

M.Engg. Thesis

**A LARGE SCALE EMPIRICAL STUDY ON FRONT-END
SOFTWARE DEVELOPMENT TECHNOLOGY FROM
COMMUNITY QA SITES**

by

Ahmad Rahman Farabi

Student Id: 0417052108

Session: April, 2017

Supervised by

Dr. Anindya Iqbal

Professor, Dept. of CSE

Thesis to obtain the degree of

**MASTER OF ENGINEERING in
COMPUTER SCIENCE AND ENGINEERING**




Department of Computer Science and Engineering
BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
Dhaka-1000, Bangladesh

May 2023

The thesis titled "**A LARGE SCALE EMPIRICAL STUDY ON FRONT-END SOFTWARE DEVELOPMENT TECHNOLOGY FROM COMMUNITY QA SITES**" submitted by Ahmad Rahman Farabi, Student Id: 0417052108, Session: April-2017, has been accepted as satisfactory in fulfillment of the requirements for the degree of M. Engg. in CSE on May 21, 2023.

Board of Examiners

1.  _____

Dr. Anindya Iqbal

Chairman

Professor

Dept. of CSE, BUET, Dhaka-1000

(Supervisor)


2.  _____

Dr. A. B. M. Alim Al Islam

Member

Professor

Dept. of CSE, BUET, Dhaka-1000

3.  _____

Dr. Rifat Shahriyar

Member

Professor

Dept. of CSE, BUET, Dhaka-1000

CANDIDATE'S DECLARATION

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

Farabi

Ahmad Rahman Farabi

DEDICATED TO PROPHET HAZRAT MUHAMMAD (SWM)

Acknowledgements

Firstly, I would like to thank Almighty for keeping me strong, patient, and determined over the years to fulfill my destiny.

Then I would like to acknowledge my dissertation supervisors Prof. Anindya Iqbal and industry expert Mr. Sohel Aman Khan for their valuable insight, guidance, support, and sharing of knowledge that has made this possible.

I would also like to thank the Department of CSE, BUET, and the office staff of the department for their support with logistics and facilities.

I would like to thank my parents- Safiqur Rahman and Sufia Rahman, wife Fahmida Sultana and son Faraz Zafir Rahman, for their inspiration, encouragement, and support which always motivated me to keep going.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life, I thank you.

To each and every one of you - Thank you.

-Ahmad Rahman Farabi

Abstract

Front-end software development technology has been rapidly evolving for the past decade and dominating over other technologies. The major part of this technology is possessed by the JavaScript (JS) frameworks which are introducing newer technologies day by day to make the front face of software more appealing to the users. To keep up the pace, front-end developers face many challenges which lead to discussions in community Q&A sites such as Stack Overflow (SO). In this study, we analyze SO posts to give an insight into the challenges and difficulties the front-end community faces as well as their interests. This study gives a better understanding of the most discussed and trending topics in SO. To conduct the study, we consider the top 3 popular JS frameworks and develop a tag set to extract the posts from the SO dump. Then we run a topic modeling algorithm on the posts to generate the topic set, label the topics, categorize them, and create a hierarchy of the categories. After that, we analyze the popularity and difficulty of the topics based on the data. Moreover, we investigate the evolutionary trends of the topics over time. We choose our approach carefully to mitigate all the possible risks and threats that come along the way. This study can help future practitioners and researchers to come up with new techniques to overcome the challenges and difficulties at an early stage of front-end development.

Contents

Cover page	i
Board of Examiners	ii
Candidate's Declaration	iii
Dedication	iv
Acknowledgements	v
Abstract	vi
Contents	vii
List of Figures	x
List of Tables	xi
Acronyms	xii
1 INTRODUCTION	1
Overview of Problem Domain.....	2
Motivation.....	4
Scope of Work.....	5
Organization of the Document.....	6
2 RELATED WORKS	7
Preliminaries.....	7
Stack Overflow.....	7
Topic Modeling.....	9
Latent Dirichlet Allocation (LDA).....	10
Angular, React and Vue.js Frameworks.....	12

Research on Topic Modeling in Software Engineering.....	13
Research on Programming Languages and Frameworks.....	15
3 METHODOLOGY	17
Dataset Preparation.....	18
Topic Modeling.....	23
Result Analysis.....	24
4 EMPIRICAL EVALUATION	26
Experimental Setup.....	26
Data Collection and Preprocessing.....	26
Topic Modeling Implementation.....	29
Evaluation Metrics.....	31
RQ1: Front-end Topics.....	32
Motivation.....	32
Approach.....	32
Results.....	35
RQ2: Front-end Topic Hierarchy.....	38
Motivation.....	38
Approach.....	38
Results.....	39
RQ3: Front-end Topic Popularity and Difficulty.....	41
Motivation.....	41
Approach.....	41
Results.....	41
Correlation between Topic Popularity and Difficulty.....	45
RQ4: Topic Evolution.....	47
Motivation.....	47
Approach.....	47
Results.....	47

5 DISCUSSIONS	51
Implications.....	51
Threats to Validity.....	54
Internal Validity.....	54
External Validity.....	55
Construct Validity.....	56
6 CONCLUSION AND FUTURE WORKS	57
What we learn from this study.....	57
Future Works.....	58
Bibliography	60

List of Figures

1.1	Top 3 JS frameworks.....	2
2.1	A sample question (on top), answer (on bottom), and comments (on middle).....	9
3.1	An overview of the methodology of our study.....	18
	Optimal K value based on coherence score.....	30
	Front-end topics and number of their questions.....	36
	Front-end topics and number of their questions.....	40
	Most popular languages in SO.....	48
	Relative impact for top 5 topics over 6 months interval from 2016 to 2020.....	49
	Absolute impact for top 5 topics over 6 months interval from 2016 to 2020.....	50
5.1	Front-end topics popularity vs difficulty for top 10 topics.....	52

List of Tables

1.1	Total posts count in SO for top languages from 2008 to March 2023.....	3
	The Tag set that is used to identify the Angular related posts.....	19
	The Tag set that is used to identify the React related posts.....	21
	The Tag set that is used to identify the Vue.js related posts.....	21
	Threshold values of TST and TRT for tag sets.....	27
	Tag sets based on relevance and significance threshold values.....	28
	Raw score achieved by our approach.....	30
	Topic labels, categories, separated by and top 10 keywords of front-end JS topics	32
	Topics with no of posts and their percentage.....	37
	Topic popularity.....	42
	Topic Difficulty.....	44
	Topic Correlation.....	46

Acronyms

API	Application Program Interface
DE	Differential Evolution
JS	JavaScript
LDA	Latent Dirichlet Algorithm
NLP	Natural Language Processing
RQ	Research Questions
SO	Stack Overflow
TRT	Tag Relevance Threshold
TST	Tag Significance Threshold
UI	User Interface

1

INTRODUCTION

Front-end development, nowadays, basically focuses on client-side application development with the new-age JavaScript (JS) language and frameworks. Most of the JS frameworks are open source and these are being widely used by developers across the world. These frameworks are similar but provide different functionalities to the developers and end users. Most modern websites and web applications use JS frameworks in some form to make the application more interactive, faster, and user-friendly. A recent survey [1] has shown that 81.7% of the top 10 million websites use JS for some functionalities. According to the developer survey by SO [2], JS continues to make the top of the list as the most popular programming language with over 68% of respondents stating that they use it at different phases of development. Among the top 10k global websites, 68% use the top 3 JS frameworks - React, Angular, and Vue.js [3]. Their proportion of use is presented in Figure 1.1. JS is moving so rapidly and evolving so fast that bin uzayr et al. [4] called it a renaissance brought by the rising sophistication of browser-based applications and the increasing popularity of JS on the server. So developers all around the world are searching for solutions on the web. To help these developers, it is necessary to determine the popular and difficult topics. This can help developers as well as researchers to understand where to put their efforts and time.

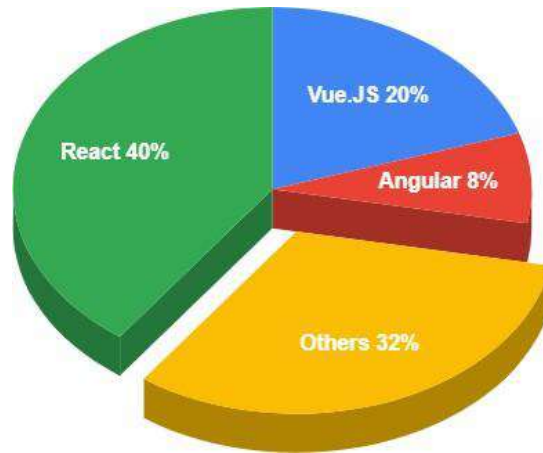


Figure 1.1: Top 3 JS frameworks

Overview of Problem Domain

Topic modeling on Stack Overflow (SO) posts is a common research domain in recent years. This domain mainly focuses on analyzing the questions and discussions available in SO. JS is a widely used programming language for web development, and the popularity of JS and its related technologies continues to grow. As a result, there is a large volume of text data available on Stack Overflow related to JS programming languages. Table 1.1 represents a tabular view of the total posts of top languages in SO. It is quite clear that JS has the topmost posts count in SO, though several languages like Python, Java, C#, etc are quite close in the race. Around 2.47M posts are available in SO from 2008 to March 2023, this huge data processing is a lot of time and power consuming [5]. Table 1.1 represents the posts count from SO for the most popular languages based on the data from 2008 to 2023.

The goal of this research is to gain a deeper understanding of the most important topics and challenges for JS developers and to identify trends, evolutions, and patterns in the discussions on Stack Overflow. This information can be used to improve the quality and relevance of technical documentation and support resources, guide the development of new software tools and platforms and benefit the design of developer support programs and initiatives.

Table 1.1: Total posts count in SO for top languages from 2008 to March 2023

Language	Total posts count
Javascript	2474745
Python	2104031
Java	1887115
C#	1580924
PHP	1455491
Android	1398480
HTML	1163889
Jquery	1032826
C++	788313
CSS	785377
React	443005
Angular	291658
Vuejs	101696

Topic modeling, specifically Latent Dirichlet Algorithm (LDA) [6], is a commonly used technique for identifying the topics covered in a large corpus of text data. However, this technique can be challenging to implement, especially when the data is large and unstructured. There is also a need to pre-process the data and remove irrelevant information, such as stop words and noise, before applying the topic modeling algorithms.

This problem domain requires interdisciplinary knowledge, drawing on techniques from computer science, data science, computational linguistics, and software engineering. It requires a deep understanding of the JS programming languages, SO data, and the limitations of topic modeling algorithms. A major challenge of this domain is to extract meaningful insights from a large and complex dataset as well as to identify the most important topics and challenges for JS developers.

Motivation

The previous empirical studies have been a major motivation for this study. Reference [7] focused on the categorization of SO data in new programming languages which has some similarities with this study. Around 10.5% of all SO posts are tagged with **javascript** which is the top tag in SO [8]. However, no empirical study has been done to analyze the posts, trends, or evolution of the front-end development life-cycle.

The motivation is to provide a more efficient and effective way for developers to access and analyze the vast amount of information available on the platform. With millions of questions and answers related to JavaScript frameworks, it can be challenging to manually go through all the contents to find the most relevant information.

This study can help developers quickly identify the most common topics or themes related to a specific framework, making it easier for them to find the information they need. This can be especially useful for developers who are new to a particular framework and need to learn the basics quickly. It can also help experienced developers stay up-to-date with the latest trends and best practices related to the framework.

Another motivation for this study is to identify patterns and trends in the questions and answers related to the framework. For example, the analysis may reveal that a particular error or issue is frequently reported by developers using the framework. This information can be used to improve the documentation, identify areas for improvement in the framework itself, or develop new tools or resources to help developers address the issue.

Moreover, there are many different JS frameworks available, each with its own strengths and weaknesses. Topic modeling can help to compare and contrast different frameworks, providing insights into the specific use cases and scenarios where each framework is most appropriate.

Overall, this study can help developers save time and improve the quality of their work by providing access to the most relevant and useful information related to the framework. It can also help developers stay up-to-date with the latest trends and best practices in the field.

Scope of Work

This study presents an analysis to understand the challenges and difficulties of front-end developers especially JS developers by explaining the following Research Questions (RQ):

- **RQ1:** What types of topics are discussed about front-end JS frameworks in SO?
- **RQ2:** Can these topics be organized into a hierarchy of topics?
- **RQ3:** What topics are the most popular and difficult and their correlation?
- **RQ4:** How do the topics evolve over time?

To answer the questions, a topic hierarchy needs to be prepared to analyze the topic categories and it is done in 3 phases. In the first phase, a tag set is developed to extract the front-end JS language related posts. The posts are preprocessed using different tools. In the second phase, a topic modeling algorithm is used to group posts into topics. Finally, a hierarchy is created by manually categorizing the topics into different categories. Besides, different statistics such as topic popularity, difficulty, and their correlation are determined and analyzed. The outcomes of this study are given below:

- **Front-end topics:** (i) Identifying the challenges faced by the front-end developers.
- **Front-end topic hierarchy:** (ii) Categorizing topics into a hierarchy of topics.
- **Front-end topic popularity:** (iii) Determining the most popular topics.
- **Front-end topic difficulty:** (iv) Determining the most difficult topics.
- **Topic popularity and difficulty correlation:** (v) Determining the correlation between topic difficulty and popularity.
- **Topic evolution:** (vi) Conduct an evolutionary study of topics over time.

For this study, we have considered the top 3 popular JS frameworks- Angular, React, and Vue.js. As the front-end development domain is quite large, we need to narrow down the domain for an accurate study. The main reason behind choosing these 3 is that they are the most popular worldwide among all other available JS frameworks. Each of them has large and active communities of developers, which means that there are many resources available for learning and troubleshooting. These frameworks also have many third-party libraries and plugins available, which can help to speed up development. Moreover, they are highly effective, simple to use, and employed by many big companies and organizations.

Organization of the Document

This thesis is organized as follows: Chapter 1 introduces our study along with the problem definitions. Chapter 2 describes the preliminaries and related works of our study. Chapter 3 represents the overall methodology of our study. Chapter 4 describes the experimental setup and results of our study. Chapter 5 indicates the implications and threats to the validity of this study. Chapter 6 concludes the study by describing what we learn from our study and some future enhancement scopes of this study.

2

RELATED WORKS

A lot of new JS frameworks are introducing in recent days and continuously growing. The demand for searching right solutions to a problem among developers is rising. Modeling these questions and solutions can be a huge help for them. Topic modeling for SO data has been a popular research topic over the last few years. A lot of empirical studies have been done on various domains. Some studies have been conducted to help the community by finding out the challenges, popular discussions, and trends. On the other hand, some research has been done to help developers by finding out bugs, defects, and vulnerabilities. This chapter describes some basic preliminaries and explains some works related to our research.

Preliminaries

Stack Overflow

Stack Overflow is one of the top community Q&A sites which has been popular among software developers all over the world for almost a decade. On this site, a person can register and ask questions about any problems they faced with some textual or visual descriptions and include tags to categorize the question. S/he can also include code snippets which are shown separately with the questions on the page. The common developer's activities in SO are given

below:

- **Asking questions:** Developers often use SO to ask technical questions related to their work. This can include questions about programming languages, frameworks, tools, or best practices.
- **Answering questions:** Responding to questions is a core activity on SO. Experienced developers often help answer questions and provide solutions to technical problems.
- **Commenting:** Developers can leave comments on posts to clarify or expand on the information provided in the question or answer.
- **Voting:** SO allows users to upvote or downvote questions and answers based on their usefulness or accuracy. Developers often vote on posts to indicate the quality of the content or to help other users find helpful information.
- **Editing:** SO allows users to edit posts to improve the quality and clarity of the content. Developers can suggest edits to questions and answers to make them more readable or to fix errors.
- **Tagging:** Developers can assign tags to their questions to help categorize the content and make it easier to find relevant information. This is also helpful for other developers who are searching for information on a specific topic.
- **Browsing and searching:** Developers can browse existing questions and answers on Stack Overflow to learn about specific topics or to find solutions to technical problems. They can also use the search function to look for specific keywords or phrases.
- **Contributing to documentation:** SO has a feature called Documentation that allows developers to contribute to community-written documentation on various programming topics.

On the site, statistics, charts, and graphs can be found on the overall data. Stack overflow is open source and all the posts with their data can be downloaded publicly if needed. A snapshot of a stack overflow page has been attached as a reference in Figure 2.1.

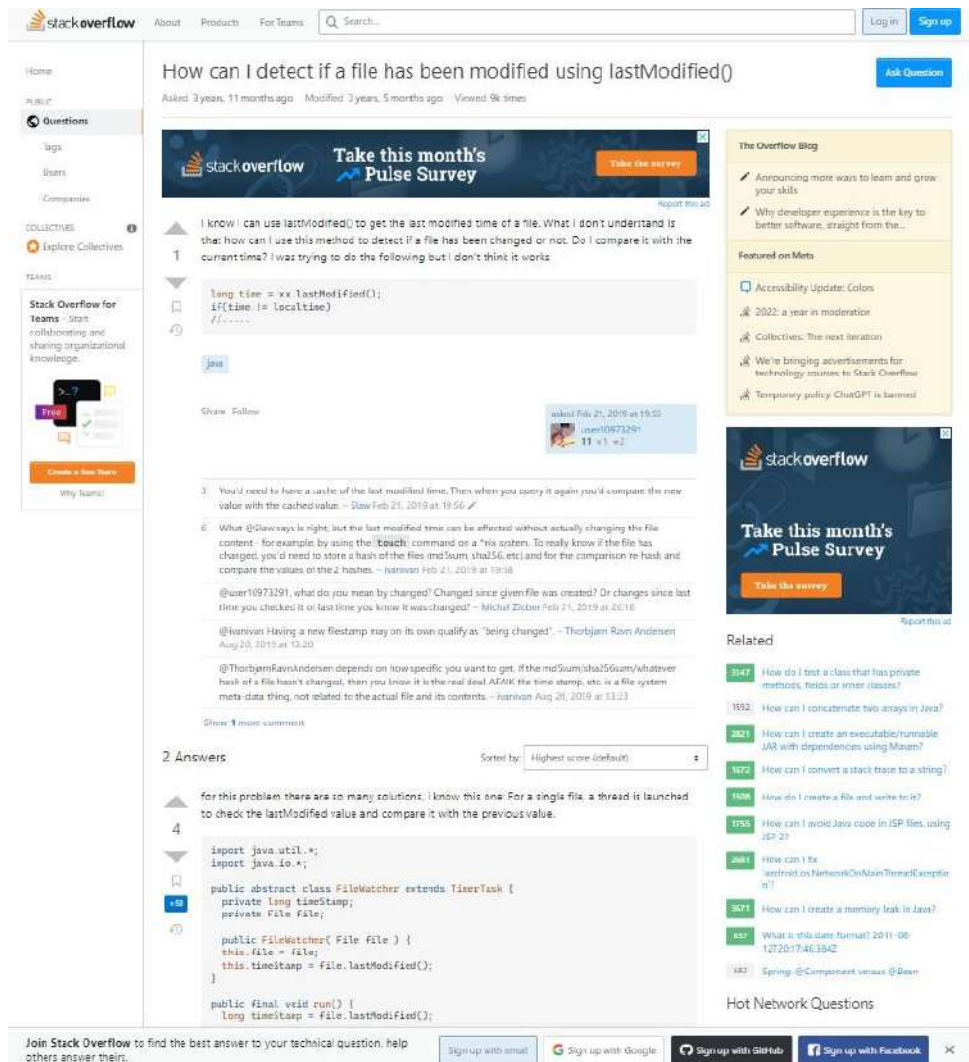


Figure 2.1: A sample question (on top), answer (on bottom), and comments (on middle)

Topic Modeling

Topic modeling is a statistical and machine learning technique used to identify hidden or latent topics in a large corpus of text data. It involves analyzing the patterns of words and phrases in a collection of documents and grouping them together into topics or themes that represent the

underlying concepts or ideas being discussed in the text.

The goal of topic modeling is to automatically identify the main topics or themes that are present in a large corpus of text data, without the need for human annotation or labeling. This can be useful for a wide range of applications, including text classification, information retrieval, content recommendation, and more.

The most common approach to topic modeling is to use a probabilistic model such as Latent Dirichlet Allocation (LDA) or Non-negative Matrix Factorization (NMF). These models work by representing each document as a mixture of topics, where each topic is a distribution over words or phrases. The model then infers the underlying topic structure by estimating the topic distributions that best explain the observed words in the corpus.

Once the topics have been identified, they can be interpreted by humans to gain insights into the content of the text data. For example, in the context of news articles, topics might represent themes such as politics, business, or sports. In the context of social media, topics might represent conversations around specific events or hashtags.

Overall, topic modeling is a powerful technique for identifying the main themes and concepts in a large corpus of text data. By automatically identifying these topics, it can help humans make sense of complex, unstructured data and gain insights into the underlying patterns and trends in the text.

Latent Dirichlet Allocation (LDA)

The most common approach to topic modeling is LDA [6]. There are other approaches like Hierarchical LDA [9], Discriminative Topic Modeling [10], Locally-consistent Topic Modeling [11] etc. In this study, LDA has been used as it is the most popular approach and has been used effectively in previous studies [7, 12-16].

The basic idea behind LDA is that each document in the corpus is a mixture of different topics, and each topic is a probability distribution over the words in the corpus. LDA assumes that there are a fixed number of topics in the corpus, and each topic is characterized by a

distribution of words. The goal of LDA is to estimate the topic distribution for each document, as well as the word distribution for each topic. LDA works by first randomly assigning each word in the corpus to a topic and then iteratively refining the topic assignments to maximize the likelihood of the observed data. The process involves calculating the conditional probability of each word given the current topic assignments and then using these probabilities to update the topic assignments. This process is repeated until a convergence criterion is met. Once the model is trained, LDA can be used to infer the topic distribution for new documents or to identify the most probable topics for a given set of words.

There are two types of popular LDA available on the web. One is Gensim LDA and other is Mallet LDA. We have used Mallet LDA in our study.

MALLET (MAchine Learning for LanguagE Toolkit) [17] is a Java-based open-source package for Natural Language Processing (NLP) and machine learning. MALLET's LDA algorithm is similar to other implementations of LDA, but it has some unique features that make it useful for topic modeling on large text corpora. One key feature is its ability to efficiently process very large datasets, thanks to its implementation of multi-threading and distributed computing. Another useful feature of Mallet's LDA is its ability to handle multiple text processing options, such as stemming, stop word removal and n-gram tokenization. This allows users to customize the text preprocessing steps to optimize their topic modeling results. Mallet's LDA implementation also includes a variety of options for specifying the number of topics, the alpha and beta hyperparameters, and the number of iterations, allowing users to fine-tune the model to their specific needs.

Alpha and beta hyperparameters basically control the distribution of topics and words. Alpha determines the topic distribution for each document in the corpus. A higher value of alpha will result in a more uniform distribution of topics across all documents, while a lower value will result in more distinct topic distributions for each document. On the other hand, beta controls the distribution of words in each topic. A higher value of beta will result in a more uniform distribution of words across all topics, while a lower value will result in more distinct word distributions for

each topic. The number of topic identification is another critical hyperparameter. LDA can not determine it automatically. Identifying optimal value is a challenge.

Even if LDA has a lot of good abilities, it suffers from "order effects" [18]. It means that different output is generated every time the order of the data is changed or shuffled. This causes instability in the output of LDA and produces inaccurate results. In our study, we have used a differential evolution approach by tuning the hyperparameters of LDA to overcome this problem and enhance stability in the output.

Angular, React and Vue.js Frameworks

Angular, React and Vue.js are the 3 most popular JS frameworks, used to build front-end web applications.

Angular framework is launched in 2016 and developed by Google [19]. It is a comprehensive framework that includes a lot of built-in features and tools for building complex, large-scale applications. Angular uses a template-based approach to define the UI, which allows developers to easily create reusable components. It uses TypeScript, a statically-typed version of JavaScript, which provides type-checking and other features to improve code quality and maintainability. It has a strong focus on modularity and uses a hierarchical structure of components and services to organize code.

React is a lightweight library that focuses on the view layer of an application and is often used in conjunction with other libraries or frameworks to build complete applications. React framework is launched in 2013 and developed by Meta [20]. It uses a component-based approach to define the UI, which allows developers to create reusable UI components that can be combined to build complex interfaces. React uses a virtual DOM, which improves performance by minimizing the number of updates required to the actual DOM. React can be used with JSX, a syntax extension that allows developers to write HTML-like code in their JavaScript files.

Vue.js is launched in 2014 and developed by Evan You, a developer from Google [21]. It is a progressive framework that can be used for both small and large-scale applications. It

uses a component-based approach to define the UI, similar to React. It allows developers to use templates, JSX, or plain JavaScript to define their components. Vue.js has a simple and intuitive API, which makes it easy to learn and use.

There are some similarities as well as differences between the 3 frameworks. Angular has a challenging learning curve due to its complexity and features, whereas React and Vue.js are relatively easier to learn. React and Vue.js has better performance because of their virtual DOM which makes the rendering process faster. Angular is known for its scalability because it has a built-in architecture for large-scale applications. React and Vue.js is also scalable but for designing large-scale application, more planning and effort is required than angular. Moreover, all of these 3 frameworks have large active community support and so they have a growing number of open-source libraries and plugins, which makes development easier.

Research on Topic Modeling in Software Engineering

In recent years, there has been a growing interest in topic modeling for analyzing large-scale text data. In the field of software engineering, topic modeling has been used to analyze source code and understand software development practices. In particular, several studies have used LDA for topic modeling in Stack Overflow posts to understand developers' challenges and improve the quality of programming-related discussions.

Recent research has shown the potential of topic modeling to enhance performance in software development. Abdellatif et al [12] focused on chatbot development challenges faced by the developers based on SO data. They collected chatbot-related posts and preprocessed the data to remove irrelevant information. They have yielded a dataset containing 3,890 posts to work with. Then they used LDA for topic modeling and labeled the posts to form topic categories and a hierarchy of topics. They have identified 12 topics grouped into 5 categories. They also identified the topic popularity and difficulty measurements and the correlation between them. They have also done some evolutionary studies on chatbot-related trends over last few years.

The study found that the most common challenges in chatbot development include natural language processing, dialog management, and integration with external systems.

In the paper regarding big data [13], the authors conducted a large-scale study on big data related posts to identify the challenges and trends. They have represented popularity, difficulty and their correlation. In this study, 28 topics have been identified and grouped into 9 categories. In the study on microservices [14], researchers extracted 20 topics from related posts and represented similar measurements like others. Rosen and Shihab [15] conducted a study on mobile development topics and analyzed the popularity and difficulty of the posts. They have identified 40 topics by categorizing the posts. Ahmed et al. [16] used LDA topic modeling to find out the topics related to concurrency development. They have identified 27 topics and investigated the challenges, difficulties, and correlations. A similar methodology has been followed to produce the results in all of the above studies but only the domain of the topics is different.

Besides these, topic modeling is used in various research for different purposes. Li et al [22] proposed a technique to extract logging information from the codebase for understanding system usage in the production environment and debugging system failures. They have used LDA for topic modeling and showed proof of their concept in the paper. Topic modeling is also used for detecting defects or buggy code. Nguyen et al. introduced a tool named "BugScout" for detecting buggy codes using topic modeling on bug reports [23]. Another research [24] detects defects from code by modeling source code. Andrzejewski et al [25] uses an optimized version of LDA "Delta LDA" on program execution traces for identifying two topics- normal usage topics and bug topics.

Stack Overflow posts are now being used in many studies to identify topics in various aspects of software engineering. The main reason is to understand the difficulties and extract the most popular discussions. Previous studies [12-16] have been a major motivation for us to find out the challenges and popular trends of frontend JS frameworks.

Research on Programming Languages and Frameworks

Chakraborty et al [7] focus on 3 new languages (Go, Swift, Rust) and identified challenges based on the SO posts. They use topic modeling for each language separately and have also conduct some comparative studies. To simulate developers' activities and evolution over time, they also extract GitHub issues. They use Differential Evolution (DE) to solve the common issues with LDA. The methodology is similar to our study. The study uses a mixed-methods approach, combining quantitative analysis of Stack Overflow data with qualitative analysis of forum posts and developer comments. The authors find that the adoption of new programming languages on Stack Overflow follows a similar pattern across languages, with a rapid increase in questions and answers in the first year followed by a slower but steady growth over time.

They also identify key topics related to each language, such as performance and concurrency for Go, syntax and interoperability for Swift, and memory management and safety for Rust. The authors note that the communities around each language have distinct characteristics, with the Go community being more focused on practical applications, the Swift community emphasizing collaboration and sharing, and the Rust community valuing correctness and safety. In addition, the authors find that SO plays an important role in the adoption and growth of new programming languages. They found evidence that developers use Stack Overflow to learn about new languages, to ask and answer questions about them, and to share knowledge and expertise.

Overall, the paper provides insights into how developers use Stack Overflow to discuss and support new programming languages, and how the characteristics of different languages affect the nature of the discussion.

Another study [26] uses both SO and GitHub textual data to predict optimized configurations for LDA. They work on 3 main goals- (i) to set proper guidelines on how the LDA hyperparameters should be set, (ii) to analyze the characteristics of text corpus related to 8 programming languages- C, C++, CSS, HTML, Java, JavaScript, Python and Ruby, (iii) to analyze corpus

feature importance via per-corpus LDA configuration. They explore diverse text corpus for various languages from different sources and introduced an approach to automatically configure LDA parameters. Their per-corpus LDA configuration approach is a unique feature of this study which can be very helpful for other studies.

To predict good configurations of topic models, the authors propose a novel approach that combines feature selection and machine learning. They use a set of pre-defined features such as the number of topics, the number of words per topic, and the coherence of the topics. They then use a machine learning algorithm to predict the best configuration for each platform. The authors evaluate their approach on both platforms and compare the predicted configurations with the ground truth configurations. They find that their approach is able to accurately predict good configurations for both platforms. The paper has important implications for software developers and researchers who use topic modeling to analyze software development platforms. The proposed approach can help them identify the most important topics discussed on these platforms and optimize the configuration of their topic models for better results.

In summary, these works demonstrate the potential of topic modeling for analyzing and extracting insights from Stack Overflow data, and highlight the importance of selecting the appropriate topic modeling techniques for different types of analysis. These studies inspired us to conduct our study to help the developers, practitioners, and researchers community worldwide.

3

METHODOLOGY

As our study focuses on front-end software development technology and JS frameworks are on top of this technology, we consider only the JS frameworks for our study. JS is a very robust technology for developing the front-end part of any software. Developers all around the world are asking JS questions constantly. Approximately, 2.4M questions are posted using “javascript” tag [8]. So, we need to be more specific to build our dataset. For this reason, we are considering Angular, React, and Vue.js frameworks.

We consider only the posts with one or more answers as the posts with no answers are useless for our study. Moreover, comments are not considered as they are not always informative in response to the questions.

Initially, only posts with the “javascript” tag are considered as the initial dataset of our study. In SO, a user can add multiple tags to a question to describe it better. As a result, a lot of other tags are found in posts with the initial tag but we only need our 3 chosen frameworks related tags. So, from the initial tag set, we have extracted 3 different tag sets using the tags- Angular, React, and Vue.js. In this way, only the relevant posts to our study were extracted from the huge number of posts in SO.

To achieve the main goal of our study, we need community Q&A discussions and SO provides the richest dataset among all Q&A sites online. We follow a methodology similar to

other empirical studies on different technologies, such as chatbot [12], big data [13], microservices [14], study on new programming languages [7], mobile development [15] and concurrency development [16]. The methodology of our study consists of 7 steps in 3 sections. Figure 3.1 shows all the steps sequentially. The following are the steps described along with the sections.

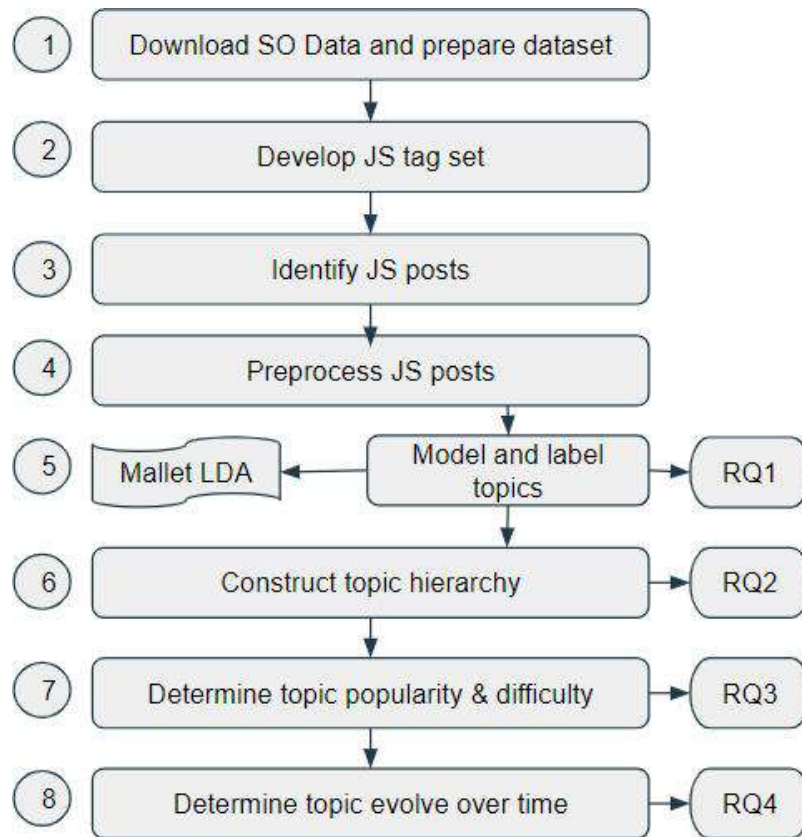


Figure 3.1: An overview of the methodology of our study

Dataset Preparation

The data collection and preparation section contains the steps to download data, develop tag sets, identify related posts, and preprocess the posts. This section is the prerequisite of the whole study.

Step 1: Download SO data and prepare the dataset. SO data dump can be downloaded from SO torrent [27]. The dataset consists of questions and answers along with some metadata (e.g., title, body, tags, view and favorite count, accepted answers identifier, creation date, etc.). The metadata needs to be collected based on the requirements. As the dataset is unstructured text data, this may need to be inserted into a relational database to make it structured. Structured data can be easily accessible and we can extract data by using a query as per the requirements.

Step 2: Develop JS tag set. In this step, the initial tag/tags need to be identified and all other tags that coexist with the initial tags should also be considered. Then two heuristic metrics are calculated that have been used in prior works [7, 12-14]. The first one is Tag Relevance Threshold (TRT) which indicates how relevant a specific tag is to the other JS related tags. TRT is calculated using eq. (3.1).

$$TRT_{tag} = \frac{\text{No of JS posts for the tag}}{\text{Total no of posts for the tag}} \quad (3.1)$$

Table 3.1: The Tag set that is used to identify the Angular related posts

Tag name	TRT	TST
typescript	0.4095	0.2248
angular-material	0.8169	0.0485
rxjs	0.6487	0.0408
angular6	0.9796	0.0302
angular-cli	0.9667	0.0249
angular5	0.9667	0.0246
ionic2	0.4587	0.0216

- continued on next page

Table 3.1 - continued from previous page

Tag name	TRT	TST
angular7	0.9824	0.0204
observable	0.6168	0.0195
angular8	0.9852	0.0187
angular2-routing	0.9679	0.0177
ionic3	0.4028	0.0163
angular-reactive-forms	0.9915	0.0131
primeng	0.9022	0.0119
ngrx	0.854	0.0115
angular-material2	0.9851	0.011
karma-jasmine	0.4858	0.0108
angular2-template	0.9617	0.0106
ionic4	0.455	0.0101

For example, Table 3.1 presents that “typescript” is a tag from the angular tag set with a TRT value of 41%. It means that 41% of the posts tagged with “typescript” are also tagged with the “angular” tag. Thus, by only considering higher TRT values, we are able to remove the posts with irrelevant tags. There can be exceptions when a tag with high TRT values has a small number of posts and this may include irrelevant posts. Hence, we use another metric Tag Significance Threshold (TST) which indicates how significant a specific tag is. TST is calculated using eq. (3.2).

$$TST_{tag} = \frac{\text{No of JS posts for the tag}}{\text{Total no of posts for the most popular tag}} \quad (3.2)$$

From Table 3.1, “typescript” from the angular tag set has TST value of 22.4% which means posts tagged with both “typescript” and “angular” are equal to 22.4% of the total posts tagged with “angular”. Table 3.2 and Table 3.3 presents the TRT and TST values of React and Vue.js.

Table 3.2: The Tag set that is used to identify the React related posts

Tag name	TRT	TST
redux	0.7636	0.0782
react-redux	0.8007	0.0518
react-router	0.9131	0.0498
react-hooks	0.9206	0.0393
material-ui	0.8555	0.0349
jestjs	0.4218	0.0222
axios	0.4025	0.0211
jsx	0.8374	0.0173
next.js	0.6053	0.0149
enzyme	0.8664	0.0124
create-react-app	0.7408	0.0114
react-router-dom	0.9395	0.011
state	0.4521	0.0107

Table 3.3: The Tag set that is used to identify the Vue.js related posts

Tag name	TRT	TST
vuejs2	0.9999	0.9561
vue.js	0.2154	0.7104
vue-component	0.598	0.2222
vuex	0.3001	0.0821

- continued on next page

Table 3.3 - continued from previous page

Tag name	TRT	TST
vue-router	0.3567	0.0713
vuetify.js	0.1953	0.0543
vuejs3	0.999	0.046
nuxt.js	0.1138	0.0335
axios	0.0455	0.0297
bootstrap-vue	0.2238	0.0115
vue-cli	0.1847	0.0107

This approach has been taken by some of the previous works [12-14]. The main goal of this step is to filter out all the irrelevant and insignificant posts as well as reduce the noise from the data.

Step 3: Identify JS posts. Now we have the tag sets to extract all the related posts. We extract the data by querying all the posts that are tagged with one of the selected tags from the tag sets. The size of the final data set can be more or less based on the size of the tag sets. The data will include some metadata along with the post questions and answers. The metadata includes title, body, view count, favorite count, accepted answer count, creation date, scores, hours to get accepted answer etc. We need the metadata for later analysis of our study.

Step 4: Preprocess JS posts. In this step, we preprocess the data before proceeding to the next steps. Preprocessing is needed to filter out all the noise and irrelevant information. We only focus on the titles of the posts but not the body contents as it can add noise to our data. This approach has been used in prior works [12, 15]. After extracting the titles, we apply different techniques to remove noise from the data. Then we build a bigram and trigram model as there are multiple words that generally appear together such as ‘auto complete’ and ‘node

module'. The bigram model analyzes the word pairs whereas the trigram model analyzes word triplets. So topic modeling must consider the bigrams and trigrams.

Topic Modeling

This section describes the process for modeling and labeling topics as well as building the hierarchy based on the output of previous steps.

Step 5: Model and label topic. Now, we proceed to identify the JS topics and label them. We use the LDA modeling algorithm [6] to model our data. All the prior works have used this technique to model data [7, 12-16]. LDA groups the SO posts into a set of topics based on their probabilistic distribution. This probability is calculated based on the word frequencies and co-occurrences of words in the posts. If a word is present in a post multiple times, then the probability of that word is high and it can be the topic of the post. LDA assigns a series of topic keywords with their probabilities to each post. The most dominant topic is the topic with the highest probability for a particular post. We use Mallet implementation [17] of LDA. Mallet LDA groups posts into a defined number of topics after completing a certain number of grouping iterations. It returns a set of topic keywords and their probability for each post.

Step 6: Construct topic hierarchy. To create a topic hierarchy, we use the open card sort technique [28]. We group similar topics into lower and higher level categories. The standard levels of categories are: higher, middle, and lower level categories. The hierarchy is constructed based on the opinions of the author and an industry expert (introduced in the Acknowledgements) who have prior experience in this type of work. Categorizing topics is a very critical part of this study. For this reason, this must be conducted based on the constructive argument of the responsible persons. This step answers research question **RQ2**.

Result Analysis

This section contains 3 steps for analyzing the extracted data. We produce topic popularity, difficulty, correlation, and evolutionary statistics.

Step 7: Determine topic popularity and difficulty. We follow the standard approach [12-14] to determine the topic popularity and difficulty.

To find out the popularity, we use 3 known metrics. The first metric is the average view count of the posts for a topic. The view count includes both the registered and guest users. The second one is the average favorite count. A user can mark a post as a favorite. The final metric is the average score of posts on a topic. The score is calculated by SO based on views, responses, and other criteria. A topic is popular if the posts have higher average views, favorites, and scores.

For determining difficulty, we use 2 common metrics. One is the percentage of posts without an accepted answer for a topic. If a post has less number of accepted answers or no accepted answer at all, then we can say that the difficulty level of the post is high. The other metric we consider is the median of hours to get an accepted answer. If the time is high for a topic, then we can say that the topic is difficult. Since the median is more resistant to outliers and skewness than the mean, it is used to measure difficulty. When the data contains outliers or is skewed, the median measures central tendency better since it is less impacted by extreme values than the mean. This step responds to **RQ3**.

Step 8: Determine the correlation between popularity and difficulty. We determine the correlation between topic popularity and difficulty. We use **Kendall correlation** [29] with a confidence interval of 95%. The reason behind choosing this measure is that it is less sensitive to outliers and that makes it more stable to produce robust results. This analysis describes any significant correlation between popularity and difficulty. This method is well known to measure

correlation in similar studies [13]. This step also responds to the last part of **RQ3**.

Step 9: Determine topic evolution over time. Finally, we determine the evolution of topics over time based on two metrics. One is the topic absolute impact and the other is relative impact. These metrics are popular and used in some of the previous studies [7, 30]. These two metrics are determined to understand the need for specific information by a developer or companies that want to understand the trends in the software development industry. Moreover, we have generated some statistical data from existing SO trends for top JS frameworks. This step answers **RQ4**.

In summary, following above mentioned steps lead to our final result of the study. These steps must be executed maintaining the exact order. It is important to prepare the experimental setup based on the above steps. The steps here are designed to explore all the research questions (RQ1-RQ4) of our study. We need to focus on the best available approach to implement the steps.

4

EMPIRICAL EVALUATION

In This section, we describe the experimental setup of the study that includes the analysis to find out the best possible approach for our study. We then produce the results and present our analysis of the results to answer the research questions **RQ1** to **RQ4**. We present graphical and textual representations of the data we generate throughout the study. We also analyze our findings to relate our study with prior works and investigate all the threats.

Experimental Setup

For the experimental setup, we used a computer with a core i3 processor, 16GB of RAM, 256GB of SSD, and a stable internet connection. For data preparation, we used the MSSQL database. For data processing, we used Python language in IDLE/PyCharm IDE. For the code repository, we have used GitHub and Kaggle. For generating visual statistics, Microsoft Office and some other tools are used.

Data Collection and Preprocessing

The initial dataset size was 88GB XML and contains approximately 20.864M posts. Before 2015, JS languages were still in their preliminary phase and there is much noise in those data.

So we only consider data for the period between January 2015 and February 2021. The posts are inserted into an MSSQL database to run SQL queries over the data. This helps to narrow down the data based on our desired time frame. We extract a total number of 1,460,841 posts related to our initial tags (javascript, angular, react, vuejs). Among these posts, 659,835 (45%) are questions and 801,006 (55%) are answers.

we develop 3 initial tag sets for 3 JS frameworks using the initial tags we consider. For the angular tag set, we consider all tags that coexist with “angular” tags. Similarly, we also extract tag sets for “react” and “vuejs”. A tag must be relevant and significant if the TST and TRT value of that tag is above certain threshold values. To select optimal threshold values for the 3 frameworks, we have inspected and analyzed the values from Tables 3.1 to 3.3. To verify the significance and relevance manually and select an optimal threshold individually, another software engineering professional (mentioned in the acknowledgment) was consulted. Then, by discussion and constructive argument, we reach an optimal threshold value for each tag set that covers the most relevant and significant posts. Then we calculate two heuristic metrics following the standard approach [12-15]. Table 4.1 shows the chosen threshold values.

Table 4.1: Threshold values of TST and TRT for tag sets

	Angular	React	Vuejs
TST	0.01	0.01	0.01
TRT	0.4	0.4	0.04
Total Tags	20	13	11

We consider optimal threshold values for all 3 tag sets. Now, we use these threshold values to extract our desired tag sets for the 3 JS frameworks. Table 4.2 represents the selected tag sets based on TST and TRT threshold values. We have a total number of 44 tags from 3 tag sets.

Table 4.2: Tag sets based on relevance and significance threshold values

Angular tag set	React tag set	Vuejs tag set
typescript	redux	vuejs2
angular-material	react-redux	vue.js
rxjs	react-router	vue-component
angular6	react-hooks	vuex
angular-cli	material-ui	vue-router
angular5	jestjs	vuetify.js
ionic2	axios	vuejs3
angular7	jsx	nuxt.js
observable	next.js	axios
angular8	enzyme	bootstrap-vue
angular2-routing	create-react-app	vue-cli
ionic3	react-router-dom	
angular-reactive-forms	state	
primeng		
ngrx		
angular-material2		
karma-jasmine		
angular2-template		
ionic4		

From the selected tag sets, we have extracted the final data set consisting of total 1,25,659 posts (Angular 70,519; React 42,283 and Vue.js 12,857) and their corresponding metadata. For the preprocessing, We use Python NLTK library [31] to preprocess the titles. We remove HTML tags, URLs, special characters, stop words, code snippets etc. We use stemming to keep the word in its original form. For example, ‘building’, ‘builds’, and ‘build’ all come to ‘build’.

After that, we build the bigram and trigram models to consider grouped words. We also ignore some biased words which commonly come to the posts' titles but have no significant meaning to the posts' objectives. For example, 'angular', 'react', and 'vuejs' -are the tags of the posts and should not be included in the titles. Furthermore, 'add', 'show', 'load', 'find' etc are the common words that can appear in any title. So, removing these words will result in more accuracy.

Topic Modeling Implementation

Now we need to implement the major part of this study. All our experimental data is dependent on this section. This section can be divided into multiple sub-sections as below:

Finding the optimal number of topics: The main challenge is to find the optimal number of topics K . K value cannot be too high or too small. The high K value can cause redundancy of topics whereas the small K value can allow multiple different topic groups into a single topic. To solve this problem, we use a broad range of values for K and l as below:

$$K = \{4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48\}$$

$$l = \{500, 1000, 1500, 2000\}$$

We calculate the coherence score of the topics which measures the understandability of the topics. The author and the software professional run the LDA with different K values and store the coherence scores. We find that K values 30 to 40 have very similar scores and $K=32$ has the peak score. We manually examine 15 posts for K values between 30 to 40 and find that $K=32$ provides the optimal set of topics according to Figure 4.1.

Handling topic instability problem: LDA has problems with hyperparameters tuning and sample order. Every time the hyperparameters are tuned or the ordering of the sample data is changed, a different output is generated. This issue is called the order effect and it introduces instability in the development of topics. To overcome the issue, we use the **Differential Evo-**

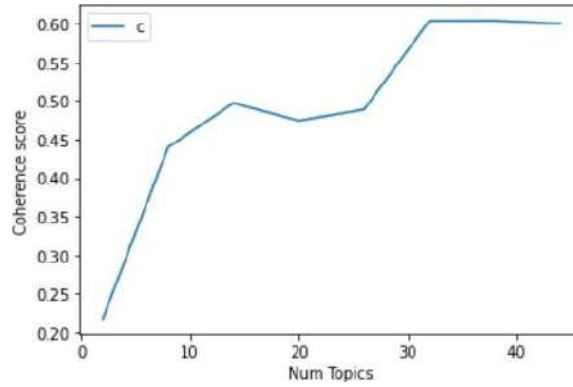


Figure 4.1: Optimal K value based on coherence score

lution (DE) approach [18]. For each different set of hyperparameters, we use 10 sample input sets after changing their order arbitrarily. After running LDA, 10 sets of topics are generated. We use **Jaccard Similarity Index** [32] on the output sets to score the parameter sets. It is a measure to determine the similarities between the topic sets. The set of parameters with the highest score is passed to the next generations for 10 more shuffled inputs. After running 3 generations like this, we have a more stable set of hyperparameters that generate a stable set of topics.

Table 4.3: Raw score achieved by our approach

Run	Raw Score
Gen 1	0.62
Gen 2	0.61
Gen 3	0.58

Table 4.3 represents the raw scores of the 3 generations. We can see Gen 1 has the highest score here. The parameter set for this generation is:

{no of topics: 32, alpha: 40, iterations: 700, optimize interval: 50}

By using this parameter set, we can reduce the instability problem of LDE, and ordering change will not have any serious effect on output. This optimization is very useful but it draws

a lot more processing times than usual. It takes almost double the time of the usual process. However, the outputs are much more stable and easily understandable using this optimization.

Topic Labeling: To label the topics, we use the open card sort process [28]. In this process, topics are manually labeled by analyzing the top 10 words from a topic set and reading through at least 15 random posts to come up with a label that describes the topic best. During the process, the author and a software industry professional individually assign labels to the topic first and then agree on a label through constructive discussions. After that, the supervisor reviews the topic labels and provides constructive feedback. Based on that, a final label is developed for each topic. The author and the industry professional are software engineers with expertise in programming languages and systems and the supervisor is a professor with relevant research experience (introduced in the Acknowledgements).

Evaluation Metrics

We construct a 3 level hierarchy based on the topic categories. The third level is the lower level and the first level is the higher level. For example, lower level category *Material UI* and *Datatable* are grouped into a mid-level category *UI Component* which is under a higher-level category *User Interface*. Table 4.4 and Figure 4.3 shows the hierarchy textually and graphically.

Topic popularity and difficulty are determined based on the data. By analyzing these with topic hierarchy, we can determine which topic categories are beginner level and which are advanced level. Table 4.6 and Table 4.7 represents the popularity and difficulty of topics. A correlation between them (Table 4.8) is also considered as an evaluation metric for determining inter-relation between popular and difficult topics.

Topic evolution over time is another important factor for this study. We analyze the topics with their creation date to figure out their trend and evolution of them over the period of time. We determine absolute impact and relative impact metrics which represents topic evolution over time. These metrics are well known and commonly used in this type of empirical studies

[12-16].

RQ1: Front-end Topics

Motivation

To answer the first research question, we need to know about the topics discussed in SO. It is the initial step of our evaluation. The front-end development domain has some basic differences from other domains. For example, this domain includes user interface design, event handling, state management, and routing, which are often unnecessary for other domains.

Approach

We have 32 topics generated from topic modeling of 1,25,659 sample posts. The topics are then labeled and organized. The topic keyword sets are briefly presented by the topic labels. Table 4.4 represents the topic labels and corresponding keywords. The topic modeling and labeling approach has been discussed in Section 4.1.2.

Table 4.4: Topic labels, categories, separated by and top 10 keywords of front-end JS topics

No	Topic Labels	Categories	Keywords
0	Material design	User Interface	material, content, dialog, container, chart, md, width, height, screen, design
1	External library management	Development \ Asset management	module, webpack, find, resolve, import, node, config, bundle, node_module, configure
2	String manipulation	Basic concepts	string, input, number, convert, format, time, date, expect, datepicker, invalid

- continued on next page

Table 4.4 - continued from previous page

No	Topic Labels	Categories	Keywords
3	Uses of variables	Basic concepts	variable, promise, define, template, provider, typescript, global, instance, throw, reference
4	Event trigger	User actions\ Events handling	element, event, trigger, directive, specific, jsx, custom, fire, attribute, onclick
5	Http request handling	Integration\ API integration	request, axios, post, http, api, axios, token, header, httpclient, laravel
6	Unit testing	Unit testing	test, jest, testing, unit, enzyme, mock, jasmine, karma, fail, library
7	Error handling	Debugging\ Troubleshooting	property, undefined, typeerror, undefine, bind, return, compute, length, push, foreach
8	Form inputs	User actions\ Inputs	form, input, field, validation, control, reactive, submit, reactive.form, validator, formgroup
9	State data handling	State management	state, change, redux, update, set, vuex, initial, detect, previous, part
10	Dynamic UI component	User Interface\ UI Component	component, dynamically, create, style, single, generate, functional, wrap, exist, svg
11	Service injection	Development\ Framework	service, datum, user, data, subscribe, implement, inject, constructor, root, share
12	Ionic development	Development\ Cross platform development	ionic, native, ion, plugin, push, storage, work, mobile, io, page
13	React hook	State management	hook, loading, call, useeffect, usestate, dependency, lazy, inside, custom, animation
14	Angular condition checking	Development\ Frameworks	ngif, ngmodel, check, checkbox, work, dynamic, working, deploy, search, simple

- continued on next page

Table 4.4 - continued from previous page

No	Topic Labels	Categories	Keywords
15	Serverside rendering	Serverside rendering	server, ngrx, side, rendering, return, client, express, selector, conditional, effect
16	React redux	State management	redux, store, action, dispatch, reducer, connect, saga, thunk, persist, middleware image,
17	Image file operation	Development\ Asset management	file, library, index, upload, external, window, include, static, folder
18	JSON data manipulation	Integration\ API integration	json, nest, bootstrap, template, object, modal, error, ngx, parse, position
19	Exchange data between components	User Interface\ UI Component	component, child, prop, parent, pass, core, slot, asp_net, data, mvc
20	CLI build tools	DevOps\ Build	cli, build, ng, fail, npm, production, serve, run, script, package
21	Material UI	User Interface	material, ui, color, icon, theme, grid, textfield, stepper, override, font
22	Routing	Navigation\ Routing	route, url, parameter, param, router, query, match, argument, path, assignable
23	Selectable inputs	User actions\ Inputs	text, option, base, select, remove, dropdown, autocomplete, condition, box, label
24	Observable events	User actions\ Events handling	observable, rxjs, async, pipe, result, subscription, operator, complete, combine, chain
25	Router links and redirects	Navigation\ Routing	router, route, link, redirect, dom, navigate, outlet, history, location, navigation
26	Contextual view rendering	User Interface\ UI Component	component, render, view, load, context, good, single, conditionally, execute, function

- continued on next page

Table 4.4 - continued from previous page

No	Topic Labels	Categories	Keywords
27	Datatable	User Interface	table, mat, primeng, filter, column, datatable, sort, display, material, tree
28	Issue solving	Debugging \ Troubleshooting	issue, custom, show, add, model, correctly, message, problem, give, work
29	API response handling	Integration \	datum, access, api, response, bind, http,header,
30	Typescript language	API integration Basic concepts	fetch, localhost, origin typescript, method, call, callback, type, interface, support, subscribe, class, extend
31	Click events	User actions	button, click, tab, mat, menu, hide, toggle, radio, group, icon

Results

Front-end development basically focuses on developing the front face of an application. So the major topics are user interface design, user action, Application Program Interface (API) integration, routing, unit testing, and debugging. To illustrate, *User Interface (UI)* topic is basically defined based on the keywords material design, datatable, chart, component, style, etc. which are materials or components to design the UI. User action includes form action development, button events, click events, observables, etc. and these are developed for interaction between the system and end users.

Finding 1: Front-end developers ask questions in SO about a broad area of 32 topics including External Library Management, Unit Testing, Material UI, Datatable, Click Events, API Integration, and Routing.

Figure 4.2 represents the topics with the number of posts and their percentage. The highest number of posts for a topic is 5632 (4.48%) and the lowest is 1942 (1.55%), with percentages

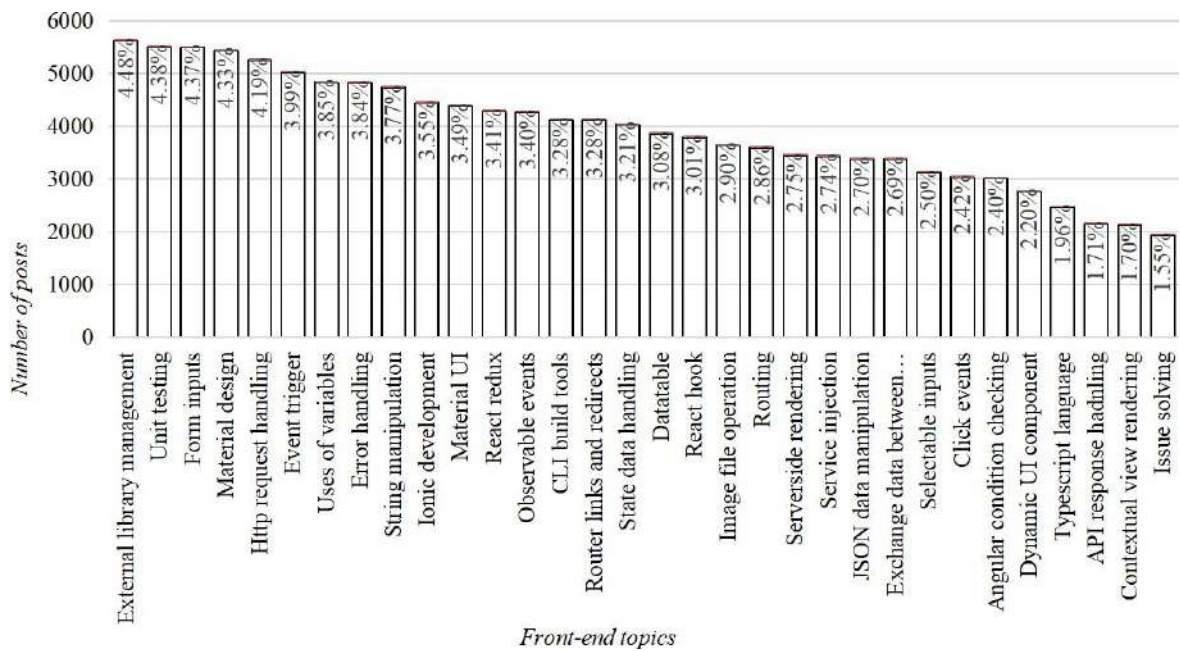


Figure 4.2: Front-end topics and number of their questions

varying from around 1.5 to 4.5% of total posts. It is clear that the discussions regarding the use of external libraries like *Material UI* or *Datatable*, user interface, or action development is higher than the rest of the topics. On the other hand, language related issues, data manipulation, and API integration – these issues are discussed less because these are rather difficult topics and need high expertise. So the number of developers working on these issues is fewer than the developers working on the UI.

Finding 2: The most discussed topic is External Library Management and the least is common Issue Solving topic.

Table 4.5: Topics with no of posts and their percentage

Topic No	Topic Label	No of Posts	% of total posts
0	Material design	5443	4.33%
1	External library management	5632	4.48%
2	String manipulation	4741	3.77%
3	Uses of variables	4838	3.85%
4	Event trigger	5018	3.99%
5	Http request handling	5259	4.19%
6	Unit testing	5509	4.38%
7	Error handling	4827	3.84%
8	Form inputs	5496	4.37%
9	State data handling	4034	3.21%
10	Dynamic UI component	2767	2.20%
11	Service injection	3437	2.74%
12	Ionic development	4456	3.55%
13	React hook	3787	3.01%
14	Angular condition checking	3014	2.40%
15	Serverside rendering	3456	2.75%
16	React redux	4289	3.41%
17	Image file operation	3643	2.90%
18	JSON data manipulation	3387	2.70%
19	Exchange data between components	3379	2.69%
20	CLI build tools	4121	3.28%
21	Material UI	4389	3.49%

- continued on next page

Table 4.5 - continued from previous page

Topic No	Topic Label	No of Posts	% of total posts
22	Routing	3596	2.86%
23	Selectable inputs	3138	2.50%
24	Observable events	4278	3.40%
25	Router links and redirects	4120	3.28%
26	Contextual view rendering	2137	1.70%
27	Datatable	3868	3.08%
28	Issue solving	1942	1.55%
29	API response handling	2149	1.71%
30	Typescript language	2464	1.96%
31	Click events	3045	2.42%
Total		125659	100%

RQ2: Front-end Topic Hierarchy

Motivation

Multiple topics can be grouped into the same category if they point to a similar domain. For example, **React Redux** and **React Hooks** both are used for state management in React framework. Hence, they can be grouped under the category **State management**. Multiple categories can also be grouped together under a super or higher-level category. Categories help to make the topics domain-specific and more understandable.

Approach

Categorizing topics follow a similar approach to labeling topics described in section 4.1.2. The author and the industry professional categorizes topic sets through constructive arguments with

the help of the supervisor.

Results

Figure 4.3 represents a graphical view of topic categories. It shows the topics in gray and their categories in white. The hierarchy of categories goes outwards from a higher level to lower level categories and topics.

The topics are grouped into categories based on their subject domain. Here, Topic *Event trigger*, *Observable events*, *Click events* all are grouped into category *Events handling*. Multiple categories can represent the same domain of any application. So these categories are grouped into another high-level category for the purpose of representing the categories as more domain-specific and less redundant. Here, the categories *Events handling* and *Inputs* both are actually parts of user actions in any system. Hence, both categories are being placed under the category *User actions*. The hierarchy finally consists of a total of 11 high-level categories and 15 categories.

According to Figure 4.3, *User Interface* category has the highest 17.49% of posts. As most of the front-end developers are UI and UX engineers, they often ask interface design related questions such as component rendering, styling, view rendering integration, etc. The next highest 16.68% questions are under *User Actions* category. It includes topics like *Events handling* and *Form actions* which are the most vital part of the front-end development of any system. The third is the *Development* category which comprises 16.07% questions of all posts including topics like *External library management*, *Service injection*, *Image file operations*, etc. Most of the language and JS framework related questions are under this category. So around half of all the questions are from the first 3 categories: *User Interface*, *User Actions*, and *Development*. *State management* and *Basic concepts* are almost equal in percentage which is around 9.6% and have questions regarding state handling and basic language related posts. Integration has

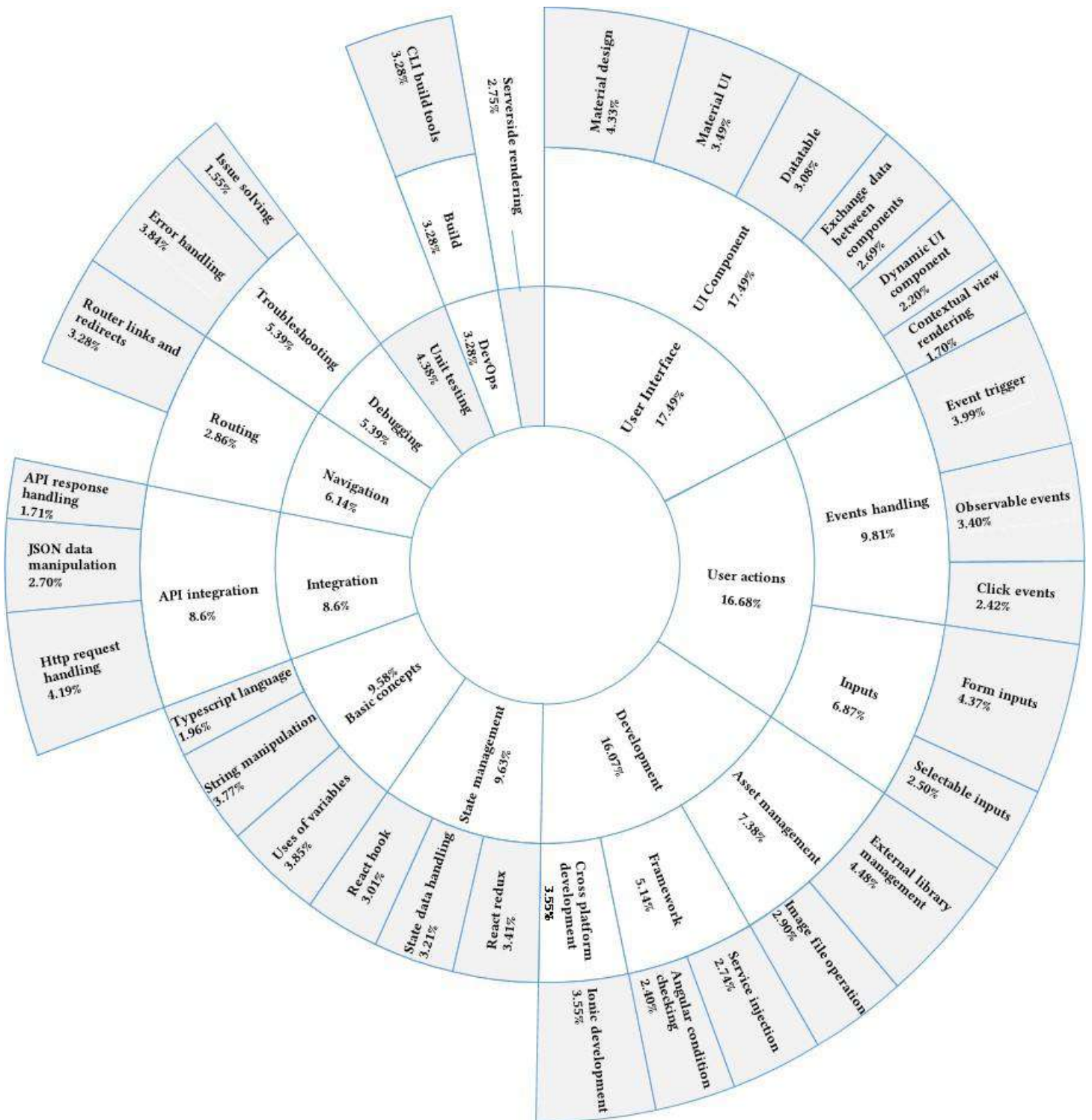


Figure 4.3: Front-end topics and number of their questions

8.6% questions regarding API and service integration. The rest of the categories have percentages less than 7% and have questions about routing, testing, debugging, and other issues.

Finding 3: Around 1/2 of all questions are from top 3 categories: *User Interface*, *User Actions* and *Development*.

RQ3: Front-end Topic Popularity and Difficulty

Motivation

In order to understand the trends, we must know the most popular as well as difficult topics. A popular topic can be more demandable for the developers and needs to be answered immediately with the best possible solutions. A difficult topic needs more attention from the community. Moreover, highlighting difficult topics can suggest to the developers whether a better tool or framework is needed or not.

Approach

The approach for measuring popularity and difficulty is already discussed in Section 3.3. We use the standard approach for measuring this. We consider a topic popular if the post has a high average view count, favorite count, and score. We consider a topic difficult if it has less or no accepted answers and took a long time to get an accepted answer.

Results

Table 4.6 represents the popularity measurements of the topics. The data is sorted by average views in descending order. So the most viewed topic is *CLI build tools*. This topic is at the top in all 3 measures. So it is altogether the most viewed, most favorite, and top-scored topic. On the other hand, *Serverside rendering* is the least viewed, 10th from least favorite, and 6th from

the lowest score.

Finding 4: *CLI build tools* is the most popular topic and *Serverside rendering* is the least popular topic.

Table 4.6: Topic popularity

Topic	Avg Views	Avg Favorites	Avg Scores
CLI build tools	5241.4	1.01	4.37
Routing	5129.2	0.87	4.27
Router links and redirects	4872.3	0.91	4.03
Error handling	4675.0	0.57	3.55
Event trigger	4453.1	0.70	3.41
API response handling	4390.3	0.61	2.61
String manipulation	4189.6	0.52	2.66
Material design	4133.9	0.69	3.08
Uses of variables	4131.8	0.73	3.50
Selectable inputs	4104.9	0.53	2.60
Click events	4083.2	0.49	2.20
External library management	4034.1	0.87	3.98
Form inputs	3970.4	0.61	2.75
Observable events	3893.9	0.96	3.85
React hook	3880.7	0.92	4.26
JSON data manipulation	3846.5	0.54	2.36
Angular condition checking	3768.1	0.63	2.60
Material UI	3742.1	0.52	2.88

- continued on next page

Table 4.6 - continued from previous page

Topic	Avg Views	Avg Favorites	Avg Scores
Typescript language	3700.9	0.74	3.62
Http request handling	3509.8	0.64	2.21
Image file operation	3455.4	0.66	2.62
Datatable	3356.5	0.46	1.86
Exchange data between components	3313.2	0.65	2.95
Contextual view rendering	3288.6	0.62	3.01
Service injection	3271.2	0.81	3.29
Unit testing	3101.4	0.59	3.43
State data handling	2992.8	0.72	2.84
Dynamic UI component	2939.9	0.65	2.71
React redux	2763.1	0.84	3.41
Issue solving	2651.2	0.52	2.31
Ionic development	2457.2	0.46	1.88
Serverside rendering	2258.7	0.59	2.33
Average	3737.5	0.68	3.04

Table 4.7 shows the topics in consideration of their difficulties. The data is sorted in ascending order by the percentage of questions without any accepted answer. From the table, we see that the topic *Observable events* has the highest 64% questions without accepted answers. The questions on this topic usually take 21.22 hours which is the 5th highest time to get an accepted answer. So we can state that this topic has the most difficult questions. On the other hand, *Ionic development* topic has the least percentage of questions without accepted answers and also 6th from lowest time to get accepted answers.

Table 4.7: Topic Difficulty

Topic	% w/o accepted answers	Hrs to accepted answers
Observable events	64%	21.22
Exchange data between components	59%	9.54
Error handling	58%	12.17
Typescript language	58%	14.93
State data handling	57%	6.98
Event trigger	57%	16.26
React hook	56%	13.01
Selectable inputs	56%	13.57
Click events	56%	14.19
Uses of variables	56%	9.48
Contextual view rendering	55%	11.80
React redux	55%	10.48
API response handling	55%	7.11
Material UI	55%	24.37
Form inputs	55%	11.29
String manipulation	54%	12.00
JSON data manipulation	54%	11.65
Service injection	54%	16.41
Dynamic UI component	54%	12.70
Routing	53%	14.20
Router links and redirects	52%	24.44

- continued on next page

Table 4.7 - continued from previous page

Topic	% w/o accepted answers	Hrs to accepted answers
Material design	52%	18.52
Issue solving	51%	13.80
Angular condition checking	51%	11.88
Datatable	51%	18.44
Serverside rendering	48%	21.06
Http request handling	48%	13.66
CLI build tools	47%	32.48
Image file operation	46%	20.02
Unit testing	46%	25.24
External library management	45%	20.56
Ionic development	45%	10.46
Average	53%	15.22

Finding 5: *Observable events* is the most difficult and *Ionic development* is the least difficult topic.

Correlation between Topic Popularity and Difficulty

From Table 4.6, we see that *Serverside rendering* is the least popular topic but Table 4.7 presents that it is a difficult topic. This leads to the discussions of correlation between topic popularity and difficulty. In some studies [12, 13], no significant correlation is found between these. However, in some other studies [16], significant negative correlations were reported. So we need to confirm our hypothesis that there can be correlations between them. We use the Kendall correlation [29] for confirming this. We calculate correlation based on 3 popularity

metrics and 2 difficulty metrics. Table 4.8 represents the result of correlations at 95% confidence level. We verify our assumptions by calculating the correlation coefficient and p-value for the metrics' values. For two variables, X and Y, the correlation coefficient serves as a measure of how much the dependent variable (Y) tends to change when the independent variable (X) changes. It measures both the relationship's strength and its direction. A positive correlation coefficient indicates a direct link between the variables, whereas a negative correlation coefficient indicates an inverse relationship. We can therefore differentiate between 3 types of correlation as below-

- No correlation - the coefficient is exactly 0.
- Positive correlation - the coefficient is between 0 and 1.
- Negative correlation - the coefficient is between -1 and 0.

Table 4.8 presents that 5 out of 6 coefficients are positive and 1 is negative but all the coefficients are near 0 which suggests that the correlation exists but it is not much significant.

We calculate the p-values to verify our findings. A correlation result can be said statistically significant if the p-value is below the evidential threshold decided for the statistical test. For example, if a hypothesis is true for 1 out of 20 times, then the threshold value to be considered sufficient evidence to reject the hypothesis, is 0.05. For our study, we consider the threshold of p-value 0.05 as it is standard and used in previous studies [7, 12, 13, 16].

Here all the *p-values* are greater than 0.05 which rejects our assumptions. This clearly confirms that there is no significant correlation between topic popularity and difficulty. So, the difficult topics are not necessarily popular among developers and the opposite is also true.

Table 4.8: Topic Correlation

Correlation coefficient / p-value	Avg Views	Avg Favorites	Avg Scores
% w/o accepted answers	0.133/0.142	0.077/0.269	0.194/0.060
Hrs to accepted answers	0.0121/0.461	-0.028/0.410	0.032/0.398

Finding 6: The correlation between topic popularity and difficulty is not statically significant.

RQ4: Topic Evolution

Motivation

The trends in SO posts have always been changing over time. The software industry has been evolving with new technologies. To keep pace with it, developers' choice of questions has been changing over time. We need to analyze the trends to get a basic idea of the growth of JS frameworks over time.

Approach

For determining topic evolution, we consider two metrics- topic absolute impact and relative impact. The topic absolute impact is determined by the number of posts over a period of time and relative impact is calculated by the percentage of posts among total posts for the timeframe. In 2016, with the launching of some top popular JS frameworks, they are becoming more popular. So we consider the timeframe from 2016 to 2020 over the period of 6 months. For calculating absolute impact, we consider the top 5 categories: *User interface*, *User actions*, *State management*, *Development and Integration*. For relative impact calculation, we consider our 3 chosen frameworks. We also investigate the growth of all JS posts in SO for comparison and analyze the trends with other languages.

Results

From fig. 4.4, we can see that JS is the second most trending language in SO after Python [5]. It was at the top for many years. Recently after 2020, Python language is gaining popularity as it has been widely used for big data handling and machine learning. Python is also popular in NLP and AI which is now trending all over the world. However, the popularity of JS is still on

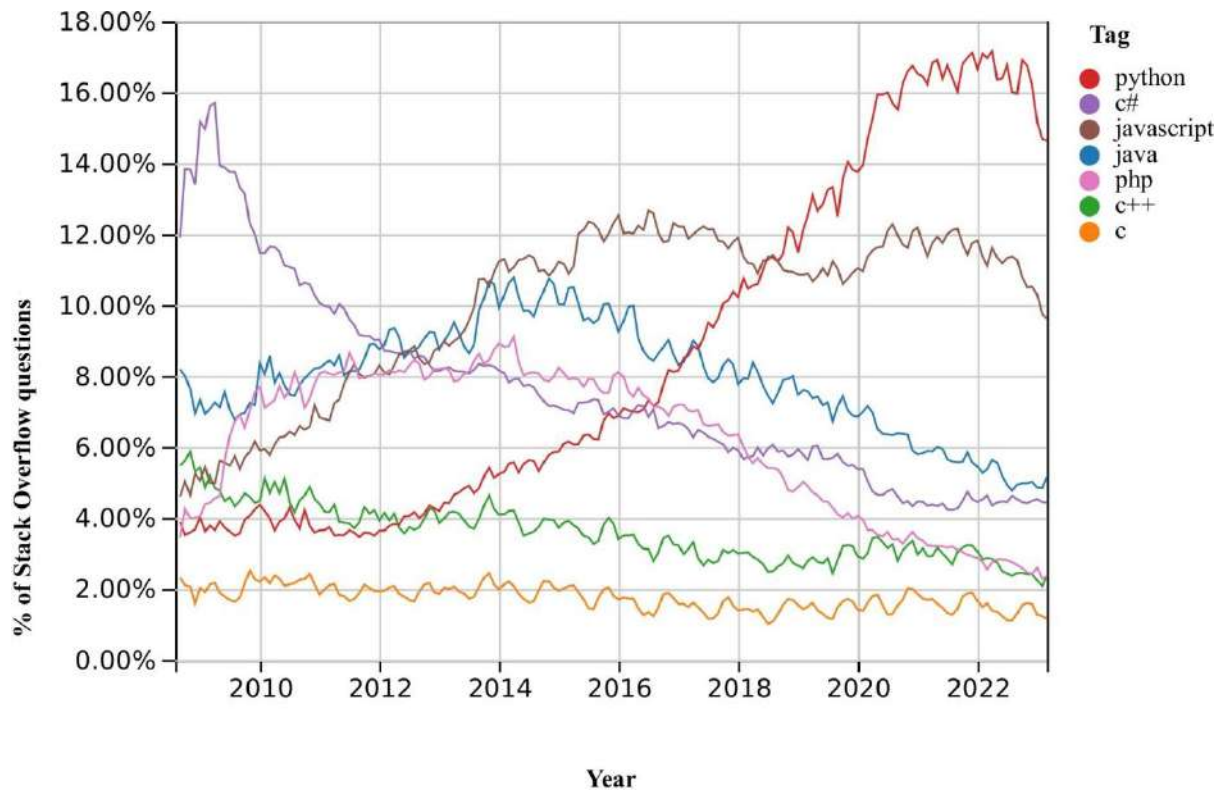


Figure 4.4: Most popular languages in SO

top as most front-end developers still love the JS frameworks. Since 2014, with the launching of multiple popular JS frameworks, it has been ruling the SO over all other languages. We can see that JS posts are continuously growing in SO starting from 5% in 2008 to around 12% in 2022 and increasing.

Figure 4.5 shows the relative impact or growth of the JS frameworks related posts compared to all the SO posts in the corresponding time frame. Since 2014, angular, react and vue.js have been gaining popularity. They are now the top 3 JS frameworks according to SO trends [5]. Since 2016, all 3 frameworks have been on top. React has been the most popular and most used among them. The popularity of the Angular framework dropped after 2019 whereas React has been rising since then. Vue.js is trying to catch up with them after its launch in 2014 [21]. To be noted, the Angular framework is launched in 2016 [19] and React in 2013 [20].

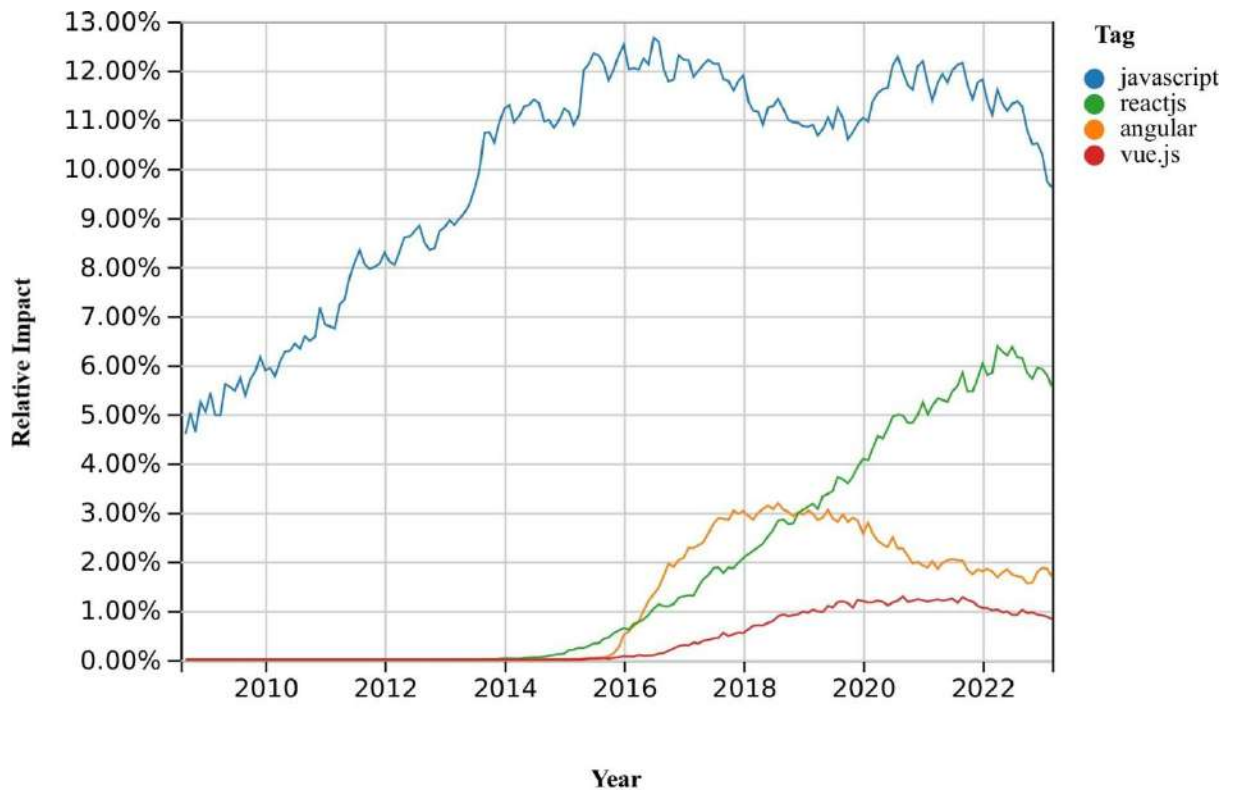


Figure 4.5: Relative impact for top 5 topics over 6 months interval from 2016 to 2020

Figure 4.6 represents the graphical view of the absolute impact of the top 5 topics. As Figure 4.5 shows that the significant increase for our JS frameworks started in 2016, so we consider the 2016 to 2020 timeframe for this analysis. We can see that *User Interface* is the most popular among all and *User Actions* is right after it. As frontend development focuses on user interface and interaction related things, these two have been the most popular category in SO in recent times. *Development* and *State Management* is below them. These two are important for front-end development. Overall development like framework, library, file management related questions is covered in this category. State management is necessary for managing in-memory state and data. *Integrations* is at the bottom of the list but it is gaining popularity in recent times. This topic is important in terms of frontend and backend integration developments.

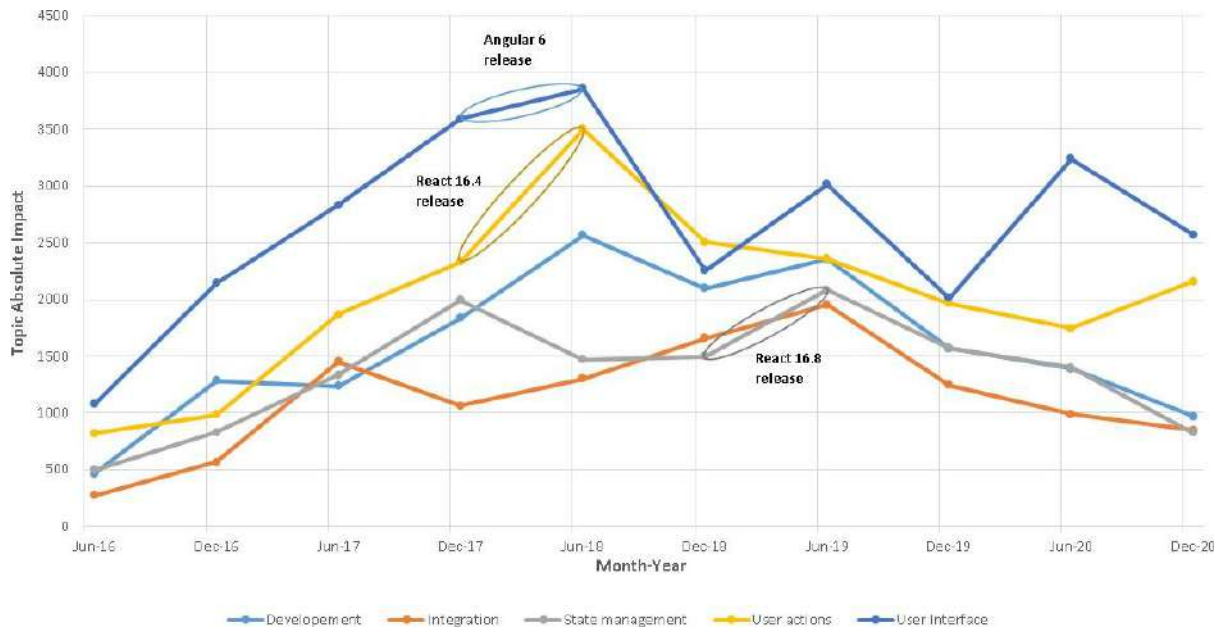


Figure 4.6: Absolute impact for top 5 topics over 6 months interval from 2016 to 2020

There are some sudden hikes in the data and we need to analyze the root cause behind these. The rise in *User Interface* from January 2018 to June 2018 is because of the Angular 6 release on May 4, 2018. Angular 6 release was a major release for the Angular framework which included Angular Material and CDK components, CLI workspaces, Library supports, and RxJS v6 [19]. The peak in *User Actions* from January 2018 was caused by React 16.4 release (24 May 2018) which included some major improvements in events handling. React 16.8 was released on February 6, 2019 [20]. It included React hooks and Redux and it caused a huge rise in *State Management* from January 2019 to June 2019.

Finding 7: *User Interface* and *User Actions* are the most popular topic categories over the last few years.

To conclude, the results we have shown answer all the research questions of our study. The data, figures, and tables are represented for a better understanding of the discussed topics, categories, their popularities and difficulties, and their evolutionary trends over time.

5

DISCUSSIONS

There can be many implications for our findings. These results can help certain people and organizations who are involved with software development, especially front-end development. In this chapter, the implications have been discussed briefly along with the types of threats we need to validate.

Implications

The results, we produce by this study, can help practitioners, educators, and researchers all over the world to easily decide on what topics they should focus on. It can save a lot of time and effort. They may consider the popularity and difficulty of topics and use their resources based on that. This analysis can guide them to the best practice of relevant technology.

Figure 5.1 represents the popularity and difficulty of the top 10 popular topics from Table 4.6. It shows the difficulty vs popularity of the topics. The circle represents the total number of posts on the topic. We use average views as a proxy for popularity and percentage without accepted answers as a proxy for difficulty. This study can help the following people in different ways.

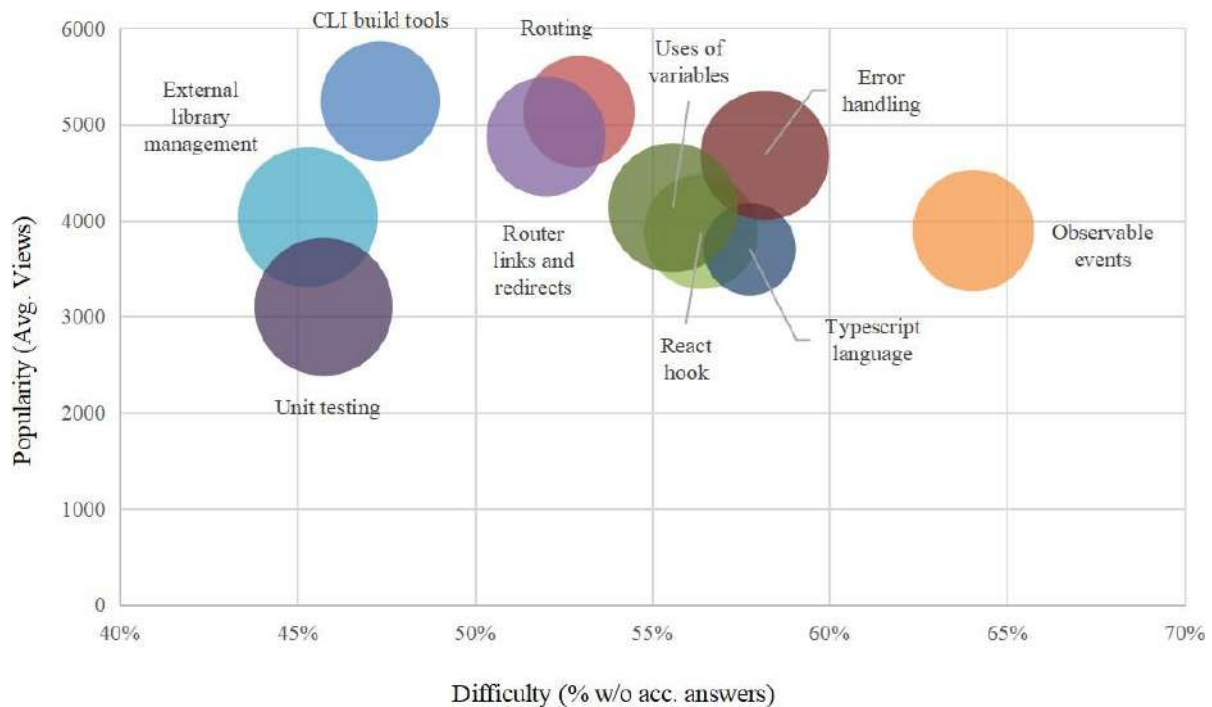


Figure 5.1: Front-end topics popularity vs difficulty for top 10 topics

Practitioners: Figure 5.1 states that *External library management* and *CLI build tools* are popular but less difficult topics because they have more views and accepted answers. These are kind of ideal topics and practitioners should find them at the top of their search. SO can use these findings to make their search results better.

Topics like *Routing* or *Uses of variables* are both popular and difficult. So, the mass developer community should focus more on these topics. They can make tutorials and documentation to reduce the difficulty levels.

On the other hand, *Typescript language* and *Observable events* are less popular and more difficult as the questions have average views but are mostly unanswered. *Observable events* is the most difficult among others. These topics are under the *User actions* category. So topics related to user actions like events and inputs are likely to take more resources (manpower, time and processing) than usual. A software manager should take that into consideration while assigning any tasks on features related to the topic category. Moreover, he can assign these topics to expert resources to handle. Thus, a lot of development time can be saved.

The findings for difficult topic categories like *User actions* can be used to motivate practitioners to improve the frameworks. They can add simple features and documentation to reduce the difficulty.

External library management is the largest topic in our dataset. It is part of the *Development* category and this holds the major part of questions from our dataset. Practitioners should invest more resources in this category than others.

Software organizations: Popular and difficult topics like *Routing* or *Uses of variables* are often most searched but mostly unsolved. Software organizations can provide training programs to developers who work on these. Training can help them to come up with solutions or alternative approaches on their own.

Researchers: Researchers may decide to give efforts to either the most popular less difficult topics or vice versa. An expert researcher can focus on the difficult topics first like *Observable events*. In that way, s/he may work on the improvement to reduce the difficulty level. Moreover, less popular but difficult topics are likely to be less explored area. So, researchers may find more scopes to improve in this area. *Observable events* or *Error handling* may be a good choice to start with.

Educators: Educators or trainers may decide based on the data in Figure 5.1 whether to start teaching the student difficult or popular topics. They may start with a more popular and less difficult topic at first like *External library management* or *CLI build tools*. Then they can move on to the more difficult topics. It also may help the students to focus on the topics. Educators can design their courses as per this. More attention should be given to difficult topic categories like *User actions*. So detailed course materials, tutorials, and examples should be added for this.

Practitioners, researchers, educators, and software organizations should consider the aspects and factors to which they should give their efforts. We believe that these findings can help to make better decisions.

Threats to Validity

There are 3 types of threats to validate here- Internal, External, and Construct validity. Internal validity refers to the degree to which the results of a study accurately reflect the underlying relationships and patterns in the data. External validity refers to the degree to which the results of a study can be generalized to other contexts or populations. Construct validity is about how well a test measures the concept it was designed to evaluate.

Internal Validity

Internal validity considers the facts that could have influenced our results. Using initial tags may lead to irrelevant and insignificant data as front-end development has a very vast knowledge base. Also, many questions are posted with wrong tags and sometimes the questions are not fully understood. We mitigate this by considering all tags that coexist with the initial tag and analyzing the TST and TRT values. Another threat is to label the topics manually. The author and involved persons have expertise in this area to validate the labeling but there can be arguments over the choice of labels. We minimize this by continuously discussing to reach a consensus and considering the feedback from the supervisor.

The choice of topic modeling algorithm and parameters can affect the internal validity of the results. It is important to select an appropriate model that accurately captures the underlying structure of the data. The evaluation metrics used to assess the quality of the topic model can affect the internal validity of the results. Choosing appropriate metrics is important to ensure that the model is accurately representing the data.

Finding an optimal number of topics is another issue we need to focus on. A number of

topics K and iterations I is chosen based on the approach of prior works [12-14]. We consider different K values and examine the coherence score to find out the optimal value. The selection of hyper-parameters set causes topic instability problems which is a serious threat. It may cause inaccuracy in our statistics. We solve the problem using DE [18].

We conduct this study with the SO dump of February 2021 and considered the data between 2016 and 2020. We found that the vast data from the inception of SO- 2008 to 2022 is very noisy and difficult to manage. Also, the data from the early days of JS is very immature and unstable. So there can be different outputs if we consider them from the beginning. Moreover, this data is a little old, so the newer data can provide different outputs. Given that our primary focus was to understand the topic's behavior and categorize them, our analysis from 2016 to 2020 provides a good insight into the 3 JS frameworks.

External Validity

External validity concerns the generalization of the findings of the study. We consider only the 3 JS frameworks for our study. It has no dependency on other languages and so, our results can not be applied to other languages. Another point is, SO has a large and diverse user base, but the extent to which the findings apply to other developer communities may vary. Topic modeling on SO is affected by changes in the platform and in the development community over time. The extent to which the results remain valid over time is an important consideration.

Furthermore, SO has a global user base, and cultural differences may affect the validity of the results. For example, the way that developers ask and answer questions may vary across cultures. Moreover, we only consider SO as our community Q&A site but there are other popular sites that can be used for this type of study and our results will not be the same for those sites. We believe that SO is the best choice for the generalization of our results. However, this study can be improved if we include posts from other popular forums.

Construct Validity

Construct validity considers the connection between theory, observations, and outcomes. It checks if the result proves the concept or not. Manual topic labeling might not reflect the actual posts related to the topics. To mitigate this, we individually examine 15 random posts for each topic and discuss until we reach an agreement for the proper label. Also, calculating popularity, difficulty and their correlation is vulnerable to construct validity. We use the known metrics for determining these results from previous studies [[12](#), [13](#), [15](#), [16](#)].

In summary, we try to mitigate all the threats we have faced in our study using the known approaches. We try to cover all the major threats but there can still be some factors that can be improved in the future. We have also discussed how our findings can help people for finding better solutions.

6

CONCLUSION AND FUTURE WORKS

In this research, we conduct an empirical study on front-end development technology using Stack overflow data. We develop front-end topics that developers all around the world ask for, create a hierarchy of topics, determine popularity and difficulty and their correlation, and study evolutionary behavior. The purpose of our study is to understand the topics that developers mostly discuss and study the evolution of these topics over time. We present our findings and results to evaluate the study. We state how our study can be beneficial to practice, research, and educational purpose.

What we learn from this study

In our study, we have applied topic modeling on 1,25,659 posts for the 3 front-end frameworks- Angular, React, and Vue.js. We have extracted 32 topics by running LDA on the posts. The topics are grouped into a total of 11 categories. *User interface* is the category with the most number of topics (6) which covers the highest percentage of posts (17.49%) from our dataset. This topic is followed by *User actions* (16.68%) and *Development* (16.07%). *External library management* is the topic with the highest number of posts (5632) which is 4.48% of all the posts, followed by *Unit testing* (4.38%) and *Form inputs* (4.37%). *CLI build tools* is the most

popular topic with the highest number of views (5241.4) and *Observable events* is the most difficult topic with the most percentage of questions without accepted answers (64%). These findings can help front-end developers and practitioners all around the world find solutions and design their resources in a more effective way.

Future Works

Topic modeling on Stack Overflow is an active area of research, and there are several future work scopes that could be explored. Here are some potential areas for future research:

Integration with other data sources: Topic modeling on SO could be integrated with other data sources, such as GitHub or social media platforms, to provide a more comprehensive view of the development community. By combining data from multiple sources, researchers could gain insights into how development practices and trends evolve over time.

Surveying developers: We shall conduct a large-scale survey on developers, managers, and other involved persons from different locations to validate our findings. We shall collect their valuable feedback on our 32 topics and 11 categories. This will give more insights to analyze each of the topics deeply. We shall also aim to find out the differences in development practices between the first-world software engineering industry and the industry in developing countries in the survey.

Incorporating available user data: Our study is based solely on the content of the questions and the title. Future work could incorporate the answers, code snippets, and user behavior data, such as voting patterns or browsing behavior, to provide a more complete picture of the development community. We can also include geographical data of the users to know more about the evolutionary pattern of the topics geographically.

Overall, there are several exciting future work scopes for topic modeling on Stack Overflow. By exploring new techniques and integrating with other data sources, researchers can

gain deeper insights into the development community and help developers stay informed about emerging trends and best practices.

Bibliography

- [1] w3techs. (2022, March) Usage statistics of JavaScript libraries for websites. Accessed 6-March-2022. [Online]. Available: https://w3techs.com/technologies/overview/javascript_library
- [2] Stack Overflow. (2021, December) 2021 Developer Survey. Accessed 12-December-2021. [Online]. Available: <https://insights.stackoverflow.com/survey/2021#most-popular-technologies-language-prof>
- [3] builtWith. (2022, March) Javascript Usage Distribution. Accessed 6-March-2022. [Online]. Available: <https://trends.builtwith.com/javascript>
- [4] S. Bin Uzayr, N. Cloud, and T. Ambler, *JavaScript Frameworks for Modern Web Development*. Springer, 2019.
- [5] Stack Overflow. (2022, December) Stack Overflow Trends. Accessed 12-December-2022. [Online]. Available: <https://insights.stackoverflow.com/trends/?tags=r%2Cstatistics>
- [6] D. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation: the journal of machine learning research, v. 3," 2003.
- [7] P. Chakraborty, R. Shahriyar, A. Iqbal, and G. Uddin, "How do developers discuss and support new programming languages in technical q&a site? an empirical study of go, swift, and rust in stack overflow," *Information and Software Technology*, vol. 137, p. 106603, 2021.

- [8] Stack Overflow. (2021, December) Tags and Questions. Accessed 12-December-2021. [Online]. Available: <https://stackoverflow.com/questions/tagged/javascript>
- [9] D. M. Blei, T. L. Griffiths, and M. I. Jordan, "The nested chinese restaurant process and bayesian nonparametric inference of topic hierarchies," *Journal of the ACM (JACM)*, vol. 57, no. 2, pp. 1-30, 2010.
- [10] S. Huh and S. E. Fienberg, "Discriminative topic modeling based on manifold learning," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 4, pp. 1-25, 2012.
- [11] D. Cai, X. Wang, and X. He, "Probabilistic dyadic data analysis with local and global consistency," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 105-112.
- [12] A. Abdellatif, D. Costa, K. Badran, R. Abdalkareem, and E. Shihab, "Challenges in chatbot development: A study of stack overflow posts," in *Proceedings of the 17th international conference on mining software repositories*, 2020, pp. 174-185.
- [13] M. Bagherzadeh and R. Khatchadourian, "Going big: a large-scale study on what big data developers ask," in *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 432-442.
- [14] A. Bandeira, C. A. Medeiros, M. Paixao, and P. H. Maia, "We need to talk about microservices: an analysis from the discussions on stackoverflow," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 255-259.
- [15] 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.

- [16] S. Ahmed and M. Bagherzadeh, "What do concurrency developers ask about? a large-scale study using stack overflow," in *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*, 2018, pp. 1-10.
- [17] A. K. McCallum, "Mallet: A machine learning for language toolkit," <http://mallet.cs.umass.edu>, 2002.
- [18] A. Agrawal, W. Fu, and T. Menzies, "What is wrong with topic modeling? and how to fix it using search-based software engineering," *Information and Software Technology*, vol. 98, pp. 74-88, 2018.
- [19] Wikipedia. (2023, March) Angular (Web Framework). Accessed 6-March-2023. [Online]. Available: [https://en.wikipedia.org/wiki/Angular_\(web_framework\)](https://en.wikipedia.org/wiki/Angular_(web_framework))
- [20] Wikipedia. (2023, March) React (Software). Accessed 6-March-2023. [Online]. Available: [https://en.wikipedia.org/wiki/React_\(software\)](https://en.wikipedia.org/wiki/React_(software))
- [21] Wikipedia. (2023, March) Vue.js. Accessed 6-March-2023. [Online]. Available: <https://en.wikipedia.org/wiki/Vue.js>
- [22] H. Li, T.-H. Chen, W. Shang, and A. E. Hassan, "Studying software logging using topic models," *Empirical Software Engineering*, vol. 23, pp. 2655-2694, 2018.
- [23] A. T. Nguyen, T. T. Nguyen, J. Al-Kofahi, H. V. Nguyen, and T. N. Nguyen, "A topic-based approach for narrowing the search space of buggy files from a bug report," in *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE, 2011, pp. 263-272.
- [24] T. T. Nguyen, T. N. Nguyen, and T. M. Phuong, "Topic-based defect prediction (nier track)," in *Proceedings of the 33rd international conference on software engineering*, 2011, pp. 932-935.

- [25] D. Andrzejewski, A. Mulhern, B. Liblit, and X. Zhu, "Statistical debugging using latent topic models," in *Machine Learning: ECML 2007: 18th European Conference on Machine Learning, Warsaw, Poland, September 17-21, 2007. Proceedings 18*. Springer, 2007, pp. 6-17.
- [26] C. Treude and M. Wagner, "Predicting good configurations for github and stack overflow topic models," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 84-95.
- [27] Stack Exchange. (2021, February) Stack Exchange Data Dump download. Accessed 5-February-2021. [Online]. Available: <https://www.brentozar.com/go/querystack>
- [28] S. Fincher and J. Tenenber, "Making sense of card sorting data," *Expert Systems*, vol. 22, no. 3, pp. 89-93, 2005.
- [29] H. Abdi, "The kendall rank correlation coefficient," *Encyclopedia of Measurement and Statistics*. Sage, Thousand Oaks, CA, pp. 508-510, 2007.
- [30] Z. Wan, X. Xia, and A. E. Hassan, "What is discussed about blockchain? a case study on the use of balanced lda and the reference architecture of a domain to capture online discussions about blockchain platforms across the stack exchange communities," *IEEE Transactions on Software Engineering*, no. 01, pp. 1-1, 2019.
- [31] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.
- [32] S. Niwattanakul, J. Singthongchai, E. Naenudorn, and S. Wanapu, "Using of jaccard coefficient for keywords similarity," in *Proceedings of the international multiconference of engineers and computer scientists*, vol. 1, no. 6, 2013, pp. 380-384.