Classification of Stack Overflow Questions Based on Difficulty

by

Maliha Noushin Raida(170042001)
Zannatun Naim Sristy (170042043)
Sheikh Moonwara Anjum Monisha (170042057)
Nawshin Ulfat(170042081)

Supervised By:

Md. Jubair Ibna Mostafa
Lecturer, Dept. of Computer Science & Engineering
Md. Nazmul Haque
Lecturer, Dept. of Computer Science & Engineering

A thesis submitted to the Department of CSE in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Islamic University of Technology (IUT)
Board Bazar, Gazipur-1704, Bangladesh.
May, 2022.

Declaration of Authorship

This is to certify that the work presented in this thesis, titled **"Classification of Stack Overflow Questions Based on Difficulty"**, is the outcome of and research carried out by Maliha Noushin Raida, Zannatun Naim Sristy, Moonwara Anjum Monisha and Nawshin Ulfat under the supervision of Md. Jubair Ibna Mostafa and Md. Nazmul Haque. It is also declared that neither this thesis nor any part of it has been submitted anywhere else for the award of any degree, diploma or other qualifications. Information derived from the published or unpublished work of others has been acknowledged in the text and a list of references is given.

Authors:

Maliha Noushin Raida

Maliha Noushin Raida

Student No.: 170042001,

Date: May 12, 2022.

[illegible]

Zannatun Naim Sristy

Student No.: 170042043,

Date: May 12, 2022.

SMAMonisha

Sheikh Moonwara Anjum Monisha

Student No.: 170042057,

Date: May 12, 2022.

Nawshin

Nawhin Ulfat

Student No.: 170042081,

Date: May 12, 2022.

Supervisors:

[illegible]

Md. Jubair Ibna Mostafa

Lecturer

Dept. of Computer Science and Engineering,

Date: May 12, 2022.

Nazmul

Md. Nazmul Haque

Lecturer,

Dept. of Computer Science and Engineering

Date: May 12, 2022.

Acknowledgment

At the beginning, we want to express our heartfelt gratitude to Almighty Allah for his blessings to bestow upon us which made it possible to complete this thesis research successfully. Without the mercy of Allah, we wouldn't be where we are right now. All thanks and praises be to Allah. It is a blessed moment for us to submit our thesis work, which will ultimately bring our Bachelor of Science studies to a close.

We would like to express our gratitude to Md. Jubair Ibna Mostafa and Md. Nazmul Haque for their encouragement, recommendations, and valuable opinions for our thesis have been tremendous. Without their help and direction, the thesis would not be on the right track in the scientific field. Their insightful judgment, time, and input were contributed throughout the thesis study, from the first phase of thesis topics introduction, to implementation, helped us to do our thesis work in proper way. Their active participation and help with data set labeling and quality checking is proven to one of the most vital for the advancement of our work. And with that thanks to all the experts and reviewers for their constructive comments.

We would like to extend our vote of thanks to the jury members of my thesis committee for their insightful comments and criticism, which helped us better our manuscript.

Finally, we are deeply grateful to our friends and family for their unwavering support. Without their constant support and encouragement throughout the program, this job would not have been finished.

Table of Contents

List of Figures

List of Tables

Abstract

Technical question answering sites, like Stack Overflow, are gaining enormous attention from the learners and practitioners of specialized fields to exchange their programming knowledge. Question answering on different topics has engaged all levels of programmers. All the developers don't have the same level of expertise, and the question differs among them in terms of complexity and context. However, the existing approach of Stack Overflow models primarily filters out the questions based on tags, which is inefficient for predicting the difficulty level. Due to the limitation of the process, a large part of these posts fails to attract the attention of appropriate users, resulting in valid questions having no answer or significant delay in response time. Therefore, to address these limitations, we proposed three different supervised models using TF-IDF, Topic Modeling(LDA), and Doc2Vec that build more complicated relationships by extracting context-dependent features between the user and the question. Each of the models builds an informative relationship that helps classify the difficulty of a question. Extensive experiments on different variations of the datasets demonstrate the improved efficacy of our proposed models over contemporary models. The experiments find out that even with limited information, the models performance scores are satisfactory and the Doc2Vec model outperforms the other models under consideration.

Chapter 1

Introduction

In this chapter, we present the overview of the whole research work. The outline includes a comprehensive motivation problem explanation as well as the objective of our study. Also it discussed the contribution made with our study. Thesis organization is noted at the last part of this chapter for readers easiness.

1.1 Overview

Developers frequently use community Q&A sites like Stack Overflow to solve programming challenges. Every day, over 6,000 new questions are posted to Stack Overflow, and approximately 10 million users[1]. The users ranging from beginners to skilled, participate in constructive exchanges of knowledge on this site, forming a dynamic programming community. Anyone can ask questions about a variety of topics to fix their issues, and other users can respond or offer their thoughts on the same. To make this procedure more user-friendly, Stack Overflow offers several filtering and preference choices such as Interesting[2], Bountied [3], Watched list[4], and Ignored Tags4 for suggesting appropriate ones.

However, querying the live server[5], we found that it takes around 16 days to get an answer whilst the standard deviation varies up to 113 days. Besides, 30% of the total questions remain unanswered, which hinders the efficacy of Stack Overflow.

Many researchers have been drawn to this concern, and they have addressed it from many angles. To better understand the problem, Wang et al. [1] conducted an empirical study on four Stack Exchange websites to find out the reasons for not getting the

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

answers fast from the QnA systems. Whereas Mondal et al. [2] explored the factors responsible for remaining a question unanswered and suggested four models predicting potential unanswered questions.

Still, this challenge has not yet been resolved and researchers are now looking at it from a different perspective by assessing the difficulty of a question to solve this problem. Neung and Twitte [3] took the help of concept hierarchy to measure the question difficulty as they considered difficulty identification related to words and not the applicational logic. While Hassan et al. [4] used supervised learning considering various features and questions of three predefined topics from Stack Overflow and D. Thukral et al. [5] generated a graph network with the assistance of temporal effect to estimate the relative question difficulty.

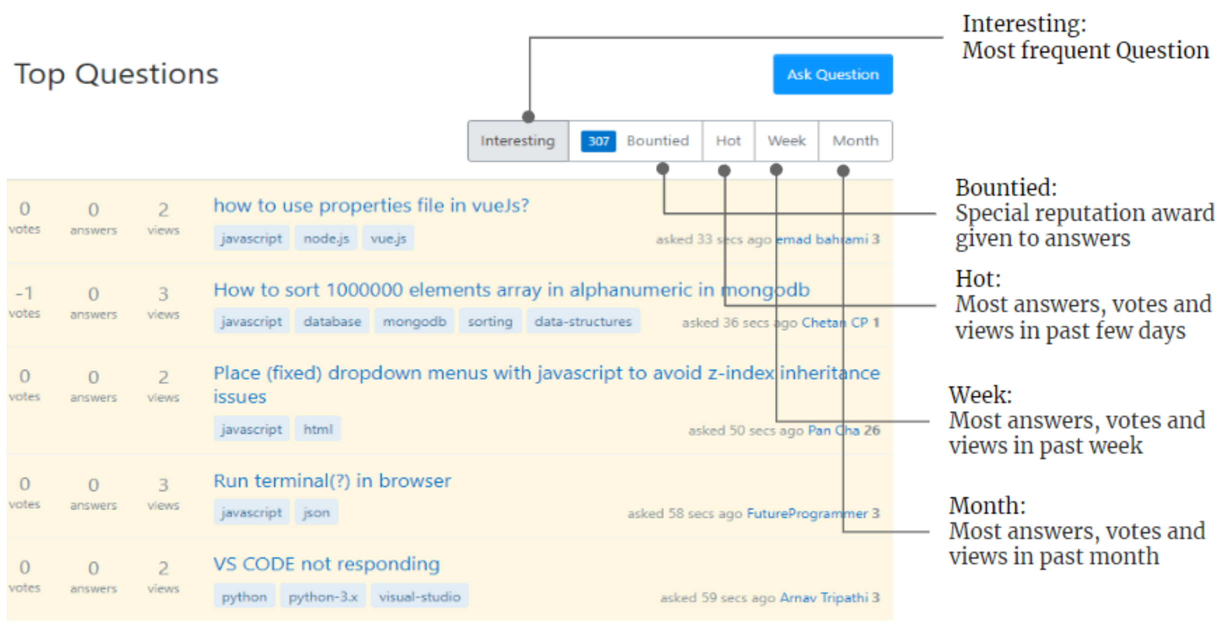
Now, we approached to estimate the difficulty of Stack Overflow posts by taking the contextual features of users and posts into account. Our work proposes and evaluates models that would work for the post classification according to difficulty.

To accomplish our objective of the proposed models, we first manually labeled the randomly collected 738 Stack Overflow questions on Java which would work as an extension to the existing dataset of 507 posts. The labelling was done into three classes basic, intermediate, and advanced. The whole dataset is generalized as it has no dependency on the sub-topics of Java. The questions were thoroughly labeled by each of the labelers while mentioning the reason for categorizing the post in a certain class. And the major voting approach was fol]owed for the final label. The whole validation of manual labeling was executed by the experts. After the labeling process, three overlapping feature sets were formulated, and each of the features was extracted to complete the dataset. We implemented three supervised learning models, Tf-Idf, Topic Modeling, and Doc2Vec, with different classifiers to find the most appropriate one for Stack Overflow data and evaluate them with the metrics of accuracy, precision, recall, and f1-score.

1.2 Motivation

Stack overflow, starting its journey in 2008, has gained tremendous popularity among programmers. The Q&A site helps the users to find the most suitable answer for their problems. As developers works quickly evolve, all it needs to meet the need of the users by lower the response time. All programmers irrespective of their background, can question about any topic related to programming on the Stack Overflow site or answer any query of their fellow programmers that they can find fit for interest and experience. From beginner to expert, any level of user has the chance to answer the same questions. This way each question is a complete environment of constructive discussion where each of the answers gives their opinion about solving the question and contributes to the knowledge bank, namely, Stack Overflow. Knowledge sharing does come with a prize in form of reputations and question upvotes. If any answer resolves the questioners doubt, the questioner would accept the answer and it would be added to the answers accept rate.

The questioning process is hard, as the wording of questions matters to make any question understandable, this depends on the questioners largely. Still, Stack Overflow supplies tools to guide users to increase straightforward. Questions in stack overflow have median time of 11 minutes [6]. On the contrary, 10.8% of the questions have no answers within 24 hours [7] which makes it necessary to recommend the questions to the user that would be able to make the best of it. Another fact, about 21.6% of questions are not answered because of failing to attract the appropriate user [8]. As a solution, many researchers worked on [6, 9] recommendation systems that would be helpful to reach out to potential users, thus less question response time and better chance to have accepted answer in questioners hour of need. But question recommendation needs to be aware of the content that is being recommended to users, whether it mates users interest and answering history. These types of characteristics have already been seen in the current Stack Overflow figure 1.1


Top Questions
Ask Question
Interesting
307 Bountied
Hot
Week
Month
Interesting: Most frequent Question
Bountied: Special reputation award given to answers
Hot: Most answers, votes and views in past few days
Week: Most answers, votes and views in past week
Month: Most answers, votes and views in past month

0 votes 0 answers 2 views — how to use properties file in vueJs? — javascript node.js vue.js — asked 33 secs ago emad bahrami 3

-1 votes 0 answers 3 views — How to sort 1000000 elements array in alphanumeric in mongodb — javascript database mongodb sorting data-structures — asked 36 secs ago Chetan CP 1

0 votes 0 answers 2 views — Place (fixed) dropdown menus with javascript to avoid z-index inheritance issues — javascript html — asked 50 secs ago Pan Cha 26

0 votes 0 answers 3 views — Run terminal(?) in browser — javascript json — asked 58 secs ago FutureProgrammer 3

0 votes 0 answers 2 views — VS CODE not responding — python python-3.x visual-studio — asked 59 secs ago Arnav Tripathi 3

Figure 1.1: Stack Overflow Recommendation System

But implemented recommendation system only works with the watchlist 1.2 selected by users. It does not ensure that the level of complexity of the questions matches with the users. Because each level of difficulty requires a different level of cognitive demand.

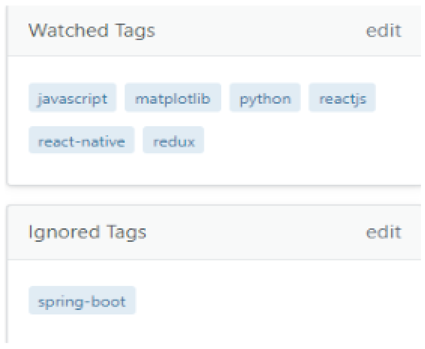


Figure 1.2: Stack Overflow WatchList System

The questions having a certain level of difficulty associated with it are beneficial for the Stack Overflow user community in multiple folds. The most intuitive ones are,

- Self-learns would have a better chance to involve in the community and increase their knowledge by participating in their level posts or discussions so that a learner might not feel demotivated while seeing their profile full of questions above their expert level. As most of the users are mostly relying upon online resources about 60%[6], Stack Overflow would be a more effective resource if it can uphold the users recommendations.

- Suggesting users according to their expertise a way to increase the chance of getting any faster than the normal system in Stack Overflow. It would be able to make sure that beginners also have a chance to answer or gain knowledge from the discussions and expert users can challenge their expertise level by trying to solve harder questions

- Topic wise difficulty categorization can help with the documentation of libraries or APIs. QnA sites like Stack Overflow can have the user perspective of the API and the need that documentation needs to fulfil.

1.3 Objective

Our study explores the difficulty estimation models, in order to determine the best model to work with the recommendation system. Till now only three research work has been published, addressing the question of difficulty. [10] works with Stack Overflow questions to create a hierarchy of topics in Javascript, providing an idea that each

[6]https://insights.stackoverflow.com/survey/2021#developer-profile-developer-roles

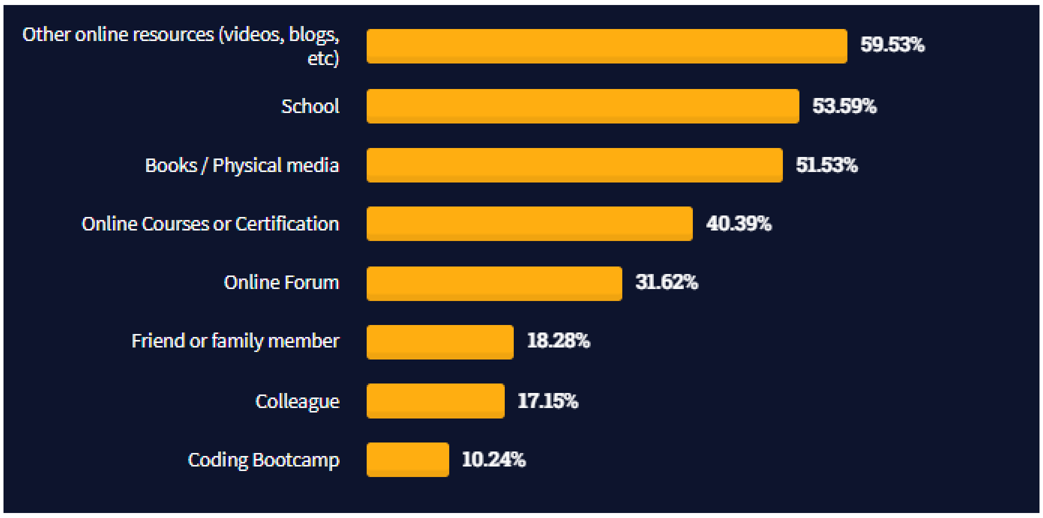


Figure 1.3: 2021 Survey of Stack Overflow users

level of the tree represents a higher complexity than its previous level and too many topics related questions also tend to be harder. Then we encounter a research work [4] that was part of our comparison, works with Stack Overflow posts with its features to create an estimation model with TF-IDF and a certain list of features related to questioners and the question itself. And last but not least, we gathered our knowledge on [5] estimating relative difficulty for Stack Overflow data, they associate temporal effect on the questions difficulty estimating network graph. But the complexity of the models does not fit well to create a recommendation system.

Our work tries to investigate different textual analysis models with three variations of the dataset having three feature lists for testing each of them to its limit. three set of features are, **semantic features** list relating to only post body, **pre-hoc** features includes features that are made public with the posting and lastly **post-hoc** features are the list consists of features that available after the answering of the posts are done. The whole approach to estimate the difficulty of Stack Overflow questions by taking the contextual features of users and posts into account. We evaluated our proposed methods in details to answer this given research questions.

- **RQ 1: Which model performs well to define question difficulty level?**
  By evaluating the proposed models, we would be able to identify the best model and extract most important features from post context.
- **RQ 2: How do pre-hoc features perform in comparison to post-hoc features in the best model?**
  Realizing importance of pre-hoc features for a recommendation system, we proceeded to find the models' performance to compare it to the post-hoc features.

- **RQ 3:How do different features correlate with the question's difficulty level?** The taken features have different inherent qualities that makes it susceptible to difficulty of Stack Overflow questions. And we answer the research question by exploring the relation the features for estimating the difficulty.

1.4 Contribution

For the question difficulty estimation model, we first gathered a dataset of posts from the data dump of Stack Overflow from 2017, because of the consistency and completeness of the dump. The dataset we have chosen is of length 738 and combined with the existing dataset from [4] , we get the total length of the merged dataset, 1245. Then we randomly selected posts from the firstly selected data list and distributed them among the authors of this paper for manual labeling them into three categories according to the certain ruleset. The labeled dataset was then fulfilled by considering the feature list of three types, one would take the syntactic features(title, body, tags) only, the second feature list is consist of the pre-hoc features, the features available with post-publication without any answers and last we considered features of post-hoc which is available for posts with answers. First, only syntactic features are important for questions having no answers and users having no previous history. The second feature list is designed for posts features and the questioners history, the last feature list is targeted toward the questions that need to have quality answers and constructive discussions. Then we chose three models for textual analysis and extract the fixed number of features from these three models. And after extracting and combining all the features (textual, post and user details). We used k=10, for k cross-validation and applied the classifiers. After generating all the metrics for evaluation, We came to the conclusion that the textual features are well extracted by Doc2vec model with vector size 36. The contribution of our research work can be list like,

- We analyzed not only the contextual features of posts but also the users and proposed three state-of-the-art models that performed better than the existing supervised model. The proposed models are evaluated and compared with one another to find an ultimately generalized difficulty estimation model with different feature sets limiting the details of the posts and users.
- For the cold start problem, we identified a well-tuned feature set that only includes question-based characteristics when the questioner is new or unfamiliar.
- We expanded the existing pre-hoc and post-hoc features to make them more comprehensive and self-contained.

- We demonstrated the changes in different features concerning the difficulty level of the question to understand the nature of different level questions, their questioners and the user's attitude towards the question.

1.5 Thesis Organization

The rest of the thesis is organized into chapters given as follows,

In chapter 2, we included all the necessary background studies that would be used further in the paper, it describes TF-IDF, the Topic Modeling technique called LDA and lastly Doc2Vec.

In chapter 3, we presented all the related works that have influenced our study directly or indirectly.

In chapter 4, we described the whole methodology, starting from the data collection, and the whole dataset creation to the last step of methodology, classification.

In chapter 5, we discussed the result generated in our methodology stage. The elaboration of the result is also included in this chapter.

In chapter 6, we mentioned all the threats and the explanation, validating all the threats from our study.

In chapter 7, the thesis works future pursuit and our plan to follow it through. And with that, it concludes the thesis work.

Chapter 2

Background Study

Inside this chapter, we discuss about the background studies that are related to our research.

2.1 Topic Modeling

Topic modeling is a type of text analysis approach in which "bags" or combinations of words are analyzed collectively, instead of individually, to identify the variety of the meaning of words depending on the context in natural language.
In 1998, Papadimitriou, Raghavan, Tamaki, and Vempala first described Topic modeling in Latent semantic indexing: a probabilistic analysis. [11]
A machine learning technique in which text data can be automatically analyzed to detect clusters of words for a set of documents. As the previously categorized and a predefined list of tags, training data are not necessary, it is referred unsupervised machine learning. Topic modeling is counting words and classifying them according to their affinitive word pattern in order to infer topics from unstructured data. By analyzing patterns like word frequency and proximity between words, the topic model can cluster relevant feedback, and also words and expressions that appear frequently. With this information, it can be immediately identified the topic of each group of texts. It is a sort of statistical modeling that is used to uncover the abstract "topics" that appear in a collection of documents by analyzing the data.

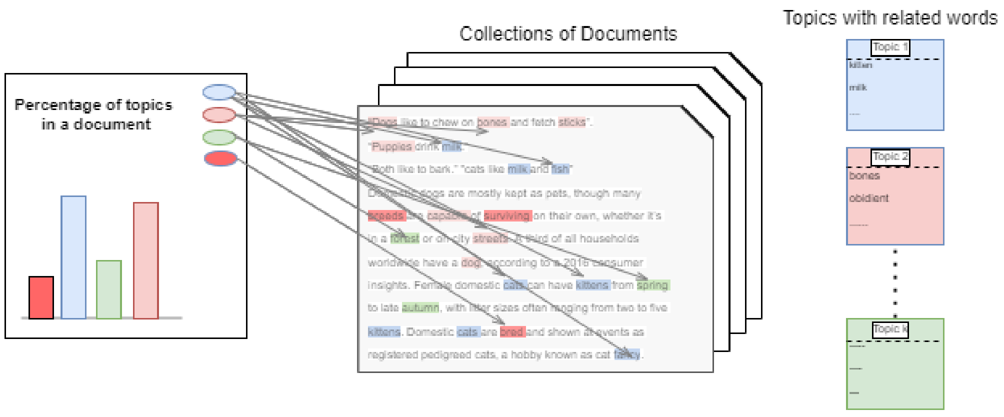


Figure 2.1: Topic Modeling Overview

Topic Modeling can be performed by various algorithms or methods. Some of them are LDA (Latent Dirichlet Allocation), NMF (Non-Negative Matrix Factorization), LSA (Latent Semantic Analysis), PLDA (Parallel Latent Dirichlet Allocation), PAM (Pachinko Allocation Model), etc [12].

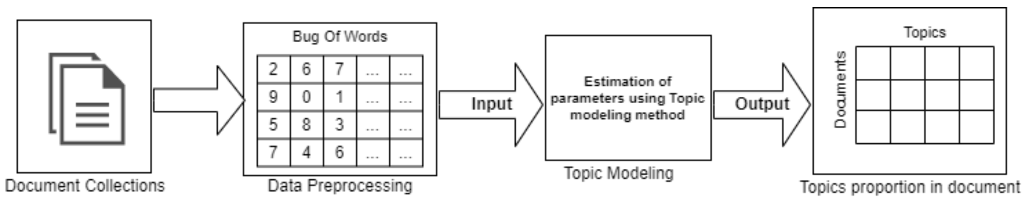


Figure 2.2: Topic Modeling Steps

2.1.1 LDA:

When it refers to fitting a topic model, Latent Dirichlet allocation (LDA) is a method that is particularly well-known. Each text is considered a collection of topics, and each topic is treated as a collection of vocabulary words. Rather than being divided into discrete sections, this permits documents to "overlap" one other in terms of content, in a manner that reflects the way natural language is generally used [13–15].
In 2000, J. K. Pritchard, M. Stephens, and P. Donnelly first proposed LDA [16, 17],, and David M. Blei, Andrew Y. Ng, and Michael I. Jordan rediscovered it in 2003. [18] Rather than the mathematical terms, the LDA model can be described using two main principles to obtain a much clearer understanding of the interrelations between topics. They are:

- Every document is a mixture of topics: A document can contain words of a few topics in different percentages/ratios. As an explanation, if document X contains 70% words of the topic Climate Change, it may contain 30% of the topic Global Warming.

- Every topic is a mixture of words: If we consider two topics for example Mathematics and Biology, equation, addition, summation etc. will be the most common words in Mathematics; on the other hand genetic, DNA, class, organism etc. There can be some common words between these two topics such as research, science, accuracy etc.

The first principle, Every document is a mixture of topics is applied by Bag Of Words. But for the second principle, the calculation is needed to determine the probabilistic distribution of words that corresponds to a topic.
In LDA firstly words of each of the documents are randomly assigned to the predefined K topics and then PTD needs to be calculated for each of the words(W) of every document (D). Here PTD represents counting the number of words that correspond to the topic T in a given document D, leaving out the current word. The probability of a word (W) corresponding to a topic depends on how many words of document (D) correspond to a topic(T). Then PTW , the counting of the number of documents to be corresponding to a topic (T) by the consideration of word(W), needs to be calculated.

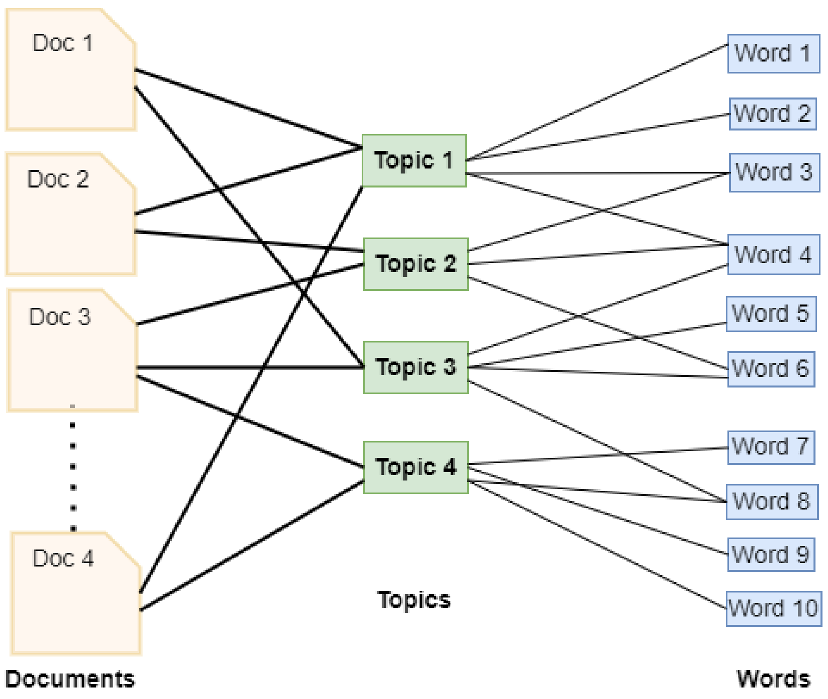


Figure 2.3: LDA Overview

In the figure, the left-most level is represented as documents, the intermediate level as topics, and the right-most level as words. As a result, it clearly demonstrates the concept that documents are represented as the proportion of topics, and topics are represented as the proportion of words [12].

2.2 Doc2Vector:

Doc2Vector is a method to numerically represent the documents, which is based on Word2Vector. Using the concept of Word2vec, vector representations of each word in a document can be built. Word2vec can be implemented using a number of algorithms, including Continuous Bag-of-Words, Skip-Gram, and others [19, 20].
In 2013, Mikolov, Tomas; et al. published paper. [21, 22] about first creation of Word2Vec. The concept of Doc2Vec was first introduced by Mikilov and Le in 2014 [23].

To learn the weights, Word2Vec utilizes a basic neural network with only one hidden layer. Rather than focusing on the predictions that this neural network might be able to provide, it is concerned about the hidden layers weights because these weights are basically the word vectors. It produces word-vectors that are adjacent to each other in vector space that contain similar definitions depending on the context, while word-vectors that are far from each other have different meanings depending on context. The Continuous Bag-of-Words(CBOW) model and the Skip-Gram model are the two most used algorithms for constructing Word2Vec representations [19, 20, 24].

Word2Vec aims to generate representation vectors from words. Doc2vec [25], on either hand, is intended to produce a numerical presentation of a document independent of its length. As documents do not appear as logical frameworks like words, hence another way must be devised in its absence. Quoc Le and Tomas Mikolov [23] suggested the Doc2Vec method for processing arbitrary length text. Excluding the addition of a paragraph vector, the above approach is nearly identical to Word2Vec. Distributed Memory (DM) and Distributed Bag of Words (DBOW) are the two different methods of Doc2Vec, like Word2Vec. Estimating the probability of a word occurring in a given context and paragraph vector is the work of DM. A sentence or document's paragraph ID is fixed, which shares the same paragraph vector during the training process. On the other side, in the presence of simply a paragraph vector, DBOW indicates the probability distribution of a paragraph containing a collection of random words [20, 25].

2.3 TF-IDF:

TF-IDF, the short form of the term frequency-inverse document frequency, it is a statistical measurement that can determine the importance or relevance of words or any string representations in a document amongst a collection of documents.This collection of documents is also called a corpus. There are some contents that remain underval-

ued. So to understand those contents better, search engines use TF*IDF.
To determine the importance of words in the document and corpus, a score is assigned to each of the words. In the fields of information retrieval and text mining, this method is commonly employed. Tf-IDF was first introduced by Spärck Jones, K in 1972 [26,27].

Term frequency is calculated by examining the frequency of occurrences of a specific term that is of interest in relation to the document. To determine how frequent (or infrequent) a word is within a corpus, inverse document frequency method is used. TF-IDF score can be achieved by multiplying the TF and IDF values together.

$$TF = \frac{(Number\, of\, times\, term\, t\, appears\, in\, a\, document\, d)}{(Total\, number\, of\, terms\, in\, the\, document\, d)} \tag{2.1}$$

To determine how frequent (or infrequent) a word is within a corpus, inverse document frequency method is used. Calculating the IDF is as follows: divide the total quantity of documents by the quantity of documents in the collection containing the phrase in question.

$$IDF(t) = log_e \frac{(Number\, of\, documents\, in\, total)}{(The\, number\, of\, documents\, containing\, the\, term\, t)} \tag{2.2}$$

As TF-IDF is a weighting system, according to the term frequency (tf) and the inverse document frequency (tf) of the word in consideration (idf), it allocates a weight to each word in a document.

Chapter 3

Related Works

Community question answering(CQA) services are places where learners and experts gather to share their knowledge as well as their problems with each other. In the twenty-first century, CQAs have opened all the boundaries for self-learning. Anyone from anywhere around the world can learn about anything they are enthusiastic about, and the other people sharing the same interest would help him to gain his goal. While the popularity of community question answering (CQA) services is increasing day by day, the number of users is also rising. They are asking more and more questions each day, making the number of incoming questions grow in a rapid way. If we take the example of Stack Overflow, we can see that till now, the number of questions asked in one day is 7600 for Stack Overflow, which denotes the escalation of question counts.

In the past decades, numerous research works have been done focused on different problems of Stack Overflow. The domains related to our thesis work are the question difficulty estimation and user expertise. Question routing often comes as an integrated part of these two domains.

3.1 Understanding the problem

The root problem here is the concerning duration between the question asked and being answered, as the questions are not being answered at the same pace the questions are being asked, which often leaves many questions unanswered.

In order to better understand the problem, Wang et al. [1] conducted an empirical investigation on four of the most popular stack exchange websites to determine the reasons behind slow responses from CQA systems. They listed 46 factors and four dimensions: question, asker, answer, answerer, and applied logistic regression to them to find out the relation between the factors and the reason for receiving a proper answer.

After conducting the analysis, they found that: When all other factors are taken into consideration, the time it takes to acquire an accepted answer has the biggest link with the individual answering the inquiry. Slow-answered questions account for 61.386.9 percent of all answers given by non-frequent answerers. So questions that were answered slowly would have gone unanswered if they had not been responded to by the non-frequent answerers. Their findings put light on the fact that the non-frequent answerers who usually answer questions that frequent answerers fail to answer aren't acknowledged under the existing incentive system, which is unfair. However, despite the fact that the existing incentive system is found to be effective at motivating frequent answerers, frequent answerers tend to answer questions that are simple. Their research outcome recommends that the CQA services need to enhance their incentive system in order to encourage non-frequent responders to be more engaged in giving answers as well as to answer questions quickly in order to reduce the amount of time that users need to wait for an answer [1].

However, the researchers approached the problem from various perspectives and came up with numerous methods and techniques for the solution. The main goal behind those approaches is to make the users interested to answers the questions.

3.1.1 Question Difficulty

To measure the questions' difficulty level, Neung V. and Twittie S. [3] suggested a different concept. They focused on the priority of users' difficulty in finding the right question to answer and proposed a method to measure the question difficulty based on the concept hierarchy. For their research, they chose only the JavaScript-related questions. They represented their proposed method in three parts. In the first part, they measured the difficulty level with the help of a concept hierarchy, for which they constructed a concept hierarchy following the method proposed by a previous study by consulting with multiple books and websites. After which, they extracted the keywords that represent the concepts or domains from the questions dataset and measured the question scope with the function they came up with. In the second part, they considered the features of the question-answer communities. They followed a previous research work to find out the most correlating five features and calculated the feature scores for the questions concerning the question asker. Their final approach was to combine the previous two approaches to determine the question's difficulty level. Moreover, what they found from their research is that applying only the first approach, which is the core contribution of their work, gives results similar to the previous studies, but after combining them with the feature scores, the result improved. Even though they acknowledged that this process has some limitations as difficulty may not always be related to words but also the applicational logic.

Unlike these works, Hassan et al. [4] proposed a supervised learning-based, difficulty-aware scoring system while considering multiple features from Stack Overflow. They focus on developing a tool to estimate the difficulty level of a SO question automatically with and without an answer. Initially, they only considered 936 questions related to Java Strings, Threads, and Inheritance and categorized them into Basic, Intermediate, and Advanced level. As the first step of supervised learning, they manually labeled the question from the viewpoint of learners. Basic questions include simple problem-solving questions or comparisons of two functions or something that can be answered directly from beginner-level books or basic API documentation. Intermediate questions most likely contain why-type questions, or how to solve more efficiently, questions involving time complexity, resource constraints, conceptual reasoning, underlying philosophy, or design principle. However, Advanced questions require in-depth programming knowledge, critical thinking, well understanding of internal language structure, or dealing with questions about rarely used frameworks or APIs. Then they preprocessed the questions and selected different features under two categories: Pre-Hoc, the features which can be used immediately after the question is posted, and Post-Hoc, which can be retrieved at a later stage after the question is answered. Using the Pre-Hoc features separately and with the inclusion of Post-Hoc features, they categorized the question with the following classifiers (Adaboost, Naive Bayes, k-nearest Neighbour, Bayesian Network, j-48, Random Forest, Random Subspace, Simple Logistic, Logitboost, SVM) and evaluated the accuracy. Finally, they found a satisfactory outcome with Random Forest of highest mean accuracy, 0.671 with only Pre-Hoc and 0.752 with the inclusion of Post-Hoc using 10-fold cross-validation.

The aforementioned studies tried to estimate the question difficulty from a bunch of questions, whereas D. Thukral et al. [5] proposed the concept of the relative difficulty of questions. They acknowledged the previous works done by other researchers and presented their own way to measure the relative difficulty level of questions from community question answering services. They were motivated by the limitations of previous works and tried out a novel approach to measure relative question difficulty automatically. For the research work, they considered three of the CQA traits, textual descriptions, temporal effects, and users' information. Among those, the temporal effect was first suggested in their work to estimate difficulty level. Their work's main approach was to build a network assuming the questions as the vertices, where the edges were assigned based on the predefined hypotheses. They mapped that network with the prediction problem of directionalities of the edges. And using those edges, they estimated the relative difficulty between the two questions. Their proposed method significantly outperformed four of the state-of-art methods and was also able to overcome the limitations like the interruption of data noises, multidomain datasets, cold starts, etc.

3.1.2 User Expertise

Alongside measuring and estimating the question difficulty level, many researchers have worked from the perspective of the users' skills and expertise level [9], [28], [29].

In one study, Li et al. [28] 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. For that, they made users' performance profiles based on their answering history. Then estimated their expertise level considering the answer quality and rated them. Finally, after checking for availability, the questions are routed to the top answerers.

Wang et al. [9] conducted an empirical study and found that more than 66% of the users have their daily comment activities over 30% and active users on the previous day are likely to be active the next day. So they proposed an answerer recommendation system approach that would find the suitable answerers on the basis of the answerer's topical expertise, interest, and activeness, known as IEA. For topical expertise and interest, they used a model called TEM (Topic Expertise Model), from which they collected variables like user topic distribution, user topical expertise distribution, and expertise-specific vote distribution. For calculating the user activeness, they used the historical activity, meaning questions, answers, and comments which are provided by the user. For a new incoming question, the recommendation scores of its candidate answerers are calculated. All candidate answerers are ranked according to that score and can obtain a rank list of answerer recommendations. Then, the answerers with the top N highest recommendation score are selected as the recommended answerers for the new question. To measure the evaluation metrics, they used nDCG@N, Pearson rank correlation coefficient, and Kendall rank correlation coefficient. Not only the IEA, they compared the recommendation system with the previous studies, too, like TEM, TTEA, and TTEA-ACT. And, IEA outperformed others. They also tested the usefulness of comments by measuring the comment-less IEA.

Where Wang et al. [9] signified the usefulness of comments, Diyanati et al. [29] actually applied comment mining in their research. They took both the questioner's and responder's credibility and expertise of Stack Overflow into consideration. Their intuition was that to find out valid answers to a question, the expertise level of the individuals can be useful. To determine the user expertise, they extracted 18 months' worth of data from a data dump that includes questions, answers, and comments, as well as data about the user's profile. They tested out two methods for their research. The first method is based on the scoring system of SO, and the second one uses comment mining. For the first method, the main hypothesis was that the questions' and answers' scores are linked. Stack Overflow provides scores based on the user's activity and the activity history. So, the hypothesis is that a user with more experience is more likely to ask questions with higher scores, and the responses to those questions appear to be higher-scored as well. They considered four cases based on the mean/max scores of the questions and answers. Then, to find the correlation between the scores of questions and answers, linear regression was applied. And it was discovered that there is no significant association between these two scores after examining over 15000 users and the scores of their queries and replies. Then they proceeded with the second method. They included comments in this method for a better result, as most of the comments seem to be informative. To aid their process, they used two features from the dataset, reputation(score given by SO) and userID for each user. The comment mining starts with finding similarities between words of two comments and later calculating similarities between those comments. Once they found out the similarity of the comments for all the questions and answers, the comments were clustered based on the similarity where similarity is above the similarity parameter. To remove outliers, the clusters with members less than three are re-examined. Comments of these individuals are evaluated for similarities, and if the similarity is above 90%, that means these comments are variations of the same comment and don't provide any information. So, they are considered fraud/trivial and removed from the list to omit outliers. Then it comes to calculating the total score for the comments. Comments are mostly textual and hold positive and negative semantics. To calculate the total score, they used the list of positive and negative words from a past research work as well as another list that's provided by five professional programmers. The stop words removal was done, and the words from the comments were compared. And finally, the scores were calculated for the relevant sentences. The fact that when a negative verb is followed by a negative adjective, the result is eventually positive was considered. To calculate a user's total score, they used an equation that takes two parameters SO provided score and the score from comments. The users were clustered into five categories using the K-means algorithm. To determine the accuracy of the method, they took the top 20 users from each category and gave these data to 5 professional programmers. With professionals' opinions and the kappa test, they calculated the recall, accuracy, and F-Measure and compared(MRR value) the results with another research work where their method outperformed the others.

3.1.3 Question Difficulty and User Expertise

While works on question difficulty estimation and user expertise had been done separately, some researchers tried to combine these two approaches for a better outcome [30], [31], [32], [33], [34], [35].

Liu Yang et al. [30] first proposed the idea of applying both the topic modeling and the expertise level together. Their intuition about this approach was that no one is an expert on all topics, and thus expertise based on topics only seems feasible. They proposed a novel probability-based model, Topic Expertise Model(TEM). They used the tag and voting information to help the model learn topics. And finally, from the outcome of the model, they extended the PageRank algorithm and provided a new link structure algorithm called CQARank.

Liu et al. [31] worked on a technique to estimate the question difficulty level in community question answering services (CQA) and proposed a competition-based model. Unlike the past work, which was based on the PageRank algorithm, they combined the concept of user expertise with the question difficulty in their proposed model. In the model, they performed a total of four questioner-answerer pairwise comparisons. The estimation of difficulty was mapped with the concept of competition from a sense that each comparison is a competition, and the user with higher expertise would win the competition, and by winning, the users' skill increases. Then, they considered the relative skills of the questioner as the difficulty level of the question and the relative skills of other users as the user expertise score. To evaluate their work, they used accuracy as an evaluation metric, and the research outcome significantly outperformed the previous PageRank-based approach. Along with this, they further reflected on their intuition about the question bodies representing the difficulty level of the questions. After conducting an analysis of the question descriptions, the outcome of the analysis supported their intuition.

But, as this approach [31] suffered from data sparseness and cold start problems, Wang et al. [32] combined the textual descriptions of the questions with the previous method proposed by Wang et al. [31]. As far as our knowledge, this is the first research work that takes textual descriptions of questions into consideration. Their work serves a novel approach named Regularized Competitive Model (RCM). RCM solved the problems existing in the previous method [31]. The competition-based model requires each question to have an answer to be compared, and each question is compared exactly twice, which introduces the sparse data problem and can’t measure difficulty for new questions. By integrating a regularizer that prevents overfitting with the competition model, the RMC model was formulated and was used to estimate the difficulty level of well-resolved questions. On the other hand, for newly posted questions, they took the assistance of the K-Nearest Neighbor algorithm, where the k-nearest neighbors of those well-resolved questions were found out as the similarity in the textual descriptions of two questions suggests their difficulty level to be close as well. After performing RMC on Stack Overflow data, the result came out better than before, which actually signified the contribution of textual description to estimate the difficulty level.

Lin et al. [33] used a probability-based model to find the hard questions and proposed a question difficulty rank(KG-DRank) algorithm based on knowledge gap. The idea is that,the users with higher knowledge base are more likely to answer a hard question. So, from the users history, or in other word, from the knowledge or expertise of a user, hard questions can be detected.

In another study, Wang et al. [34] tried to make a personalized recommendation system that would work with new questions on CQA sites. And for their research, they chose the popular CQA site, Stack Overflow. They worked with a Stack Overflow dataset that includes new questions, questions with answers, and the answering history of users. They considered both topic modeling and link structure for their work. At first, they prepared question profiles from both answered and unanswered questions with the help of topic modeling based on the Twitter-LDA model. Then from the answered questions, they constructed the user profile with the help of their answering history. And the final concept was to match the question profile with the user profile and recommend the new questions to a group of experts for which they applied the NEWHITS algorithm.

After analyzing the previous works, J. Sun et al.(2018) [35] proposed a framework for question difficulty and expertise estimation in CQA sites that will help to appropriately route and assign questions to users with the suitable expertise. By analyzing previous studies, they propose an approach to avoid overfitting, address the cold-start problem, and also to improve the scalability of the solution. The focus of the research work was fixed on question difficulty estimation, user expertise estimation, and question routing. In the process, they first built the competition graph to incorporate their intuitions. Then they examined the use of different heuristic-based algorithms to estimate question difficulty and estimated difficulty levels for newly posted questions using EGA and language-aware features. And routing newly posted questions to possible answerers based on textual features, question difficulty, and user expertise rank generated by QDEE's [35] prior processes. In this process, they have used various methods and found: Leveraging the expertise gain assumption (EGA) to handle graph sparseness can significantly improve the question difficulty estimation results; the difficulty estimation model cannot be applied to cold-start questions, only 3% - 10% of users are considered as Owls (users who prefer to answer hard questions) in Stack Overflow.

Although the primary focus of research works mentioned here was suggested to have more questions answered in CQAs, Allamanis et al. [36] tried to retrieve insights from the Stack Overflow's questions. To give guidelines and improve the IDE technologies and smart documentation systems, they did an analysis of stack overflow's questions. They have done topic modeling after categorizing questions into programming concepts and types of information in these two perspectives. At first, they tried to analyze different concepts of the questions by applying topic modeling with 150 topics and 2000 iterations as concepts focus on the question's confusion, not the questioner's need. The result was described using the orthogonality(cosine similarity) between the language, tools, and platforms and found similarities between several languages' programming tasks. As an extension of this part, they considered the code identifiers along with the question's texts and applied the new LDA and found that, though SO questions are related to code, only code cannot describe concepts. After that, the question types came into their focus as they indicated the reasons for which questions were asked. The question and answer texts were chunked, and noun phrases were removed; topic modeling was run on the remaining active phrases with 100 topics and 2000 iterations. By analyzing this result, it is found that for retrieving code-related information and understanding code better, the question type is very impactful. Then they combined the results of these results of two topic modeling and analyzed them to find correlation through covariance. As a result, they found top correlations; using the concepts and identifiers, the most problematic issues can be determined easily. There was another conclusion that the type of the asked questions can be estimated based on a few terms about the domain, which is beneficial for integrated development environments (IDEs).

3.1.4 Limitations

A huge portion of the related work is done on the resolved questions. Even though many researchers worked with new questions [28], [32], [34], [1] and considered cold start problems [32], [5], they worked on questions of specific topics. So, none of these works provides a generalized approach to estimate the question difficulty.

Chapter 4

Methodology and Dataset Generation

4.1 Dataset Generation

In this section, the dataset extraction method from Stack Overflow for the text classification models will be discussed. Also, the discussion will continue on what and how the features are selected about a post and its owner. And lastly, what are the inherent properties of the dataset that are linked to the difficulty classification will be presented.

The posts and their details were collected from the Stack Exchange data dump of 2017[1]. We only considered the posts from Stack Overflow and its users as it has gained a sky-high reputation in the past years from its establishment in the field of programming, continuing to maintain over 18 million registered users and more than 22 million questions. It would be safe to say that the Stack Overflow site has posts about any available topic of programming. And like any other Q&A site, the questions on Stack Overflow deal with a certain level of difficulty per post as it relates to certain topics of programming. And this paper is going to continue its discussion on how the machine learning approach can determine question difficulty and how accurately. The dataset selected for the experiment contains the overall picture of the questions for a particular language.

4.1.1 Data Extraction from Stack Overflow

Java is certainly one of the popular languages that are being used right now. And according to the Stack Overflow 2017 survey, which includes data from 2013 to 2017, illustrates the fact that about 39.7%[2] of the developers are using Java, putting it in the third position. The java language contains professionals, where 37.9% of them are web developers, 39.9% of desktop developers, and 41.4% of them are part of DevOps. This

[1]https://archive.org/details/stackexchange
[2]https://insights.stackoverflow.com/survey/2017#technology

information undoubtedly makes it a beneficial language for learners. Also, both academics and jobs both sectors have a higher requirement for Java programming. And it is the basis for a largely used framework like Spring which takes the dominant part of the usage of Java. Throughout the world, the language Java is broadly used stating from websites, system software, data storage, IT infrastructure and complicated programs like data science according to a survey done in 2020 by Jetbrains[3]. So it can be effortlessly advocated why Java is chosen as a representative for further work on Stackoverflow questions.

Now to take a step forward, we need to assemble a dataset that is stable and evenly distributed to present the overall scenario of posts generally answered in Stack Overflow. In the research field, a significant number of works depend on Stack Overflow data like [37], where the researchers tried to extract useful information for API documentation from Stack overflow posts which would add useful insights for developers. Then again, in another paper [38], the researchers tried to use the Stack Overflow data to connect with IDE for the developers' convenience. The java language has a special place where researchers used their methods for answering many of the unsolved problems and represent the posts, questioners, and answerers for its own societal context using legacy data of Java Community in Stack Overflow [39].

Following the consistency, we used the Stack Overflow data dump of December 2017[4] and used the Microsoft SQL Server database for the data storage facilities. The data dump is of size 19GB formatted in .7z as it was compressed, and after extraction, the database took about 137 GB of the hard disk space. So using this type of dataset is useful for representing a deep-rooted effect of users' and posts' characteristics but laborious in the case of manual labeling, which will be further explained later in the current section. We considered a simple query to get a subset that would satisfy the requirements for a certain post which would make the labeling and further the machine learning process much easier. The query result returned about 2000 rows of certain characteristics. The query is limited to answered questions, the interval of the first answer to a question, and the question score. Query constraints were used to make sure that the upcoming manual labeling operation could get an unbiased and overall equal number of instances. And with that, it also helped construct a dataset of Java with its diversified topics. And selecting randomly 700 rows having post Id, post title, post body, and post tags for labeling and moving on to the next step to difficulty labeling.

[3]https://www.jetbrains.com/lp/devecosystem-2020/java/
[4]https://archive.org/details/stackexchange

4.1.2 Difficulty Labeling

After the first set for manual labeling had been constructed, we went ahead to label the data manually by analyzing the post body. The categories to consider while labeling was decided to be Basic, Intermediate, and Advanced. And the whole labeling process was executed by four of the authors, and the validation of the labeling was done by the experts. The reason for marking a question in a certain label was mentioned by each of the labelers, which made the process of validating the labels a bit easier and parallelly it worked as the labeler's perspective for the question that might not match with other labelers. The distribution of the posts among the researchers was done in random order, and we were clueless about any post coming to us for labeling. So, this more or less eliminates the bias of any topic related to Java, as upcoming posts might not be on the same topic. The forwarded question to each of the labellers had common posts between them so that each of the posts was allocated to at least three of the labelers. This would give a chance to rule out any chance of mislabeling a post. For final labeling, we followed the majority voting approach. Both labelers would put their suggestive label with their approach of thinking, and the conflict would be discussed with the remaining team members and take our stand on the post. If the label of a post could not be determined by the authors, the experts were consulted to resolve the dispute.

The idea of labeling the question by the human examiner is originally inherited from the work [4], where the researcher used supervised learning on the manually labeled dataset. Our idea was set on a similar sort of suggestion with an extension of breaking down each of the rules on a more granular level so that it provides enough clarity of the label a question is getting. The rule set that was mentioned in [4] was discussed among the supervisors and us, and we found this ruleset is not enough to label correctly. So we divided each of the rules into fragments, Table 4.1 .Subdivided rules are each connected to questions that were incorporated into the dataset.

Table 4.1: Labeling Rule Set

Difficulty Class	General Rule set	Granular Breakdown
Basic	Questions on simple built-in functions/API documentation/beginner level knowledge	Simple Built-in-funtion
		Simple Operator
		API documentation
		Beginner level Theory Question
		Basic OOP problem
		Simple Program Understanding
	Questions related to comparison between functions of various languages	Analysis of various languages' functions
		Beginner level query difference
		Simple problem solving
	Questions with simple problem-solving	Simple query problem solving
		Simple functionality related
	Questions with simple exception, error and other problem	Solve for nullpointer exception
		Simple Error Handling
		Simple configuration problem
Intermediate	Questions demanding deeper understanding of the programming language to answer	Built-in function deep understanding
		Need more knowledge about the algorithms
		Multiple questions
		Need knowledge on Advanced Programming topics
		Difference between two packages
	Questions stating the answer of the problem but still inquires about more efficient answer	Looking for Appropriate way
		Analyzing different alternatives
	Questions about a system's computational cost, space utilization, or other resource usages	Efficient way
		Performance, optimization, accuracy
		Memory related
	Questions requiring conceptual thinking in response to any programming structure, API, or design principle	Reverse programming
		Underlying philosophy of any programming construction
		Design pattern
		Feasibility study
		Question about built-in documentation in details
	Required Testing Related Knowledge	Automated testing related problem
		Requires knowledge of Testing
Advanced	Questions about critical challenges that require in-depth technical expertise or logical reasoning to solve	Critical problems where solution needs in-depth programming knowledge or conceptual thinking.
		Multiple question, Solution needs in-depth programming knowledge or logical thinking.
		Multiple question and in depth knowledge needed
	Questions that require advanced in-depth knowledge of internal language structure	In-depth knowledge of internal language structure.
		In-depth knowledge of packages
	Questions that deals with infrequently/rarely used framework/API	Deals with infrequently/rarely used framework
		Deals with deprecated framework
	Related to real life scenario	Efficiency related Question in real life scenario
		Optimization in Real life scenario
	Other Rules	Mentioned having the answer, asking for suggestion
		In-depth knowledge on Garbage Collection Algorithm
		In-depth testing and security knowledge
		Need in-depth knowledge multiple topics
		Large amount of study already known
		Need deep knowledge about design architecture , SW maintenance, and SDLC,new plugin
		Works on large dataset, artificial intelligence

After labeling the first selection of questions, we measured three classes that we divided the questions into. And established that we did not have enough questions in Advance class. To prevent the bias factor, we add about 50 more rows to the labeling process and start our iteration all over again. And at the end, we had 738 posts labeled and verified. Table 4.2 shows the dataset after labeling, with the dataset already existing in [4]. And the whole extended dataset would be used for posts feature extraction in the following procedure.

Table 4.2: Dataset wise class Distribution

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

4.1.3 Feature Extraction

The dataset after labeling only consists of post Identification number, post body, and post tags, but still pre-hoc features, the features that will be available just with most of the post publications, and post-hoc features that will be available when answerers start interacting with the published post. The feature list Table 4.3 was decided on having a notion that these will be having crucial for determining the difficulty of a question, and this conception was built from [4] where they considered the question body size, response time, score of a question, view count and answer count. We not only considered the body but also the title and tags with it. 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. We also considered details extracted from the post body like Line of Code snippet, URL, and image count.

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 of the textual and code sections 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 snippet, URL, and image count.

Table 4.3: The Features and their definition with associated feature list

Feature Name	Defination	Included in
Processed Body	Post full textual body excluding code snippets and the html tags like <p>, <code>,<href>	Semantic Features,Pre-hoc,Post-hoc
Tags	Post tags, decided at the time of posting, e.g. <java> <oop><multithread>	Semantic Features,Pre-hoc,Post-hoc
Title	Post title, decided by questioner	Semantic Features,Pre-hoc,Post-hoc
Question Length	Length of the whole Processed Body	Semantic Features,Pre-hoc,Post-hoc
Url+Image_Count	Number links the post	Semantic Features,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	Semantic Features,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 viewed by users	Post-hoc
Favorite_Count	Number of times save as favorite	Post-hoc
Up_vote_Count	Number of up votes for being useful and appropriate	Post-hoc
Answer Count	Number of answers in a question thread	Post-hoc
Question_Score	The total number of upvotes it received minus the total number of downvotes it received	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

Code snippet was extracted for the post body and kept for further counting. The line of code was calculated using [5]Pygout, 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. If a post has more than one code snippet, the line number of codes is summed together. And for the feature of URL and image count refers to the number of hyperlinks in the post, as we can not identify which hyper references are related to which image and which are related to code or general links about the questioning

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

concept. The relationship between the features would be explored while working on the models and their contribution to the result predicting the difficulty of posts.

4.1.4 Dataset Quality

To present the quality of our dataset, it is important to understand what we intended for the dataset to achieve. The dataset represents the overall picture of all Java-tagged questions. The questions are consistent with tags from Java varied from development, testing, and deployment. In other words, the total number of unique tags in the dataset is 758, which indicates it is certainly exploring all renowned areas of the Java programming language and the concepts used mostly by the developers. This certainly helped us to gain a better understanding of user behavior in the Java community in Stack Overflow. The dataset contains a balanced number of rows for each of the categories described in the table 4.2.

Now, as it can be seen, we have an almost similar number of rows for two of the classes but are a little behind in the Advance class. As the advanced topics are not usually used by Stack Overflow users or not many experts rely on a Q&A site for their problems. We also added the dataset from [4] to increase the volume of our dataset, and it does not validate as it is already validated by the paper's authors, which was extracted from the 2017 data dump and correlates with our labeling rules. The dataset from the [4] paper has collected only posts on three of the topics related to Java. But the dataset we collected had far more topics and posts, so the dataset after merging with the published [4] dataset does make it biased to some extent but not overwhelming to the model considered in this paper. Our dataset acts as an extension to the obtained artifact.

4.2 Methodology

For predicting the difficulty class for a post, we proceeded to work on our textual features that would be inherited from the post body for the document analyzing model. This section will describe all comparing methodologies that can have a satisfactory effect on the difficulty classification for the documented questions. We extracted three essential sections from a certain post. One part referred to the code snippet; the second one indicated the paragraph texts, and lastly, the hyperlinks. And preprocessing had been operated only for textual elements from the questions. These preprocessed bodies would be used for analysis by TF-IDF vectorizer, topic modeling by LDA and Doc2Vec, and document vectorizer for extraction of features for classification of posts.The codes and result artifacts are kept in open repository[6]

4.2.1 Preprocessing

After the dataset preparation, we took the next step to preprocess the new post body, excluding the code snippets. The semantic relationships between abstract concepts are the main feature that needs to be measured by the document analytical models.

- **Title, Tags and Textual Body Composition**
  The title and tags from a question describe the main aspect and straightforward concept of the post. Usually, the title is the part where the major question is described with the least wording. After the inclusion of the title and tags to the body, the new body would represent the whole post alphabetically, which would help us to discover a further semantic association between body, title, and tags.

- **Tokenization**
  In the step of tokenization, we tokenized each of the posts into smaller units(words) on a space delimiter so that it would be easier to extract important terms and their occurrences. And the word tokenizer used for the Stack Overflow dataset was from the Gensim [40] library's simple_preprocess function, which not only tokenizes but also includes a lower casing, removing any accent marks from the sentence. And lastly only storing strings with a minimum length of 3 to a list of tokens.

- **Stop words Removal**
  After getting the list of tokens, the words that are most commonly used but bare minimum information about the sentence are the stopwords, like articles, pronouns and prepositions. So in classifying the posts the stopwords would only take space on the contrary provide barely any importance. We used the most commonly used NLTK [41] library for using its documented English stopwords appending it to the Stanford CoreNLP stopword[7] stopword list.

- **Stemming & Lemmatization**
  The first list of tokens is joined with space delimiter and then the examination of each word to convert it to its original form starts. And for this task, we used the NLTK librarys well defined Snowball stemming function. And before lemmatization, we used extracted the only specific part of speech using the SpaCy [42] open-sourced NLP library of python. We used the `en_code_web_sm` model for

[6]https://github.com/SMAMonisha/Difficulty-wise-Stack-Overflow-Question-Classification-for-Recommendation-System

[7]https://github.com/stanfordnlp/CoreNLP/blob/main/data/edu/stanford/nlp/patterns/surface/stopwords.txt

tagging the words, allowing only the selected parts of speech to be included in the list of each document. We only allowed 'NOUN', 'ADJ', 'VERB', 'ADV' to be part of the further calculation as in a sentence these parts would most likely carry the valuable meaning. Now, lemmatization would take place to the words to dictionary form of words with the same SpaCy library.

- **Bag Of Word**
  Each of the list 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 librarys corpora package. Using this dictionary, we created the BOW for each sentence.

- **Frequency Limitation**
  We limited word frequency to eliminate the outliers so that uncommon words and meaningless words would not be added to the models in further steps. To identify the frequency limit, we needed to plot the frequency of each word in all of the documents cumulatively. And decided to exclude words that appear less than 30 times in all of the posts.

4.2.2 Document Analysis Model

For semantic analysis, we encounter many papers that used [4] TF-TDF for information retrieval from each of the post bodies, and Topic modeling [7, 36] for extracting the comprehensive information from posts and lastly Doc2Vec is used to vectorize a document as a representation of that list of tokens.

The first model to execute the datasets was TF-IDF [43], the whole process starts after preprocessing of the data and we calculate the step by step according to the main equation of term frequency and inverse document frequency. Lastly multiplied it together to get a vector representation of TF-IDF. After executing the model, we get more than 1600 features(words) for each of the documents.

And the second model that we considered was Topic Modelings popular technique Latent Dirichlet Allocation [44]. We used LDA to seek the unseen documents in a particular document. For yielding an LDA model, we applied Gensims 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 post body. The topic number is a variable that we had the chance to choose for our model. We tried to set topic numbers from 20 to 40, and each of the models was measured using the accuracy, recall, precision and F-1 score. Ultimately, we found that our model performs the best using the number of topics set to 23.

At last, we built our Doc2Vec [22] with Gensims doc2vec model with parameters of vector size, min count set 2 which would remove words having a frequency less than 2 and epoch number over the whole dataset. Given the whole dataset, the first task was to construct a dictionary of vocabulary from the stream of documents. Then vocabulary dictionary and the preprocessed corpus were passed to train the Doc2Vec model. Now any document, this model can infer the vector representation of that document. Vector length can be set by users, so we tried vector sizes from 20 to 40 and the best score of accuracy was given by the vector size 36.

4.2.3 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 pre-hoc and post-hoc categories. To determine the best model for extracting textual features and their correlation with the other features, we applied different multi-class classifiers to the dataset. The most used classifier for text classification [43], e.g., Random Forest [45], XGBoost [46], Adaboost [47], and lastly, SVM [48].

To perform classification on the dataset, we executed K-fold cross-validation where K=10, using the Sklearn [49] library of python. And the whole models with the classifier were run 10 times to train and test. The aforementioned classification models were tuned using the parameters to get a better classifier. Here we discuss all necessary variables that were set to different classification algorithms.

- We set the **Random Forest** to have 15 n_estimators meaning to create that number of decision trees, a depth of 8(max_depth) for each of the trees, and the criterion for trees was chosen to be entropy.
- Then for **Xgboost** , we used the boosting rounds(n_estimators) of 40 with a learning rate of 0.05. And the maximum tree depth was set to 8 as before.
- **Adaboost** was implemented with the parameters set to n_estimators 1000, and the learning rate was kept constant like Xgboost.
- Lastly, the single vector classification, **SVC** model was built for the one-vs-rest (ovr) decision function of shape and enabling the probability estimation to fit the training data with measuring the metrics of performance.

And for each fold, we calculated the accuracy, precision, recall, F-1 score, and AUC-ROC using Sklearns metrics package. After completing all folds, the average for each of the metrics was calculated for comparing the text analyzing models performances.

Chapter 5

Results and Discussion

In this section, the performance of each model with different feature sets is evaluated using different machine learning classifiers. First, 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. Secondly, the efficacy of the pre-hoc feature set will be analyzed in contrast to the post-hoc feature set for different datasets. Then the relationship between features and question difficulty level will be addressed in order to provide insights into the characteristics of questions of various levels of complexity, as well as their questioners and how they influence other users. Finally an overall discussion will be presented to connect all the findings.

5.1 Performance of Question Classification Models

The performance of three models using different classifiers on semantic features of question is shown in Table 5.1. As we can see, for almost every classifier, the Doc2Vec based model outperforms the other two models. With the classifier XGBoost, the Doc2Vec based model can classify the question based on difficulty with an accuracy of 0.656 and F1-score 0.626 whilst Tf-Idf based model performs with an accuracy of 0.653 and F1-score 0.622 and TM based model achieve an accuracy and precision of 0.62 and 0.579 respectively. Among the classifiers, AdaBoost has slightly better accuracy than XGBoost but relatively lower coverage, and the longer time requirement makes it less preferable.

Table 5.2 and Table 5.3 show performance metrics of textual models for all the classifiers with pre-hoc features and post-hoc features, respectively. For pre-hoc features, with accuracy and F1-score of 0.657 and 0.629, the Doc2Vec-based model using XGBoost classifier provides better performance. The Tf-Idf based model performs relatively poorly than the Doc2Vec based model with an accuracy of 0.646 and F1-score of 0.629. Lastly TM based model performs comparatively with the lowest accuracy and f1 score of 0.622 and 0.586.

Table 5.1: Performance metrics of Tf-Idf based model, TM based model and Doc2Vec based model with different classifiers using Semantic Features of Question

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

Considering the post-hoc feature set, usually, most of the classifiers perform better with the Doc2Vec model except for the XGBoost classifier, where the performance of the Tf-Idf based model clearly surpasses both TM and Doc2Vec based models. That is accuracy of 0.663 and F1-score of 0.638 for Tf-Idf based model where an accuracy of 0.659 and an F1-score of 0.632 for Doc2Vec based model and accuracy of 0.644 and F1-score of 0.617 for TM based model.

But the dataset of the Tf-Idf based model was split after calculating the Tf-Idf score for overall data to keep the feature set constant whilst for the other two models, we partitioned the dataset before performing any preprocessing in order to keep it as realistic as possible. As a result, the Tf-Idf based model gains the advantage of learning the test set, 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 based model, and in comparison to the TM-based model, we can infer that the Doc2Vec-based model performs better.

However, it is seen in Table 5.4 as the feature set grows, the overall performance of all question classification models improves. Unlike the TM-based model, where adding features boosts performance significantly, Doc2Vec and Tf-Idf-based models are strongly context-dependent, slowing performance improvement with nearby less important features.

Table 5.2: Performance metrics of Tf-Idf based model, TM based model and Doc2Vec based model with different classifiers using Pre-hoc Features

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

Answer to the RQ1: In general, the Doc2Vec based model with XGBoost classifier outperforms all other models in classifying difficulty wise questions, with an accuracy of 0.657. Even for any unknown or new user with no prior information about the questioner, it can filter questions with an accuracy of 0.656.

5.2 Comparative Analysis of Pre hoc and Post hoc Features

Now, this comparison is significant since the goal of our research is to forecast the difficulty level of a question in order to assist in appropriate question filtering for answering, hence reducing 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.

Table 5.5 and Table 5.6 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 disregard the Tf-Idf model based model since it is unsuitable for question filtering in a real-world setting.

Table 5.3: Performance metrics of Tf-Idf based model, TM based model and Doc2Vec based model with different classifiers using Post-hoc Features

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

Table 5.4: Performance of different models with growing feature sets

Models	Features	Accuracy	Precision	Recall	F1-score	Auc-Roc
Tf-Idf Based	Semantic	0.653	0.61	0.653	0.622	0.746
	Pre-hoc	0.646	0.602	0.645	0.615	0.741
	Post-hoc	0.663	0.637	0.663	0.638	0.762
TM Based	Semantic	0.62	0.582	0.62	0.579	0.68
	Pre-hoc	0.622	0.587	0.622	0.586	0.679
	Post-hoc	0.644	0.61	0.644	0.617	0.734
Doc2Vec Based	Semantic	0.656	0.625	0.656	0.626	0.746
	Pre-hoc	0.657	0.64	0.657	0.629	0.744
	Post-hoc	0.659	0.636	0.659	0.632	0.738

According to the tables, in both models, post-hoc features surpass pre-hoc features by no more than 0.04 points for any performance indicator, with slightly better coverage. Even for a data set with a wide range of topics, pre-hoc features can classify questions with an accuracy of 0.58 and 0.598 using TM and Doc2Vec based models, respectively, while post-hoc features can classify with an almost identical accuracy of

Table 5.5: Performance comparison of Pre-hoc and Post-hoc features for TM based model using different data sets

TM Based Model						
	Existing Dataset (Topic Wise)		Our Dataset (Random)		Merged Dataset	
Metrics	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
AUC-ROC	0.632	0.703	0.692	0.72	0.679	0.734

Table 5.6: Performance comparison of Pre-hoc and Post-hoc features for Doc2Vec based model using different data sets

Doc2vec Based Model						
	Existing Dataset (Topic Wise)		Our Dataset (Random)		Merged Dataset	
Metrics	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
Auc-ROc	0.693	0.725	0.721	0.746	0.744	0.738

0.597 and 0.629 using TM and Doc2Vec based models, respectively.

In general, with pre-hoc features, the Doc2Vec based model conducts difficulty wise question classification with an accuracy of 0.657 and F1-score of 0.629, whereas the TM based model performs it with an accuracy of 0.622 and F1-score of 0.644, demonstrating the efficiency of pre-hoc features

As for the importance of different features, from the semantic features of the question, we extract the competency of the question title, question content, tags, question size, and LOC for each question classifier. Apart from semantic features, the most important pre-hoc features are questioner reputation and users' bronze badge counts,

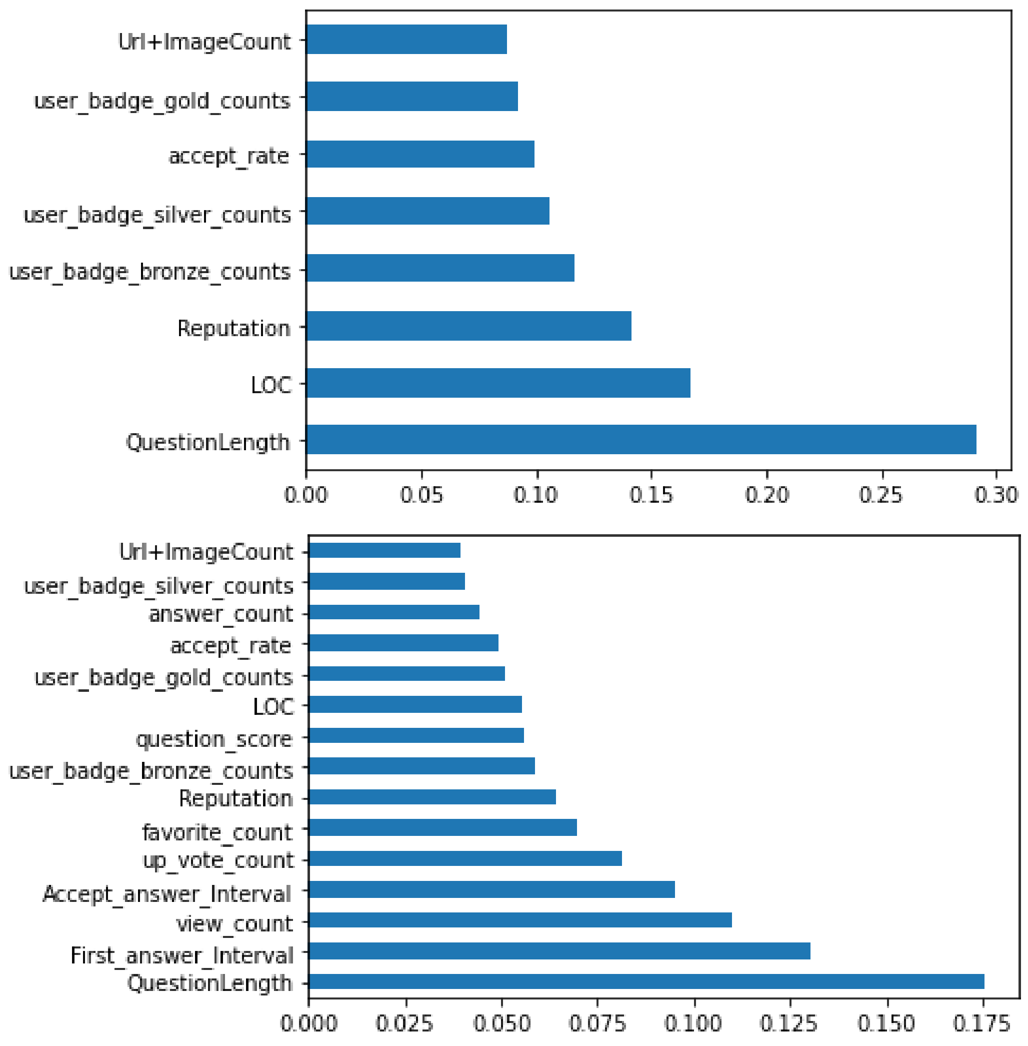


Figure 5.1: Importance of Pre-hoc and Post-hoc Features

whereas post-hoc features include time interval between questioning and first answer, view count, time interval between questioning and answer acceptance, upvote count, and favorite count.

Answer to the 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 .04 for any performance metric.

5.3 Correlation between Features and Question Difficulty Level

To understand the relationship between features and question difficulty level, we look at Table 5.7 to see how the value of features changed as complexity increased and got some interesting insights.

Table 5.7: Changes of different features according to question difficulty

Features	Basic (736)		Intermediate (410)		Advanced (102)	
	Avg	Median	Avg	Median	Avg	Median
Question size	66	50	115	80	212	164
LOC	7	4	13	6	23	13
User Reputation	22193	4810	25352	7660	16397	4510
User_Bronze_Badge	96	41	112	50	90	60
User_Gold_Badge	17	6	18	6	11	5
User_Silver_Badge	62	29	75	32	56	31
Accept Rate	58	0	63	0	64	89
View Count	189697	56300	99439	11700	21408	12400
Answer Count	9	5	8	4	4	5
Favorite_Count	81	12	124	11	24	3
Question Score	281	60	289	45	77	46
Up_Vote_Count	283	60	290	47	78	46
First_Answer_Interval	4956	4	6959	9	21278	11500
Accepted_Answer_Interval	27902	5	40154	15	73601	11
Url+Image_Count	0.3	0	1	0	2	1

Researchers have suggested that the complexity of the question is proportional to the length of the question. We noticed, however, that Code Size is proportionate to the difficulty of the questions after adding the code snippet for classifying complexity. After more analysis, we uncover two plausible reasons. 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, as the number of lines of code increases, so does the difficulty.

With rising question complexity, both the View_Count and the Answer_Count rapidly fall as users choose to go through the questions that they can understand. And the more difficult the question, the more experience is required to answer it, limiting the number of answer counts for difficult questions while allowing users from beginner to expert to answer the easiest, increasing the number of answer counts for basic level questions.

As for Favorite_Count, Question Score, and Up_Vote_Count, we can see that the intermediate level questions gain the highest score while advanced level ones receive the lowest. 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, as well as 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. Knowing that upvotes, accepted answers, and bounty contribute to user reputation, it is easy to see why people who ask intermediate-level questions have a greater reputation than those who ask tough ones.

Though on average, questioners of intermediate level inquiries have the most user badges and questioners of advanced level queries have the least, the median value reveals a proportional relationship between user badge and question difficulty. This is because active users frequently ask efficiency-related or why-type questions which are of intermediate level difficult questions, whereas infrequent users occasionally ask about critical/rare situations that belong to most difficult ones.

The Accept Rate, 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, and these intervals lengthen significantly for advanced questions. One possible cause is that only a few users have the level of knowledge and competence required to answer difficult issues. Considerably understanding the difficult question takes time, and discovering the answers takes even longer. Another reason could be that because the number of challenging questions and potential answerers is limited, attracting the attention of appropriate users for the responses takes time.

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, owing to the fact that difficult questions require more information to convey them clearly. Another possible reason is that the questioner actually analyzes the question himself to find the solution and then includes his conclusions with the questions to provide some guidance to others.

Answer to the RQ3: While question size, LOC, accept rate, URL, and image count, first_answer, and accepted_answer interval are proportionally connected to question difficulty, view_count and answer_count are inversely proportional to difficulty. However, up to the intermediate level difficulty, user reputation and badges, favorite count, question score, and upvote count increase, but drop for advanced level questions.

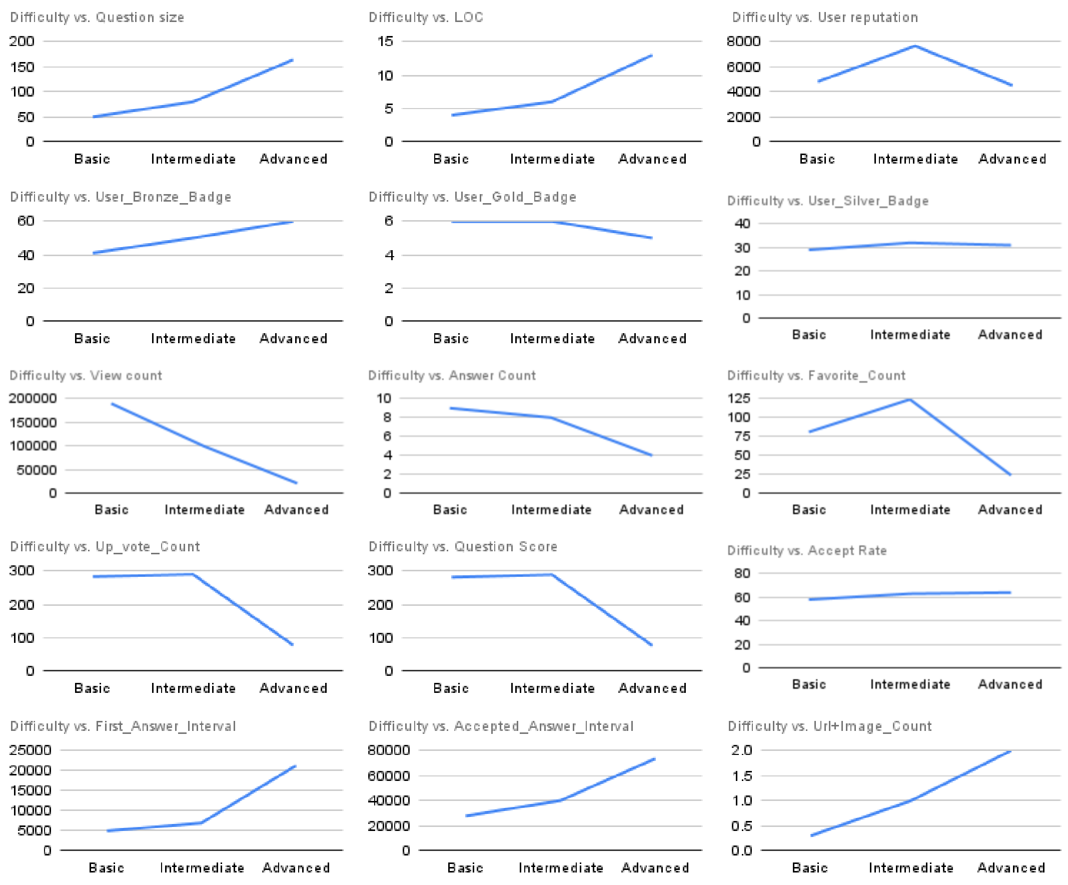


Figure 5.2: Question difficulty vs. different features

5.4 General Discussion

Many filtering and preference options are available on Stack Overflow, including tags, most watched, and most recent. However, failing to draw the attention of appropriate users results in many valid questions remaining unanswered, or replies arriving after a long period of time. Our study's purpose is to predict a question's difficulty level in order to assist in question filtering for answering, lowering response time and unanswered questions.

To support our research, we first focused on developing a model that can classify questions based on difficulty while maintaining optimal performance and applicability for real-world circumstances. We had to rule out the Tf-Idf based model from our three options owing to its intrinsic constraint of sparse vectorization. For the other models, Doc2Vec based model performs better for all sorts of data sets and feature sets than TM based model and is more context-dependent than TM based model, which is more feature dependent.

After comparing different classifiers, we discovered that XGBoost outperforms the others in terms of performance, coverage, and time. Therefore we may conclude that the Doc2Vec based model combined with the XGBoost classifier could be a viable model for detecting question difficulty.

However, until a self-competent feature set is extracted, the model will not be enough to solve the problem. When it comes to extracting features, one crucial consideration is to focus on those features that are available when a new question arises, because otherwise it will be unable to resolve difficulties with unanswered questions or delay response time.

Here, post-hoc features are evidently not available when a question is posted; rather, they are available after the question is answered. But it is vital for understanding the characteristics of different difficulty level questions and users, as well as examine the competency of pre-hoc features that are suited for question filtering.

Because post-hoc features provide more information than pre-hoc features, performance increase with post-hoc features is natural. However, while comparing the performance indicators of the two feature sets, we discovered that the result for pre-hoc features did not decrease by more than 0.04, demonstrating the robustness of pre-hoc features.

To cope with cold start problems where the questioners' information is absent, we used semantic features to examine the model's performance and achieve satisfactory results.

Finally, in order to better grasp the characteristics of difficult questions, we looked at the relationship between attributes and question difficulty. We discovered that basic level questions are modest in size and have fewer subsidiary components, which grows as the difficulty level increases. And this complexity level proportional affects the resolving and approved answer time intervals.

Advanced level questions, on the other hand, have the lowest view count and answer count because they need in-depth knowledge and a high level of competence. However, intermediate level questions have the greatest favorite count, question score, and up_vote count, as users choose to solve the questions to learn more about their degree of expertise. And among the most reputed and active users, the intermediate level questions are more popular due to their real-world applicability.

Chapter 6

Threats to Validity

In our study, some threats and challenging aspects raised a question on the validity of our work. These are:

- Internal validity
- External Validity
- Constructive Validity

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 data set in which questions were chosen at random regardless of the topic. At first we extracted the data dump of stack overflow 2017 then arbitrarily picked the questions. Our data set 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 didnt use any language specific feature to keep it extendable for other languages. Moreover, we selected Java because it is very popular and mature. According to Stack Overflow survey we found 39.7% of the developer using this language. Besides it covers a lot of programming philosophies from testing to object oriented programming which 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 pre-hoc features for usual situations and semantic features for the cold start problem where a user is anonymous or new in the community. Then we train our models accordingly on different types of data sets for assessment. Finally, we found that, for pre-hoc features, all model preforms with an minimum accuracy of 0.615 whilst with no prior information of users, all models have an accuracy of at least 0.60, which is fairly satisfactory.

6.3 Construct Validity

The main challenge regarding the construction validity is labeling the dataset. With anonymous majority voting, we attempted to mitigate this threat. At least four researchers categorized each item separately, and conflicts were resolved through expert judgment. Only the question body, title, and tags were evaluated for labeling to minimize the bias of other features.

To ensure the validity of the rules set, we focused on Hassan et al. [4]s rules and divided them into granular levels for better understanding. We sought advice from specialists to adjoin any new rules.

Chapter 7

Conclusion and Future Work

In this research work, we have done a comparative analysis and found that the Doc2Vec model works the best for difficulty based question classification. The motivation behind this work was to recommend questions to suitable users based on the question difficulty level. So, for that, we needed to categorize the questions into different difficulty levels, which we have already done. In our work, we mostly considered Java-related questions. As Java is the most popular programming language with a well-built developers' community, we could gather diverse types of questions in Stack Overflow to support our research work.

So, our future works will explore the performance of our model on other programming languages like C#, Python, Pearl, Ruby, etc. Other than that, to build the desired recommendation system, the user base is also needed to be categorized for getting the question answered. The users can be categorized based on their activities on Stack Overflow(asking and answering questions), expertise level, and activeness. To measure the users expertise level, further works can be done, like proposing some framework or model. The temporal data related to users activities are also essential for recognizing the active users because recommending questions to an inactive user will be futile.

And finally, we need a recommendation system that would recognize the hidden relations between the question types and the users and recommend the questions to users with enough expertise to answer the questions. So, there are a lot of scopes for future works from this research work.

REFERENCES

[1] S. Wang, T.-H. P. Chen, and 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.

[2] S. Mondal, C. M. K. Saifullah, A. Bhattacharjee, M. M. Rahman, and 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)*, ser. ISEC 2021. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3452383.3452392

[3] N. Viriyadamrongkij and 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)*, pp. 1–6, 2017.

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

[5] D. Thukral, A. Pandey, R. Gupta, V. Goyal, and T. Chakraborty, "Diffque: Estimating relative difficulty of questions in community question answering services," *ACM Trans. Intell. Syst. Technol.*, vol. 10, pp. 42:1–42:27, 2019.

[6] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and B. Hartmann, "Design lessons from the fastest qamp;a site in the west," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 28572866. [Online]. Available: https://doi.org/10.1145/1978942.1979366

[7] L. Wang, B. Wu, J. Yang, and 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), 2016, pp. 901–908.

[8] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, and K. A. Schneider, "Answering questions about unanswered questions of stack overflow," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, 2013, pp. 97–100.

[9] L. Wang, L. Zhang, and J. Jiang, "Iea: an answerer recommendation approach on stack overflow," *Science China Information Sciences*, vol. 62, 2019.

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

[11] C. H. Papadimitriou, H. Tamaki, P. Raghavan, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," in *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, ser. PODS '98. New York, NY, USA: Association for Computing Machinery, 1998, p. 159168. [Online]. Available: https://doi.org/10.1145/275487.275505

[12] "A beginners guide to latent dirichlet allocation(lda)," https://iq.opengenus.org/topic-modelling-techniques/, accessed: 9.05.2022.

[13] "A beginners guide to latent dirichlet allocation(lda)," https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2, accessed: 25.04.2022.

[14] "Topic modelling techniques in nlp," https://iq.opengenus.org/topic-modelling-techniques/, accessed: 25.04.2022.

[15] "6 topic modeling," https://www.tidytextmining.com/topicmodeling.html, accessed: 25.04.2022.

[16] J. K. Pritchard, M. Stephens, and P. Donnelly, "Inference of population structure using multilocus genotype data," *Genetics*, vol. 155, no. 2, pp. 945–959, 2000.

[17] D. Falush, M. Stephens, and J. K. Pritchard, "Inference of population structure using multilocus genotype data: linked loci and correlated allele frequencies," *Genetics*, vol. 164, no. 4, pp. 1567–1587, 2003.

[18] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, no. null, p. 9931022, mar 2003.

[19] "Understanding word2vec and doc2vec," https://shuzhanfan.github.io/2018/08/understanding-word2vec-and-doc2vec/, accessed: 25.04.2022.

[20] "A gentle introduction to doc2vec," https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e, accessed: 25.04.2022.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[23] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, p. II1188II1196.

[24] Y. Goldberg and O. Levy, "word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[25] "Doc2vec," https://blog.birost.com/a?ID=00600-e831ba42-3d77-495c-baa3-dba970172e91, accessed: 25.04.2022.

[26] K. S. Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, 1972.

[27] M. T. Maybury, *Karen Spärck Jones and Summarization*. Dordrecht: Springer Netherlands, 2005, pp. 99–103. [Online]. Available: https://doi.org/10.1007/1-4020-3467-9_7

[28] B. Li and 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*, ser. CIKM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 15851588. [Online]. Available: https://doi.org/10.1145/1871437.1871678

[29] A. Diyanati, B. S. Sheykhahmadloo, S. M. Fakhrahmad, M. H. Sadreddini, and M. H. Diyanati, "A proposed approach to determining expertise level of stack-overflow programmers based on mining of user comments," *J. Comput. Lang.*, vol. 61, p. 101000, 2020.

[30] L. Yang, M. Qiu, S. Gottipati, F. Zhu, J. Jiang, H. Sun, and Z. Chen, "Cqarank: jointly model topics and expertise in community question answering," *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013.

[31] Q. Wang, J. Liu, B. Wang, and L. Guo, "Question difficulty estimation in community question answering services," in *EMNLP*, 2013.

[32] ——, “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)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1115–1126. [Online]. Available: https://aclanthology.org/D14-1118

[33] C.-L. Lin, Y.-L. Chen, and 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)*, 2014, pp. 336–339.

[34] L. Wang, B. Wu, J. Yang, and S. Peng, “Personalized recommendation for new questions in community question answering,” *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 901–908, 2016.

[35] J. Sun, S. Moosavi, R. Ramnath, and S. Parthasarathy, “QDEE: question difficulty and expertise estimation in community question answering sites,” *CoRR*, vol. abs/1804.00109, 2018. [Online]. Available: http://arxiv.org/abs/1804.00109

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

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

[38] A. Bacchelli, L. Ponzanelli, and M. Lanza, “Harnessing stack overflow for the ide,” in *2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE)*, 2012, pp. 26–30.

[39] G. Blanco, R. Pérez-López, F. Fdez-Riverola, and 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*, vol. 105, pp. 446–454, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19311884

[40] R. Rehurek and P. Sojka, “Gensim–python framework for vector space modelling,” *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[41] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O’Reilly Media, Inc.", 2009.

[42] M. Honnibal and I. Montani, “spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing,” 2017, to appear.

[43] J. Ramos *et al.*, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the first instructional conference on machine learning*, vol. 242, no. 1. Citeseer, 2003, pp. 29–48.

[44] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[45] T. K. Ho, “Random decision forests,” in *Proceedings of 3rd international conference on document analysis and recognition*, vol. 1. IEEE, 1995, pp. 278–282.

[46] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” *CoRR*, vol. abs/1603.02754, 2016. [Online]. Available: http://arxiv.org/abs/1603.02754

[47] R. E. Schapire, “Explaining adaboost,” in *Empirical inference*. Springer, 2013, pp. 37–52.

[48] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[49] 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,” *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.