B.Sc. in Computer Science and Engineering Thesis

Question Difficulty Estimation of Stack Overflow Posts

Submitted by

SK. Adnan Hassan
201205059

Dipto Das
201205080

Supervised by

Dr. Anindya Iqbal

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

Dhaka, Bangladesh

September 2017

CANDIDATES' DECLARATION

This is to certify that the work presented in this thesis, titled, "Question Difficulty Estimation of Stack Overflow Posts", is the outcome of the investigation and research carried out by us under the supervision of Dr. Anindya Iqbal.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

SK. Adnan Hassan
201205059

Dipto Das
201205080

CERTIFICATION

This thesis titled, **"Question Difficulty Estimation of Stack Overflow Posts"**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in September 2017.

Group Members:

SK. Adnan Hassan

Dipto Das

Supervisor:

Dr. Anindya Iqbal
Assistant Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

ACKNOWLEDGEMENT

We are thankful to the Almighty for the ability that He has granted us to complete this thesis. Without His grace, it could never be possible to complete this study successfully.

Our heartiest gratitude goes to our thesis supervisor, Dr. Anindya Iqbal for his guidance, insight and motivation. His unwavering enthusiasm in this topic kept us engaged to our research work. His personal generosity has made our research duration enjoyable. Without his valuable and constant guidance along with the words of encouragement, we could not take this thesis to cross the height of our expectation.

We are also grateful to Dr. Rifat Shahriyar for his insightful comments and encouragement. We want to thank our teacher Toufique Ahmed for his selfless assistance and valuable suggestions.

We would like to thank the research participants - students from different terms in CSE, BUET and several BUET alumni for their numerous help. We are indebted to the wonderful teachers whom we have met in BUET and whose teachings helped us in this journey.

We would also like to thank our friends in CSE, BUET, encouraging us all this time and for helping us time to time regarding different technical things.

Finally, we are grateful to our families: our parents, without whose constant support and blessings we would not have progressed this much; and our other family members for their encouraging words which supported us spiritually throughout writing this thesis and our lives in general, whose value increases to us with time.

Dhaka
September 2017

SK. Adnan Hassan

Dipto Das

Contents

List of Figures

List of Tables

ABSTRACT

The use and significance of community Q/A sites such as Stack Overflow for learning programming languages as a supplement to traditional study materials are continuously increasing. Consequently, the number of posts of different levels of difficulty is also growing rapidly. It would be very helpful for the users if this huge collection of posts on Stack Overflow are categorized according to their difficulty level. This would benefit teachers easily collect suitable examples for the level they are teaching, learners identify examples that match their knowledge level, routing decision of questions to appropriate potential experts considering, etc. However, this significant and challenging problem is rarely addressed. In this work, we have presented a supervised learning based tool to automatically estimate the difficulty level of questions. We manually labeled randomly selected 900 questions (300 from each of Java String, Inheritance and Thread categories) to build a training dataset and used them to train different supervised learning models. The best performing model (Random Forest) achieves 72.71% validation accuracy and 72.6% f-measure estimated using 10-fold cross validation. For manual validation of the results, we have used students and lecturers as well as professional programmers and achieved an almost similar result. Using the Random Forest based model, we classified all available (around 45K) questions of these three categories from Stack Overflow and studied some properties of questions with respect to their difficulty levels. Results of the study suggest that basic questions receive higher view, higher rating, and contain lower lines of code. We also found that the reputation of answerer is usually significantly higher than that of the corresponding asker.

Chapter 1

Introduction

1.1 Motivation

Stack Overflow is a widely visited community Q/A site very popular with the programmers. Over 92% questions are answered in a median time of 11 minutes [1]. Programmers irrespective of expertise levels, from novice learner to expert professionals, do constructive discussions on this site. The questions are asked by the users and experts from the community come with possible answers among which one is marked as accepted by the user who asked the question. The registered users have ratings based on their various services to Stack Overflow community according to certain rules and regulations[1].

Due to the potentiality to be used as an immensely popular supplement to traditional tutorials and learning materials of different programming topics, it has attracted researchers to investigate research issues such as answer quality estimation, user expertise or rating measurement, question routing, etc. [2]- [3]. However, the problem of Question Difficulty Estimation (QDE) i.e., categorizing questions with respect to difficulty level has received little attention. We believe this categorization will benefit a number of tasks such as: (i)Identifying posts/discussions easily that match the level of the target audience by instructors of programming languages to show examples or design practice problems. The learners themselves will be benefited in the same way. It may be noted here that a survey by Stack Overflow Business suggests that 69% developers are self-taught [4], while other categories in the survey are shown in Figure 1.1. (ii)Routing questions to appropriate users in consideration of the level of expertise and that of question difficulty. It is likely to generate a better response as the experts will not get bored and emerging contributors will also be encouraged. (iii) Another recent trend is to augment API documentation and tutorials with contents from Q/A sites like Stack Overflow [5]. For this purpose, along with the topic-wise classification of discussions, categorization of questions ac-

[1] https://stackoverflow.com/help/whats-reputation

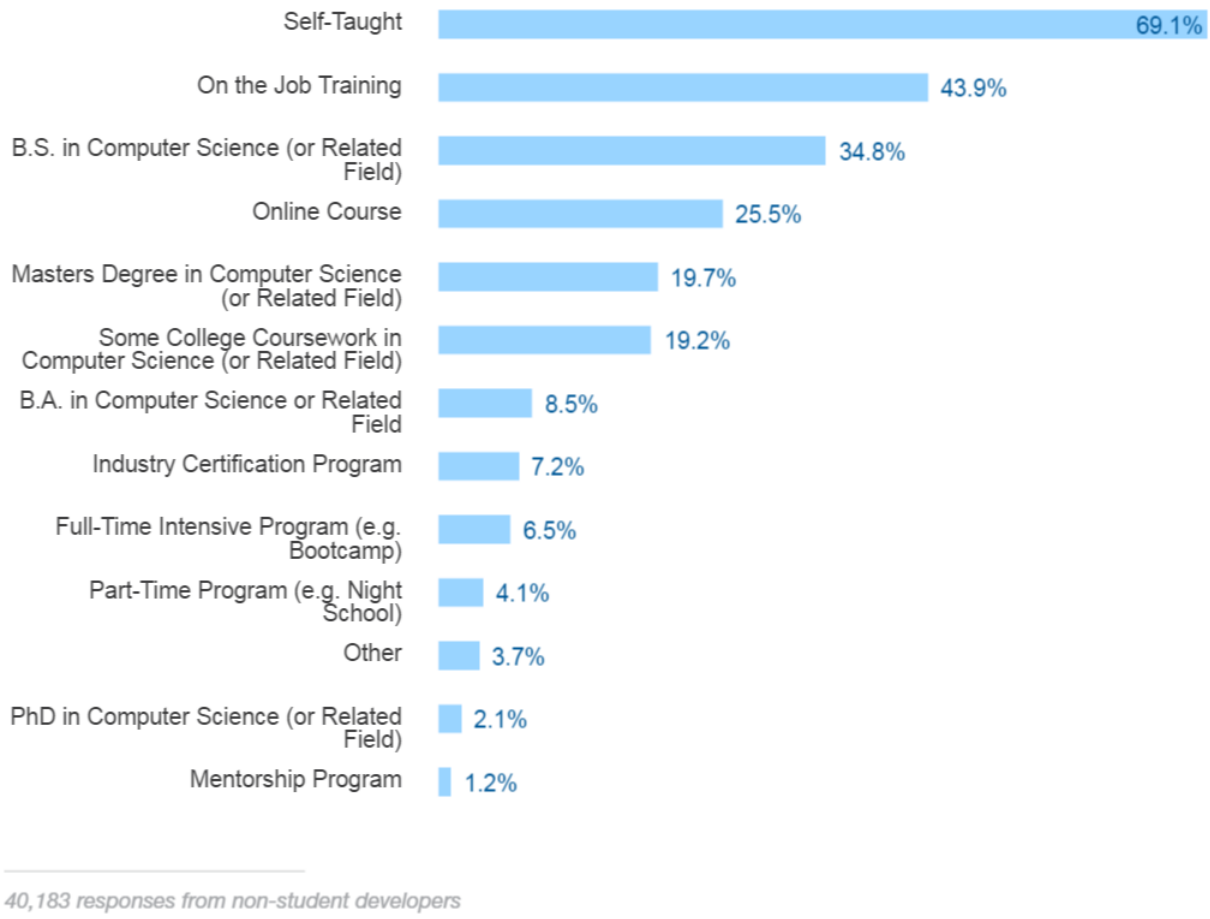

Figure 1.1: How programmers learn to code

cording to difficulty level will be very helpful. (iv) If the Q/A sites plan to offer incentive based on the response of answerers both in terms of quantity and quality (difficulty), this system will be of benefit.

.

1.2 Objectives

To the best of our knowledge, only two attempts have been made to address the problem of estimating the difficulty of questions asked on community question answering services from the NLP community. Liu et. al. [6] proposes a competition-based model for the sake of estimating question difficulty. They exploited pairwise comparison of users and questions. Their subsequent work to estimate question difficulty in [7] incorporates questions' textual description with question-user comparison into a single framework. Both these works [6] and [7] correlate users' rating with the questions to determine the questions' difficulty level. However, their approach

is not readily usable by non-technical users as complex input is required by the users. On the other hand, our supervised learning based QDE tool is released for public and can be used in a very straightforward way with comparable performance.

Our research is guided by two research questions:

- **RQ1.** With how much accuracy and precision can the automated tool predict the difficulty level of a question?
- **RQ2.** Do the ratings of human beings match our automatically estimated difficulty?

1.3 Research Hypotheses

From the automatically measured difficulty level of almost 45000 Stack Overflow questions on Java String, Inheritance and Thread, we would like to study the correlations of rating of questioners and answerers and properties of questions such as the number of lines of code with respect to their difficulty levels.

1.3.1 Rating of Accepted Answerers

When a questioner asks a question in Stack Overflow, s/he can choose between the answers given by community users by "accepting" it. When an answer is marked accepted the answerer is given +15 rating points. The most obvious way of getting higher ratings is by giving a quality answer. It is likely that rating of an answerer increases if more people view the answer. As basic questions are viewed more (supported with t-test, $p < .001$), they should be more rewarding than non-basic questions. Therefore, we pose the following hypothesis:

Hypothesis 1 (H1): The rating of accepted answerer of a basic question is more likely to be higher than the rating of accepted answerer of a non-basic question.

1.3.2 Rating comparison of questioner and accepted answerer

In Stack Overflow, a questioner asks a question when he stumbles upon a problem. The community users who presumed know to a solution to the problem give their solution. As previously mentioned, the questioner can choose between these answers by marking one of them accepted. There is a natural conception that who gives an accepted answer to a question has more expertise/experience than the user who asked it and therefore should have a higher reputation than

the later. Therefore, we pose the following hypothesis:

Hypothesis 2 (H2): The reputation point of accepted answerer for a question is likely to be higher than the reputation point of the questioner who asked the question.

1.3.3 Lines of Code (LOC) in Questions

A user asking a question on Stack Overflow describes his/her problem and the user-specific situation in the body of the question besides the slightly-generic and small description of the problem in the title. In the body, to illustrate his/her situation, s/he can add code snippets in the question. We assume that simple questions can be described with small or no code snippet. On the other hand, in case of conceptually challenging problems' better illustration needs a code example. Non-basic questions are expected to have more lines of code in their body than that of basic questions. Therefore, we present the following hypothesis:

Hypothesis 3 (H3): It is likely that the questions that contain a large number of lines of code are Non-Basic.

1.4 Contribution

In order to determine the questions' difficulty level automatically using supervised learning technique, first, we manually labeled randomly selected 900 Stack Overflow questions. Among these, 300 were on Java String, another 300 on Inheritance and the remaining 300 on Thread categories. The selected questions were labeled by the first two authors with high inter-rater agreement measured by Cohenś Kappa. When there was disagreement it was settled by the third author. Thus a question difficulty training dataset is built which is then used to train different supervised learning models. The best performing model (Random Forest) achieves 72.71% validation accuracy and 72.6% f-measure estimated using 10-fold cross validation. We have also determined which features/characteristics of a question contribute more while deciding a question's difficulty level. Using the Random Forest based model, we classified all available (around 45K) questions of these three categories from Stack Overflow and studied some properties of the questions with respect to their difficulty levels.

To answer the second research question, we have conducted manual validation involving subjects from almost all the groups of users of Stack Overflow. We have randomly selected students of different levels (sophomore, junior, and senior) at Computer Science and Engineering department and lecturers having experience of teaching programming languages. We have also recruited professional programmers. The manual rating of the subjects on 60 questions to each (20 from each type, i.e., String, Inheritance, and Thread) closely match our automatic estima-

tion of question difficulty. Six human raters participated in the study. They were divided into three groups - (i) UG students, (ii) Lecturers, (iii) Professional Developers. These three groups' members labeling were validated against the predictions of our QDE tool. It had 65% accuracy with UG students, 65% accuracy with lecturers and 61.25% accuracy with professionals.

1.5 Thesis Organization

The organization of the rest of the thesis is as follows.

In Chapter 2, we have also presented relevant background studies on topic modeling, LDA, supervised machine learning algorithms, inter rater agreement and statistical significance tests.

In Chapter 3, we have discussed about previous works related to our work.

Chapter 4 briefly explains proposed framework, data set generation process and feature selection process.

Chapter 5 focuses on the experimental setups and results. It also illustrates the implication of the results.

Chapter 6 presents threats to validity and how we encounter them.

Chapter 7 concludes our thesis. This chapter also includes the outlines of some future works related to this dissertation.

Chapter 2

Background Study

In this chapter, we present the background studies related to our research.

2.1 Topic Modeling

In machine learning and natural language processing, a topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Topic modeling is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently: "dog" and "bone" will appear more often in documents about dogs, "cat" and "meow" will appear in documents about cats, and "the" and "is" will appear equally in both. A document typically concerns multiple topics in different proportions; thus, in a document that is 10% about cats and 90% about dogs, there would probably be about 9 times more dog words than cat words. The "topics" produced by topic modeling techniques are clusters of similar words. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

Topic models are also referred to as probabilistic topic models, which refers to statistic algorithms for discovering the latent semantic structures of an extensive text body. In the age of information, the amount of the written material we encounter each day is simply beyond our processing capacity. Topic models can help to organize and offer insights for us to understand large collections of unstructured text bodies.

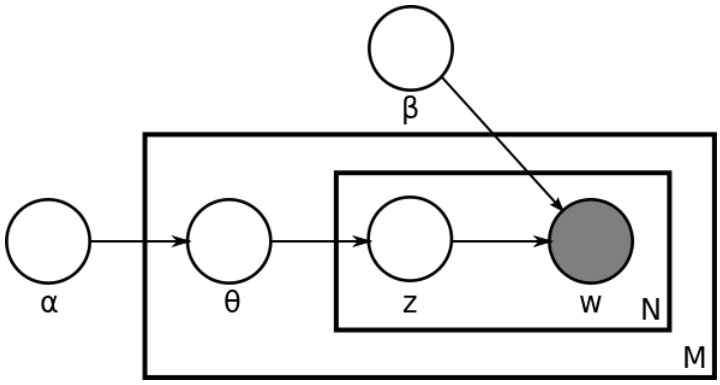

Figure 2.1: Plate notation representing the LDA model.

2.1.1 Latent Direchlet Allocation

In natural language processing, latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's creation is attributable to one of the document's topics.

With plate notation in Figure 2.1, the dependencies among the many variables can be captured concisely. The boxes are "plates" representing replicates. The outer plate represents documents, while the inner plate represents the repeated choice of topics and words within a document. M denotes the number of documents, N the number of words in a document. Thus: .

α is the parameter of the Dirichlet prior on the per-document topic distributions, β is the parameter of the Dirichlet prior on the per-topic word distribution, θ_m is the topic distribution for document m, φ_k is the word distribution for topic k, z_{mn} is the topic for the n-th word in document m, and w_{mn} is the specific word. Here, The words w_{ij} are the only observable variables, and the other variables are latent variables.

The generative process is as follows. Documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA assumes the following generative process for a corpus D consisting of M documents each of length N_i:

1. Choose $\theta_i \sim \mathrm{Dir}(\alpha)$, where $i \in \{1, \ldots, M\}$ and $\mathrm{Dir}(\alpha)$ is a Dirichlet distribution with a symmetric parameter α which typically is sparse $(\alpha < 1)$
2. Choose $\varphi_k \sim \mathrm{Dir}(\beta)$, where $k \in \{1, \ldots, K\}$ and β typically is sparse.
3. For each of the word positions i, j, where $j \in \{1, \ldots, N_i\}$, and $i \in \{1, \ldots, M\}$.
   (a) Choose a topic $z_{i,j} \sim \mathrm{Multinomial}(\theta_j)$.

(b) Choose a word $w_{i,j} \sim \text{Multinomial}(\varphi_{z_{i,j}})$.

The lengths N_i are treated as independent of all the other data generating variables (w, z).

2.2 Supervised Learning Techniques

Supervised learning is the machine learning task of inferring a function from labeled training data [8]. The function aims to predict based on those labeled evidences in presence of uncertainty. A collection of reasonable number of labeled evidences is called a training data set. Each of these labeled evidences is called a training example. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value or label. A supervised learning algorithm analyzes the training data, "learns" from the observations by identifying patterns in them and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. Its performance will improve with the increasing number of training data. Supervised learning are very much promising in case of classification and regression problems.

Given, a data set with n training instances: $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, we want to learn a rule: $(f : x \to y)$ to predict output y for new unseen input x. If we want to predict real valued output(e.g.: house rent) with supervised learning, it is called **Regression**. On the contrary, if we want to predict discrete valued outputs(e.g.: label of hand-written digits) then it is called **Classification** problem. No matter what we do, we must make the prediction function or model general so that it can perform well in case of unseen instances.

Since difficulty level prediction is a classification problem, we have applied different supervised learning algorithms on our data set(e.g.:). In this section, we briefly describe the supervised learning techniques that we have applied on our data set.

2.2.1 Primary Supervised Machine Learning Algorithms

Primary machine learning approaches try to to learn one hypothesis from the training data. In this subsection, we describe some primary supervised machine learning algorithms.

Naive Bayes

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in

the number of variables (features/predictors) in a learning problem. In the statistics and computer science literature, Naive Bayes models are known under a variety of names, including simple Bayes and independence Bayes.

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $x = (x_1, x_2, x_3, \ldots, x_n)$ representing some n features (independent variables), it assigns to this instance probabilities $p(C_k|x_1, x_2, \ldots, x_n)$ for each of k possible outcomes or classes C_k.

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is unfeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as:

$$p(C_k|x) = \frac{p(C_k) * p(x|C_k)}{p(x)} \tag{2.1}$$

This equation can be written using Bayesian probability terminology as,

$$posterior = \frac{prior * likelihood}{evidence} \tag{2.2}$$

Support Vector Machine

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall into.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

In linear SVM, kernel is considered as a linear equation. Given, a training data set with n points of the form: $\{(\vec{x_1}, y_1), \ldots, (\vec{x_n}, y_n)\}$ and $y_i \epsilon \{1, -1\}$, indicating which class the point x_i belongs to. We want to find the "maximum-margin hyper-plane" that divides the group of points $\vec{x_i}$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyper-plane and the nearest point $\vec{x_i}$ from either group is maximized. Any hyper-plane can be written as the set of points $\vec{x_i}$ satisfying,

$$\vec{w} \cdot \vec{x} - b = 0 \tag{2.3}$$

where $\vec{w}$ is the normal vector to the hyper-plane. The parameter $\frac{b}{\|\vec{w}\|}$ determines the offset of the hyper-plane from the origin along the normal vector $\vec{w}$.

Linearly non-separable problems can be separated using Non-linear SVM that does uses dimension transformation to map $\phi : (x_1, x_2, \ldots) \rightarrow (f(x_1), g(x_2), \ldots) \rightarrow (y_1, y_2, \ldots)$. Here, the main idea is linear-separability increases as the feature dimension increases. The problems associated is that increase in dimensionality leads to high computation. Hence, we introduce kernel trick. Some common kernels are: $K(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v} + 1)^2$, $K(\vec{u}, \vec{v}) = (\vec{u} \cdot \vec{v} + 1)^p$, $K(\vec{u}, \vec{v}) = e^{-\|\vec{u} - \vec{v}\|^2 / 2\sigma^2}$ etc.

2.2.2 Ensemble Machine Learning Algorithm

In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. In contrast to primary machine learning approaches which try to learn one hypothesis from training data, ensemble methods try to construct a set of hypotheses and combine them to use. Usually, it performs better in sense of accuracy and at the same time the model it provides is more general.

Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.

In random forest, each tree in the ensemble is built from a sample drawn with replacement (i.e. a bootstrap sample) from the training set. While splitting a node during the construction of the tree, the split that is chosen is not the best split among all features. In stead, the split that is picked is the best among a random subset of the features. As a result of this randomness, the bias of the forest slightly increases, with respect to the bias of a single non-random tree, but due to averaging its variance decreases. Decreases in variance usually compensates the increase in bias, hence yielding a better predicting model.

Ada Boosting

The core principle of Ada Boosting is to fit a sequence of weak learners on repeatedly modified versions of data. The prediction from all of them are combined through a weighted majority vote to produce the final prediction. The data modification at each so-called boosting iteration consists of applying weights $w_1, w_2, \ldots, w_N$ to each of the training samples. Initially, those weights are all set to $w_i = 1/N$, where N denotes the number of weak learners. It means that the first step simply trains weak learners using original data. For each successive iteration, the sample weights are individually modified and the learning algorithms are applied to reweighted data. At a given step, those those training examples that were incorrectly predicted by the boosted model in the previous step have their weights increased, whereas the weights are decreased for the training instances that were predicted correctly. As iterations proceed, instances that are hard to predict receive ever-increasing influence. Each subsequent weak learner is therefore forced to concentrate on the examples that are missed by the previous ones in the sequence.

2.3 Measuring Inter Rater Agreement

When a dataset is manually labeled is to be applied for supervised machine learning,it is necessary to measure the agreement among the raters. In statistics, inter-rater reliability or inter-rater agreement is the degree of agreement among raters. It gives a score of how much homogeneity, or consensus, there is in the ratings given by judges. It is useful in refining the tools given to human judges, for example by determining if a particular scale is appropriate for measuring a particular variable. If various raters do not agree, either the scale is defective or the raters need to be re-trained.

Fleiss' kappa (named after Joseph L. Fleiss) is a statistical measure for assessing the reliability of agreement between a fixed number of raters when assigning categorical ratings to a number of items or classifying items. This is used when three individuals label the dataset separately. This contrasts with other kappas such as Cohen's kappa, which only work when assessing the

Table 2.1: Interpretation of κ value

κ	Interpretation
<0	Poor agreement
0.01 - 0.20	Slight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 1.00	Almost perfect agreement

agreement between not more than two raters. The measure calculates the degree of agreement in classifications.

If a fixed number of people assign numerical ratings to a number of items then the kappa will give a measure for how consistent the ratings are. The kappa, κ can be defined as,

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{2.4}$$

The factor $1 - \bar{P}_e$ gives the degree of agreement that is attainable above chance, and, $\bar{P} - \bar{P}_e$ gives the degree of agreement actually achieved above chance. If the raters are in complete agreement then $\kappa = 1$. If there is no agreement among the raters (other than what would be expected by chance) then $\kappa \leq 0$.

Landis and Koch (1977) [9] gave the Table 2.1 for interpreting κ values.This table is however by no means universally accepted, but this is used as suggested interpretation.

2.4 Statistical Significance Test

In statistical significance test, generally two statistical data sets are compared. Alternatively, a data set obtained by sampling is compared against a synthetic data set from an idealized model. It is done to identify whether the two groups originate from same population To select the proper test method, it is important to identify whether the data follow normal distribution. In Figure 2.1, we have presented different steps of statistical test. Depending on the result of normality test (Shapiro-Wilk test), a test is selected to identify whether two groups significantly differ from each other. Finally, determine the magnitude of difference between the two groups, effect size is calculated. The selection of test not only depends on the normality test, but also on the type of variables. For ordinal(continuous) dependent variable (e.g., review interval, code churn etc.), Mann-Whitney U test is applied, whereas for dichotomous (categorical) variable (e.g, rating of questioner, rating of answerer) chi-square test is applied. In the next sub-section, we present brief discussion on statistical tests relevant to our research.

2.4.1 Shapiro-Wilk test

To make a choice between t-test and Mann-Whitney U test, we need to know whether the data set follows normal distribution. The Shapiro-Wilk test, proposed in 1965, calculates a W statistic that tests whether a random sample, x1,x2,...,xn comes from (specifically) a normal distribution. Small values of W are evidence of departure from normality.

The W statistic is calculated as follows:

$$W = \frac{\sum_{i=1}^{n} a_i x_i}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \tag{2.5}$$

where x_i are ordered sample values (x_1 is the smallest) and a_i are constants generated from the means, variances and co-variances of these order statistics of a sample of size n from a normal distribution.

2.4.2 Mann-Whitney U test

Mann-Whitney U test is a non-parametric test that is used to compare two population means that come from the same population.It is also used to test whether two population means are equal or not. It is used for equal sample sizes, and is used to test the median of the two populations. Usually the Mann-Whitney U test is used when the data is ordinal. Rank of the sample size is used to estimate the U value. Wilcoxon rank sum, Kendalls, and Mann-Whitney U test are similar tests and in the case of categorical data, it is equivalent to chi-square test.

Mann-Whitney U test, being a non-parametric test, does not make any assumptions related to the distribution. There are, however, some assumptions as stated below:

- The sample drawn from the population is random.
- There exists independence within the samples.
- Ordinal measurement scale is applied.

It is estimated as:

$$U = n_1 n_2 + \frac{n_2(n_2+1)}{2} - \sum_{i=n_1+1}^{n_2} R_i \tag{2.6}$$

where, $U =$ Mann-Whitney U test,
$n_1 =$ sample size one,
$n_2 =$ sample size two,
$R_i =$ Rank of the sample size,

Mann-Whitney U test is frequently used in psychology, medical/nursing and business. For example, in psychology, it is used to compare attitude or behaviour. In medicine, it is used to analyze the effect of two medicines and whether they are equal or not. It is also used to analysis whether or not a particular medicine cures ailment. In business, it can be used to know the preferences pf different people and also to see if that changes depending on location. Different confidence levels can be considered for Mann-Whitney U test and that is indicated by variable p (e.g., $p = .05$ means 95% confidence level).

Chapter 3

Related Works

In the learning process, the learners of programming languages or even programmers take resort to tutorials and Q/A sites. The number of such kind of existing tutorials and Q/A sites is pretty huge. Among the technical tutorial and Q/A sites, Code Project and Stack Overflow are noteworthy. To analyze the interactions in these sites and to design make them more helpful to users, the researchers are carrying out various studies on the usage pattern and different characteristics of learning. Some studies such as [10] have tried to correlate the search trends on Google and Stack Overflow and show that technical terms searched and asked have strong correlation over time. Searching and asking of newer, specific technical terms have a stronger correlation, compared with older, general technical terms.

When the official documentation of any product is sparse or even non-existent, answers on Stack Overflow becomes a substitute for official documentation. To measure the interest of users on a different topic, research has been conducted on what kinds of questions are asked on Q/A websites for programmers, which questions are answered, and how best answers are selected in [11]. It found several categories of questions that are asked on Stack Overflow frequently. This study found that "How-to" questions are asked most frequently. Other categories of questions asked on Stack Overflow include about discrepancy, development environment, error, conceptual, code review, and novice inquiries. Another study [12] discusses the use of code examples while learning a new programming language or API. It shows that programmers often depend on code examples to support their learning.

A good amount of effort has been put to investigate the contents of questions on Stack Overflow. Arora et al. [13] aim to flag questions good or bad on Stack Overflow. Basically what they did is to assess if a question asked conforms to the standard of Stack Overflow. If a question matches the expected standard, it is marked as good and bad otherwise. Jiarpakdee et al. [14] discover the relation between question quality and effective features of that question. Here, by saying effective features, they mean positive sentiment, negative sentiment and politeness. They also used textual, community-based information extracted from the dataset they used.

They found that though effective features alone cannot build a model good enough to predict question quality, one of the mentioned effective features - politeness ranking second important feature of the model, as a group can improve the performance of a model to identify the question quality. Baltadzhieva et al. [15] studied upon the individual terms or words used in writing the title and body of the question and studied to which extent do these term can help to predict the probability of a question to receive an answer and the score that the question may have. Studying on two indicators of question quality: question score and number of answers on that questions they came to a decision that their model performed better when terms were included.

A study [5] on Stack Overflow tries to augment API documentation from the insights in Stack Overflow posts. They applied a novel supervised machine learning approach SISE that uses the sentences as features. This study shows a way to consider Stack Overflow metadata and parts-of-speech tags to significantly improve extraction processes. Another study by Toba et al. [16] studies on the quality of the answers provided by the users on CQA sites. They proposed a hybrid hierarchy-of-classifiers framework to model QA pair with a view to identifying high-quality answers. They also found out a number of novel features to distinguish answers' quality. Rocha et al. [17] showed the feasibility of automatically generated tutorials. They developed and evaluated several methods to generate tutorials for APIs with the contents of Stack Overflow. They also organized those contents according to their complexity level that had better result regarding the generation methodology.

Competition based difficulty level estimation of the questions was done in [6]. This study compares between users and questions. They adopted some assumptions. They assumed that the difficulty level of a question is higher than that of the expertise level of the user who asks the question in general. They also assumed that the user who provides the accepted answer has a higher expertise level than the user who posted the question. This is a hard assumption. Also, possible absence of answers at the early stage after a question is posted is beyond applicability of this approach. An improvement over this is proposed in [7]. They added a textual description along with the user-question comparison.

Chapter 4

Generation of Dataset and Feature Selection

4.1 Generation of Dataset

First, we discuss how we have gathered data from Stack Overflow. Then we shall discuss the training set generation process we adopted prior to implementing the supervised learning. This whole process is shown in Figure 4.1.

4.1.1 Harnessing Stack Overflow Data

There are two ways to collect Stack Overflow data. some works such as [5] use Stack Overflow API and some others such as [18] use Stack Overflow data dump. Both the processes have respective advantages and disadvantages. The advantage of using Stack Overflow API is that it always reflects the latest content available on Stack Overflow. However, the query mechanism does not offer much pruning facility. On the other hand, if we use Stack Overflow data dump and store it locally in a database, it is possible to use full query facility which is required in our case. Hence, we adopted the second means accepting that the data is not real-time. For an empirical study, this is not a matter of concern. All the data dump we used were downloaded between 20 September to 24 September 2016 from data dump website. [1]

4.1.2 Topic Modeling

Java is one of the most popular programming languages in the world. According to a survey [19] conducted by Stack Overflow, 41.6% back-end developers and 54.2% mobile developers prefer

[1] ia800500.us.archive.org/22/items/stackexchange/

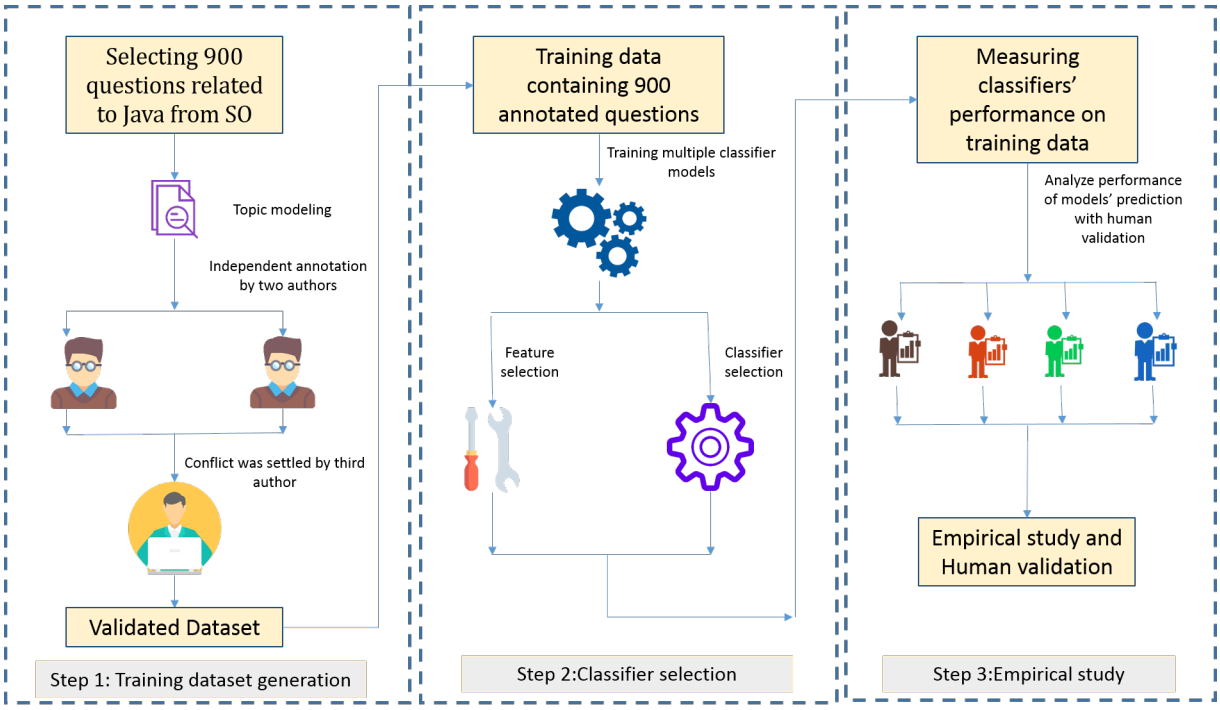

Figure 4.1: An overview of our data processing

Java which makes it one of the top three languages used by these kinds of programmers. 33.6% mathematical and data analysis are done in Java. Java is the most popular language among students since 51.1% of them using this language. 30.7% full-stack engineers use Java which is in the top five of their choice. It is also the second most popular technology on Stack Overflow. Therefore it can be said that Java is one of the most used programming language and the most preferred language for learning purpose. In order to find the issues facing most by the Java programmers, we used LDA for topic modeling. LDA has been proved to be the best technique for topic modeling in natural language documents in [20]. It has been used also in software engineering genre. Lukins et al. [21] used LDA to extract topics from bug reports. The same technique was applied to identify topics in source code by the work of Kuhn et. al. [22]. A work on frequently asked questions by mobile developers by Rosen et. al. [23] applied LDA to the titles only because title summarizes the question.It eliminated the body of the question including code snippet as it might add noise. However, as our main concern is the questions on Stack Overflow, we decided to apply LDA upon the questions using the concept from the aforementioned work. Since our research questions relates to the questions only, answers are not consulted. We used MALLET 2.0.7 [24] for applying LDA upon the question titles of our data set. We had to study with different values of hyper-parameter (K). According to Rosen et. al [23], the value of sampling iteration has such an effect that increasing it increases precision. However, increasing it too much has a bad effect if we want to generalize the topics to some extents. We used the same value $K = 40$ as the work in [23]. We separated the titles from each question and put them in 900 different text files. We excluded common English adverbs,

Table 4.1: **Characteristics of Questions of Different Categories**

Type of Question	Characteristics
Basic	Questions that can be answered with simple built-in functions/API documentation/beginners level books. Questions with comparison between constructs/functions of two languages (for better understanding of the language or for learning a new language). Questions with simple problem-solving.
Intermediate	Questions that require a relatively deeper understanding of the language to answer, for example Why type questions. Questions where the questioner knows about the answer/solution of the question/problem but wants to know more efficient answer/solution. Questions related to time complexity, memory usage or other different resource usages of a system/solution. Questions that require answers with conceptual reasoning/underlying philosophy of any programming construct/API or syntax/design principle.
Advanced	Questions that deal with hard/critical problems where solution needs in-depth programming knowledge or conceptual/logical thinking. Questions that require advanced in-depth knowledge of internal language structure. Questions that deals with infrequently/rarely used framework/API.

pronouns, conjunctions and prepositions provided by [24] from our consideration.

4.1.3 Training Set Generation

In this work, we only considered questions regarding three popular Java related topics: Strings, Threads and Inheritance. They are among the popular Java topics [25]. However, the methodology followed in our work is generalized and will work for any type of question. We randomly selected 300 Q/A threads from each of aforementioned Java topic categories. We did not consider any proportion of posts with respect to the total number of posts of the corresponding category in Stack Overflow.

After extracting the questions, the first two authors labeled them into 3 categories, i.e., basic, intermediate, and advanced. During this labeling process, they were only allowed to view the titles and descriptions of the questions. Table 4.1 describes the characteristics of the types of questions we observed for these 3 categories.

Here we discuss the properties of questions in Stack Overflow that we considered while labeling the data set. A thorough and careful prior observation of a large number of questions identified these properties. In short, the answers to the basic questions require knowledge from traditional learning materials such as books, basic API documentation, tutorials, and simple programming logic. The intermediate questions require conceptual knowledge of the language/platform, designing efficient solutions and understanding resource usage of the system/program. The advanced ones deal with advanced logical thinking to solve a hard/difficult problem and in-depth

knowledge of the internal structure of the language/platform.

We used Cohen's kappa coefficient [26] to measure the inter-rater agreement. The result is $\kappa = 0.738$, indicating high agreement in labeling. In the case of disagreements, the 3rd author provided a decisive vote. After these preprocessing steps, we had 900 labeled questions in the training data set. Of this 900 questions, 598 (66.44%) questions were labeled into the basic category, 244(27.11%) questions belong to the intermediate category, and the remaining 58(6.44%) questions were categorized as advanced. The hypotheses of this study require the classification of questions into either basic or non-basic category. Therefore, to improve the performances of difficulty classifiers, we combined our three class data set into a two-class data set by relabeling both intermediate and advanced questions as non-basic questions. All the data used in this research and the test and validation results have been uploaded to this link [2]

4.2 Feature Selection

After generating the training data set, we derive some features to be used in the supervised learning process. We mainly considered the attributes of the questions. No attribute of the answers was taken into account except the response time of the first answer and median response time of all the existing answers. Following the guidelines to select features offered by Yang et al. [27] and Joachims et al. [28] we try to remove features that are either redundant or irrelevant and can thus be removed without incurring much loss of information [29].The selection helps to reduce over-fitting and memory requirement of the classifier. Moreover, it will give an idea about the correlation of different features with classification. We used information gain [27] to select features. Information gain is a measure of the reduction in entropy of the class variable(Labeling) after the value for the feature is observed and is widely used for feature selection.

To select features we used WEKA's[3] select attribute [30,31] option. In Table 4.2, we present the features we used in decreasing order of their respective information gain. Most of these are self-explanatory. The score of the question implies the difference between upvotes and downvotes of a question indicating the extent of research effort, clarity and usefulness of the question. Favorite counts number shown on a question indicates how many community users marked it as 'favorite' under their profile that lets one find it easily later.

[2] https://goo.gl/tFpvZV
[3] http://www.cs.waikato.ac.nz/ml/weka/

Table 4.2: **Features with highest Information Gain**

Ranking	Feature
1	Fastest Response time to the Question
2	Median Response time to the question
3	Question Body Size
4	Favorite Count of the question
5	Age of the Questioner
6	Score of the Question
7	View Count of the Question
8	Reputation of Questioner
9	Comment Count of the Question
10	Answer Count of the Question

Chapter 5

Empirical Studies

In this chapter, we present the empirical evaluation of our approach and evaluate the performance of different machine learning classifiers in the real environment. We also investigate the research questions we proposed previously.

5.1 Performance Evaluation of Supervised Learning Techniques

After preprocessing data and selecting the features, we see that our data is imbalanced towards basic questions. Hence, to remove the imbalance we applied SMOTE [32] and increased the number of non-basic questions by oversampling. Next, we have applied different machine learning techniques to select the best model for this problem for subsequent use. We tested different machine learning algorithms including ensemble approaches that are commonly used for text classification in software engineering (e.g., [33]) on our training set, e.g., k-nearest Neighbour (Ibk) [34], Decision Tree (J48) [35],Random Subspace [36], Naive Bayes [37], Bayesian Network [38], Logistic Regression [39], Decision Table [40], Random Forest [41], AdaBoost with Decision Stump [42], Multilayer Perceptron [43], LogitBoost [39] and Support Vector Machines (SMO, the sequential minimal optimization implementation in WEKA) [44]. We used WEKA's default setting for each classifier. We have applied "10-fold cross-validation" to train our data and test them 100 times. Table 5.1 shows accuracy, precision, recall, and f-measure for all the classifiers mentioned above using 10-fold cross validation.

We take both accuracy and f-measure as a performance metric. The logic behind taking f-measure as a performance metric is that it gives us an opportunity to remove any choice biases as most of the questions were annotated as basic. We may observe from the table that only three classifiers achieved accuracy and f-measure above 0.7. After comparing results for all the classifier mentioned above with 10-fold cross validation, we found that Random Forest clas-

Table 5.1: **Performance of Different Classifiers**

Classifier	Accuracy	Precision	Recall	F-measure
k-nearest neighbour (Ibk)	.559	.585	.559	.525
Decision Trees (J48)	.6522	.652	.652	.652
Naive Bayes	.5508	.617	.551	.474
Random Subspace	.7221	.722	.722	.722
Decision Table	.6847	.697	.685	.680
Random Forest	**.7271**	**.729**	**.727**	**.726**
Adaboost with Decision Stump	.6514	.651	.651	.651
Multilayer Perceptron Neural Network	.617	.617	.617	.617
LogitBoost	.693	.693	.693	.693
Logistic	.6798	.650	.650	.650
Support Vector Machines	.6015	.621	.601	.586
Bayesian Network	.7088	.709	.709	.709

sifier achieved both highest accuracy and f-measure score, i.e., 0.727 and 0.726, respectively. Therefore, for the rest of our work, we consider Random Forest as our chosen classifier. This is referred as QDE tool in the rest of the work.

The confusion matrix created by this tool on our training data is shown in Table 5.2.

Table 5.2: **Confusion Matrix of Random Forest on Training Set**

		Classified As		
		Basic	Non-Basic	Total
True Class	Basic	406	192	598
	Non-Basic	136	468	604
	Total	542	660	1202

As we see from Table 5.2, out of 1202 instances 874 instances were classified correctly achieving an accuracy of 72.712% with 10-fold cross-validation. It is evident from the analysis that the QDE tool is quite successful in both identifying basic and non-basic questions. Therefore, to answer our first research question regarding the ability to give a nominal response to a Stack Overflow question, we conclude that our approach was able to categorize questions into the two categories with reasonable accuracy and precision.

5.2 Ground Truth Validation of the Results of QDE

In order to see how the QDE tool performs in the real environment, we tried to verify its performance in comparison with six human raters. The population for this experiment was recruited from the students of various study levels and the professionals of various experience levels. The students who were considered for this study were from junior and senior levels at Department

of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET). We recruited professionals who have experience of Java programming for a long period of time. Two lecturers from BUET who have experience in teaching programming were also recruited for this study.

In case of recruiting students, we randomly selected one student from each level, i.e., junior and senior. For this purpose, we used student website [1]. We contacted some randomly selected students who have made their email address public. It took 10 emails to recruit those two students (response rate 20%). We have consulted the university curriculum to have a better understanding of their academic pipeline. The junior students had taken two programming language courses on structured programming and object oriented programming, one course on data structures and one algorithm course. They also have the experience of using java in small scale software development and also had taken courses on compilers, databases and had experience of doing practical projects based on teachings of those courses. The senior students had taken advanced software engineering courses, had experience of reasonable-scale software development. The lecturers who were recruited for this study had experience of teaching different programming languages including Java (mentioned on their website) for three years and also took part in consultancy projects on behalf of BUET.

For professionals, we used Facebook and a mailing list of professional programmers in Bangladesh. These professionals had experience in the software industry for more than five years.

The study was conducted using Google Forms and there were no time constraints. To ensure minimum bias when we recruited participants for this study, we informed the participants neither the aim of this study nor the process of this experiment. We selected these participants in a way so that they can be compared as a pair according to the same or close level of experience with Java.

Then we verified the performance of the model with respect to the labeling of the human raters. The question given to the participants were then predicted by our classifier model. The accuracy (match between results of the human and QDE tool) of our QDE tool is shown in Table 5.3.

Table 5.3: **Result of the Study**

Rater	Average Accuracy
UG students	65%
Lecturers	65%
Professional Developers	61.25%

As previously mentioned, all the validation and test results are uploaded in the web link[2]. As

[1] http://cse.buet.ac.bd/moodle/
[2] https://goo.gl/tFpvZV

we can see from Table 5.1, The average accuracy was 72.71%. It is evident that our classifier model achieved almost the same accuracy it had in the training set.

To answer to our second research question, we conclude that our QDE tool had almost the same accuracy of prediction it had on the training set.

5.3 Hypotheses Testing Results

We have collected all the questions of Java String, Thread and Inheritance category from Stack Overflow which have an accepted answer. There were 5505 Inheritance, 16829 Thread and 22578 String questions. We classified these questions with our chosen classifier i.e., Random Forest. We can categorize the questions into basic and non-basic (which consists of intermediate and advanced questions) classes for the testing hypothesis as discussed before. We also collected the reputation points of the accepted answerers for these questions. The question distribution of three categories is shown in Table 5.4.

Table 5.4: **Question Distribution for three categories**

	Basic	Non-Basic	Total
Inheritance	2928	2577	5505
Thread	8038	8791	16829
String	14601	7977	22578
Total	25567	19345	44912

We performed Shapiro-Wilks normality tests [45] on all the measures in this study and found that first two distributions following a normal distribution and the last one significantly differing from a normal distribution. Therefore, we used parametric hypothesis tests(t-tests [46])for the first two hypotheses and non-parametric test(Mann-Whitney U test [47]) for the last hypothesis introduced in section 2 and used mean for first two cases and median for the last case to report the central tendencies.

5.3.1 H1: Basic question vs Accepted answerer rating

Table 5.5 shows the mean rating of accepted answerers for both basic and non-basic questions for three categories. The mean rating of accepted answerer is significantly higher (t-test, $p < .001$ for all the categories) for basic questions on all the categories, supporting H2. Table 5.5 shows t-value, critical t-values and df (degree of freedom) for all the categories.

Table 5.5: **H1: Basic question vs Accepted answerer rating**

Category	Mean Rating of Accepted Answerers		t-value	t-critical	df
	Basic Questions	Non-basic questions			
Inheritance	112412.4	72800.21	8.25	1.65	5503
Thread	94274.6	62441.5	13.5	1.65	16827
String	85988.1	53612.5	15.574	1.65	22576

We see from Table 5.5 that for all categories t-values are greater than respective t-critical values supporting H1.

5.3.2 H2: Questioner rating vs Accepted answerer rating

Table 5.6 shows the mean rating for both questioners and accepted answerers for three categories. The mean rating of accepted answerer is significantly higher (paired t-test, $p < .001$ for all the categories) than mean rating questioner on all the categories, supporting H2. Table 5.6 shows t-value, critical t-values and df(degree of freedom) for all the categories.

Table 5.6: **H2: Questioner rating vs Accepted answerer rating**

Category	Mean Rating		t-value	t-critical	df
	Accepted Answerers	Questioners			
Inheritance	93869.14	3719.64	37.38	1.65	5503
Thread	77645.88	3203.01	62.87	1.65	16827
String	74549.5	2845.17	71.79	1.65	22576

We see from Table 5.6 for all categories t-values are greater than respective t-critical values supporting H2.

5.3.3 H3. Non-basic Questions vs Lines of Code (LOC)

In Stack Overflow, sometimes code snippets are found both in questions and answers. These snippets help to present the problems and the solutions as well. Non-basic questions contain larger code snippets (line of code) than the basic ones since only text description sometimes fails to present the complex problems found in non-basic questions. To evaluate the hypothesis, we first excluded the questions without any code snippets from the dataset. Therefore, our statistical test result solely depends on questions with code snippets. We performed Shapiro-Wilk test on the dataset and found that it does not follow normal distribution. Since the variable type of LOC is ordinal and the dataset does not follow normal distribution, we applied Mann-

Whitney U test to evaluate the hypothesis. We found $p < .001$ which implies that the result is statistically significant to support H3.

5.4 Implications of the study

This study on the difficulty level categorization has several implications on the users of Stack Overflow.

1. Benefiting the users' teaching material collection from Stack Overflow.
2. Benefiting the users' learning process from Stack Overflow.
3. Providing helpful code examples
4. Useful for Routing Questions.

These following subsections describe these implications.

5.4.1 Benefiting the users' teaching material collection from Stack Overflow

Nowadays, teachers/instructors of programming courses often take examples from Stack Overflow posts to demonstrate real world issues and sometimes for practice exercise. The level of students varies from sophomore year to graduate level. Hence, it is very useful to have a classification of posts/discussions that match the level of target audience. Manually identifying them matching desired difficulty level is cumbersome as numerous entries are being posted in every minute and few of them are of a particular level, e.g. advanced. Thus, the developed QDE tool is expected to benefit teachers select suitable examples more conveniently and appropriately.

5.4.2 Benefiting the users' learning process from Stack Overflow

The users on Stack Overflow are of different expertise levels. Their grasp of a particular topic may vary. While learning a particular thing from Stack Overflow, s/he may be confused about the topics and solutions available to the specific problem the user is facing. There may exist many solutions to a problem that troubles a user. Efficient and better solution may require the user to have knowledge on advanced topics whereas the same problem may be solved with basic concepts in a less efficient and less appreciated way. If a user has to use a significant amount of time in identifying questions that match his/her level, s/he may get discouraged and the learning

process is hampered. Therefore categorizing the questions according to their difficulty levels may help a lot to improve the learning practice of the users of Stack Overflow.

5.4.3 Providing helpful code examples

People often prefer to take help through viewing code examples especially in case of learning how to use new APIs or a new language [12]. Often *basic* questions on Stack Overflow are accompanied by some code snippet or code examples. The person who is attempting to answer the question often mentions a single API along with a line of code or a block of code in order to simplify the use of that API. This can be very helpful for the people who like to rely upon code examples while learning.

5.4.4 Useful for Routing Questions

When a new question is asked in Stack Overflow, it is routed to some potential users who are experts on the topic so that answers are posted quickly. On the common topics, there are many users capable of asking, however, with different depths of knowledge. The users' capability is represented by his/her rating. If the difficulty level of a question is identified, it may be routed to the user with a rating that is supposed to match the difficulty level of the question. Consequently, experts will be relieved of dealing with trivial/simple questions. On the other hand, new users may be encouraged by being asked to reply to questions matching his/her level.

Chapter 6

Threats to Validity

In this empirical study, there can be several threats that can challenge the validity of this study. There are four common threats to validity. They are:

1. Internal validity
2. External validity
3. Construct validity
4. Conclusion validity

All these are discussed in the following subsections.

6.1 Internal validity

The primary threat to internal validity is choosing a programming language and tags. We chose to work with a dataset about Java Strings, Inheritence and Threads. Though questions about these topics or having *Java* as tags with the corresponding questions are very much common on Stack Overflow, they do not necessarily cover all the possible topics or questions available here. However, characteristics related to the difficulty of a topic may not be related to the language or topic highly. We think this threat is minimal for the following reasons:

- Java, being a very much popular, vast, and mature programming language, covers a lot of things common to other programming languages. 51.1% back-end code is done with Java according to a survey by Stack Overflow that is shown in Figure 6.1.
- Java unifies many of the programming philosophies such as object orientation and structured programming which brings many modern concepts under the same hood.

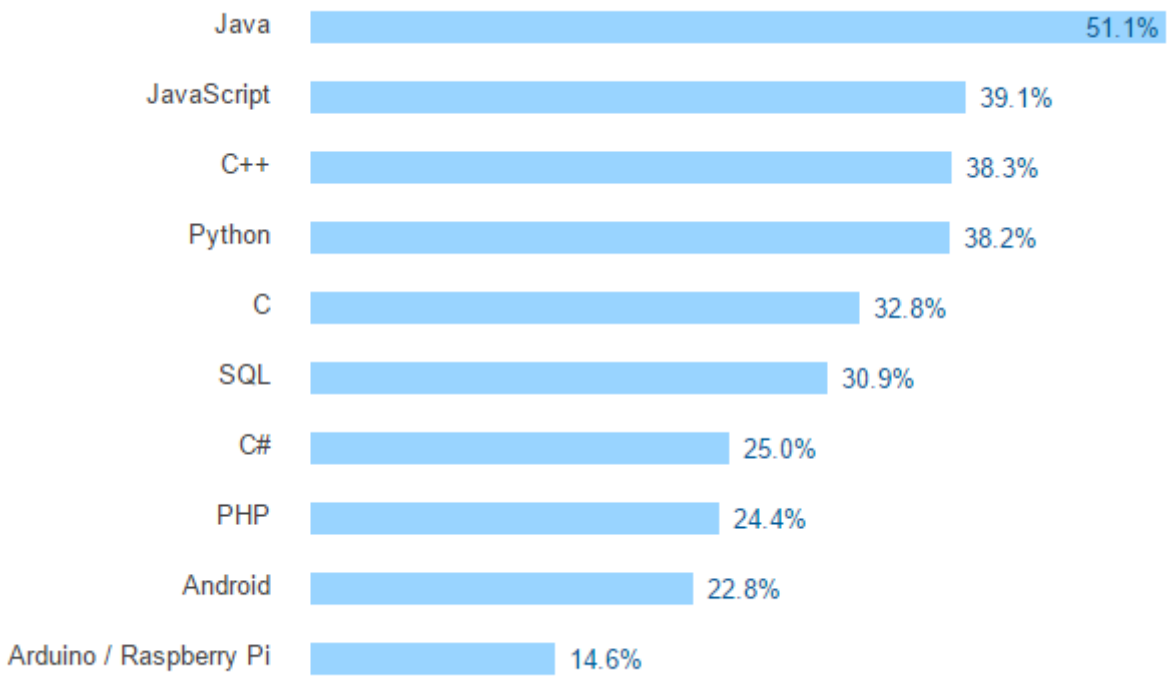

Figure 6.1: Popularity of programming languages in back-end development

- We did not use any feature specific to Java to train our models.

.

Therefore, we believe that our internal threats to validity are not a matter of concern.

6.2 External validity

The consideration of difficulty levels of programming related questions may vary greatly from person to person according to their expertise level. However, our dataset was labeled by first two authors who are senior year undergraduate students who have hands-on experience in application development with Java. Tie-breaking of their conflicts was done by the third author who is a faculty member having experience of teaching Java at undergraduate level. Again, among the six human raters participated in the validation process, two were selected randomly from the undergraduate students (junior and senior). The other human raters are lecturers or professionals in software industry. Hence, we may say that the annotation of the questions according to the difficulty being biased by one's expertise level is minimal. Another threat may come from the impact of age of the raters. According to a survey conducted by Stack Overflow [19], we know that average developers' age is 29.6 years with a median of 27 years. Figure 6.2 shows the graphical representation of Stack Overflow survey participants' age. In our case, persons who

labeled the dataset initially and the human raters participating in manual validation are aged between 20 to 36 years with a median of 30 years which is very close to that of the general users of Stack Overflow. This similarity implies that our methodology is not vulnerable to possible external threats.

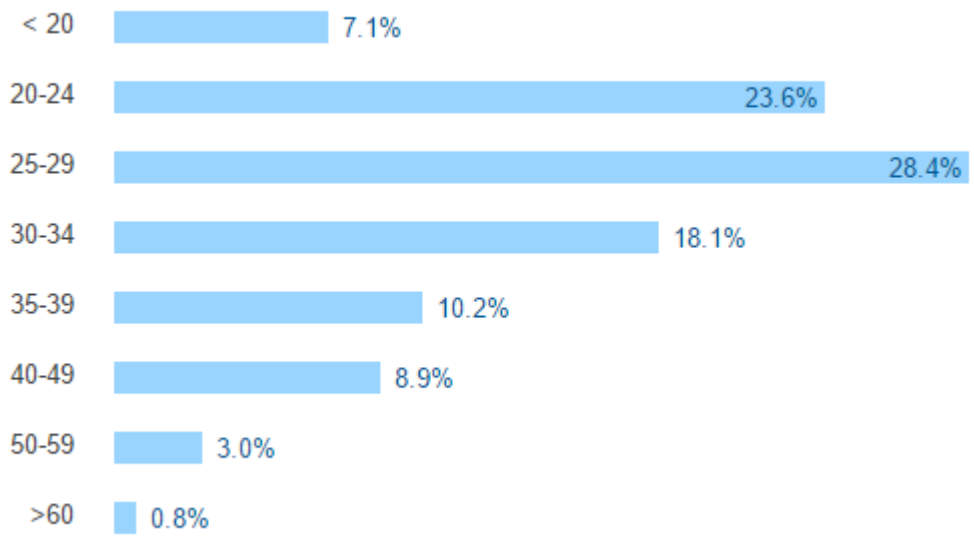

Figure 6.2: Developers' profile: **Age** [Source: http://StackOverflow.com/insights/survey/2016]

Gender of raters may have impact upon the labeling process. 92.8% developers are male as we know from a survey by Stack Overflow, shown in Figure 6.3. Majority of our raters were male to reflect this observation.

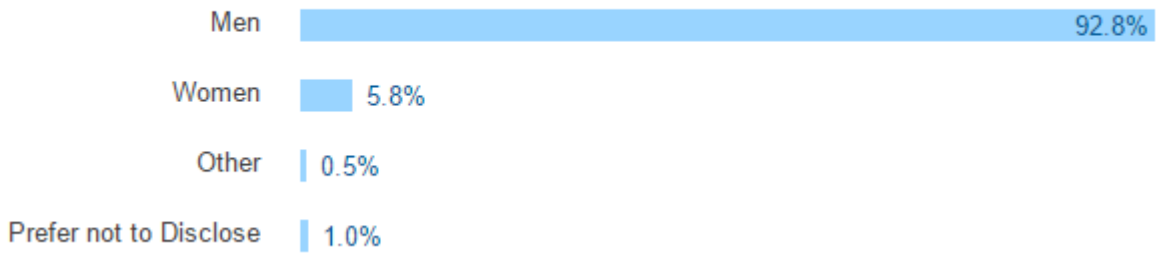

Figure 6.3: Developers' profile: **Gender** [Source: http://StackOverflow.com/insights/survey/2016]

6.3 Construct validity

In absence of any dataset available for our context, we had to build our own using the standard methodology as discussed in Section 4.1. In short, we had the data labeled by the first two authors independently. The third author settled the disagreements independently. Thus, the preparation of dataset is not likely to suffer from any bias. This dataset consisting of 900 questions performed well under different classifiers with accuracy up to 72.71%. We used 10-fold-cross validation while validating the performance.

The predicted data by our model was also manually validated by programmers of different expertise levels. Their result matched with our model's predicted data with similar accuracy. Hence, our manual rating did not face any sort of bias due to variation in expertise levels of the human raters. Therefore, construct validity in our study is expected to be non-existent.

6.4 Conclusion validity

The used dataset was collected from Stack Overflow data-dump from late 2016. This dataset reflects the current trends of Stack Overflow very well. The size of our dataset is close to the sizes of other studies recently done on Stack Overflow data. For the imbalanced nature of dataset, we used oversampling using SMOTE [32]. We have adopted widely used implementations of the commonly used machine learning techniques for this study. Therefore, our study does not have any serious threat to conclusion validity.

Chapter 7

Conclusion and Future Work

In this study we have presented, to the best of our knowledge, the first scheme that determines the difficulty level of programming questions in Q/A sites such as Stack Overflow without requiring complex input pre-processing. We have used supervised learning techniques by developing standard annotated dataset, tried different classifiers, and selected the best one in terms of accuracy and f-measure. The result has also been validated with independent human raters. The measured difficulty level of questions is likely to help the learning process of programming to identify the posts that match the expertise level of target audience and also the site admin to route questions to appropriate persons. Thus it is expected to benefit numerous users who use this alternative, yet, very strong learning medium. Using the best performing model, we classified all available questions of relevant types from Stack Overflow and studied some properties of questions with respect to their difficulty levels. Results of the study suggest that basic questions receive higher view, higher rating, and contain lower lines of code. We also found that the reputation of answerer is usually significantly higher than that of the corresponding asker.

In future, we plan to work on different types of questions on various programming languages and other technical topics such as security.

References

[1] L. Mamykina, B. Manoim, M. Mittal, G. Hripcsak, and Hartmann, "Design lessons from the fastest q&a site in the west)," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2857–2866, ACM, 2011.

[2] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu, "Searching questions by identifying question topic and question focus.," in *ACL*, vol. 8, pp. 156–164, 2008.

[3] M. A. Suryanto, E. P. Lim, A. Sun, and R. H. Chiang, "Quality-aware collaborative question answering: methods and evaluation," in *Proceedings of the second ACM international conference on web search and data mining*, pp. 142–151, ACM, 2009.

[4] R. Ferrigno, "How today developers are learning to code." https://www.stackoverflowbusiness.com/blog/how-todays-developers-are-learning-to-code. Accessed: 2017-03-28.

[5] C. Treude and M. P. Robillard, "Augmenting api documentation with insights from stack overflow," in *Proceedings of the 38th International Conference on Software Engineering*, pp. 392–403, ACM, 2016.

[6] J. Liu, Q. Wang, C.-Y. Lin, and H.-W. Hon, "Question difficulty estimation in community question answering services," 2013.

[7] Q. Wang, J. Liu, B. Wang, and L. Guo, "A regularized competition model for question difficulty estimation in community question answering services," 2014.

[8] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2012.

[9] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[10] C. Chen and Z. Xing, "Towards correlating search on google and asking on stack overflow," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 1, pp. 83–92, IEEE, 2016.

[11] C. Treude, O. Barzilay, and M.-A. Storey, "How do programmers ask and answer questions on the web?: Nier track," in *Software Engineering (ICSE), 2011 33rd International Conference on*, pp. 804–807, IEEE, 2011.

[12] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns, "What makes a good code example?: A study of programming q&a in stackoverflow," in *Software Maintenance (ICSM), 2012 28th IEEE International Conference on*, pp. 25–34, IEEE, 2012.

[13] P. Arora, D. Ganguly, and G. J. Jones, "The good, the bad and their kins: Identifying questions with negative scores in stackoverflow," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pp. 1232–1239, ACM, 2015.

[14] J. Jiarpakdee, A. Ihara, and K.-i. Matsumoto, "Understanding question quality through affective aspect in q&a site," in *Proceedings of the 1st International Workshop on Emotion Awareness in Software Engineering*, pp. 12–17, ACM, 2016.

[15] A. Baltadzhieva, G. Chrupala, G. Angelova, K. Bontcheva, and R. Mitkov, "Predicting the quality of questions on stackoverflow.," in *RANLP*, pp. 32–40, 2015.

[16] H. Toba, Z.-Y. Ming, M. Adriani, and T.-S. Chua, "Discovering high quality answers in community question answering archives using a hierarchy of classifiers," *Information Sciences*, vol. 261, pp. 101–115, 2014.

[17] A. M. Rocha and M. A. Maia, "Automated api documentation with tutorials generated from stack overflow," in *Proceedings of the 30th Brazilian Symposium on Software Engineering*, pp. 33–42, ACM, 2016.

[18] A. Bacchelli, L. Ponzanelli, and M. Lanza, "Harnessing stack overflow for the ide," in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, pp. 26–30, IEEE Press, 2012.

[19] "Stack overflow developer survey results." http://stackoverflow.com/insights/survey/2016. Accessed: 2017-04-06.

[20] 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.

[21] S. K. Lukins, N. A. Kraft, and L. H. Etzkorn, "Source code retrieval for bug localization using latent dirichlet allocation," in *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*, pp. 155–164, IEEE, 2008.

[22] A. Kuhn, S. Ducasse, and T. Gírba, "Semantic clustering: Identifying topics in source code," *Information and Software Technology*, vol. 49, no. 3, pp. 230–243, 2007.

[23] C. Rosen and E. Shihab, "What are mobile developers asking about? a large scale study using stack overflow," *Empirical Software Engineering*, vol. 21, no. 3, pp. 1192–1223, 2016.

[24] A. K. McCallum, "Mallet: A machine learning for language toolkit," 2002.

[25] "All java language topics." https://stackoverflow.com/documentation/java/topics. Accessed: 2017-06-01.

[26] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[27] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Icml*, vol. 97, pp. 412–420, 1997.

[28] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," *Machine learning: ECML-98*, pp. 137–142, 1998.

[29] M. L. Bermingham, R. Pong-Wong, A. Spiliopoulou, C. Hayward, I. Rudan, H. Campbell, A. F. Wright, J. F. Wilson, F. Agakov, P. Navarro, *et al.*, "Application of high-dimensional feature selection: evaluation for genomic prediction in man," *Scientific reports*, vol. 5, 2015.

[30] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1289–1305, 2003.

[31] P. C. Rigby and M. P. Robillard, "Discovering essential code elements in informal documentation," in *Proceedings of the 2013 International Conference on Software Engineering*, pp. 832–841, IEEE Press, 2013.

[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[33] L. B. de Souza, E. C. Campos, and M. d. A. Maia, "Ranking crowd knowledge to assist software development," in *Proceedings of the 22nd International Conference on Program Comprehension*, pp. 72–82, ACM, 2014.

[34] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[35] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[36] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[37] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

[38] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[39] J. Friedman, T. Hastie, R. Tibshirani, *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[40] R. Kohavi, "The power of decision tables," in *European Conference on Machine Learning*, pp. 174–189, Springer, 1995.

[41] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[42] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[43] D. W. Ruck, S. K. Rogers, M. Kabrisky, M. E. Oxley, and B. W. Suter, "The multilayer perceptron as an approximation to a bayes optimal discriminant function," *IEEE Transactions on Neural Networks*, vol. 1, no. 4, pp. 296–298, 1990.

[44] B. Schölkopf and C. J. Burges, *Advances in kernel methods: support vector learning*. MIT press, 1999.

[45] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[46] B. J. Winer, D. R. Brown, and K. M. Michels, *Statistical principles in experimental design*, vol. 2. McGraw-Hill New York, 1971.

[47] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

Appendix A

Codes

A.1 Sql query for generating questions

We use this sql query to generate 300 questions randomly from Java Strings, Thread and Inheritance

```
SELECT
    top 300 Title,score,tags,body

FROM
    Posts
where
    Tags like '%java%' and Tags like '%string%'
    and Tags not like '%javascript%' and Tags not like '%c++%'

ORDER BY
   rand()
```

A.2 Script for LDA in Mallet

We use the following script to do LDA up on the titles of Stack Overflow questions.

```
#!/bin/bash
./bin/mallet import-dir
--input labeling1000_dataset --output labeling1000_dataset.mallet
--keep-sequence --remove-stopwords

bin/mallet train-topics  --input labeling1000_dataset.mallet
```

```
7
8 bin/mallet train-topics  --input labeling1000_dataset.mallet
9 --num-topics 40 --output-state topic-state.gz
10 --output-topic-keys
11  labeling1000_dataset_keys.txt --output-doc-topics
12  labeling1000_dataset_compostion.txt
13
14 bin/mallet train-topics  --input labeling1000_dataset.mallet
15 --num-topics 40 --optimize-interval 40 --output-state topic-state.gz
16 --output-topic-keys
17 labeling1000_dataset_keys.txt --output-doc-topics
18 labeling1000_dataset_composition.txt
```

A.3 Sql query for getting the feature values of a question

We use this for getting the feature values of a question.

```
1 SELECT
2 p.Title,p.score,p.ViewCount,p.AnswerCount,p.CommentCount,LEN(p.body) as ques_siz
3 p.favoritecount,p.id,
4 u.reputation as user_repo,u.creationdate as user_join_date,
5 datediff(MINUTE, p.CreationDate, a.CreationDate) as QATimeGap
6 from posts as p, posts as a, users as u
7 where
8 p.id=a.parentId and
9 p.owneruserid=u.id and
10 a.creationdate =
11 (select min(tau.creationdate)
12 from posts tau
13 where tau.parentID=p.id
14 group by tau.parentID) and
15 p.title like '%Java␣HttpServer/HttpExchange␣GET%'
```

A.4 Sql query for getting median and age values of questions

We use this query to get the median response time of the question and age of the questioner.

```
1 WITH my_table AS
2 (
3   SELECT
```

```
    P.title,U.age,p.score,p.ViewCount,p.AnswerCount,p.CommentCount,LEN(p.body) a
p.favoritecount,u.reputation as Question_user_repo,
acuser.reputation,
    DATEDIFF(minute, P.CreationDate, A.CreationDate) AS ResponseMinutes
  FROM
    Posts P
    INNER JOIN Posts A ON P.Id=A.ParentId
    INNER JOIN Users U on p.ownerUserId=U.id,
    posts as ac,
    users as acuser

  WHERE
    p.AcceptedAnswerId=ac.id and
    ac.OwnerUserId=acuser.id and
    P.PostTypeId=1 AND
    p.tags like '%java%' and
    p.tags like '%multithreading%' and
    p.AnswerCount > 0
)

SELECT DISTINCT title,
    PERCENTILE_CONT(0.5)
    WITHIN GROUP (ORDER BY ResponseMinutes)
    OVER (PARTITION BY title) AS Median,
    age,score,ViewCount,AnswerCount,CommentCount,ques_size,favoritecount,
    Question_user_repo,
    min(ResponseMinutes)
    OVER (PARTITION BY title ORDER BY ResponseMinutes) AS QATimeGap,reputation
FROM my_table
```

A.5 Python code for getting number of lines of code (LOC) in question

We use the following Python script to know the number of lines of code in each of the question body.

```
import re
import math
import xlrd
import csv
```

```
5
6 data = list(csv.reader(open('filename.csv')))
7 for row in range(0, len(data)):
8         text = data[row][1]
9         starts = []
10        ends = []
11        sizes = []
12        for m in re.finditer('<code>', text):
13                starts.append(m.end()+1)
14        for m in re.finditer('</code>', text):
15                ends.append(m.start())
16        for i in range(0, len(starts)):
17                newcode = text[starts[i]:ends[i]]
18                linecount = newcode.count(';')+newcode.count('//')+newcode.count
19                sizes.append(linecount)
20
21        print(math.ceil(float(sum(sizes)) / max(len(sizes), 1)))
```

Generated using Undegraduate Thesis LaTeX Template, Version 1.3. Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh.

This thesis was generated on Saturday 9th September, 2017 at 8:31am.