation step utilizes different performance metrics (accuracy, precision, recall, F1-score, AUROC) to evaluate the effectiveness of different models along with the classifiers, which are discussed in subsection 3.4.

3.1. Dataset Generation

This section discusses the whole dataset generation process from the SO for the question classification models. Firstly, a stable curated data dump of SO is selected, and based on some informative criteria, a specific portion of data is extracted, which is discussed in subsection 3.1.1. Secondly, the researchers execute the labeling process, which is thoroughly described in the Data Labeling subsection 3.1.2. And lastly, in subsection 3.1.3, various features are identified which could be relevant to understanding difficulty wise question classification.

3.1.1. Data Extraction

As SO contains a diverse area of programming knowledge, including technology, domain, and language, it is not feasible to cover all aspects of programming knowledge for difficulty assessment. So, we confined ourselves to focusing only on Java related questions and answers and extracted those datasets from SO. Java is one of the popular languages that developers are using for building enterprise solutions [26]. The SO 2020 survey illustrates the fact that about 40.2%[26] of the developers are using Java, putting it in the fifth position. 38.4% of respondents who are professional developers have chosen Java as their programming language.

This information undoubtedly makes Java a beneficial language for learners. A significant amount of research has been conducted on Java to identify the key traits of the development process [27, 28]. Throughout the world, the Java programming language is broadly used, starting from websites, system software, data storage, IT infrastructure, and data science, according to a survey done in 2020 by Jetbrains [29]. Hence Java is chosen as a representative of SO questions.

To capture the overall scenario of the posts generally answered in Stack Overflow, a stable and timely distributed dataset is needed to be extracted. A significant number of research works leverage such data for the research purpose, including API documentation from SO posts [11], and connecting to Integrated Development Environment (IDE) [30], comprehensive research on the legacy data of Java community [31] and so on.

Following the data consistency, we used the SO data dump of December 2017[5], which was stored in the 2008 SQL Server database for running queries locally and extracting SO posts. The whole data dump consists of posts from 2012 to 2017, which is informative for expressing a deep-rooted effect of users' and posts' characteristics. Here, incorporating more recent data is not applicable for two reasons. First, we used only textual information of a question to extract **A Priori** and **Pre-Hoc** features. Thus, recent questions do not add any value. Second, more recent data are not stable enough for extracting Post-hoc features as the number of views, answers, comments, and the overall score may increase gradually. Moreover, we compared our approach with the SOQDE approach, which is also based on the same data dump of Java programming language. We developed a series of queries constraining on Java tag, question score, owner id, answer count, and such to get a subset of a relevant dataset that would satisfy the requirements for a certain question to be valid and generic. The queries of Listing 1, 2 and 3 are three of those queries. Other queries related to our works will be enclosed after the acceptance of the manuscript.

```sql
select Id, ViewCount,Title,AnswerCount,Tags,Score from Posts
where PostTypeId=1
and AcceptedAnswerId Is not null
and AnswerCount>0
and Score >10
and YEAR(CreationDate)= 2017
and Tags like '%<java>%'
order by AnswerCount desc;
```

Listing 1: Query to find java questions in the year of 2017.

```sql
select x.Id as QuestionId,y.Id as AnswerId
,x.CreationDate as QuestionDate,y.CreationDate as AnswerDate,
DATEDIFF(Day,x.CreationDate,y.CreationDate) as Interval
from (select * from Posts where PostTypeId=1) x LEFT JOIN
(
    SELECT *, ROW_NUMBER()
    OVER(PARTITION BY ParentId ORDER BY CreationDate) AS RowNo
    FROM Posts
    where PostTypeId=2
) y
on  x.Id=y.ParentId and y.RowNo=1
where y.Id is not NULL;
```

Listing 2: Query to calculate the difference between the creation of every question and its first answer.

```sql
select x.Id as QuestionId,x.Title,x.Score,y.Id as AnswerId,x.CreationDate as QuestionDate,y.CreationDate as AnswerDate, DATEDIFF(Day,x.CreationDate,y.CreationDate) as Interval
from (select * from Posts where PostTypeId=1) x LEFT JOIN
(
    SELECT *, ROW_NUMBER() OVER(PARTITION BY ParentId ORDER BY CreationDate) AS RowNo
    FROM Posts
  where PostTypeId=2
) y
on  x.Id=y.ParentId and y.RowNo=1
where y.Id is not NULL
and x.Tags like '%<java>%'
and DATEDIFF(Day,x.CreationDate,y.CreationDate)>30
and x.Score >0
Order by Interval desc;
```

Listing 3: Query to find java questions answered 30 days later having positive scoring.

Here, the SQL in Listing 1 represents a query to find the questions having a "java" tag in the year 2017 SO dataset, whereas Listing 3 finds the questions which have a "java" tag and the first answer was found after 30 days with a positive score. Listing 2 query was used to calculate the difference between the creation of every question and its first answer.

Out of 2000 queried questions, we randomly picked a total of 750 questions for the next step of manual labeling (described in the following subsection 3.1.2). During the manual inspection, we had to exclude 12 questions because of not having all the features needed for model training. Each post is extracted with post id, post title, and post body. To keep the labeling process unbiased, we did not extract any other information related to Q/A threads or users' history. However, after completion of labeling, contextual and user history-related features are identified and extracted for each post in the section 3.1.3.

[5] https://archive.org/details/stackexchange

Table 1
Labeling rule set for measuring question difficulty

Difficulty Class	General Rule Set	Granular Breakdown	Example Question
Basic	Questions on simple built-in functions / API documentation / beginner level knowledge	Regular Built-in-funtion	How can I pad a String in Java?
		Simple Operator / Expression	Check if at least two out of three booleans are true
		API documentation	Add a dependency in Maven
		Beginner level Theory Question	In Java, difference between package private, public, protected, and private
		Basic OOP problem	How to determine an object's class (in Java)?
	Questions related to comparison between concepts and functions of various languages	Analysis of various languages' functions	Difference between StringBuilder and StringBuffer
		Beginner level difference related query	The difference between the Runnable and Callable interfaces in Java
		Simple problem solving in other languages	What is the JavaScript version of sleep()?
	Questions about simple problem-solving or random topic	Simple problem solving	How to create a method to return 1 if input is provided 0 and 0 if provided 1 without using conditions?
		Random query	Is String.length() invoked for a final String?
	Questions with simple exception, error and other problem	Solve for nullpointer exception	Boolean.valueOf() produces NullPointerException sometimes
		Simple Error Handling	getSupportFragmentManager() shows a compile time error
		Configuration problem solved by documentation	maven build failed: Unable to locate the Javac Compiler in: jre or jdk issue
Intermediate	Questions that require a relatively deeper understanding of the language to answer, for example Why type questions	Advance features of a language that require deeper understanding	How do I use a PriorityQueue?
		Need more knowledge about the algorithims	Consistency of hashCode() on a Java string
		Multiple questions in a single post	How to launch a java program with precisely controlled execution time?
		Need knowledge on Advanced Programming topics	Logback to log different messages to two files
		Difference between two packages	Joda Time and Java8 Time difference
	Questions where the questioner knows about the answer / solution but wants to know more efficient one	Looking for contextually suitable solution despite having a solution	Copying TIMESTAMP to DATETIME on MySQL with Hibernate
		Analyzing various solutions for execution time	Java Timer vs ExecutorService?
	Questions related to time complexity, memory usage or other different resource usages of a system/solution	Efficient way	Fastest way to determine if an integer's square root is an integer
		Performance, optimization, accuracy	Memory allocation problems with android application
		Memory related	Freeing memory wrapped with NewDirectByteBuffer

Continued on next page

Table 1 – *Continued from previous page*

Difficulty Class	**General Rule Set**	**Granular Breakdown**	**Example Question**
	Questions need conceptual reasoning of programming construct /design principle	Reverse programming	How do I "decompile" Java class files?
		Underlying philosophy of any programming construct	Why use getters and setters/accessors?
		Design pattern	What's Alternative to Singleton
		Feasibility study	Setting up Java + svn + Eclipse+ Tomcat , development environment with docker
		Question about built-in documentation in details	Java 9: How to find every new method added
		Required Testing Related Knowledge	What is the proper way to setup and seed a database with artificial data for integration testing
Advanced	Questions that deal with hard/critical problems where solution needs in-depth programming knowledge or conceptual/logical thinking	Solution needs in-depth programming knowledge or conceptual thinking	JPMS and can't understand its dynamism, Can it be done in Java 9 module system
		Despite of large analysis, several unsolved issues	Incompatible types, equality constraints and method not found during Java 9 Migration
		Improvement of existing answers	How to implement retry policies while sending data to another application?
	Questions that require advanced in-depth knowledge of internal language structure	In-depth knowledge of internal language structure	Spring RedisConnectionFactory with transaction not returning connection to Pool and then blocks when exhausted
		In-depth knowledge of packages	Publish a bom from a multi-module-project
		In-depth knowledge on Garbage Collection	Latencies issues which G1GC
	Questions that deals with infrequently used functions	Deals with infrequently/rarely used framework /API /functions	Using Ebean for persistence
		Deals with depricated framework/functions/API	How do I properly map a 'MagImageScalingCallback' using JNA?
	Question that requires in-depth knowledge about software architecture & SDLC	In-depth knowledge of software architecture	JPA Clean Architecture
		In-depth knowledge of software maintenance	How do I upgrade to jlink (JDK 9+) from Java Web Start (JDK 8) for an auto-updating application?
		In-depth testing and security knowledge	RESTful Authentication via Spring
	Related to production environment	Efficiency related question	Why is AES encryption / decryption more than 3x slower on Android 24+?
		Deployment related question	GWT Deployment on Tomcat 5.5
	Question that deals with large data set and diversified topics	Data mining/ Deep learning/ Artificial Intelligence	Deeplearning4j - using an RNN/LSTM for audio signal processing
		Need in-depth knowledge of multiple topics	Unable to identify source of java.lang.ClassNotFoundException BaseDexClassLoader

3.1.2. Data labeling

Before the manual data labeling process started, it was necessary to define the rules that would be followed by every annotator. The main ruleset was retrieved from SOQDE [7], which had divided all rules into three different categories: Basic, Intermediate and Advanced. However, the majority of the time, the chosen rule set was general, incomplete, and included rules of various types. The ambiguous rules can confuse and raise questions among the annotators. So, after analyzing the whole set of guidelines for labeling, the given rules were broken down into more granular degrees for better understanding and a clearer detection of class in the labeling process. In the table 1, the first column indicates the labels to be given to the posts, the second column depicts the ruleset from the existing literature, the third column represents the breakdown of each of the main rules, and the last column exhibits an example that falls under each of the granular rules.

Figure 2: Stack Overflow question & answer thread indicating various features

. After the ruleset was made, it was given to the annotators so they could start the labeling process. By solely examining the title and the entire post body, the entire categorization process was carried out manually. The main annotation process was done by the first four authors, and experts were contacted to validate the labels. The posts were given to the annotators at random so that there wouldn't be any biases based on the topics of the posts. Each post was forwarded to more than one evaluator, and the final label was decided through anonymous majority voting. From measuring Cohen's Kappa coefficient [32] for calculating inter-rater agreement, the score $\kappa = 0.72$ showed higher agreement between the raters. The score of 0.72 lies within the (0.60 to 0.80) range, which denotes the strength of inter-rater agreement is substantially good, according to Landis et al. [33]. Furthermore, in case of any dispute or further clearance, the experts were consulted. This would rule out any chance of mislabeling a post.

In the paper [7], the dataset consisted of only three specific topics of Java: threads, inheritance, and strings. As

Table 2
Dataset wise class distribution

Dataset Name	Total No. of Samples	Class Distribution		
		Basic	Intermediate	Advanced
SOQDE	507	375	104	28
Generalized	738	360	305	73
Merged	1245	735	409	101

SOQDE [7] labeling and the labeling process mentioned above primarily follow the same generic rules, these datasets could be merged to create a larger dataset. In the Table 2, three dataset names were mentioned: one is named after the existing paper, the second one is the newly labeled dataset, and the last one is the one that has accumulated the previously named two sets. The merged dataset is used for the posts' feature identification step, described in the following sections (3.1.3).

3.1.3. Feature Identification

The labeled dataset only consists of three features: post identification number, post body, and post title. However, it is difficult to make a decision from this limited number of features. Hence, analyzing existing literature and manual inspection motivated us to extract more features. From the existing literature, [7], various features, including the question body, response time, question score, view count, etc., are identified and considered as the prominent features (Table 3). In addition, after manual inspection, some contextual information, like post owner history and answer-related features described in Table 3, are also considered.

In total, 18 features represented in Table 3 were utilized to determine the difficulty of a post. These features cover both cold-start situations having only question text and warm-start situations having contextual information for difficulty assessment. The cold-start is applicable for new questions that the users will ask, and the warm-start is applicable for tagging the existing vast amount of questions based on difficulty. Figure 2 presents an actual SO question-answering thread along with the identified features.

For the "processed Body" feature, after extracting the code snippets, the considered post body was appended to the title and tags. The code snippets are the section that is written between the anchor tag of `<code></code>`. So, each textual and code section could be featured separately for document analysis models. The user profile details were considered for every questioner and answerer, and features like reputation, accept rate, and badges were scraped from the data dump at that timestamp. We also considered details extracted from the post body, like Line of Code (LOC), code snippets, URLs, and image counts.

Extracted code snippet kept for further calculation of the line of code using Pygout[6], a python command-line tool that counts only physical lines of source code. For each snippet in a particular post, we measured the LOC metrics. The line number of codes is combined if a post has more than one code snippet. Moreover, the feature of $URL + image$ count refers to the number of hyperlinks in a certain post that might be used to clarify the posted question, including images, live codes, or questioning concepts. These features, along with the dataset, were used as input in the following Model Adaptation step 3.2.

[6] https://pygount.readthedocs.io/en/latest/

3.2. Model Adaptation

To predict a question's difficulty class, we preprocessed the post body, including the title and tags (removing code snippets), to harvest the textual features and assemble them with the features coming from Table 3. This section will describe all document analyzing models that can have a satisfactory effect on the difficulty classification for the documented questions. The overall process is described in Figure 1 (2.1, 2.2). Here, we first preprocessed the data related to the question (3.2.1). Then, the contextual information is extracted using semantic analysis (3.2.2). Lastly, We extracted the semantically high-level features and projected them into our dataset to make a distinguishable version.

3.2.1. Preprocessing

After the dataset preparation, we preprocess the new post body, excluding the code snippets. This preprocessed data is used in the following Semantic Analysis (3.2.2).

- **Title, Tags, and Textual Body Composition**:
  The title and tags from a question describe the post's main aspect and straightforward concept. After the title and tags are included in the body, the new body would represent the whole post narrative, which would help us discover a further semantic association between body, title, and tags.

- **Tokenization**:
  We tokenized each post into smaller units (words) to extract meaningful terms and their occurrences. And the word tokenizer used for the SO dataset was from the Gensim[34] library's simple_preprocess function that includes a lower casing, removing any accent marks from the sentence and storing only words with a minimum length of 3 to a list of tokens.

- **Stop words Removal**:
  Generating the list of tokens, the elimination of the stopwords, like articles, pronouns, and prepositions, was performed using the most commonly used NLTK[35] library for its documented English stopwords appending it to the Stanford CoreNLP stopwords list.

- **Stemming & Lemmatization**:
  The examination of each word to convert it to its original form was also executed using the NLTK library's well-defined Snowball stemming function. And before lemmatization, we used SpaCy[36], an open-source NLP library, with the `en_code_web_sm` model for tagging the words and allowing only 'NOUN,' 'ADJ,' 'VERB,' 'ADV' to be part of the further calculation. Now, lemmatization would take place to the words to dictionary form of words with the same SpaCy library.

Table 3
The features and their definition with associated feature list

Feature Name	Definition	Included in
Processed Body	Post full textual body excluding code snippets and the html tags like <p>, <code>,<href>	A Priori,Pre-hoc,Post-hoc
Tags	Post tags, decided at the time of posting, e.g. <java> <oop><multithread>	A Priori,Pre-hoc,Post-hoc
Title	Post title, decided by questioner	A Priori,Pre-hoc,Post-hoc
Question_Length	Length of the whole Processed Body	A Priori,Pre-hoc,Post-hoc
Url+Image_Count	Number links the post	A Priori,Pre-hoc,Post-hoc
LOC	Line of Code, counting only physical lines of source code in snippet extracted from post body Summing up all the LOCs from a certain post	A Priori,Pre-hoc,Post-hoc
User_Reputation	User Reputation Point given by Stack Overflow activities like answering, questioning	Pre-hoc,Post-hoc
User_Bronze_Badge	Number of awards for basic use of the site	Pre-hoc,Post-hoc
User_Gold_Badge	Number of awards for important contributions from members of the community	Pre-hoc,Post-hoc
User_Silver_Badge	Number of awards for being experienced users who regularly use Stack Overflow	Pre-hoc,Post-hoc
Accept_Rate	The percentage of answers accepted based on the questions asked by the user.	Pre-hoc,Post-hoc
View_Count	Number of time a certain question is viewed by users	Post-hoc
Favorite_Count	Number of times a certain question is saved as favorite	Post-hoc
Up_vote_Count	Number of up votes on a certain question for being useful and appropriate	Post-hoc
Answer Count	Number of answers in a certain question-answer thread	Post-hoc
Question_Score	The total number of upvotes it received minus the total number of downvotes it received in a question	Post-hoc
First_Answer_Interval	Interval in days between question creation date to first answer creation	Post-hoc
Accepted_Answer_Interval	Interval in days between question creation date to accepted answer creation	Post-hoc

- **Bag Of Word (BOW)**:
  The list of filtered tokens would be converted to a dictionary, having each of the words mapped to a unique identity number. And for this purpose, we used the Gensim library's corpora package. Using this dictionary, we created the BOW for each sentence.

- **Frequency Limitation**:
  We limited word frequency to eliminate the outliers not to be added to the models in further steps, which may affect the results. To identify the frequency limit, we needed to plot the frequency of each word in all the documents cumulatively. And decided to exclude words that appear less than 30 times in general after plotting the total frequency of each word appearing in the documents (i.e., posts) under consideration. We found a very small number of words having an appearance less than the set limit. And each of these words was added to the stop words and eliminated from each of the documents. For example, some of the eliminated words were related to numbers which had a rare probability of appearing in other questions in the same manner.

3.2.2. *Semantic Analysis*

In this part, we experimented with three well-known models: the TF-IDF [7], a numeric statistical measure; Latent Dirichlet Allocation (LDA) [8, 21] and the documentation vectorization technique, Doc2Vec [37].

TF-IDF considers each question as a document and each of the unique words in that document as terms to compute document-wise term distribution. To do so, we used the preprocessed data from subsection 3.2.1 and calculated the TF-IDF using Eq. 1.

$$TF\text{-}IDF_{i,j} = TF_{i,j} \times log(\frac{N}{df_i}) \tag{1}$$

Here, $TF_{i,j}$ denotes the frequency of the unique word, j in the question/post, i. N is the total number of questions. The value df_i represents the number of questions that contain the word i. The result of the TF-IDF was used to get a vector representation of features, which had more than 1600 features for each question.

Since TF-IDF can not be capable of capturing a composite representation of information in terms of topic, we considered topic modeling, a popularly used model for representing relevant topics from the question. Specifically, we used LDA [38] to find the abstract topics from any question using Eq. 2.

$$P(\beta, \theta, z, w) = (\prod_{i=1}^{K} p(\beta_i|\eta))(\prod_{d=1}^{D} p(\theta_d|\alpha)) \prod_{n=1}^{N} p(z_{d,n}|\theta_d)p(w_{d,n}|\beta_{1:K}, z_{d,n})) \tag{2}$$

Here, β,θ, and z denote the distribution of words (1^{st} part of the equation) for K number of topics, the topic proportion of a question (2^{nd} part of the equation) and topic assignment of a word in a question (last part of the equation), respectively. For yielding an LDA model, we applied Gensim's LDA model, which took the created corpus, id to word mapped dictionary from the preprocessing step, and lastly, the number of topics for finding from each of the questions. The topic number is a hyperparameter we had the chance to choose for our model. We experimented by setting topic numbers from 20 to 40, and we found that our model performs the best using the number of topics set to 23.

LDA discards some contextual information from the question with its BOW approach, as it prefers to represent the statistical relationship of occurrences. Consequently, it may lose some possible good representation and semantic information of the question because of the consideration of word order. Doc2Vec, a modified version of Word2Vec, allows learning the real semantic information representing the whole question as a vector. The architecture of the Doc2Vec is shown in Figure 3.

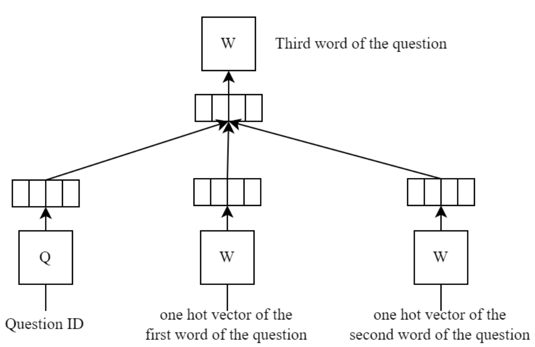


Figure 3: Architecture of Doc2Vec.

We built it with Gensim's Doc2Vec model with parameters of vector size as a variable and min_count set to 2, which would remove words having a frequency less than two and epoch number over the whole dataset. Given the training set, the first task was to construct a vocabulary dictionary from the stream of questions. Then the vocabulary dictionary and the preprocessed corpus were passed to train the Doc2Vec model. For any question, this model can infer the vector representation of that question. Users can set vector length, so we tried vector sizes from 20 to 40, and the best score of accuracy was given by the vector size 36.

3.2.3. *High Level Feature Extraction*

As it is inconvenient to classify the difficulty of a question from the low-level features, we transform these low-level features into high-level semantically rich features which have a higher classification, recognition, and segmentation capabilities. Generally, high-level features were built on top of the low-level feature, like a scratch from an image. It contains the information from the raw text (e.g., a different topic from a large document, a contextual summary of a document) that is easy to understand and recognize.

To extract the high-level, semantically rich features from the low-level dataset, after getting the high-level features from the Semantic Analysis 3.2.2 subsection, we projected these features into our dataset to make a transformed version of our dataset, which is shown in Figure 1 (2.2). This transformed dataset had high-level features with their corresponding transformed values. This transformed dataset was used for classification, described in the next subsection 3.3.

3.3. Question Classification

As all textual analysis models provided the features representing the post body in vectorized format, we incorporated the extracted textual features with the features divided into ***A Priori***, ***Pre-Hoc*** and ***Post-Hoc*** categories. We applied different multi-class classifiers to the dataset to determine the best model for extracting textual features and their correlation with the other features. The most used classifier for text classification e.g., Random Forest [39], XGBoost [40], Adaboost[41], and lastly, SVM[42].

To perform classification on the dataset, we executed K-fold cross-validation where K=10, using the Sklearn[43] library of python. And the whole models with the classifier were run ten times to train and test. The aforementioned classification models were tuned by a trial-error process using the parameters to get a better classifier. Here we discuss all the classifiers with the necessary variables that were set to conduct our experiment.

- **Random Forest:** It is an ensemble machine learning classification technique having multiple decision trees. We set 15 decision trees with a depth of 8 for each of the trees, and the criterion for trees was chosen to be entropy.

- **XGBoost:** It is a parallel tree boosting classification algorithm. We used the boosting rounds of 40 with a learning rate of 0.05. And the maximum tree depth was set to 8 as before.

- **Adaboost:** It is an iterative ensemble classification technique that combines multiple classifiers to increase the accuracy of classifiers. In our experiment, we used 1000 classifiers and a learning rate of 0.05.

- **SVM:** It is a supervised classification technique with the objective of finding a hyperplane in an n-dimensional space that distinctly classifies the questions based on their difficulties. It was built for the one-vs-rest ('ovr') decision function of shape and enabled the probability estimation to fit the training data by measuring performance metrics.

We calculated five performance metrics for each fold, such as accuracy, precision, recall, F-1 score, and AUROC using Sklearn's metrics package. After completing all folds, the average for each metric was calculated to compare the text analyzing models' performances.

3.4. Performance Evaluation

To assess the performance of different adaption models and classifiers, we considered five performance metrics: accuracy, precision, recall, F-1 score, and Area Under the Curve & Receiver Operating Characteristic curve (AUROC). Accuracy is the ratio of the samples that are predicted as true from the total samples. It is used to measure how close a given question is to its actual difficulty. The following equation is used to measure it.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

Here, *TP* and *TN* are the numbers of positive and negative samples that are correctly classified. *FP* is the number of negative-class samples misclassified as the positive class, and *FN* is the number of positive-class samples misclassified as the negative class. In contrast, accuracy is a measurement of both positive and negative samples, and precision measures only the positive samples. Precision measures the relevancy of results by computing $\frac{TP}{TP+FP}$ while recall measures truly relevant results by computing $\frac{TP}{TP+FN}$.

Having uneven class distribution, we computed F1-score and AUROC, well-known metrics for class imbalance problems. We computed F1-score to capture the weighted average of correctly identifying difficulty and total identified difficulty using the following equation.

$$F1\ Score = \frac{2 * (Recall * Precision)}{(Recall + Precision)} \tag{4}$$

Along with F1-score, *AUROC* represents the degree or measure of separability between classes, and it can be used in both balanced and imbalanced datasets, especially imbalanced datasets. *ROC* is a probability curve of a classifier at various thresholds. It plots a curve based on the true positive rate (*TPR*) and false positive rate (*FPR*) represented in Eq. 5 and Eq. 6.

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

$$FPR = \frac{FP}{FP + TN} \tag{6}$$

In these equations, *TP*, *TN*, *FP* and *FN* denote the same meaning of equation 3. To compute the points in a *ROC* curve, *AUROC* computes an aggregate measure of various thresholds. Using these metrics, we analyzed the performance of the various models and classifiers in Subsection 4.1 and 4.2. The value of these metrics lies within the range of 0 to 1. A score of 0 indicates the poor performance of the classifier, and a score of 1 indicates good performance.

Table 4
Performance metrics of Tf-Idf model, TM model and Doc2Vec model with different classifiers using A Priori Features of Question

A Priori Features of Question						
Classifier	Model	Accuracy	Precision	Recall	F1-score	AUROC
Random Forest	Tf-Idf	0.615	0.545	0.615	0.509	0.698
	TM	0.624	0.569	0.624	0.549	0.702
	Doc2Vec	0.643	0.608	0.643	0.594	0.713
XG-Boost	Tf-Idf	0.653	0.61	0.653	0.622	0.746
	TM	0.62	0.582	0.62	0.579	0.68
	Doc2Vec	**0.656**	**0.625**	**0.656**	**0.626**	**0.746**
Ada-Boost	Tf-Idf	0.643	0.577	0.643	0.59	0.712
	TM	0.632	0.586	0.632	0.581	0.612
	Doc2Vec	0.659	0.633	0.659	0.625	0.674
SVM	Tf-Idf	0.63	0.548	0.63	0.56	0.71
	TM	0.63	0.548	0.63	0.56	0.71
	Doc2Vec	0.63	0.548	0.63	0.56	0.722

Table 5
Performance metrics of Tf-Idf model, TM model and Doc2Vec model with different classifiers using Pre-hoc Features

Pre-hoc Features						
Classifier	Model	Accuracy	Precision	Recall	F1-score	AUROC
Random Forest	Tf-Idf	0.626	0.553	0.626	0.525	0.711
	TM	0.623	0.55	0.623	0.549	0.673
	Doc2Vec	0.655	0.628	0.655	0.605	0.746
XG-Boost	Tf-Idf	0.646	0.602	0.645	0.615	0.741
	TM	0.622	0.587	0.622	0.586	0.679
	Doc2Vec	**0.657**	**0.64**	**0.657**	**0.629**	**0.744**
Ada-Boost	Tf-Idf	0.644	0.577	0.644	0.591	0.71
	TM	0.621	0.582	0.621	0.576	0.61
	Doc2Vec	0.663	0.64	0.663	0.63	0.673
SVM	Tf-Idf	0.586	0.423	0.586	0.446	0.62
	TM	0.586	0.423	0.586	0.446	0.637
	Doc2Vec	0.586	0.456	0.586	0.448	0.635

4. Results & Discussion

In this section, we investigated and answered our three research questions by evaluating the performance of each model with different feature sets (***A Priori***, ***Pre-Hoc*** and ***Post-Hoc*** features) using different machine learning classifiers. Firstly, in section 4.1, we will compare the three text analysis models for each type of feature set separately and then calibrate the complete understanding to find the best model (answer of **RQ1**). In section 4.2, the efficiency of the ***Pre-Hoc*** feature set will be analyzed in contrast to the ***Post-Hoc*** feature set for different datasets (answer of **RQ2**). Finally, the relationship between features and question difficulty level will be described in section 4.3 to provide insights into the characteristics of questions of varying levels of complexity (answer of **RQ3**).

4.1. Performance of Question Classification Models

Table 4, 5 and 6 summarized the comparative results of different classification models on the merged dataset using three types of features (***A Priori***, ***Pre-Hoc*** and ***Post-Hoc*** features) and the values in boldface represent the best performing method for a particular classifier.

For ***A Priori*** features, the performance of three models using different classifiers is shown in Table 4. As we can see, for almost every classifier, the Doc2Vec model outperforms

Table 6
Performance metrics of Tf-Idf model, TM model and Doc2Vec model with different classifiers using Post-hoc Features

Post-hoc features							
Classifier	Model	Accuracy	Precision	Recall	F1-score	AUROC
Random Forest	Tf-Idf	0.618	0.554	0.618	0.52	0.743
	TM	0.647	0.613	0.647	0.597	0.721
	Doc2Vec	0.648	0.62	0.648	0.603	0.719
XG-Boost	Tf-Idf	0.663	0.637	0.663	0.638	0.762
	TM	0.644	0.61	0.644	0.617	0.734
	Doc2Vec	**0.659**	**0.636**	**0.659**	**0.632**	**0.738**
Ada-Boost	Tf-Idf	0.652	0.59	0.652	0.608	0.729
	TM	0.654	0.629	0.654	0.628	0.658
	Doc2Vec	0.667	0.643	0.67	0.646	0.692
SVM	Tf-Idf	0.594	0.487	0.594	0.454	0.677
	TM	0.594	0.482	0.594	0.454	0.676
	Doc2Vec	0.594	0.482	0.594	0.454	0.68

the other two models. This is because topic modeling finds the topics (that incorporate some words) from a document. Whereas Tf-Idf finds the most important words of the document to find the key concept of the document. Since the neighboring words and word sequencing are not taken into account by the last two models, these two models sometimes miss some relevant insights of the question. On the other hand, Doc2Vec considers the concept of a word in the surrounding of other words in a document which helps to draw more insightful conclusions to estimate the question's complexity based on tags or topics in accordance with the question scenario.

Among the classifiers, AdaBoost has the highest accuracy because of its sequentially growing learnability. But it has relatively lower coverage due to overfitting and longer time requirement from an algorithmic perspective. However, XGBoost additionally employs parameter regularization with sequential adaptation, which greatly lowers overfitting and improves the classifier as a whole, making it a better option. XGBoost with the Doc2Vec model can classify the question based on difficulty with an accuracy of 0.656, F1-score 0.626 and AUROC 0.746.

Table 5 and 6 show the performance metrics of textual models for all the classifiers with ***Pre-Hoc*** features and ***Post-Hoc*** features. For ***Pre-Hoc*** features, the Doc2Vec model using XGBoost classifier provides better performance, with accuracy, F1-score and AUROC of 0.657, 0.629 and 0.744 respectively. But surprisingly ***Post-Hoc*** feature set, the performance of the Tf-Idf model clearly surpasses that of both the TM and Doc2Vec models.

However, the datasets of the Tf-Idf model were split after calculating the Tf-Idf score for overall data to keep the feature set constant. As a result, this model gains the advantage of learning the test set earlier, which is not the case for any classification or filtering system and renders it ineffective as a question classifier in real life. Hence, we can overlook the Tf-Idf model, and in comparison to the TM model, we can infer that the Doc2Vec model performs better, with an accuracy of 0.659, an F1-score of 0.632, and an AUROC of 0.738.

Finally, as the feature set grows, the overall performance of all question classification models improves. Unlike the TM model, where adding features boosts performance significantly, the performance improvement of Doc2Vec with less important features is somewhat slower as it is more context-dependent.

Table 7
Performance comparison of Pre-hoc and Post-hoc features for TM model using different data sets

TM Model						
Metrics	Existing Dataset (Topic Wise)		Our Dataset (Random)		Merged Dataset	
	Pre-hoc	Post-hoc	Pre-hoc	Post-hoc	Pre-hoc	Post-hoc
Accuracy	0.733	0.747	0.58	0.597	0.622	0.644
Precision	0.662	0.674	0.578	0.587	0.587	0.61
Recall	0.733	0.747	0.58	0.597	0.622	0.644
F1-Score	0.683	0.696	0.567	0.585	0.586	0.617
AUROC	0.632	0.703	0.692	0.72	0.679	0.734

Table 8
Performance comparison of Pre-hoc and Post-hoc features for Doc2Vec model using different data sets

Doc2vec Model						
Metrics	Existing Dataset (Topic Wise)		Our Dataset (Random)		Merged Dataset	
	Pre-hoc	Post-hoc	Pre-hoc	Post-hoc	Pre-hoc	Post-hoc
Accuracy	0.739	0.732	0.598	0.629	0.657	0.659
Precision	0.668	0.648	0.592	0.625	0.64	0.636
Recall	0.739	0.732	0.598	0.629	0.657	0.659
F1-Score	0.689	0.675	0.578	0.617	0.629	0.632
AUROC	0.693	0.725	0.721	0.746	0.744	0.738

Answer to RQ1. In general, the Doc2Vec model with XG-Boost classifier surpasses all other models in classifying questions based on difficulty, with an accuracy of 0.657 and AUROC of 0.738. It can also filter questions with a high degree of accuracy for any unknown or new user with no prior information.

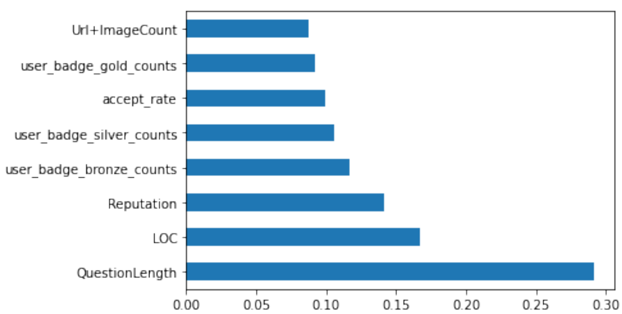
4.2. Comparative Analysis of *Pre-Hoc* and *Post-Hoc* Features

To understand the change in performance of ***Pre-Hoc*** and ***Post-Hoc*** features, we analyzed the performances of different models on topic-wise, generalized, and merged datasets. The performance comparison is important since the goal of our study is to forecast a question's difficulty level and filter them accordingly. As a result, it reduces the response time and unanswered questions. ***Pre-Hoc*** features are chosen such that they are available whenever a new question emerges, making them better for filtering purposes, whereas ***Post-Hoc*** features are only available after the question has been resolved. Since the main purpose of this comparison is to analyze the performance of the models for routing, ***A Priori*** features are not considered separately.

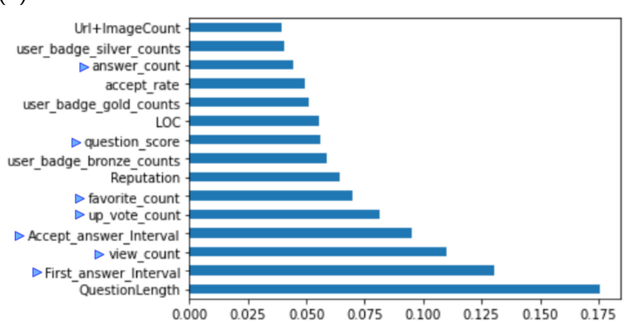
Table 7 and 8 represent the performance metrics of each textual model for ***Pre-Hoc*** and ***Post-Hoc*** features side by side in the context of different data sets. We dropped the Tf-Idf model since it is unsuitable for question filtering in a real-world setting.

According to the tables, in both models, ***Post-Hoc*** features surpass ***Pre-Hoc*** features by not more than 0.04 for any performance indicator, with slightly better coverage.

Even with random data set which incorporates a wide range of topics, ***Pre-Hoc*** features can classify questions with an accuracy of 0.58 and 0.598 using TM and Doc2Vec models, respectively. In contrast, ***Post-Hoc*** features can classify with an almost identical accuracy of 0.597 and 0.629 using TM and Doc2Vec models, respectively. This is because the extra information that ***Post-Hoc*** contain is less cohesive to other features. As a result, the effect of including those features is somewhat insignificant.



(a) ***Pre-Hoc*** Features



(b) ***Post-Hoc*** Features

Figure 4: Importance of ***Pre-Hoc*** and ***Post-Hoc*** Features

Figure 4a & 4b demonstrate the importance of each feature in Pre-hoc and Post-hoc feature sets except for question title, question content, tags, question size as they are used in the semantic analysis model (TF-IDF, TM, Doc2Vec) which in combined actually has the highest importance. In the figures, importance refers to how much a feature contributes to classifying questions based on their difficulty. The horizontal bars reflect the percentage values of the importance of features.

Among the ***Pre-Hoc*** features, the most important features are question length, LOC, and reputation. Whereas for ***Post-Hoc*** features, question length, first_answer_interval, view_count, accept_answer_interval, up_vote_count are the most significant.

Answer to RQ2. ***Pre-Hoc*** features perform nearly identical to ***Post-Hoc*** features in both the TM-based and Doc2Vec-based models, with a maximum difference of 0.04 for any performance metric. So, it can be inferred that ***Pre-Hoc*** characteristics are sufficient for filtering questions based on difficulty.

Table 9
Changes of different features according to question difficulty

Index	Features	Basic (736)	Intermediate (410)	Advanced (102)
		Avg	Avg	Avg
1	Question size	66	115	212
2	LOC	7	13	23
3	User Reputation	22193	25352	16397
4	User_Bronze_Badge	96	112	90
5	User_Gold_Badge	17	18	11
6	User_Silver_Badge	62	75	56
7	Accept Rate	58	63	64
8	View Count	189697	99439	21408
9	Answer Count	9	8	4
10	Favorite_Count	81	124	24
11	Question Score	281	289	77
12	Up_Vote_Count	283	290	78
13	First_Answer_Interval	4956	6959	21278
14	Accepted_Answer_Interval	27902	40154	73601
15	Url+Image_Count	0.3	1	2

4.3. Correlation between Features and Question Difficulty Level

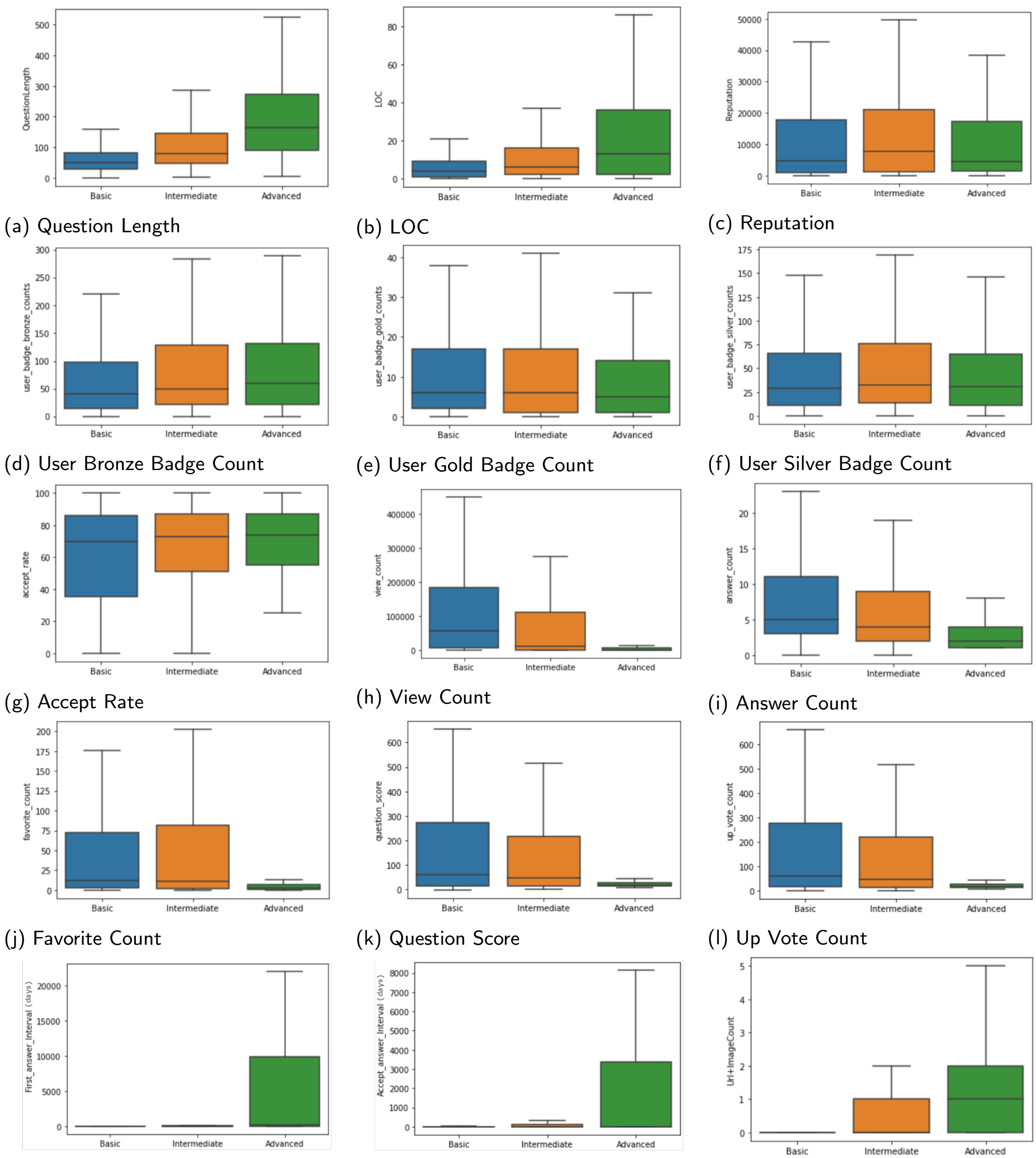
Table 9 and Figure 5 depict how the value of features change as complexity increases. It provides some interesting insights into the relationship between features and question difficulty levels which can be useful for selecting features to filter questions in future studies.

Previously researchers [7] suggested that the complexity of the question is proportional to the length of the question. Our study took a step further in this direction and elaborated that the code size measured in LOC Table 9[1, 2] & Figure 5[a, b]) is also proportionate to the difficulty of the questions. To identify the reason behind this relationship, we uncover two plausible explanations. The first is that deciphering lengthier codes is more difficult. The second issue is that people tend to include as much content (textual and code) as possible to communicate a complex subject properly. As a result, while the number of code lines increases, so does the difficulty.

With the rising complexity of a question, both the View_Count and the Answer_Count (Table 9[8][9] & Figure 5[h][i]) rapidly fall as users choose to go through the questions that they can understand.

As for the features, Favorite_Count, Question_Score, and Up_Vote_Count (Table 9[10][11][12]), we can see that on average, the intermediate level questions gain the highest score while advanced level ones receive the lowest. However, the density distribution of Question Score and Up_Vote_Count (Figure 5[k][l]) of the questions gradually falls towards a lower value as the difficulty arises. So it is safe to infer that users prefer a certain amount of brainstorming to solve a problem, but they do not want to spend too much time and mental energy comprehending a single question.

Other strong measures of the complexity of the questions are the User_Reputation and the User_Badges. However, one essential point to note is that people with a higher reputation ask intermediate-level questions, while those with a lower reputation ask the most challenging questions (Table 9[3] & Figure 5[c]). Knowing that upvotes, accepted answers, and bounty all contribute to user reputation, it is easy



(a) Question Length (b) LOC (c) Reputation

(d) User Bronze Badge Count (e) User Gold Badge Count (f) User Silver Badge Count

(g) Accept Rate (h) View Count (i) Answer Count

(j) Favorite Count (k) Question Score (l) Up Vote Count

(m) First Answer Interval (n) Accept Answer Interval (o) URL and Image Count

Figure 5: Boxplot analysis of question difficulty and density distribution of different features.

to see why people who ask intermediate-level questions have a greater reputation than those who ask tough questions.

Before moving into user badges, let us figure out what kind of badge is provided for what purpose. For basic site usage, bronze badges are awarded. Silver badges are awarded to experienced users who use the site frequently. The most dedicated users earn gold badges. On average, questioners of intermediate level inquiries have the most user badges, and questioners of advanced level queries have the least (Table 9[4][5][6]). This is because active users frequently ask efficiency-related or why-type questions, whereas infrequent users occasionally ask about critical/rare situations. On the other hand, the badges' density distribution depicted in the Figure 5[d][e][f] differs from the averages of the Table 9 data while tacitly conveying the same. According to the distribution, users who ask the more challenging questions have a higher number of bronze badges (Figure 5[d]) than those who ask the easy ones. In contrast, the questioners of intermediate level questions have more gold and silver badges (Figure 5[e][f]), whereas questioners of advanced level questions have fewer. This implies that frequent and motivated users are more interested in intermediate level questions, while fundamental users are more interested in difficult questions.

The Accept_Rate (Table 9[7] & Figure 5[g]), on the other hand, rises with difficulty level, implying that users improve their ability to ask questions based on their domain expertise. They ask less irrelevant, redundant, or unclear inquiries the more knowledgeable they are.

However, as the difficulty of the question increases, so does the First_Answer_Interval and Accepted_Answer_Interval (Table 9[13][14] & Figure 5[m][n]), and the maximum of these intervals lengthen significantly for advanced questions.

Finally, although the relationship between URL and image counts and difficulty is not particularly strong, simple questions typically contain fewer subsidiary resources than difficult questions (Table 9[15] & Figure 5[o]), indicating the fact that difficult questions require more information to convey them clearly.

Answer to RQ3. While question size, LOC, accept rate, URL+image count, as well as first and accepted answer interval, are all proportionally related to question difficulty, view and answer count is inversely proportional.
However, user reputation and badges, question score, favorite, and upvote count improve up to the intermediate level difficulty but drop for advanced level questions.

5. Implications

Difficulty-wise, SO question classification, has significant importance in facilitating coordination between knowledge seekers and appropriate responders. We discuss implications for practitioners and researchers.

5.1. Implications for Practitioners

Our analysis of questions' difficulty measurement offers both questioners and answerers to share their knowledge in the relevant domains. It reduces the chance of not getting any answers from the appropriate users for knowledge limitations or lack of motivation. For example, when a user asks a question that requires moderate or high knowledge in relevant domains to answer should not be treated equally with trivial questions. This may lead to the problem of getting no answers due to insufficient knowledge. Apart from that, users who have achieved a good reputation already are likely to not engage themselves in answering trivial questions to gain more reputations. However, low-reputed users may find much enthusiasm to answer such questions having sufficient knowledge.

We proposed different semantic models to capture the latent features of questions and various classifiers that can be used to segregate questions based on their difficulty. Adopting such techniques for understanding a new question may equip to route or attract suitable users to answer the question. As SO has a vast amount of questions, maintainers can leverage the proposed technique to filter out questions based on difficulty or add a new tagging policy. This will reduce the amount of time required to get an answer and the number of questions not having any answers.

5.2. Implications for Researchers

With its growing popularity among programmers, SO has also become a fascinating research topic. The research area of this topic ranged from analyzing current programming philosophy to providing various functional and algorithmic enhancements.

Although our study mainly focuses on providing new functionality, it requires analyzing different SO features. So we prepared three different feature sets named ***A Priori***, ***Pre-Hoc***, ***Post-Hoc*** which cover different purposes. We found that the feature sets are quite efficient for representing the nature of the question. While contextual features like question body, title, and tag provide the most important insights, the auxiliary features about the question and questioners help to predict the type of the questions.

Nevertheless, it is time-consuming for researchers to manually curate difficulty-wise categorized SO posts. The labeled dataset of 1245 posts, along with its associated feature set, is a good source for analyzing the posts of SO and the characteristics of its users.

We proposed classifiers that perform well in categorizing SO questions according to their difficulty level. This difficulty-wise classification will further help to construct a difficulty-wise question recommendation system or facilitate a personalized profile setup. Moreover, our study did not use any language or topic-specific features, so it can be useful for any question-answering site as well.

6. Threats to Validity

While conducting our study, some threats and challenging aspects hinder the validity of our work. These threats need to be mentioned to establish the effectiveness of our proposed approach. Threats are mentioned by being categorized into Internal, External, and Construct Validity. All of these are explained in the following subsections.

6.1. Internal Validity

Threats to internal validity include the models' tendency towards topic-based question difficulty classification that may overlook the actual context of the inquiry. We mitigate this threat by employing a dataset in which questions were chosen randomly regardless of the topic. Our experimental dataset has 908 distinct tags, ensuring that the training and testing set is topic diverse.

Another concerning factor is the incorporation of features that are highly related to the language since we have considered only Java language. But we did not use any language-specific feature to keep it extendable for other languages. Moreover, we selected Java because it is very popular and mature and covers a lot of programming philosophies, and the Java community exhibits the characteristics of any generic programming community.

6.2. External Validity

Threats to external validity concern the generalizability of our models for any scenario. While several features are available for simply assessing question difficulty, there can only be a limited number of features accessible to enable difficulty-based question filtering, which may reduce the models' efficiency. To address this problem, we prepare different feature sets like ***A Priori*** for completely new questions (i.e., the cold start) and ***Pre-Hoc*** features for usual situations where questioner's features are available. Using these, we trained our models accordingly for assessment. Finally, we found that, even with the cold start issue, all models have an accuracy of at least 0.60, which is fairly satisfactory.

6.3. Construct Validity

The main challenge regarding construct validity is labeling the dataset. With anonymous majority voting, we attempted to mitigate this threat. At least four researchers labeled each question separately. In case of conflicts, we discussed and resolved them through expert judgment. The Cohen's Kappa value for inter-rater agreement also indicates the validity of our labeling process.

To ensure the validity of the rules set, we focused on Hassan et al. [7] 's rules and expanded those into granular levels along with some new rules. These comprehensive rules helped to reduce any changes of disagreement and ensured the validity of our approach.

7. Future works & Conclusion

Question difficulty estimation is important to improve the routing and tagging policy of Stack Overflow. For this reason, we proposed different semantic models to determine the questions' difficulty. We utilized various contextual features and classifiers to approximate the difficulty of such questions. For experimental purposes, we curated 1245 difficulty-wise labeled questions manually, which capture generalized and topic-wise data variation. It provides a good taxonomy of rule sets for difficulty-wise question labeling as a byproduct. We implemented the semantic models with different classifiers and evaluated the performance of those models along with the features sets (***A Priori***, ***Pri-Hoc*** and ***Post-Hoc***) using multiple metrics like F1-score, AUROC. It has been found that the Doc2Vec model and XG-Boost classifier achieved the best performance than other models. From the feature usefulness analysis, we found that ***Pre-hoc*** features are sufficient to classify a question with limited information. Moreover, different features have a proportional and inverse relationship concerning the question difficulty.

The proposed approach demonstrates that applying semantic analysis like Doc2Vec is useful for representing high-level features of questions. This provides the opportunity to understand a question's hidden knowledge representation without human intervention, like moderators' actions. Researchers can use other deep learning techniques to apprehend the knowledge base. Besides, to determine the questions' difficulty, our approach uses three feature sets. Researchers can consider similar features to improve the performance of the classifiers.

In the future, we plan to work on creating a recommendation system that would recognize the hidden relations between the question types, considering the information related to the question and the user's historical information of collaboration. This will enable us to recommend the questions to appropriate users with enough expertise to answer.

References

[1] Stack Exchange, All Sites - Stack Exchange, 2009.

[2] S. Wang, T.-H. P. Chen, A. Hassan, Understanding the factors for fast answers in technical q&a websites: an empirical study of four stack exchange websites, Proceedings of the 40th International Conference on Software Engineering (2018).

[3] S. Mondal, C. K. Saifullah, A. Bhattacharjee, M. M. Rahman, C. K. Roy, Early detection and guidelines to improve unanswered questions on stack overflow, in: 14th Innovations in software engineering conference (formerly known as India software engineering conference), pp. 1–11.

[4] G. Burel, Y. He, A question of complexity: Measuring the maturity of online enquiry communities, in: Proceedings of the 24th ACM Conference on Hypertext and Social Media, pp. 1–10.

[5] C.-L. Lin, Y.-L. Chen, H.-Y. Kao, Question difficulty evaluation by knowledge gap analysis in question answer communities, in: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), IEEE, pp. 336–339.

[6] N. Viriyadamrongkij, T. Senivongse, Measuring difficulty levels of javascript questions in question-answer community based on concept hierarchy, 2017 14th International Joint Conference on Computer Science and Software Engineering (JCSSE) (2017) 1–6.

[7] S. A. Hassan, D. Das, A. Iqbal, A. Bosu, R. Shahriyar, T. Ahmed, Soqde: A supervised learning based question difficulty estimation model for stack overflow, in: 2018 25th Asia-Pacific Software Engineering Conference (APSEC), pp. 445–454.

[8] M. Allamanis, C. Sutton, Why, when, and what: Analyzing stack overflow questions by topic, type, and code, in: 2013 10th Working Conference on Mining Software Repositories (MSR), pp. 53–56.

[9] F. Rabbi, M. N. Haque, M. E. Kadir, M. S. Siddik, A. Kabir, An ensemble approach to detect code comment inconsistencies using topic modeling., in: SEKE, pp. 392–395.

[10] A. Barua, S. W. Thomas, A. Hassan, What are developers talking about? an analysis of topics and trends in stack overflow, Empirical Software Engineering 19 (2012) 619–654.

[11] C. Treude, M. P. Robillard, Augmenting api documentation with insights from stack overflow, in: 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), IEEE, pp. 392–403.

[12] C. Miller, S. Cohen, D. Klug, B. Vasilescu, C. Kästner, "did you miss my comment or what?" understanding toxicity in open source discussions, in: In 44th International Conference on Software Engineering (ICSE'22).

[13] J. Cheriyan, B. T. R. Savarimuthu, S. Cranefield, Norm violation in online communities–a study of stack overflow comments, in: Coordination, Organizations, Institutions, Norms, and Ethics for Governance of Multi-Agent Systems XIII, Springer, 2017, pp. 20–34.

[14] C.-L. Lin, Y.-L. Chen, H.-Y. Kao, Question difficulty evaluation by knowledge gap analysis in question answer communities, in: 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), pp. 336–339.

[15] L. Amancio, C. F. Dorneles, D. H. Dalip, Recency and quality-based ranking question in cqas: A stack overflow case study, Information Processing & Management 58 (2021) 102552.

[16] M. N. Haque, S. Sharmin, A. A. Ali, A. A. Sajib, M. Shoyaib, Use of relevancy and complementary information for discriminatory gene

selection from high-dimensional gene expression data, Plos one 16 (2021) e0230164.
[17] B. Li, I. King, Routing questions to appropriate answerers in community question answering services, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 1585–1588.
[18] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, Z. Chen, Cqarank: jointly model topics and expertise in community question answering, in: Proceedings of the 22nd ACM international conference on Information & Knowledge Management, pp. 99–108.
[19] L. Wang, L. Zhang, J. Jiang, Iea: an answerer recommendation approach on stack overflow, Science China Information Sciences 62 (2019) 1–19.
[20] A. Diyanati, B. S. Sheykhahmadloo, S. M. Fakhrahmad, M. H. Sadredini, M. H. Diyanati, A proposed approach to determining expertise level of stackoverflow programmers based on mining of user comments, Journal of Computer Languages 61 (2020) 101000.
[21] L. Wang, B. Wu, J. Yang, S. Peng, Personalized recommendation for new questions in community question answering, in: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), IEEE, pp. 901–908.
[22] J. Sun, S. Moosavi, R. Ramnath, S. Parthasarathy, QDEE: question difficulty and expertise estimation in community question answering sites, CoRR abs/1804.00109 (2018).
[23] Q. Wang, J. Liu, B. Wang, L. Guo, Question difficulty estimation in community question answering services, in: EMNLP.
[24] Q. Wang, J. Liu, B. Wang, L. Guo, A regularized competition model for question difficulty estimation in community question answering services, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1115–1126.
[25] D. Thukral, A. Pandey, R. Gupta, V. Goyal, T. Chakraborty, Diffque: Estimating relative difficulty of questions in community question answering services, ACM Trans. Intell. Syst. Technol. 10 (2019) 42:1–42:27.
[26] Stack Overflow, Stack Overflow Developer Survey 2020, 2020.
[27] C. F. de Castro, D. de Souza Oliveira, M. M. Eler, Identifying characteristics of java methods that may influence branch coverage: An exploratory study on open source projects, in: C. Cubillos, H. Astudillo (Eds.), 35th International Conference of the Chilean Computer Science Society, SCCC 2016, Valparaíso, Chile, October 10-14, 2016, IEEE, 2016, pp. 1–8.
[28] Y. Jiang, Research on application value of computer software development in java programming language, in: Journal of Physics: Conference Series, volume 1648, IOP Publishing, p. 032152.
[29] Jet Brains, Java Programming - The State of Developer Ecosystem in 2020 Infographic, 2020.
[30] A. Bacchelli, L. Ponzanelli, M. Lanza, Harnessing stack overflow for the ide, in: 2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE), IEEE, pp. 26–30.
[31] G. Blanco, R. Pérez-López, F. Fdez-Riverola, A. M. G. Lourenço, Understanding the social evolution of the java community in stack overflow: A 10-year study of developer interactions, Future Generation Computer Systems 105 (2020) 446–454.
[32] J. Cohen, A coefficient of agreement for nominal scales, Educational and psychological measurement 20 (1960) 37–46.
[33] J. R. Landis, G. G. Koch, The measurement of observer agreement for categorical data, biometrics (1977) 159–174.
[34] R. Rehurek, P. Sojka, Gensim–python framework for vector space modelling, NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic 3 (2011) 2.
[35] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.
[36] M. Honnibal, I. Montani, Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing, Unpublished software application. https://spacy. io (2017).
[37] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ICML'14, JMLR.org, 2014, p. II–1188–II–1196.
[38] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, Journal of machine Learning research 3 (2003) 993–1022.
[39] T. K. Ho, Random decision forests, in: Proceedings of 3rd international conference on document analysis and recognition, volume 1, IEEE, pp. 278–282.
[40] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794.
[41] R. E. Schapire, Explaining adaboost, in: Empirical inference, Springer, 2013, pp. 37–52.
[42] C. Cortes, V. Vapnik, Support-vector networks, Machine learning 20 (1995) 273–297.
[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830.

Declaration of interests

- The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
- The authors declare the following financial interests/personal relationships which may be considered as potential competing interests.