# To Perform Automated Similarity Analysis between Stack Overflow Questions and Online Bug Reports

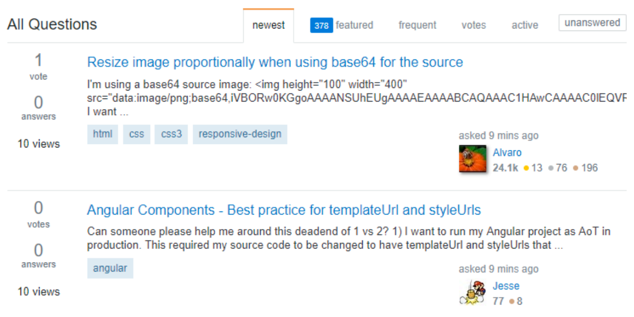
Suchina Parihar
Dept. of Electrical and Computer Engineering
University of Calgary
Calgary, Canada
suchina.parihar2@ucalgary.ca

***Abstract*— Today's software engineering field is composed of a diverse array of technologies, tools, languages and platforms [24]. Nowadays, there are various online communities for developers to learn and share their knowledge and communicate their doubts with other developers [31]. It also helps beginners who can ask their questions and receive answers for their specific queries. Online programming question and answer (Q&A) websites, such as Stack Overflow, provides its participants with rapid access to knowledge and expertise of their peers. With time, these websites have turned into repositories of software engineering knowledge [24]. There are different ways of mining such repositories for topics, trends and patterns of user behaviour. In the proposed project, Latent Dirichlet Allocation (LDA), a statistical topic modeling technique, will be used to analyze the contents of millions of questions and answers and produce relevant topics [6]. These topics are then compared to the Bug Reports of a different software to uncover the relevant questions, thus making this research more automated.**

***Keywords—Stack Overflow;Topic Modelling; Latent Dirichlet Allocation(LDA); Cosine Similarity; Jaccard Similarity***

## I. INTRODUCTION

Nowadays, software engineers use various software information sites to search, communicate, collaborate, and share information with one another [1]. Software information sites play an important role in software engineering [2-3]. Stack Overflow is one of the most popular software information sites where people ask and answer technical questions about software development and maintenance [4]. The user asking a question lacks knowledge of a specific topic and searches for an expert to provide the specific knowledge. In this way, the experts are the source of information. In Stack Overflow, access is free and answers are voted according to the questioner's satisfaction. The questioner can tag a question to indicate a specific subject. Users can vote questions, answers and edits to indicate how helpful they were. The votes determine the user's reputation [5].



Resize image proportionally when using base64 for the source

I'm using a base64 source image: <img height="100" width="400" src="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAQAAAC1HAwCAAAAC0lEQVF
I want ...

Angular Components - Best practice for templateUrl and styleUrls

Can someone please help me around this deadend of 1 vs 2? 1) I want to run my Angular project as AoT in production. This required my source code to be changed to have templateUrl and styleUrls that ...

Fig.1. Two sample Questions in Stack Overflow

Performing Descriptive analysis on Stack Overflow could help us in improving developer experience in using these sites. It would help us to better understand how information within Stack Overflow could be useful to various software development activities, and the community of knowledge is formed to benefit individual developers [6]. Treude et al. manually analyzed 385 questions and assigned them to 10 categories [7]. Treude et al. employed a manual process to analyze the content of questions. In this paper, I would like to automate the process by using textual mining that could automatically assign categories to questions [6].

In this paper, a framework is proposed, which takes data from Stack Overflow as input. Firstly, the distribution of questioners and answerers is investigated. Secondly, the behaviour of questioners and answerers is investigated i.e. whether the questioners and answerers on Stack Overflow are two different sets of people. Thirdly, millions of questions from Stack Overflow are investigated to infer the different topics that developers ask by using topic modeling. Then, to extend this research and make the automation more active, a new dataset of Bug Reports is used. The similarity measures between topics of bug reports and Stack Overflow topics are calculated i.e. for each Bug Report, the corresponding relevant Questions from Stack Overflow are discovered. To achieve the above, text analysis is performed. The descriptive analysis of data helps in understanding the relationships within the data.

Topic modelling of questions and bug reports help in discovering hidden semantic structures in data. This is followed by Similarity analysis of topics to discover similar topics thus unfolding which bug belongs to which question topic.

## II. PRELIMINARIES

In this section, Stack Overflow, LDA, Cosine Similarity, Jaccard Similarity are introduced.

### *A. STACK OVERFLOW*

Stack Overflow allows users to register, post questions, and answer posted questions. Since users are registered, one could track the questions he or she posts, and answers he or she makes [31]. For each posted question, a user can include textual description of the problem. The user can also include code snippets. Code snippets are often separated from other normal texts. Other users can answer the posted question. Multiple answers could be given by various people. The one posting question could then either post a comment or indicate one of the answers as correct. Other people could also rate whether they like either the questions and/or the answers. A snapshot of the StackOveflow page showing a question and its corresponding answer is shown in Fig.1 [6].

### *B. LDA*

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words [8].
Given a dataset of documents, LDA backtracks and tries to figure out what topics would create those documents in the first place. LDA is a matrix factorization technique. In vector space, any corpus (collection of documents) can be represented as a document-term matrix. The value of i,j cell gives the frequency count of word Wj in Document Di. LDA converts this Document-Term Matrix into two lower dimensional matrices (M1 and M2). M1 is a document-topics matrix and M2 is a topic-terms matrix with dimensions (N, K) and (K, M) respectively, where N is the number of documents, K is the number of topics and M is the vocabulary size. LDA makes use of sampling techniques in order to improve these matrices. It iterates through each word "w" for each document "d" and tries to adjust the current topic-word assignment with a new assignment. A new topic "k" is assigned to word "w" with a probability P, which is a product of two probabilities p1 and p2. For every topic, two probabilities p1 and p2 are calculated [9].

*P1 – p(topic t / document d) = the proportion of words in document d that are currently assigned to topic t.*
*P2 – p(word w / topic t) = the proportion of assignments to topic t over all documents that come from this word w.*

The current topic-word assignment is updated with a new topic with the probability, product of p1 and p2. After a number of iterations, a steady state is achieved where the document topic and topic term distributions are fairly good. This is the convergence point of LDA [9].

Parameters of LDA:

1. *Alpha*: alpha represents document-topic density. Higher the value of alpha, documents are composed of more topics and lower the value of alpha, documents contain fewer topics.
2. *Beta*: Beta represents topic-word density. Higher the beta, topics are composed of a large number of words in the corpus, and with the lower value of beta, they are composed of few words.
3. *Number of topics*: Number of topics should be extracted from the corpus. Researchers have developed approaches to obtain an optimal number of topics.
4. *Number of Topic terms*: Number of terms composed in a single topic. It is generally decided according to the requirement.
5. *Number of Iterations*: Maximum number of iterations allowed to LDA algorithm for convergence.

### *C. COSINE SIMILARITY*

The cosine similarity between two vectors (or two documents on the Vector Space) is a measure that calculates the cosine of the angle between them [10]. This metric is a measurement of orientation and not magnitude, it can be seen as a comparison between documents on a normalized space because we're not taking into the consideration only the magnitude of each word count (tf-idf) of each document, but the angle between the documents. What we have to do to build the cosine similarity equation is to solve the equation of the dot product for the cosθ:

$$\vec{a} \cdot \vec{b} = \|\vec{a}\|\|\vec{b}\| \cos\theta$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|}$$

Cosine Similarity will generate a metric that says how two documents are related by looking at the angle instead of magnitude [10].

### *D. JACCARD SIMILARITY*

Jaccard similarity addresses a task of finding textually similar documents in a large corpus such as the Web or a collection of news articles. We should understand that the aspect of similarity we are looking at here is character-level similarity, not "similar meaning," which requires us to examine the words in the documents and their uses [11]. The Jaccard similarity is defined

$$\mathsf{JS}(A,B)=\frac{|A\cap B|}{|A\cup B|}$$

Given a set A, the cardinality of A denoted |A| counts how many elements are in A. The intersection between two sets A and B is denoted A ∩ B and reveals all items which are in both sets. The union between two sets A and B is denoted A ∪ B and reveals all items which are in either set.

## III. MOTIVATION & BACKGROUND

A key aspect of software development is to deliver high quality software in a timely and cost-efficient manner. Code reuse has been widely accepted to be an important approach to achieve this [20]. There are many different sources from where reused code can be obtained, e.g., third-party libraries [20], source code of open source software [21], and Question and Answer (Q&A) websites such as Stack Overflow [22,23]. In the recent years, these Question and Answer websites have been proven successful in solving various technical problems. Thus, developers can take help from these websites and can reuse the code from the answers. The linkage of same and different types of software development documentation might save time to evolve new software solutions and might increase the productivity of the developer[27]. A primary challenge with this type of code reuse is that programmers often resort to these code snippets in an ad-hoc manner, by copying and pasting these fragments in their own source code [20]. Thus, various recommendation systems are developed to recommend the relevant questions in Stack Overflow for a particular problem. Existing research (e.g., [22,24]) has shown that developers resort to Stack Overflow for help e.g., to resolve implementation problems or examine best coding practices. In contrast to this existing work, the objective of my research is to study the relevant topics from Stack Overflow during the maintenance stage for fixing of bugs.

This study was an extension to the previous work done by Wang et al. [6]. The motivation behind this study was to better understand how information within Stack Overflow could be useful to various software development activities and to automate the process of analyzing the data. The descriptive analysis is compared to the previous paper but LDA was used differently. I tuned LDA according to my Project requirement. Apart from the previous Research, I added a new aspect of finding the relevant questions in the Stack Overflow. I used Mozilla Firefox Bug Reports and calculated the similarity of a particular Bug report with the Question topics to find out which set of Questions are related to that particular Bug Report. This research when used with 'tags' can make an excellent Recommendation System.

## IV. METHODOLOGY

### A. *DATA COLLECTION*

The input data is a set of questions posted in StackOverflow along with their answers. For questions which are answered, the list of answers and for each answer, the information of the developer who answers it is also known. For this Project, the dataset is downloaded from Kaggle. It is 10% of StackOverflow questions and answers (1264216 questions and 2014516 answers). The Bug Reports dataset is from Mozilla Firefox and is downloaded from github.

### B. *DATA PROCESSING*

Data Processor analyzes the input and generates two kinds of outputs.

1. Includes the number of times each developer answers questions, posts questions, and the proportion of a developer's posts that are questions.
2. The contents (normal text and code) of the questions will be extracted.

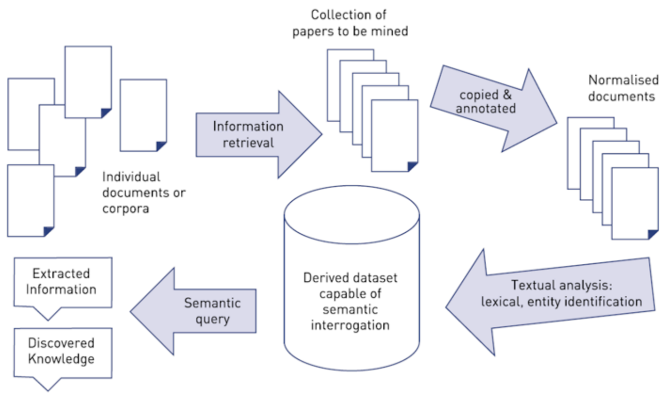

Collection of papers to be mined
Normalised documents
copied & annotated
Information retrieval
Individual documents or corpora
Derived dataset capable of semantic interrogation
Textual analysis: lexical, entity identification
Semantic query
Extracted Information
Discovered Knowledge

Fig.3. Text Mining Process

### C. *TEXT MINING*

Text mining is a process of extracting interesting and significant patterns to explore knowledge from textual data sources [12]. Text mining is a multi-disciplinary field based on information retrieval, data mining, machine learning, statistics, and computational linguistics [12]. Several text mining techniques like summarization, classification, clustering etc., can be applied to extract knowledge. Text mining deals with natural language text which is stored in semi-structured and unstructured format [13].

Generic process of text mining( Fig.3) performs the following steps:

1. Collecting unstructured data from data processor.
2. Pre-processing and cleansing operations are performed to detect and remove anomalies. Cleansing process make sure to capture the real essence of text available and is performed to remove stop words stemming and indexing the data [14].
3. Processing and controlling operations are applied to audit and further clean the data set by automatic processing.
4. Information processed in the above steps are used to extract valuable and relevant information for effective and timely decision making and trend analysis [15].

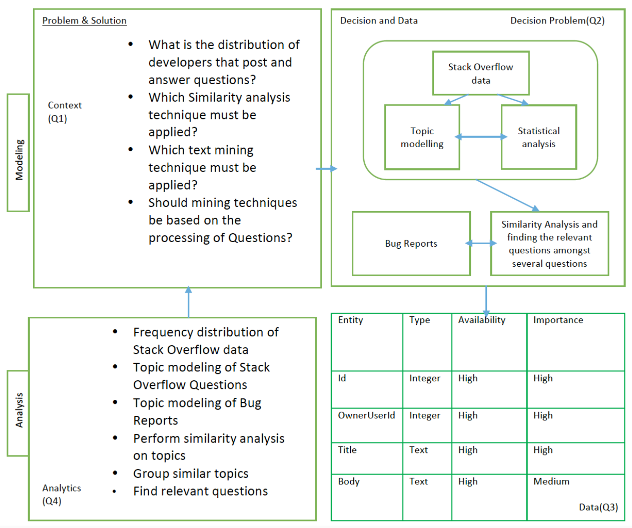
## V. EXPERIMENTS

### A. *ANALYTICS DESIGN SHEET*

The initial plan of the project can be shown with the help of the analysis design sheet. It consists of 4 parts. The first part describes the problem and its solution. There were so many questions in my mind during the initial stage of research, like which techniques to be used and how to figure out the data on which the experiments will be performed. So, I stated them in the first part of Analysis design sheet. The second part is about the data and the relationship between the data. In the third part, some attributes of the data are provided in the table which consists of Id, OwnerUserId, Title, Body etc. The fourth part is about the basic steps to be followed to achieve the goal of the paper. It is like a black box view of the process to be followed.

### B. *RESEARCH QUESTIONS*

**RQ1**: What are the distributions of developers that post questions?



- What is the distribution of developers that post and answer questions?
- Which Similarity analysis technique must be applied?
- Which text mining technique must be applied?
- Should mining techniques be based on the processing of Questions?

- Frequency distribution of Stack Overflow data
- Topic modeling of Stack Overflow Questions
- Topic modeling of Bug Reports
- Perform similarity analysis on topics
- Group similar topics
- Find relevant questions

| Entity | Type | Availability | Importance |
|---|---|---|---|
| Id | Integer | High | High |
| OwnerUserId | Integer | High | High |
| Title | Text | High | High |
| Body | Text | High | Medium |

Fig.4. Analytics Design Sheet

**RQ2**: What are the distributions of developers that answer questions?
**RQ3**: Do developers that ask questions answer questions too?
**RQ4**: What topics do developers ask about?
**RQ5**: What are the relevant topics corresponding to the Bug Reports?

### C. *DESCRIPTIVE ANALYSIS*

We should always perform appropriate Descriptive Analysis before further analysis of our data. It makes us more familiar with the data, check for obvious mistakes, learn about variable distributions, and about relationships between variables. This aspect actually contributes in giving our research more visual clarity, to say the least and also builds the foundation for Analytical software engineering. As it can be difficult to make sense of a large amount of quantitative data, an initial approach, in addition to examining the data visually, is to calculate summary measures, to describe the spread for each variable.

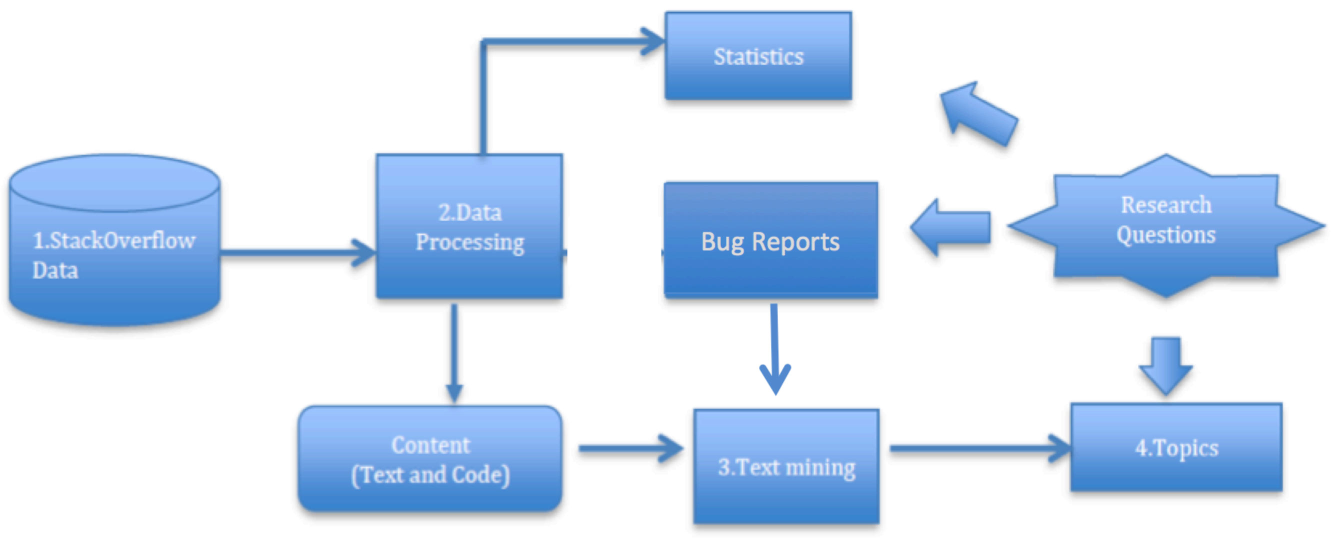


Fig.2. Methodology of the Project

These are of great interest, particularly if a comparison between groups is to be made or the results of the study are to be generalised to a larger group [16].

**RQ1: What are the distributions of developers that post questions?**

This question is answered by performing statistics on the Questions data. I have plotted the Line Graph of questioners in Fig.5. The x-axis of the graph shows Number of questions and y-axis shows number of developers. The y-axis is in log-scale. The graph shows the number of developers that ask a given number of questions. From the graph, we can notice that most developers only ask one question. The number of developers that ask questions' reduces exponentially as we consider a

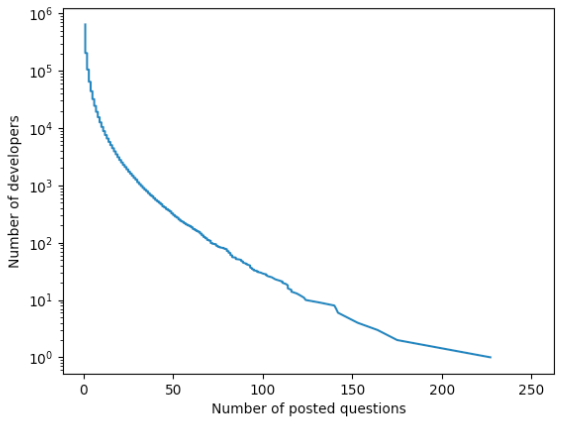


Fig.5. Line Graph of Questioners

higher number of posted questions. The result shows that there are few "regular" questioners on Stack Overflow. This is possibly because many questions have already been asked before and users could find answers to them by just looking into the various pages on Stack Overflow or other question and answer sites via search engines.

**RQ2: What are the distributions of developers that answer questions?**

For this Research Question, I again perform statistics on Answers data and I have plotted the line graph of answerers in Fig.6. X-axis shows number of answers and y-axis shows number of developers in log-scale. The graph shows the number of developers that answer a given number of questions. From the graph, we can notice that most developers only answer one question. The number of developers that answer questions reduces exponentially as we consider a higher number of answers. The highest number of questions a developer answers in this dataset is 3000. There is only one developer that answers this many questions. Compared with the distribution of questioners, the distribution of answers is different. The number of developers that answer a substantial number of questions is more than the number of developers that ask a substantial number of questions.

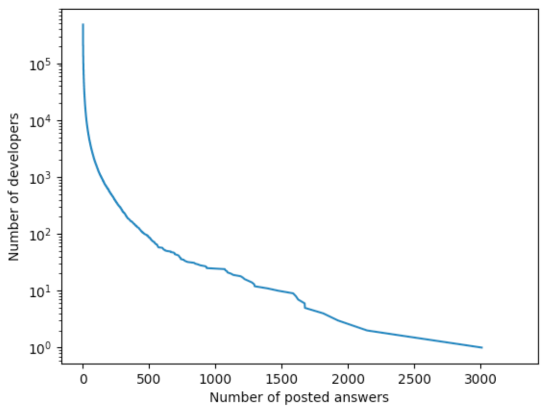


Fig.6. Line Graph of Answerers

**RQ3: Do developers that ask questions answer questions too?**

To answer this research question, I investigated the proportion of posts that various developers make that are answers to some questions. I noticed that a majority of developers only answer questions but do not ask questions. Thus, we could divide the Stack Overflow community into two groups: people that only ask questions, and those that only answer questions. The second group is the majority. These correspond to ideal developers that contribute answers to the community.

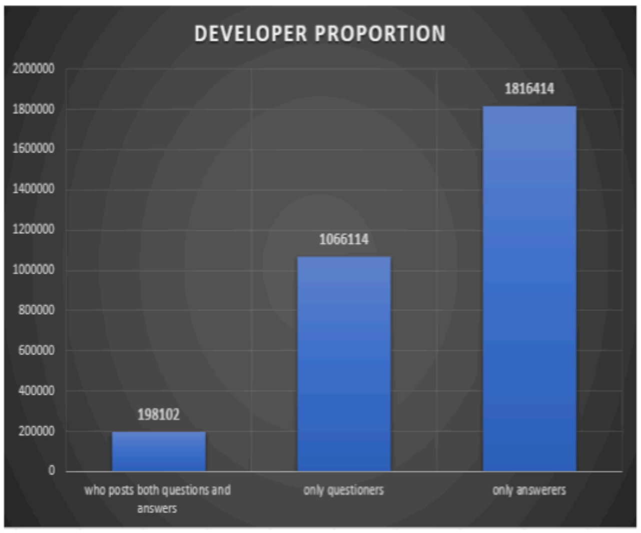


Fig.7. Post Proportion

## *D. TOPIC MODELING*

For Topic Modeling, I ran LDA on the 'Title' column of the Questions dataset. LDA extracts topics in an unsupervised manner; the algorithm relies solely on the source data, word distributions of text, with no human intervention. LDA and LSI are conceptually similar, both are transforms that map one vector space to another. I ran the full LDA transform against the BoW corpus I had, with the number of topics set to 5. And for LSI, I load up the corpus and dictionary from files, then apply the transform to project the documents into the LDA topic space. A topic analysis tool like LDA will try to find N independent word distributions within the word distributions of all the messages. These N word distributions effectively form topics. In this Research, Python as the programming language is chosen, mainly because of its straightforward, compact syntax, multiplatform nature and ease of deployment. I have used a package in Python named Gensim [17] to achieve topic modeling along with their probabilities.

One method for evaluating the number of topics is to evaluate perplexity of the model on hold out data. I did this, but there is no substitute for actually running a few different models with different number of topics and actually have humans evaluate it. You could use Mechanical Turk if you have a lot of data/topics [18]. Gensim provides a mechanism for reporting of model perplexity. While analyzing the LDA log files for Questions dataset, I monitored lines like the following:

INFO: *-12.440 per-word bound, 5555.2 perplexity estimate based on a held-out corpus of 2000 documents with 4302 words*

This means that gensim took the current mini-batch and estimated perplexity based on the adjusted variational Evidence Lower Bound (ELBO), on this held-out corpus. The adjusted bit means that likelihood is re-weighted as if the held-out corpus was the entire training corpus. This is done to make comparisons between models using a different number of topics/model settings easier, as ELBO also includes model regularization parts that can otherwise dominate the score. This held-out perplexity is an estimate, so different mini-batches will give different scores. But in general we'll see the perplexity value decrease as training progresses.

INFO: *-11.980 per-word bound, 4039.6 perplexity estimate based on a held-out corpus of 2000 documents with 4357 words*

For further analysis, I have set a threshold value of 5000 for Questions and 500 for Bug Reports and tried to figure out an ideal number of topics based on that score. For LDA tuning, I used alpha at 0.05 and eta at 8.95511695383e-06.

**RQ4: What topics do developers ask about?**

To answer this research question, I ran LDA on the text and code contents of the questions. I set the number of topics to be 20, and after LDA completes running, it outputs 20 topics: each topic is a set of words sorted in terms of their likelihood of belonging to the topic. LDA does not generate a meaningful label for each topic. We manually study the words in each topic and related questions, and assign a label to the topic. Table 1 shows the 20 topics with manually assigned labels, some representative words in each topic.

| TOPIC # | TOP WORDS | LABEL |
|---|---|---|
| 1 | *Ios, date, column, based, element, trying, group, rest, format, variable model, string, read, sort, specific, range, value, argument, bash* | IOS app system design |
| 2 | *Button, element, template, size, returning, docker, creating, id, layout, db, scala, parsing, unique, css, xml, container, long, automatically, webpack* | Distributed Apllications |
| 3 | *doesn't, work, cannot, statement, vba, activity, entity, framework, memory, invalid, casE, matlab, permission, dependency, submit, use, outside, error, times* | Matlab Framework Issues |
| 4 | *Unable, request, post, join, results, http, pandas, reading, aws, works, condition, location, displaying, instead, listview, local, sending, stream, modal* | Batch processing |
| 5 | *Filter, function, click, passing, undefined, cell, pass, adding, video, fields, reference, upload, node.js, compare, exception, option, collection, method, word* | Server side dependencies |
| 6 | *Message, show, spark, directory, mvc, print, client, asp.net, pdf, push, page, folder, properly, created, back, load, hide, classes, information* | C# and Asp.Net Model View Controller |
| 7 | *Select, bootstrap, issue, install, count, regex, link, convert, facebook, match, expression, graph, powershell, expected, pattern, fragment, recyclerview, hibernate, part* | Social Media APIs |
| 8 | *Change, null, background, values, color, mongodb, git, connection, remote, binding, functions, action, event, plot, dictionary, log, fetch, calculate, bind* | Version Control Systems |
| 9 | *Laravel, loop, node, found, index, property, website, selected, dropdown, able, thread, writing, static, existing, limit, rendering, single, apply, vector* | Multithreading Implementations |
| 10 | *Array, insert, map, error, import, result, value, email, send, replace, first, returns, type, call, instance, arrays, form, console, boot* | Electronic mail apps |
| 11 | *Module, response, key, tables, dataframe, chrome, two, download, box, order, typescript, need, errors, binary, include, file, different, setup, internal* | Distributed data Frame Tables |
| 12 | *angular2, nested, react, list, login, item, content, add, items, menu, wrong,* | Angular JS 2 Front End Web |

| | | |
|---|---|---|
| | *loading, images, grid, merge, resource, start, contains, full* | Development Framework |
| 13 | *Android, line, showing, csv, remove, empty, authentication, command, app, numbers, display, copy, device, file, names, configure, saving, generated, detect* | Android Application Configurations |
| 14 | *Angular, spring, script, package, ionic, parse, delete, test, run, strings, fails, python, xamarin, sum, testing, lines, Jenkins, network, file* | Front End Code Deployment |
| 15 | *R, view, controller, return, save, azure, parent, selenium, external, position, extract, space, npm, records, task, label, shell, cordova, frame* | Automation Testing |
| 16 | *Json, firebase, object, div, component, django, failed, variables, dynamically, header, xcode, parameter, updating, codeigniter, height, document, correctly, screen, accessing* | Model View Template Architecture of Django |
| 17 | *Angularjs, wordpress, bar, chart, syntax, execute, split, url, mobile,* changing, *requests, characters, options, large iframe, storage, navigation, implementation, interface* | Front End Mobile Web Applications using Angular and Wordpress |
| 18 | *Js, columns, input, rows, rowtable, issues, auto, animation, domain, don't, defined, gradle, multiple, performance, program, write, one, events, values* | Data base schema for Java Script animation components |
| 19 | *Studio, output, query, visual, error, missing, stop, character, sql, route, token, configuration, tab, left, end, trouble types, compile, linq* | C# LINQ queries on Visual Studio |
| 20 | *Swift, google, api, excel, validation, connect, attribute, last, tag, core, retrieve, duplicate, setting, ui, notification, sqlite, play, matrix, maps* | IOS Application Development using google api |

TABLE 1. Top 20 Questions' topics

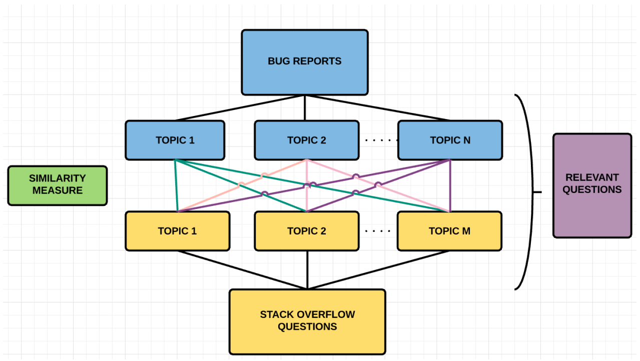


Fig.8.Automated system for finding Relevant Questions

**RQ5: What are the relevant topics corresponding to the Bug Reports?**

For this Research Question, I downloaded Bug Reports of Mozilla Firefox.
Firstly, the reports are pre-processed and fed to LDA for topic modelling. The perplexity threshold is 500 for Bug Reports. The parameter alpha is used at 0.333333333333 and eta at 0.00107066381156.
Secondly, the similarity values are measured between bug topics and question topics so as to find out the relevant question topics corresponding the bugs. The Tables of similarity values are in Appendix.
Thirdly, the relevant questions corresponding to the topics are explored.
Lastly, a comparative analysis is preformed between two similarity techniques used: Cosine Similarity and Jaccard Similarity.

## *E. COMPARATIVE ANALYSIS*

Jaccard coefficient and cosine similarity are two techniques to test the similarity between two sets of documents. In this analysis, these two techniques are compared with respect to the time complexity and relevant result according to the search query.

***Time Complexity between Jaccard Coefficient and Cosine Similarity for document similarity measure*:**

The Table 2 shows the time required by Jaccard Coefficient and Cosine Similarity for document similarity measure to retrieve the relevant result. The result for 10 topics is shown.

| **Query** | **Cosine Similarity Time(in ms)** | **Jaccard coefficient Time(in ms)** |
|---|---|---|
| Topic 1 | 8 | 10 |
| Topic 2 | 10 | 11 |
| Topic 3 | 9 | 10 |
| Topic 4 | 9 | 12 |
| Topic 5 | 8 | 9 |
| Topic 6 | 9 | 10 |
| Topic 7 | 9 | 14 |
| Topic 8 | 8 | 10 |
| Topic 9 | 10 | 13 |
| Topic 10 | 10 | 15 |

TABLE 2. Time taken fro Similarity measure

***F-measure between Jaccard Coefficient and Cosine Similarity for document similarity measure:***

The performance of identifying correct topic has been measured via three metrics: precision, recall, and F-measure. Precision is the percentage of correctly identified topics for the particular bug report over all the Question topics, while Recall is the percentage of correctly identified topics for the particular bug report over all the correctly identified topics for the Questions and unidentified topics for the Questions. Suppose the number of correctly identified topics for the particular bug report is C, the number of wrongly identified topics for the bug report is W and the number of unidentified topics for the bug report is M, then the precision of the approach is given by the expression given below

$$P = C/\ (C + W)$$

and the recall, R,

$$R = C/\ (C + M)$$

F-measure incorporates both precision and recall.
Fmeasure is given by

$$F = 2PR/\ (P + R)$$

***Relevancy of results between the Jaccard Coefficient and Cosine Similarity for document similarity measure:***

The result obtained against a search query entered is more relevant for the topics created by Cosine Similarity measure as compared to the topics generated by Jaccard Coefficient.

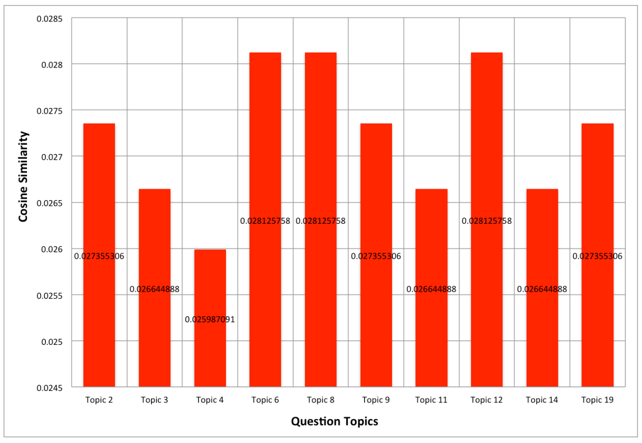


Fig.9. Cosine Similarity of Bug Report 1 with 20 Question Topics

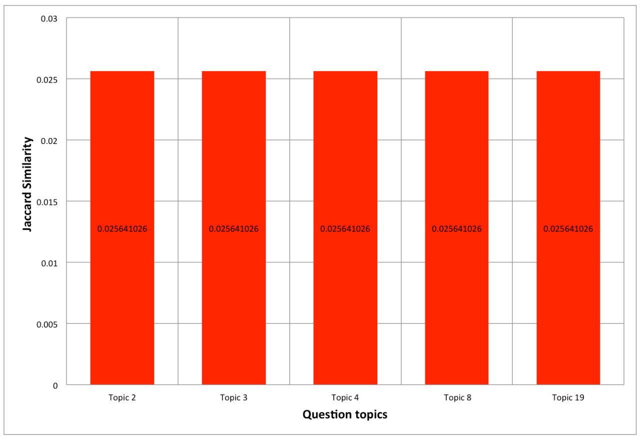


Fig.10. Jaccard Similarity of Bug Report 1 with 20 Question Topics

## F. *THREATS TO VALIDITY*

1. In this Research most of the work is automated except for topic labeling which is manual. I have assigned the topic labels based on top 20 words. Topic labeling is prone to experimenter bias. Thus, a threat to internal validity [19].
2. As the data is downloaded from the internet, it cannot be fully trusted thus increasing threat to external validity.
3. Threat to construct validity often refers to the appropriateness of the metrics used [6]. Thus, tuning of LDA might lead to different results if the parameter values are changed.
4. The data preprocessing plays a significant role in this research. If the data is not preprocessed properly, it might compromise the results.

## VI. RELATED WORK

Treude et al. is the first to analyze Stack Overflow question and answer site [7]. In their Research, they manually investigated a few hundred questions and assign them into 10 categories. They also used tags in their study and investigated how tags are used in StackOverflow. Treude et al. looked into the distribution of questions that receive zero or more answers and what kinds of questions receive more answers. Wang et al. extended the work of Treude et al. by investigating additional research questions. They looked into the distribution of questioners and answerers. They employed Latent Dirichlet Allocation, a state-of-the-art and well-known topic modeling technique, to semi-automatically infer topics from questions and assign a question to several topics with some probabilities. They used a dataset of 100,000 questions and extracted it using StackOverflow API2. In this study I have replicated their work. I have used different data and extended their work into making a recommendation system.

Blei et al. [8,25] showed the importance of topic modelling in multiple scenarios. How immensely LDA helps in topic modelling. Their contributions helped me with valuable insights for my research. My research on topic modelling is extremely crucial in real world scenarios, because to have a better way of managing the explosion of electronic document archives these days, it requires using new techniques or tools that deals with automatically organizing, searching, indexing, and browsing large collections. On the side of today‘s research of machine learning and statistics, it has developed new techniques for finding patterns of words in document collections using hierarchical probabilistic models. These models are called topic models [26]. Discovering patterns often reflect the underlying topics that united to form the documents, such as hierarchical probabilistic models are easily generalized to other kinds of data; topic models have been used to analyze things rather than words such as images, biological data, survey information and even commit messages. The four methods that topic modeling rely on are

Latent semantic analysis (LSA), Probabilistic latent semantic analysis (PLSA), Latent Dirichlet allocation (LDA) and Correlated topic model (CTM). The basic drawback of unsupervised machine learning is that labeling the topics is a tiresome prospect [26].

Mathias Ellmann looked into software documentations that have similar information included as a Stack Overflow post so as to get new inspirations on how to solve their current development problem. The linkage of same and different types of software development documentation might save time and might increase the productivity of the developer. In his study, he discussed the approach to get a broader understanding of different similarity types within and between software documentation as well as an understanding of how different software documentations can be extended [27].

A wide variety of Similarity measures are compared and their effectiveness is analyzed in Text Clustering [28,29]. Choosing an appropriate similarity measure is crucial for cluster analysis, especially for a particular type of clustering algorithms. For example, the density-based clustering algorithms, such as DBScan [30], rely heavily on the similarity computation. Density-based clustering finds clusters as dense areas in the data set, and the density of a given point is in turn estimated as the closeness of the corresponding data object to its neighboring objects [29]. Anna Huang described Cosine and Jaccard similarities. According to her study, when documents are represented as term vectors, the similarity of two documents corresponds to the correlation between the vectors. This is quantified as the cosine of the angle between vectors, that is, the so-called cosine similarity. The Jaccard coefficient, which is sometimes referred to as the Tanimoto coefficient, measures similarity as the intersection divided by the union of the objects. For text document, the Jaccard coefficient compares the sum weight of shared terms to the sum weight of terms that are present in either of the two document but are not the shared terms [29].

## VII. CONCLUSION & FUTURE WORK

The Research is influenced by Wang et al. [6]. I have tried to replicate their techniques and found some consistent results.

1. The distribution of Questioners and Answerers is consistent with the work of Wang et al. [6].
2. The proportion of posts is not similar to the results by Wang. In my study, the proportion of "only Answerers" is more than the proportion of "only Questioners"
3. Latent dirichlet allocation (LDA) was used to automatically form topics. The topics were manually labeled by me. The top 20 question topics are mentioned in the TABLE 1.
4. Comparative analysis is performed on Cosine similarity and Jaccard similary. Cosine similarity proves to be better than Jaccard similarity on the basis of time and relevant results.

In the future, this study can be extended by investigating more questions from Stack Overflow and from other information web sites. This current study only investigates the 20 high-level topics. Various number of topics and various topic modeling techniques can be used in future work. LSI can be tried and the comparison between the results of LDA and LSI can be stated. Different bug reports can be used to find the relevant topics by using different types of similarity measures. I have not used tags in this research. Thus, tags can be included to further improve this research for recommending the Stack Overflow questions related to particular bugs.

## VIII. REFERENCES

[1] Xia X, Lo D, Wang X, Zhou B. Tag recommendation in software information sites. In Proc. the 10th Working Conference on Mining Software Repositories (MSR), May 2013, pp.287-296.

[2] Begel A, DeLine R, Zimmermann T. Social media for software engineering. In Proc. the FSE/SDP Workshop on Future of Software Engineering Research, November 2010, pp.33-38.

[3] Storey M A, Treude C, Deursen A, Cheng L T. The impact of social media on software engineering practices and tools. In Proc. the FSE/SDP Workshop on Future of Software Engineering Research, November 2010, pp.359-364.

[4] Zhang, Yun, et al. "Multi-factor duplicate question detection in stack overflow." *Journal of Computer Science and Technology*30.5 (2015): 981-997.

[5] Baltadzhieva, Antoaneta, and Grzegorz Chrupała. "Predicting the quality of questions on stackoverflow." *Proceedings of the International Conference Recent Advances in Natural Language Processing*. 2015.

[6] Wang, Shaowei, David Lo, and Lingxiao Jiang. "An empirical study on developer interactions in StackOverflow." *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. ACM, 2013.

[7] C. Treude, O. Barzilay, and M.-A. D. Storey. How do programmers ask and answer questions on the web? In ICSE, pages 804–807, 2011.

[8] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research*3.Jan (2003): 993-1022.

[9] https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/

[10] http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

[11] Phillips, Jeff M. "University of Utah,"Jaccard Similarity and Shingling"

[12] W. Fan, L. Wallace, S. Rich, and Z. Zhang, "Tapping the power of text mining," Communications of the ACM, vol. 49, no. 9, pp. 76–82, 2006.

[13] Talib, Ramzan, et al. "Text Mining: Techniques, Applications and Issues." *International Journal of Advanced Computer Science & Applications* 1.7 (2016): 414-418.

[14] G. King, P. Lam, and M. Roberts, "Computer-assisted keyword and document set discovery from unstructured text," Copy at http://j.mp/1qdVqhx Download Citation BibTex Tagged XML Download Paper, vol. 456, 2014.

[15] N. Zhong, Y. Li, and S.-T. Wu, "Effective pattern discovery for text mining," IEEE transactions on knowledge and data engineering, vol. 24, no. 1, pp. 30–44, 2012

[16] Liu, Regina Y., Jesse M. Parelius, and Kesar Singh. "Multivariate analysis by data depth: descriptive statistics, graphics and inference,(with discussion and a rejoinder by Liu and Singh)." *The annals of statistics* 27.3 (1999): 783-858.

[17] Alghamdi, Rubayyi, and Khalid Alfalqi. "A Survey of Topic Modeling in Text Mining." *International Journal of Advanced Computer Science and Applications (IJACSA)* 6.1 (2015).

[18] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4597325/

[19] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, and X. Li. Comparing twitter and traditional media using topic models. In ECIR, 2011.

[20] Lim, Wayne C. "Effects of reuse on quality, productivity, and economics." *IEEE software* 11.5 (1994): 23-30.

[21] Inoue, Katsuro, et al. "Where does this code come from and where does it go ? -integrated code history tracker for open source systems." *Proceedings of the 34th International Conference on Software Engineering*. IEEE Press, 2012.

[22] Rosen, Christoffer, and Emad Shihab. "What are mobile developers asking about? a large scale study using stack overflow." *Empirical Software Engineering* 21.3 (2016): 1192-1223.

[23] Sadowski, Caitlin, Kathryn T. Stolee, and Sebastian Elbaum. "How developers search for code: a case study." *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM, 2015.

[24] Barua, Anton, Stephen W. Thomas, and Ahmed E. Hassan. "What are developers talking about? an analysis of topics and trends in stack overflow." *Empirical Software Engineering* 19.3 (2014): 619-654.

[25] Blei, David M., and John D. Lafferty. "Dynamic topic models." *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006.

[26] Alghamdi, Rubayyi, and Khalid Alfalqi. "A Survey of Topic Modeling in Text Mining." *International Journal of Advanced Computer Science and Applications (IJACSA)* 6.1 (2015).

[27] Ellmann, Mathias. "On the similarity of software development documentation." *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 2017.

[28] Gomaa, Wael H., and Aly A. Fahmy. "A survey of text similarity approaches." *International Journal of Computer Applications* 68.13 (2013).

[29] Huang, Anna. "Similarity measures for text document clustering." Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand. 2008.

[30] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of 2nd International Conference on KDD, 1996.

[31] https://stackoverflow.com