International Journal on Semantic Web and Information Systems
Volume 20 • Issue 1 • January-December 2024

Semantic Web Approaches in Stack Overflow:
Research Trends and Technological Insights

Zeeshan Anwar
https://orcid.org/0000-0002-8029-0604
Department of Computer Software Engineering, National University of Sciences and Technology, Islamabad, Pakistan

Hammad Afzal
Department of Computer Software Engineering, National University of Sciences and Technology, Islamabad, Pakistan

Seifedine Kadry
Lebanese American University, Beirut, Lebanon & Noroff University College, Kristiansand, Norway

Xiaochun Cheng
https://orcid.org/0000-0003-0371-9646
Swansea University, UK

ABSTRACT

StackOverflow (SO), a prominent question-answering site for programming, has amassed a vast repository of user-generated content since its inception in 2008. This paper conducts a thorough analysis of research trends on SO, examining 170 publications from 2008 to 2019. Utilizing qualitative and quantitative methods, the study categorizes papers using literature review and Latent Dirichlet Allocation (LDA), identifying 62 topics grouped into 8 main categories. Additionally, it highlights tools developed by researchers using SO data sets, showcasing their practical applications. The analysis also identifies research gaps and proposes future directions for each research area. This study serves as a valuable resource for practitioners and researchers interested in utilizing community data sets, offering insights into existing work, essential tools and techniques, and potential avenues for future research.

KEYWORDS

Research Trends, Stack Overflow, Techniques, Tools

1. INTRODUCTION

Stack Overflow (SO) is the largest and most popular online community of programmers. As of August 2024, there are more than 23 million registered users on SO, 21 million questions, and more than 32 million answers have been posted, with an answer rate of 71%. SO releases its data dump quarterly under the Creative Common Licence. The data set of SO is valuable for both practitioners and researchers. Programmers are using SO to solve their problems and help other programmers, whereas, researchers are using this data set for various purposes like tools development, API documentation, behavior studies, emotion mining, etc. The existing research on SO is mostly focused only on one research area at a time. Therefore we intend to fill this gap by conducting large-scale research to explore all the research areas in which SO data set is used.

DOI: 10.4018/IJSWIS.358617

While several systematic reviews and surveys have explored various aspects of research using Stack Overflow (SO) data, these studies often have notable limitations. For instance, A. Ahmad et al. (2020) conducted a systematic literature review on the use of machine learning algorithms for software requirements identification on SO, focusing primarily on Latent Dirichlet Allocation (LDA). However, their review is constrained by a limited scope, emphasizing only the identification and classification of software requirements while calling for increased collaboration between requirements engineering and machine learning communities to address unresolved issues. Similarly, Meldrum, Licorish, and Savarimuthu (2020) conducted a systematic mapping study to explore research interest in SO but highlighted the relatively low quality of existing studies, with many lacking proper validation for their proposed solutions. Another study by A. Ahmad, Feng, Ge, and Yousif (2018) surveyed research on mining SO but identified significant gaps in the current research, especially regarding the design and usage of Q&A sites like SO. These limitations underscore the need for a more comprehensive and systematic review that not only covers a broader range of research trends but also addresses the methodological gaps and validates the proposed solutions.

The work presented here intends to assist the research community in consolidating their respective research domains by identifying the gaps in existing research, suggesting ways of collecting data, the existing techniques that can be most useful to solve the problem, the tools that can help to automate the technique, analyze, and evaluate the results. Finding the solution to these problems can be timeconsuming. This research will help researchers who are interested in using SO data sets and answer all of the queries of the researcher and save their huge time. The focus of our review is to find the research trends and categorize them. Research tools and techniques that are used by researchers are also identified and listed in this paper. Research trends are determined by performing content analysis of papers. Various research trends against each category are given in this paper.

Research papers published from 2008 to 2024 are collected, neglecting the papers from low-quality sources. All papers in which ”stack overflow” appears in the title and abstract are selected. For the remaining papers, we searched the paper and included only those papers in our survey in which the SO data set is used. Before the manual review of papers, 62 topics from papers using LDA McCallum (2002) are extracted. LDA topics are labeled semi-automatically by using automatic term recognition Frantzi, Ananiadou, and Mima (2000).

This research gives an overview of various topics discussed on SO. The utilization of research tools and techniques on real problems and their results help to improve the existing tools. We also categorized tools and techniques based on their domain and usage. Based on the review of papers, we proposed various research findings against each research area by highlighting the potential and limitations of each paper. These findings give a quick overview to readers about existing research and save them time in going through various papers. Finally, we identified the various gaps in the existing research and proposed many research areas that will be helpful for future researchers.

This survey paper makes the following contributions to the existing body of knowledge:

- Comprehensive Coverage: Unlike previous surveys, which often have a limited scope and focus on specific problems or narrow research areas, our study takes a general-purpose approach. We cover all major research trends, tools, and techniques used in studies involving Stack Overflow (SO) data. This broad scope allows us to provide a holistic view of the field, offering insights across multiple domains rather than being confined to a single area.
- Advanced Methodology: Our study employs Latent Dirichlet Allocation (LDA) to automatically extract and categorize topics from the literature. This semi-automated approach not only enhances the objectivity of our analysis but also enables us to identify emerging trends and topics that might be overlooked in manual reviews. We also utilized automatic term recognition to label the LDA topics, further refining our analysis.
- Recent Data Inclusion: We have incorporated research papers published from 2008 to 2024, ensuring that our findings reflect the most current trends and developments. By including this

extended timeframe, our survey captures the latest advancements and shifts in research focus, which previous reviews may not cover.

- Identification of Gaps and Future Directions: Beyond summarizing existing research, our paper specifically identifies gaps in the literature and proposes new research areas. This forward-looking aspect of our work is designed to guide future research efforts, providing a roadmap for researchers who are interested in using SO datasets. This focus on practical application and future potential adds significant value and novelty to our review.
- Tool and Technique Categorization: We not only review research trends but also categorize the tools and techniques used in the literature according to their domain and usage. This categorization is intended to help researchers quickly identify the most relevant tools and methods for their specific needs, something that is not comprehensively addressed in existing surveys.

The remaining sections of this paper are organized as follows: Our research methodology is given in Section 2. Results of Topic Modeling and categorization of topics are given in Section 2.1. Stack Overflow research trends are identified and summarized in Section 3. Techniques and Tools used by researchers to solve their research problems are given in Section 4 and Section 5 respectively. Section 6 presents the findings of the review. Gaps in existing research are identified in Section 7 and finally, Section 8 concludes this research.

2. RESEARCH METHODOLOGY

The research methodology adopted in this paper consists of various steps as shown in Figure 1. A systematic review of the literature is performed to collect information related to our research questions. Research papers are collected from the different sources that include: IEEE Explore, ACM, Springer, Elsevier, PlosOne, DBLP, and Arvix (only selected). These databases are reputable sources in the fields of computer science and software engineering, ensuring that the papers retrieved are of high quality and relevance.

The search strings that we used to retrieve papers are:

1. Stack Overflow
2. Data Mining and Stack Overflow
3. Machine Learning and Stack Overflow
4. Stack Overflow data set
5. Stack Overflow Tools
6. Stack Overflow and Open Innovation
7. Sentiment Analysis and Stack Overflow
8. Topic Modeling and Stack Overflow

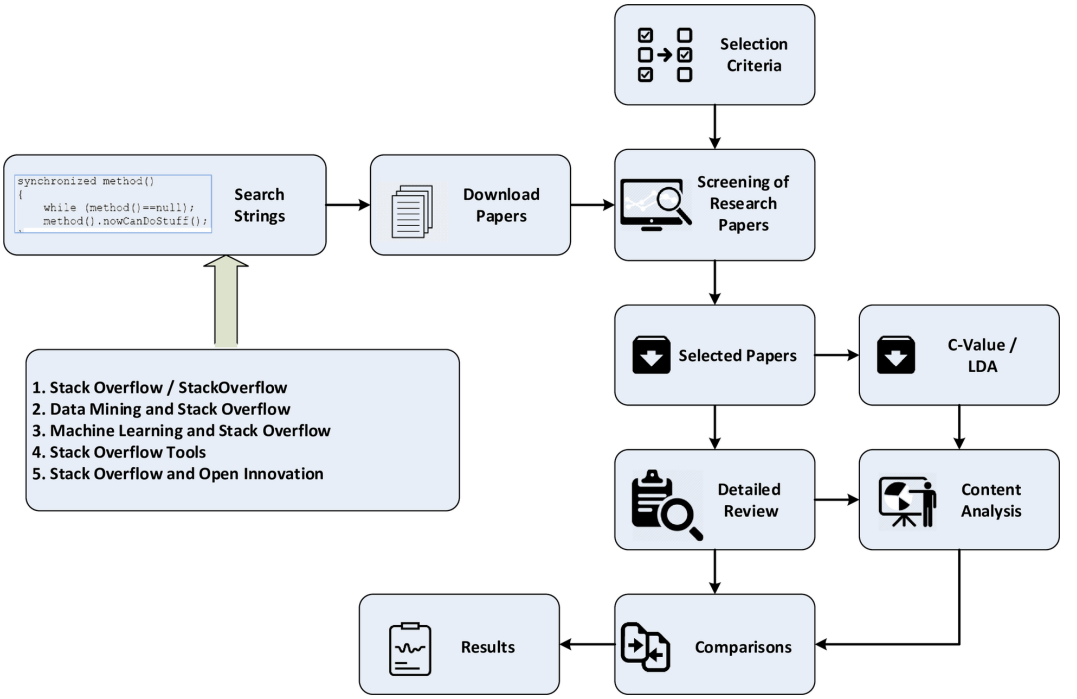
A total of 267 papers were retrieved, published from 2008 to 2024 (papers published up to August-24).

After downloading, the papers were selected for review based on the following criteria:

1. Papers should be from well-known publishers.
2. Papers are then manually screened based on the title and abstract.
3. All remaining papers are manually analyzed to select only those papers in which the Stack Overflow data set is used or that were directly related to your research questions.

As a result of the screening process, 214 papers are selected for review. Year-wise distribution of papers is plotted in Figure 2. Selected papers are divided into five types; these types include 121

Figure 1. Research methodology



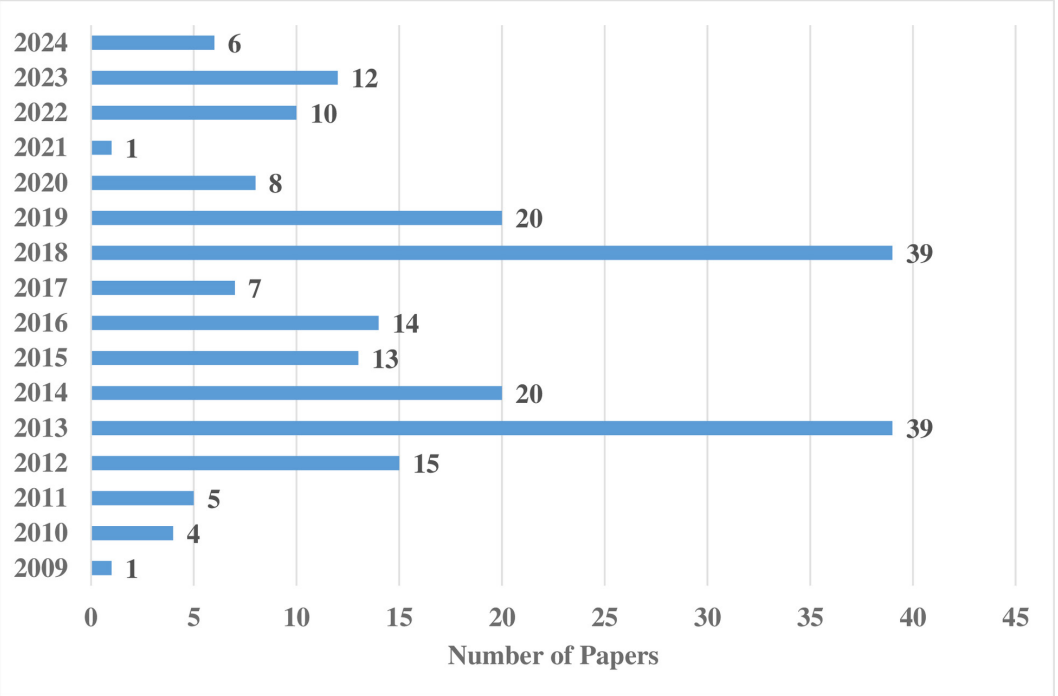
conference papers, 45 journal papers, 3 book chapters, 7 theses, and 1 technical report. Publication type-wise distribution of papers is given in Figure 3. We extracted the countries of the first author and plotted the number of publications and countries on the map by using the Google Geolocation API. The plot is given in Figure 4. 48 papers are published by USA authors, and 35 are published by Canadian authors.

Prior to detailed review, papers were automatically categorized by employing C-Value Frantzi et al. (2000) and Latent Dirichlet Allocation (LDA) McCallum (2002) methods. This process involves converting all the articles into text format and using the automatic term recognition method (C-Value) to extract the terms from the papers. These terms are used to label the topics that we extracted using LDA. LDA is a statistical topic modeling technique that is used to extract keywords related to a topic. This categorization provides unbiased and reproduce-able results to group similar papers. This categorization also speeds up the review process and also provides useful insights about the contents of the paper.

During the review of papers, we used two research and reference management software i.e. Mendeley Henning and Reichelt (2008) and Qiqqa Graham (2013). We performed a thorough review and content analysis of each paper. Every important point of the paper is annotated and a summary of each paper after highlighting the research problem, research methodology, features used, tools and techniques used to solve the problem, results, and evaluation metrics is given in our review. After writing a summary of the paper, we categorize the paper by assigning tags to the paper in reference management software. Tags are assigned based on the problem given in the paper, the research/ solution methodology used by the authors, and the results of C-Value and LDA. These tags help us to quickly extract relevant papers related

to a domain, find similarities between papers, and perform expedition analysis. Tools and techniques used to solve the research problem are given against each category of paper in the form of tables.

Figure 2. Year-wise papers distribution



Finally, we analyzed and compared the scope, research methodology, techniques, and results of each paper and identified the strengths and weaknesses of existing research. Based on scope, strengths, and weaknesses, we proposed gaps in the existing research.

2.1 Topic Modeling

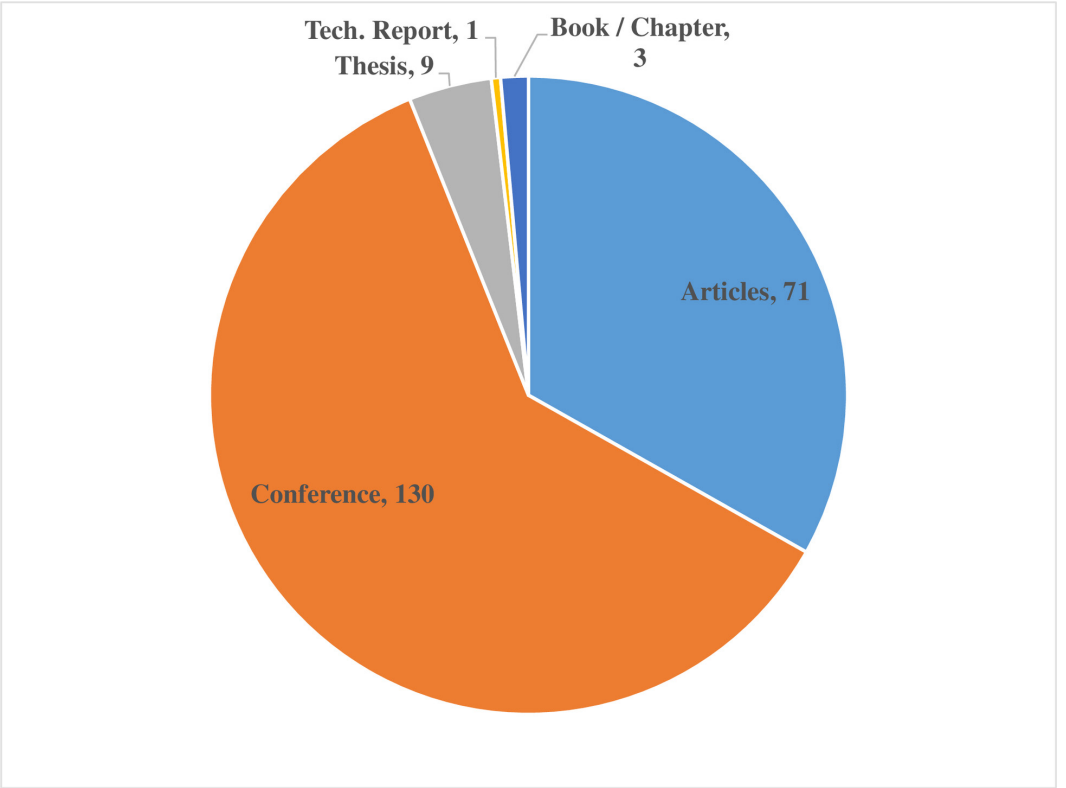
We created a corpus of all selected research papers and applied topic modeling to get an overview of topics discussed in the papers before starting our review. The intuition behind applying topic modeling is to extract topics from the corpus. These topics help us to categorize the papers based on the topics. We selected Latent Dirichlet Allocation (LDA) for topic modeling because LDA is the most popular and widely used algorithm for topic modeling. Mallet Toolbox McCallum (2002) is used to implement LDA. As LDA returns the list of top keywords, the assignment of a topic to keywords is a manual process.

We assigned topics to keywords semi-automatically by automatically extracting the terms using the CValue/NC-Value method Frantzi et al. (2000). The extracted terms and topic keywords were matched and assigned as topic-based, no contextual meanings of terms. When there does not exist a suitable term, we used the top four topic keywords to name a topic.

The C-Value method extracts the terms from the corpus and context of terms, i.e., frequent words that appear with the term are extracted by the NC-Value method. A prerequisite of the C-Value/NC-Value method is the Parts of Speech Tagged corpus. We used Stanford POS Tagger Toutanova, Klein, Manning, and Singer (2003) for POS tagging of the corpus. The C-Value method is implemented using Frantzi et al. (2000). A list of automatically extracted terms is given in Table 1.

Standard pre-processing steps, i.e., remove emails, newline characters, single quotes, punctuation, stop words, and lemmatization are performed before running LDA McCallum (2002). We determine the optimal number of topics by recursively running the LDA algorithms from two topics to 70 topics and computing the coherence score at each stop. A maximum coherence score of 0.44 is achieved

Figure 3. Type of publications



for 62 topics. It means there are 62 topics in the corpus. The graph of the coherence score is given in Figure 5. A list of topics, topic keywords, and the topic rate is given in Table LDA Results (attached as supplementary material). Accepted Answers, SO Comments, Question Categorization, Post Edits, and Q&A Score are the top five topics.

We further categorize the topics into various categories and sub-categories. The review in this paper is given based on the categories. The categorization of topics was performed to group the related papers and explain them under the same heading. We used the C-Value terms and review of a paper for categorization of the paper. 214 papers are divided into 8 main categories, and each category is also divided into sub-categories. The categorization of papers is given in Table 2.

3. A REVIEW OF STACK OVERFLOW RESEARCH TRENDS

In this section review of 214 papers is given. We have divided the review into eight main categories. Each research category is an independent research area that is referred to as a research trend. In each category, a summary of relevant papers and tools and techniques used to solve the problem is given.

3.1 Using Stack Overflow Data for Tool Development

Out of 214 papers, 37 tools were created by researchers, which are discussed in 43 papers. The name of the tool with usage is given in Table A-3 (attached as supplementary material). Whereas, techniques and tools that are used by researchers are given in Table A-4 and Table A-5, respectively.

Table 1. C-value/NC-value

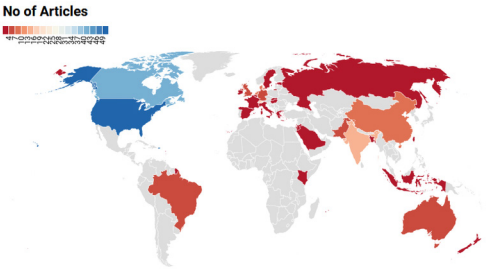
ID	Term	Tags
1	Stack Overflow	[NNP, NNP]
2	Code snippets	[NN, NNS]
3	Stack Exchange	[NNP, NNP]
4	Software development	[NN, NN]
5	Programming languages	[NN, NNS]
6	Software engineering	[NN, NN]
7	Knowledge units	[NN, NNS]
8	API usage	[NN, NN]
9	QA websites	[NN, NNS]
10	Stack Overflow posts	[NNP, NNP, NNS]
11	Stack Overflow discussions	[NNP, NNP, NNS]
12	Software developers	[NN, NNS]
13	Questions	[NNS, NNS]
14	API documentation	[NN, NN]
15	Research questions	[NN, NNS]

Ponzanelli and Luca Ponzanelli (2014) proposed a holistic recommender system for software

Table 2. Topic categorization

Category	Sub-Category	Topics
API Documentation	API Usage Patterns, Project Documentation, Crowd Documentation, API Misuse, Deficient Documentation	API Documentation, Software Engineering, CQA, Q&A Websites, API Type, API Usage, Programming Languages, Cookbook Preparation, Java Unsafe API, Project Documentation, Code Documentation, Baseline Results
Q&A Analysis	Unanswered, Deleted and Closed Questions, Data Set Creation, Questions, Answers, Comments, Code Snippets, Shared Links, Q&A Quality	Accepted Answers, SO Comments, Question Categorization, Post Edit, Code Snippet, Question Tags, Obsolete Answer Study, Migrated Questions, Data Set, Quality Prediction, Q&A Quality, Post Structure, Code Example, Q&A Edits and Quality, Mobile Development Questions, Tagging Posts, Post Features, Code Security, CQA Data, SE Data set, Tags Study, CQA Sites, Questions
Behaviour Study	User, Gender, Activity	User Behaviour, Software Developers, Gender Study, User Representative, User Activity, User Study, User Interaction, User Participation, Knowledge Sharing
Recommender System	Ranking, Prediction	Q&A Score, Question Ranking, Train Model, Prediction, Recommender System, Video Tutorial, Finding Source Code
Emotion Mining	-	Sentiment Analysis
Finding Experts	-	User Reputation, Expert Users, User Ranking
Tool Development	-	Code Generation, Query Feed, Stack Overflow, Market Platform, Tag Analysis
Topic Modeling	-	Topic Modeling, Topic Clustering

Figure 4. Country-wise distribution of papers [Created by using geolocation API by using country of first author]



development and maintenance activities. The proposed recommender system will get information from the SO, blogs, bug reports, e-mails, and other project artifacts and recommend the relevant literature to the developer in the IDE as a plugin. This will speed up the development process. Seahawk and Propter are also presented in this paper to support the concept. Seahawk is developed to integrate SO data in the Eclipse IDE. It allows users to retrieve Q&A, link relevant discussions, and attach comments. SO data is copied into SQLite and made available to developers offline. Bacchelli, Ponzanelli, and Lanza (2012) Data dump is converted into a relational database; ID, Question, Title, Answer, Tag, and Author fields are imported into the database. Seahawk is a recommender system that consists of a Recommendation Engine and Annotation Engine. Apache Solar is used to index questions & answers. Three experiments were performed to check the capabilities of the Seahawk. It is observed that it suggests useful information to the developer during development. Ponzanelli (2012); Ponzanelli, Bacchelli, and Lanza (2013a, 2013b).

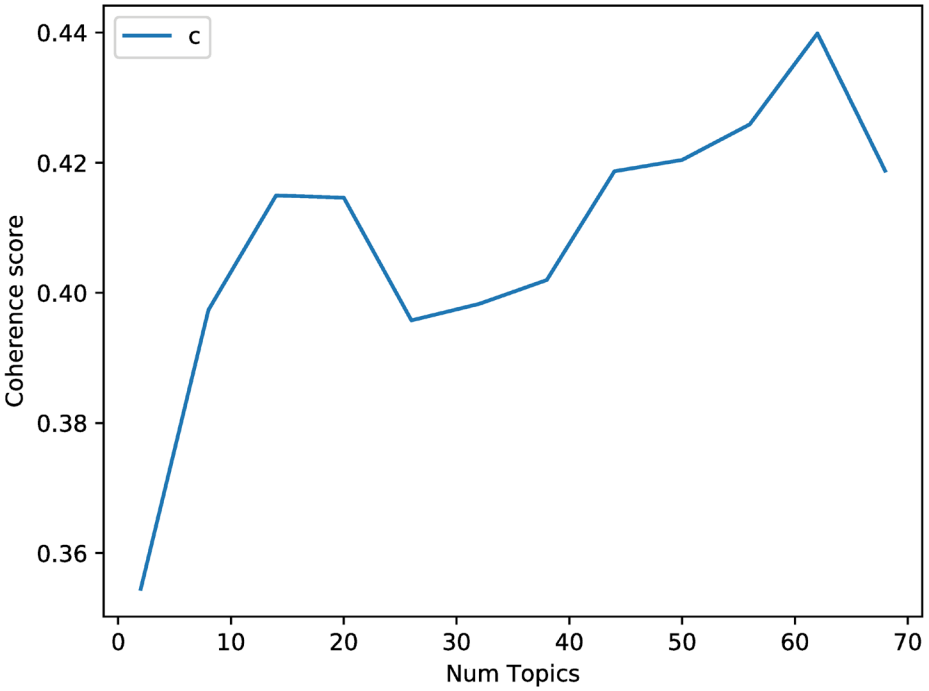
SODA is a tool to visualize the SO tag data. It helps to visualize and understand discussion trends, analysis of topics in different time intervals, and co-occurrence between the tags. Latorre, Minelli, Mocci, Ponzanelli, and Lanza (2015).

In another study, a recommender system, "Prompter" is developed that recommends SO discussing in the Eclipse IDE after analyzing the code. Related SO discussions are collected by Google and Bing search engines, and a ranking model is developed to rank the SO discussion. Text mining is used to stem code. Qualitative validation is performed on 2 studies, and study-I is repeated after one year to check the validity of the prompter. Various statistical methods are implemented to check the results. It is observed that prompter is useful in recommending code and related discussion and ease programming. After one year, its results are different but are still useful to help programmers. Ponzanelli, Bavota, Di Penta, Oliveto, and Lanza (2014, 2015); Ponzanelli, Bavota, Penta, Oliveto, and Lanza (2014).

A tool named CodeTube Ponzanelli et al. (2016) is developed that helps to extract fragments from video tutorials and related discussions on SO. Fragments can be searched by query only those fragments that include code are considered. A video crawler was developed to download videos from YouTube data API. Google2Str was used to extract audio transcripts of the video. Video Tutorial Analyzer extracts pieces of information to isolate video fragments related to specific topics. FFMPEG is used to extract frames. Tesseract-OCR is used to extract text from frames. BoofCV and Frame Segmentation are used to facilitate text extraction from complex frames. The textual similarity between frames is computed by using the Vector Space Model. Extracted video frames and SO discussion are indexed using Lucene. An assessment was made using questionnaires and interviews. It was observed that the CodeTube concept is very useful for experienced programmers.

OPAvion is a tool for mining and visualization of large graphs. It consists of three modules: summarization, anomaly detection, and visualization. Stack Overflow data is used to demonstrate the capabilities of this tool and to show that it can mine large data sets. Akoglu, Chau, Kang, Koutra, and Faloutsos (2012).

Figure 5. Number of topics determination



Confusing programming concepts can be identified by analyzing SO data. The authors used LDA for said purpose. A major finding is that the distribution of questions does not vary among languages. We can create tools to find the orthogonality of different languages, platforms, and tools. Allamanis and Sutton (2013).

The source code is retrieved from SO data to facilitate programmers. A tool is also created that recommends source code. SO data is modeled using Mallet and LDA. The snowball is used for stemming. Pre-processing was performed using a stored procedure. The cosine similarity of strings was also calculated to rank documents by the similarity of topics. Arwan, Rochimah, and Akbar (2015).

In another research, an Esclipe-based plugin, "SurfClipse" is developed to suggest error/exception solutions in the IDE. Data is taken from Google, Bing, and Yahoo Search engines and the Stack Overflow Q&A site. Data is ranked using 8 metrics. Evaluation is made by searching results of 25 errors, and an accuracy of 96% is achieved. In another user-based evaluation agreement of 64.28% is achieved. M. M. Rahman, Yeasmin, and Roy (2013).

Baker, a tool that is implemented as a browser extension, is used to link SO code snippets with the API documentation. Baker is highly precise (0.97) and supports Java and JavaScript. Barker develops AST from the code snippet and then uses the information from Oracle to deduce facts about AST. Eclipse is used to create AST for JavaBaker and JavaScript snippets are parsed by the ESPRIMA parser. Neo4j a graph database, is used to develop Java Oracle. A web service is created to automatically upload JAR files, analyze, and add to the oracle by a tool called Dependency Finder. Daniel German's list of signatures is used to bootstrap the oracle. JavaScript Oracle is built by parsing the source file of libraries by a parser ESPRIMA. Baker can be used to integrate source code into API documentation. Subramanian, Inozemtseva, and Holmes (2014).

A recommendation system is designed by using how-to posts of SO to recommend information that can assist development. Textual similarity and crowd evaluation are the criteria for recommendations. Logistic Regression is used to classify the Q&A pairs. Features for classification are selected using information gain and 5 attributes were selected. Weka was used for classification. The Apache Lucene search engine is used to index data. Qualitative manual analysis is used, and the criteria are relevance and reproducibility. L. B. L. de Souza, Campos, and Maia (2014).

In another research, SO data are analyzed, and the success of the SO reputation mechanism is studied to map it to bug tracking systems to enhance the quality, contribution, and extract value from bug tracking systems. It is observed that the game mechanism will help to make the bug-tracking system more effective. Stack Overflow API is used for downloading data, and correlation is used for the analysis of data. Lotufo, Passos, and Czarnecki (2012).

StORMeD is SO readymade data that is created to facilitate researchers. 70K Java discussions are used to construct the Heterogeneous Abstract Syntax Tree (H-AST). Island parsing is used to construct the HAST. The island parser is implemented in Scala using the parboiled2 parser. The tf vector is generated using Apache Lucene. StORMeD data dump is available online to facilitate researchers. Processing H-AST reduces the time of researchers. Ponzanelli, Mocci, and Lanza (2015a).

The number of tags increases the chances of getting answers and the number of views on the question. Therefore, there should be an automatic system for tagging the questions. To develop such a system, a discriminative model is developed to suggest tags, and the model is trained using SVM. The model is 68% accurate and can suggest tags and fill missing tags. The Mallet is used for preprocessing questions, and SVMlight is used for training the discriminative model. Saha, Saha, and Schneider (2013).

Authors Sim and Gallardo-Valencia (2013) analyzed SO from three perspectives, i.e., its design, its role in software documentation, and its uses in example-centric programming. 11 characteristics of SO design are discussed. Questions that are being asked and replied to on SO are studied to categorize the questions into various types. Coding is used for the categorization of questions, and justification is given for frequently answered question categories. A live system named as an example overflow is created. Data can be searched, for example overflow using keywords, and Apache Lucene is used, which internally uses tf-idf.

S. Martinez, Elming, and Johannsen (2013) studied regular expression to classify the documents because n-gram representation has many limitations. L1-regularized logistic regression is used in all experiments. Features, namely Bag of Words (BoW), BoW + Harvard Inquirer (BoW+HI), and BoW+Exp (Experts) are used in experiments. 7 researchers queried the database with a regular expression and 1156 RE were collected from SO and RateBeer data sets. 500 held-out examples annotated by three turkers using Amazon Mechanical Turk were also used. A sliding window of size 3 was used to extract trigrams from the annotations and these trigrams were converted into a regular expression. Experiments were performed on the baseline and proposed method. The proposed method resulted in 24 to 41% error reductions.

A tool to auto-comment code is developed. This tool finds similar code on the stack overflow and adds comments to the code. To check the validity of the tool, 23 Java and Android projects are commented. The evaluation is made in a user study where users were asked to rate the comments on accuracy, adequacy, conciseness, and usefulness. Wong, Yang, and Tan (2013).

Developers often express their opinions about APIs on different forums. These opinions are valuable for other developers who are trying to learn or use a particular API. Uddin and Khomh Uddin and Khomh (2019) developed a tool to automatically extract opinions about APIs and created a database of opinions. Opiner extracts opinion sentences from the discussion, associates these sentences with API mentions, and marks the API aspects like performance and usability.

Research on SO code snippets identified that there are various security-related issues with code snippets. A Heterogeneous Information Network (HIN) is created to model the metadata and relatedness with code snippets established by a meta-path-based approach. The Snippet2vec model

is created based on HIN and meta-path to establish the relationship between metadata and reduced dimensions. SO posts related to Java for Android application development were extracted using crawlers. More than 20,000 code snippets were labeled as secure and insecure. A multi-view fusion classifier using LibSVM is developed to classify the code snippets. The classifier is compared with DeepWalk, LINE, and metapth2vec. The classifier is also compared with NB and SVM. The developed method outperforms all the existing ones. Y. Ye et al. (2018).

Part of Speech (POS) tagging is domain-specific because the performance of the POS tagger decreases outside of the domain. To cater to this limitation and develop a POS tagger of the Software Engineering domain, a specific Part of Speech (S-POS) is developed. S-POS is trained on SO data. Annotated data, tag- sets, and tools are available online for further research. To speed up the annotation process, data was tagged by the Stanford POS tagger. The Linear Maximum Entropy Markov model is used to develop the S-POS tagger. The accuracy of S-POS is 5.8% higher than the Stanford POS tagger. D. Ye, Xing, Li, and Kapre (2016).

Silva, Paixao, and de Almeida Maia (2018) conducted a reproducibility study for duplicate question˜ detection on stack overflow. Two popular techniques, i.e., DupPredictor and Dupe for duplicate question detection, are re-implemented because their implementation is not available. New tools for duplicate question detection are named DupPredictorRep and DupeRep. The tools were tested on three data sets, including the original data sets on which DupPredictor and Dupe were developed. The recall rate is used to judge the performance of newly developed tools. The recall rate of both tools is reduced by 20.5% for DupPredictorRep and 28.5% for DupRep. It may be due to different toolsets used in the implementation of new tools.

A huge number of users are using knowledge contributed by some users here forth named Digital Heroes. In this thesis Abdur (2018), the author developed criteria for identifying heroes from stack overflow and by using data mining to identify the heroes. Selection criteria were developed after a survey with SO users. A web-based tool, Digital Hero Discovery (DHD) is developed to identify and keep a record of digital heroes. The tool performs the data collection, data cleaning, filtering, ranking, and classification functions and as a result, generates a list of digital heroes.

About 20-35% of questions on the Stack Exchange network are unanswered or answered late. Convertino, Zancanaro, Piccardi, and Ortega (2017) named these questions as Dark Side questions and proposed a tool that identifies such questions at the time of posting. The tool developed by the authors consists of a web-based interface for moderators and a logistic regression-based classifier. Classifier training was annotated by authors in five categories. The predictor was evaluated in a user study by comparing it with a naive (time-based) approach. The false Positive (FP) of the author's approach is 35.4% whereas, the FP of the naive approach is 57%.

Named Entity Recognition (NER) is an important task in any domain as it gives us valuable information about the corpus before reading it. But NER is domain-specific, and classifiers developed for NER do not work well for Software Engineering. To develop a software-specific NER (S-NER) Ye et al. developed a classifier based on machine learning. The methodology of the authors consists of five steps, i.e., data collection, pre-processing, tokenization, unsupervised word clustering, and Conditional Random Field (CRF) based supervised learning. 1520 SO posts were annotated by five annotators using the Brat tool. A custom tokenizer was developed. CRF++ ToolKit is used to train the classifier. Results are compared with the baseline method, and improvement in precision, recall, and f1-score is reported. D. Ye, Xing, Foo, et al. (2016).

3.2 Using Stack Overflow Data for Finding Experts

In this category, various papers are reviewed in which experts are identified from SO for various purposes like routing questions, finding potential answerers, profiling experts, route jobs, rewarding experts, etc. Techniques used to find experts are given in Table A-6, whereas tools used to find expert users are given in Table A-7.

The expertise of an individual can be mapped into different shapes (-, I, T, M). In this paper, graph partitioning is used to find the broad area of expertise. After creating a graph, the PageRank algorithm is applied. Expert Ranks within each area of expertise followed a power distribution law. Jenks Natural Breaks clustering is used to divide ExpertsRanks into High, Medium, and Low. Most of the SO users have hyphen shape expertise. V. Kumar and Pedanekar (2016).

There is a need to identify experts early before they leave the community. In this paper Pal, Harper, and Konstan (2012), question selection bias is used to identify experts. It is observed that experts select questions with less Existing Value (EV). Several models were proposed for the identification of experts, and predictions were made using SVM, Decision Trees, Gaussian Discriminant Analysis, and Bagging Meta Classifier. Factor analysis was used to reduce the dimensionality of features. It is observed that the Bagging meta classifier gives the best results. It is concluded that experts can be identified in the first month of their activity, but for more accurate results, activity of 3 months is required. SO and TurboTax data are used for experiments.

In another paper, a technique to find the behavior of users and find experts in private social network platforms & SO is proposed. The interaction of users on both platforms is studied, and similarity between both platforms is observed. Different metrics to find the expertise are also proposed. Experiments were performed on data from 8 months. Raj, Dey, and Gaonkar (2011).

One reason for not getting answers in the Q&A forum is not routing questions to experts. To solve this problem, a profile-based approach is proposed for routing questions to potential experts. Various approaches to create profiles from previous answers are implemented, and it is observed that Topic Modeling approaches (Latent Dirichlet Allocation (LDA), User Persona Model (UPM), and a Segmented Topic Model (STM)) outperformed word-based modeling approaches (TF-IDF, Language Models). STM gives better results as compared to LDA. "Finding expert users in community question answering" (2012); Riahi (2012).

In another research, the supervised LDA algorithm is extended as RankSLDA to model expertise. The proposed algorithm is used for question recommendations to experts. The proposed model is compared with various baseline models (Popularity Ranking, TF-IDF, Latent Semantic Indexing, and LDA-R). Several evaluation metrics (Precision at cut-off, Discounted Cumulative Gain, Mean Average Precision, and Mean Reciprocal Rank) are also proposed for analysis and comparison of the model. It is observed that RanksLDA outperforms as compared to all baseline models. San Pedro and Karatzoglou (2014).

Developer profiles on social media sites like SO, GitHub, etc. give useful insights about the activity and the ability of the developer. In this research Singer et al. (2013), members of developer profile aggregators are studied. Masterbranch and Coderwall are major profile aggregators. Surveys and interviews are used for data collection. Grounded theory is used to code and find categories from data. Reasons for participation, modes, and terms of interaction, the impact of participation, and risks & challenges of participation are highlighted in this study. It is observed that developers participate to learn and show their work. Some recruiters are also participating to find developers.

In another paper Singh and Shadbolt (2013), first features of the SO and Crowd Sourcing websites are discussed. Then SO data are annotated with the links from Wikipedia and Drupal data sets using Wikipedia Miner and Open Calais to resolve the name and topic disambiguation. This linkage provides additional information about the topic and helps in finding experts. It is observed that SO tags are not sufficient to find experts after linking experts in the cross-domain can be discovered.

Venkataramani, Gupta, Asadullah, Muddu, and Bhat (2013) developed a recommender system to find the expertise of SO users in a particular domain. This model captures the expertise of users based on the frequency of terms used in their code on GitHub; expertise is then mapped to tags of the Q&A community and finally, the expertise of users in the target domain is mapped. The proposed model is evaluated using 5000 SO questions and 20 GitHub projects. The system recommends questions to developers related to their expertise. It is observed that 7 out of 15 developers answered the question related to their expertise on SO.

An analysis of the SO reputation system is made to observe how users gain a reputation. A Random Forest classifier is used to predict the expert user. PageRank and Singular Value Decomposition are used to determine the expertise of the user and identify anomalous users. It is observed that users with high reputations ask more questions. The question of high-reputation users and their answers are mapped using a network diagram. The results are also compared with the results of Pal et al. (2012). It is concluded that expert users can be identified based on their first month's activity. Movshovitz-Attias, MovshovitzAttias, Steenkiste, and Faloutsos (2013)

In this paper Vasilescu, Serebrenik, and Van Den Brand (2013), linguistic diversity as a method is used to study the risks associated with Open Source Software (OSS) development and maintenance. OSS is mostly developed by using more than one language. A speaker of knowledge is required for the maintenance of OSS. The mutual intelligibility measure is introduced based on SO data. User-based tags from SO data are used to determine the similarity between programming languages by using association rule mining. The similarity between 160 languages was calculated. The proposed approach is validated by a case study of Emacs, a text editor, which is maintained by 446 users and developed by using 27 languages. It is observed that Emacs exhibits fewer risks because of the healthy number of Emacs Lips speakers. Vasilescu, Serebrenik, and Van Den Brand (2013)

Different features of the post and users were selected, and four experiments were conducted using Random Forest to find the best feature set for finding experts. View count and favorites are the measures to identify difficult questions with an accuracy of 95% and 0.9 f-measure. Cross-topic questions are more difficult as they require expertise from multiple domains. Similarly, finding an expert's best answer and difficult question is the best combination of features with an accuracy of more than 99% and an f-measure of 0.99. Bhanu and Chandra (2016)

L. Wu, Baggio, and Janssen (2015, 2016) analyzed 110 sites of the Stack Exchange network to study the types of users. Two types of users are identified and named as Type-A and Type-B users. Type-A users are doing simple tasks, whereas Type-B users are real knowledge producers. An Attention Network model is developed to model the growth of these communities, and it is observed that the number of Type-A users should be twice as large as Type-B users for the growth of the community.

Usually, the expertise of the user is determined by his reputation on the community sites. However, the current reputation mechanism is based on user activity, and the quality of content is given secondary importance. To tackle this, the reputation-calculating mechanism based on the quality and difficulty of contribution is proposed. The proposed method is compared with four baseline methods (original user reputation, best answer ratio, Z-score, and posted answers), including the SO reputation mechanism. The recall of the proposed method is higher as compared with baseline methods. Huna, Srba, and Bielikova (2016)

In this research Hanrahan, Convertino, and Nelson (2012), features of hard problems that are posted on stack overflow and features to find experts are developed. A question that took a longer duration to answer is difficult. Difficult questions are modeled using a state machine, and various events that occur after posting the question are captured. Experts are identified by reputation, Z-score, and average duration between up and downvotes.

Expert finding aims to recommend individuals who can answer questions effectively. Traditional methods rely on content similarity between questions and past answers, but they struggle with experts who haven't addressed similar questions. This study introduces the Recurrent Memory Reasoning Network (RMRN), which retrieves relevant historical data from experts by using perception and reasoning. A Gumbel-Softmax mechanism helps select pertinent records. Tested on large datasets from Stack Overflow and Yahoo! Answers, the RMRN outperforms existing methods Fu et al. (2020).

Finding the right expert and creating expert profiles are crucial tasks for companies, researchers, and job seekers, particularly with the rise of social networks. Expert finding focuses on identifying individuals with the necessary skills or knowledge, while expert profiling summarizes their expertise. A study explores how social tagging can enhance both processes by introducing a new expertise indicator based on tagging activity. Tags are used as indicators of interest to build candidate profiles,

and the Latent Dirichlet Allocation (LDA) algorithm is applied to analyze the distribution of tags across topics, utilizing their semantic relationships. This is further refined by filtering topics using tag depth, which serves as the expertise indicator. Experiments conducted on Stack Overflow data demonstrate the accuracy of this approach Kichou, Boussaid, and Meziane (2020).

3.3 Q&A Analysis/Mining

This category is further subdivided into eight categories, i.e. Unanswered, Deleted, and Closed Questions, Data Set Creation, Questions, Answers, Comments, Code Snippets, Shared Links, and, Q&A Quality.

Relevant papers of each sub-category are reviewed. Techniques used for Q&A Analysis are given in Table A-8 and Tools for Q&A analysis are given in Table A-9.

3.3.1 Analyzing Unanswered, Deleted, and Closed Questions

Lezina and Kuznetsov (2013) developed a classifier to predict closed questions at the time of posting. Random Forest, Support Vector Machine, Vowpal Wabbit, and the baseline model provided by Kaggle are used for prediction. The classification was carried out by using SciKit-Learn Random Forest and output is calculated by using posterior probability. User feature, post feature, and tag feature are used. Post features were calculated by using LDA and VW-varinfo, which is a component of Vowpal Wabbit.

The results were evaluated using a multi-class logarithmic loss metric.

Answers on SO become obsolete with time. In this research, authors H. Zhang, Wang, Chen, Zou, and Hassan (2019) identified the obsolete answers and observed that 58.4% answers are obsolete at the time of posting, 20.5% obsolete answers are updated, and answers with tags like node.js, Ajax, android, and objective-c are likely to become obsolete because of API evolution.

7.5% of SO questions are unanswered, and several unanswered questions have seen exponential growth in the past few years. In this paper Asaduzzaman, Mashiyat, Roy, and Schneider (2013) various reasons are identified for which a question remains unanswered. A classifier is also built using the WEKA toolkit to predict how long it will take a question to get the answer.

Deleted questions are categorized, and it is observed that it takes at least a month to get a delete vote for a poor-quality question. Therefore, an automated approach based on machine learning is proposed to predict the question that will be deleted with 66% accuracy. Linguistic inquiry and Word Count software are used to investigate the features of the question. Adaboost classifier gives the best performance. A confusion matrix is used to compare results. Correa and Sureka (2014)

In another research Correa and Sureka (2013), closed questions on SO are analyzed. It is observed that subjective questions contain highly valuable information. A machine learning algorithm is used to predict closed questions with an accuracy of 73%. Popular tags in closed questions are also given in the paper. Support Vector Machines, Naive Bayes, Logistic Regression, etc. are used. Stochastic GradientBoosted Trees gives the best performance. The confusion matrix is created for the evaluation.

3.3.2 Data Set Creation

A data set named SOTorrent is developed to study the evaluation of the contents of Stack Overflow posts. The version history of posts on the block level is maintained in SOTorrent. In this paper, the authors explain the creation process of SOTorrent and how 136 similarity metrics are used to create version histories of posts. The authors observed that on stack overflow, 86.6% of posts are edited after posting, and comments and post edits are co-related. Along with the SO post edit, the history link to GitHub repositories that are referred to in the posts is also maintained in the data set. Baltes, Dumani, Treude, and Diehl (2018); Baltes, Treude, and Diehl (2018)

In this research Z. Yao, Weld, Chen, and Sun (2018), StaQC a data set of questions and code is developed from the Stack Overflow. This data set contains more than 148K Python and 120K SQL questions. The data set is created by using a Bi-View Hierarchical Neural Network (BiV-HNN).

BiV-HNN predicts the standalone code from the accepted answers with multiple code blocks. BiV-HNN is trained on a manually labeled data set of Python and SQL How-to-do-it questions and answers. The accuracy of this model is 15% higher than baseline (select first, select all, Logistic Regression, Support Vector Machine) methods.

3.3.3 Mining and Analyzing Stack Overflow Questions

LDA and NLP are used for mining SO data after using a stemming algorithm. Issues faced by web developers in using client-side technologies are analyzed automatically and manually. To rank the issues manually, a metric is developed. Mixed-methods analysis is used. Topics are first categorized, then hot topics are identified. Temporal trends are identified, and web topics in mobile development along with the main technical challenges are identified. Bajaj, Pattabiraman, and Mesbah (2014)

SO questions are categorized, and a comparison of six classifiers is made. NB gives the best results out of multilayer Perceptron, Support Vector Machine, K-Nearest Neighbors, J4.8 Decision Tree, and Random Forests. Weka is used for classification. How to do it questions can be used to write a cookbook. Need to know could be integrated with the IDE using the plugin. Seeking something that could be used to generate state of the art about tools, books, and tutorials. Campos and de Almeida Maia (2014)

A classifier is developed using a Random forest and Support vector machine to automatically classify SO questions in various categories. Training data of 500 questions is used to train a classifier, and a random forest-based classifier can classify the Android posts into seven categories with 0.88 and 0.87 precision and recall, respectively. API usage is the most discussed category among Android developers. Beyer, Macho, Pinzger, and Di Penta (2018)

In an article Jin and Servant (2019), the relationship between question edits and the number of answers is studied. Four attributes of edits and their relationship with several answers are analyzed. It is concluded that the question owner more frequently edits the question body. However, edits made by other users get more answers.

In this paper Lal, Correa, and Sureka (2014), the temporal distribution of migrated questions is made, and machine learning is used to predict questions that will be migrated. Network analysis of migrated questions across sites of SE is made using Gephi. Reasons for the migration of questions are also identified. Mann-Whitney-Wilcoxon The RankSum test is used for comparing the reputation of migrated question owners. The reputation of migrated question owners is less; the migrated question is discussed more and is less popular. KNN, Decision Trees, AdaBoost, and Gaussian Naive Bayesian are used for prediction. Adaboost and Decision Tree give the best results, i.e., 73% accuracy.

Nasehi, Sillito, Maurer, and Burns (2012) identified various characteristics of Q&A and divided posts into different categories. It is identified that an explanation of the question is also important. Open coding is initially used to code attributes of questions/answers, and after that, focused coding is used.

Codes were distributed among categories for qualitative analysis.

In another paper Voß (2012), the mixed method is used to study the types of questions and which type of question gets answered. Open coding of questions (title and body) is performed, and the question is categorized into 10 categories. It is observed that review, conceptual, and how-to/novice questions are frequently answered.

In another study, tags of SE & categories of Wikipedia are organized in the Simple Knowledge Organization System (SKOS). Statistical analysis is used to compare computer science papers archived at arXiv.org in 2011 to maps with corresponding tags of SO. It is observed that folksonomies of tags or categories can be transformed into concept schemes in SKOS. The resulting knowledge is useful for retrieval of information, finding information, and bibliometric analysis.

Authors in D. Chang, Schiff, and Wu (2013) used four classifiers named Random Forest, Support Vector Machine, Logistic Regression, and K-Nearest Neighbors to find the features of the question that will be answered. The accuracy of the classifier was improved by adding the features of the author profile. Precision, recall, f-measure, and AUC are used for evaluation. The authors conclude

that these features alone are not good enough for prediction. Data must be carefully collected, and the questions must be ranked based on their difficulty to improve the accuracy.

A classification model to estimate the difficulty of the SO question is developed. The model used two types of features named pre-hoc (available when the question is posted) and post-hoc (available when the accepted answer is posted). Before the classification manual labeled 936 Java questions were randomly selected into basic, intermediate, and advanced categories. 10x classifiers were used to test the performance of each classifier. Using pre-hoc features, the accuracy of Random Forest is 67.6% and by using post-hoc features, accuracy is improved to 75.2%. Hassan et al. (2018)

3.3.4 Mining and Analyzing Stack Overflow Answers

Many questions on SO did not get the best answer. To cater to this, a machine learning model that works on shallow (statistical) and semantic (similarity) features is proposed. Python-related Q&A from the Stack Overflow data dump of 2016 is used in this research. The answer with the most upvotes is treated as the best answer. The machine learning model was compared with the baseline (first, last, and longest answer), and an accuracy of 8.52% was achieved. J.-W. Lin (2018)

In a paper Calefato, Lanubile, Marasciulo, and Novielli (2015), factors that increase the chance of an answer being accepted are studied. Time is the most important factor; other factors are presentation, effect, and reputation. The success probability is modeled using logistic regression using the Weka tool. SentiStrength is used for sentiment analysis Graßl and Fraser (2022). The statistical significance of each model is tested using Delong's test for correlated ROC curves using the pROC package for R. F-measure, precision, and recall are also calculated.

In another paper Lord, Sell, Bagirov, and Newman (2018), authors analyzed the SO data set to measure the time of the first answer posted and the accepted answer posted. The analysis is termed Survival Analysis. It is observed that the response rate of Python posts is greater than R, whereas the accepted answer rate of R is higher than Python.

In an article Ginsca and Popescu (2013), ranking of the answer is performed using the features from the user profile. It is observed that users with a more complete profile are likely to give better-quality answers. The tf-idf model is used for modeling user descriptions. Spearman correlation coefficients are used to report upvotes and downvotes. Various methods based on ranking Support Vector Machine (RSVM) are used. Normalized Discounted Cumulative Gain (nDCG) and Mean Reciprocal Rank (MRR) are used for evaluation. RSVM-User-Mixed gives the best results.

In a paper Gkotsis, Stepanyan, Pedrinaci, Domingue, and Liakata (2014), shallow linguistic (readability metrics) features are used to predict the best answer on 21 SE sites. Alternate Decision Trees are used, which are implemented in the Weka tool. Precision, recall, and F-Factor are used for evaluation.

Information gain is used to select the best features. 84% average precision and 70% recall are obtained.

Q&A on SO undergoes various edits to reach an optimal solution. Authors Diamantopoulos, Sifaki, and Symeonidis (2019) extracted the common edit patterns. Post table, PostHistory, and PostBlockVersion tables from SOTorrent data set for analysis. The analysis is performed on answers to Java questions. Answer text and code before and after edits were queried. The difference in posts was calculated by the Python difflib module. Text similarity was calculated by using tf.idf and snippet similarity was calculated by using the longest common sequence. Finally, hierarchical clustering was applied to group edit patterns. A technique based on deep learning is developed to rank the most suitable answer for a question. Three sets of features, i.e. (i) Q&A text (ii) user metadata (iii) Q&A text and user metadata, are used in experiments. Three data sets were created for experiments. One data set is used for training, and two data sets are used for testing. Q&A text features were selected by using doc2vector. The model with Q&A text and user metadata features gives the best result. S. Xu, Bennett, Hoogeveen, Lau, and Baldwin (2018)

In a research, a model named Context And Syntax Embedding (CASE) is a hybrid of the Recurrent Neural Network-Language Model (RNN-LM) and tag-answer co-occurrence model. CASE is trained on the QUASAR-S data set, which is a cloze-style question data set of Stack Overflow with background domain knowledge. CASE is compared with several baseline methods. Several variants of CASE were also developed to test the accuracy of CASE. CASE achieved accuracy gains over baseline methods and is effective in predicting cloze-style questions, even for the corpus of different domains, i.e., SPADES, where no background knowledge is available. The accuracy of CASE-BiGRU-CC + ft on the training set is 44% and the test set is 45%. Winston (n.d.); Winston, Dhingra, Mazaitis, Neubig, and Cohen (n.d.)

3.3.5 Mining and Analyzing Stack Overflow Comments

In an article Movshovitz-Attias and Cohen (2013), a natural language model using n-gram, LDA, and link-LDA is proposed to predict comments for Java classes. It is observed that comments prediction will save 47% of comments typing, which is a tedious activity. Three training data sets are used in the resurch study and it is observed that SO data gives better results as compared to data from other projects because more English words are used in this data.

Stack Overflow shows the top five comments along with the answer. These comments are ranked by the score. It is observed that 87% comments have zero scores; in 97.3% cases only one comment is hidden, which has the same score as the 5th comment. Therefore, the current ranking mechanism of comments should be changed to develop a new ranking system. Authors experiment with the creation date, similarity with a question, and length of comments. In case of a tie, comments should be ranked by their length. 70% of the comments (both hidden and displayed) have useful information. H. Zhang, Wang, Chen, and Hassan (2019)

3.3.6 Research on Stack Overflow Code Snippets

In an article M. Ahmad and Cinneide (2019), the effect of copying code on recipient class cohesion is´ studied. The SOTorrent data dump is used, and a random sample of 378 Java classes on GitHub that received code snippets from stack overflow is studied. Low-level similarity-based Class Cohesion (LSCC) and Class Cohesion (CC) metrics are used. Cohesion is calculated between a pre-SO and post-SO snapshot of classes. It is observed that in 70% of cases, copied snippets affect code cohesion, and in 70% of cases, lost cohesion is never gained.

Treude and Robillard (2017) conducted a web-based survey by collecting data from 321 participants to understand the usability of code fragments posted on stack overflow. Before the survey, text from posts was removed, and participants were asked to mark the understandability of 120 code fragments. The authors identified seven main issues that affect the understandability of code fragments. These issues are incompleteness, code quality, missing rationale, organization of code, naming issues, clutter, and missing domain information.

Squire and Funkhouser Squire and Funkhouser (2014) determined the percentage of code that is required in successful SO Q&A after performing various experiments. Code-to-text ratio is calculated to identify how much code is required, and Flesch-Kincaid Reading Ease (FKRE) is used to measure the readability of the posts. It is observed that 1:3 is an acceptable ratio for a good answer, and the question ratio is 1:8. 40-50 is an ideal range of FKRE for the best posts. The polarity of the best posts is also positive.

It is also observed that sentiment analysis has a limitation with vocabulary.

Source code in SO posts is often taken from tutorials. To study this, 600 Android tutorials were correlated with code snippets of SO posts by using the SOTorrent data set. The motive was to study the reasons for duplication of code and the evaluation of duplicate code over time. Big Query is used to extract data, and the code clone detection tool is used to find duplicate code in the tutorials and SO posts. Reasons for duplicating code were determined by qualitative analysis. The top reasons for

duplicating code are errors in executing code and unexpected behavior of code. Nishi, Ciborowska, and Damevski (2019)

Source code in SO posts is often used by developers in their projects. An exploratory study of 289 code files that belong to 182 open-source projects was conducted by authors in Y. Wu, Wang, Bezemer, and Inoue (2019). These code files have links to SO posts. In 31.5% of files, developers modified the code to fit into their programs. This modification is as simple as renaming variables and as complex as reimplementing the algorithms from scratch. To find the barriers associated with reuse of code, the authors surveyed 453 members of the SO. The top barriers are modification of code, incomprehensible code, and low-quality code.

A. Rahman, Farhana, and Imtiaz (2019) studied 44966 Python answers on SO and found that 7.1% posts include insecure code. Code injection is the most occurring insecure coding practice. 9.8% accepted answers also include insecure code. OpenStack insecure coding practices are used to find insecure code.

LDA is used to identify the words that are most likely to include insecure code.

Alrashedy (2018) use machine learning and NLP techniques to predict the programming language of code snippets posted with stack overflow posts. The prediction is based on three sets of features, i.e., (i) title, body, and code snippets; (ii) title and body; and (iii) code snippets. Random Forest and XGBoost classifiers were trained on all the feature sets. Maximum accuracy of 91.1% is achieved by the XGBoost classifier by using title, body, and code snippets features.

Tahir, Yamashita, Licorish, Dietrich, and Counsell (2018) used qualitative and quantitative techniques to identify the code smell and anti-pattern concepts, popularity among users, and corrective actions taken to remove smells. The authors found that most of the time developers share their code to detect its smell. SO is being used as a crowd-based platform for code smell detection, and developers rely on feedback rather than automated smell detection tools. 59% questions on code smells are related to C#, JavaScript, and Java languages.

A large-scale study is conducted in Ragkhitwetsagul, Krinke, Paixao, Bianco, and Oliveto (2019) by surveying 201 SO users and analyzing code snippets by using the code clone detection tool. It is observed that 65% of users encounter outdated code and 25% of them never fix it. 69% of users never check licensing information. Another SO user survey shows that 87 users found problems in SO posts like outdated code and incorrect and mismatched solutions. 85% users are unaware of the SO license agreement, and 66% never check licensing conflicts before using the code. 2289 clone pairs of Java extracted by Simian and SourcererCC were also analyzed; 153 of them were copied from the Qualitas project, and 66% of them were outdated. Ninka is used to extract license information; 214 code snippets that appeared in 2427 GitHub projects are violating the license of the original software.

Stack Overflow is helpful, but previous research shows that insecure cryptography code is being used in production systems that are borrowed from SO. To tackle this, Fischer et al. (2019) proposed a system based on Nude (a concept borrowed from economics) that is supported by code analysis developed using a deep neural network. The deep neural network learns the security API usage patterns and classifies the secure and insecure cryptography code on SO. The system achieved an AUC-ROC of 0.992. The system is tested in a controlled environment where users are tasked with implementing cryptography code. The participants using Nudge developed more secure systems.

Interest in combining natural language and computer code analysis has grown with the availability of large datasets like Stack Overflow's 15 million programming-related questions. However, NLP techniques for identifying code tokens and software-related entities within text are still lacking. To address this, we present a new NER corpus for programming, featuring 15,372 sentences annotated with 20 entity types. Our BERTOverflow model, trained on 152 million Stack Overflow sentences, improved F1 scores by +10 over standard BERT. Additionally, the SoftNER model, which incorporates a code token classifier, achieved a 79.10 F1 score on Stack Overflow data Tabassum, Maddela, Xu, and Ritter (2020).

3.3.7 Analyzing Shared Links in Posts

Developers share links on SO for innovation. Types of links shared, most frequent links shared, and the domain are identified in this paper. The official documentation is the most widely shared link. Carlos, Cleary, and Singer (2013)

Developers often share URLs in questions, answers, and comments. These URLs form a huge knowledge network. To study the motives of URL sharing, D. Ye, Xing, and Kapre (2016) performed a large-scale study by using qualitative and quantitative methods. Open coding is used to study the motives of developers, and the structure & dynamics of the knowledge network are studied by using qualitative methods. Internal URLs are shared most on SO. To further explore the motive of URL sharing, 1000 URLs were randomly selected, and open coding is performed by the source and destination of URLs. The authors found reference information for problem-solving and reference answers are the topmost motives of URL sharing.

3.4 Accessing Stack Overflow Post Quality

Ponzanelli, Mocci, Bacchelli, and Lanza (2014) classified SO questions as good and bad quality questions.

Readability metrics, author's popularity, and simple textual features of Stack Overflow are used. Classification is performed using decision trees and Genetic algorithms. It is observed that decision trees are good to classify only on the author's popularity. Stanford NLP Parser is used to remove code snippets from questions. JGAP is used to implement GA. Important features are identified that determine the quality of questions. A precision of 80 to 97 is achieved using GA. It is concluded that we can partially automate the identification of bad-quality questions. In another paper, Ponzanelli, Mocci, Bacchelli, Lanza, and Fullerton (2014) propose an approach to remove good-quality questions from the review queue, thus decreasing the workload of the moderators.

Votes alone do not tell us about the quality of the answer. Therefore, authors Omondiagbe, Licorish, and MacDonell (2019) used random forest and Neural network classifiers with various settings of features to determine the best feature set that can predict the accepted answer. It is observed that the length of code in the answer, the reputation of the user, the similarity of text between Q&A, and the time lag between question & answer can best predict the accepted and unaccepted answers.

Ellmann and Schnecke (2018) study different measures that will increase the quality of question and answer. A data dump from 2014 is used in this research. Error, discrepancy, and how-to question types are extracted from the data set for further research. Naive Bayes with six features, i.e. Flesh-Reading-Ease score, code-to-text ratio, word count, the image exists, the listing exists, a quote exists, and a number of tags are used to predict when a question will be answered or voted.

In another paper Y. Yao et al. (2013), a co-prediction method based on regression and classification is proposed to predict the quality of questions and answers. The purpose of this paper is not to predict the quality of individual questions or answers. The correlation between the quality of the question and answer is predicted. Features of question, answer, and author are used for this purpose. The model is trained on one million questions and 1.5 million answers. For the evaluation of model correlation, effectiveness, efficiency, RMSE, and prediction error are used. This method improves 13.13% prediction error.

In this paper G. Li, Zhu, Lu, Ding, and Gu (2015), the effect of collaborative editing on Q&A sites is calculated by proposing different models based on negative binomial regression. It is observed that editing the question title has a negative effect, editing the body does not have any effect, editing the tag has a positive effect, editing the answer has a negative effect, and editing comments has a negative effect on user motivation. But this negative effect is minimal as far as the overall quality of the content is concerned.

In another article Anwar, Afzal, Ahsan, Iltaf, and Maqbool (2023) determined the optimal set of features and developed a hybrid deep learning model that is based on CNN and LSTM to access

the quality of Stack Overflow post. The model is trained and tested on Stack Overflow posts. This work will help moderators to filter the bad quality post automatically.

In his thesis, Selleras Selleras (2020) developed a model to identify high-quality and low-quality answers on Stack Overflow. The model integrates partial least squares (PLS), natural language processing (NLP), and binomial logistic regression (BLR). Data was extracted from Stack Exchange using specific queries, focusing on six question features and five answer features. A regression equation was generated, and the model was tested on three Q&A pairs, achieving an 80% accuracy in predicting the optimal answer Selleras (2020).

This paper examines methods to enhance the effectiveness of Stack Overflow question posts in attracting answers. Through statistical data analysis and literature review, we identify three key factors common in previously successful or answerable questions. Based on these findings, we propose a prototype sidebar for the ”ask” page that dynamically (1) assesses the quality of questions as they are being composed, (2) displays answer previews from related questions, and (3) guides users in applying these key factors during the question development process Hsieh (2020).

Community Question and Answer (CQA) sites have grown to include vast quantities of questions and answers, making it impractical to review all responses on a given topic. To streamline this process, a method utilizing the Grey Wolf Optimizer is proposed. This approach employs the Biterm Topic Model to effectively summarize and identify key answers. This technique has proven effective in extracting essential information M. Li, Chen, Chen, and Wang (2020).

In a recent study, Roy and Sing Roy and Singh (2020) identified key features that predict whether a question will be closed, including question length, specific words or phrases, and the number of tags. They tested CNN and LSTM models and proposed a CNN-based approach to automatically learn these features from question text. Evaluating their method on a Stack Exchange dataset, they found it outperformed existing techniques for predicting closed questions, achieving over 80% accuracy. This approach can assist users in enhancing their questions before posting on CQA sites.

Social Question Answering sites (SQAs) enable users to ask questions and receive answers from the community, but these platforms often suffer from a rise in low-quality answers. Previous research has mainly focused on identifying high-quality answers among existing responses but has not addressed the influx of low-quality content. This study aims to address this by developing a system to detect and filter low-quality answers as they are posted. Using data from Stack Exchange, 26 features were extracted, and machine learning algorithms were applied to accurately identify 85% to 96% of low-quality answers. The key contribution is a real-time system that flags subpar answers during posting, providing an early warning to users about answer quality Roy, Ahmad, Singh, and Banerjee (2022).

In a recent study, Roy, Singh, and Banerjee (2022) identified key indicators of a question’s likelihood to be closed, such as question length, use of capital letters, and specific keywords. They proposed a method utilizing a Random Forest classifier to automatically detect these factors in question text. Evaluating their approach with Stack Exchange data, they achieved over 80% accuracy in predicting whether questions would be closed. Their analysis also revealed that factors like the number of tags and title length significantly influence a question’s closability.

3.5 Topic Modeling and Stack Overflow Data

In this category, a review of papers related to topic modeling is given. LDA is the most popular topic modeling technique, and Mallet is the most popular tool to apply LDA. A list of techniques and tools used for topic modeling is given in Table A-10 and Table A-11, respectively.

In a paper, Barua, Thomas, and Hassan (2014) created a topic set after applying LDA of SO data; the topic is a single part of a topic set. Code is removed before applying LDA as it cannot help in topic modeling. Topics of interest are identified in this research. The Mallet is used to apply LDA.

In this paper Linares-Vasquez, Dit, and Poshyvanyk (2013), an analysis of SO questions for developing´ mobile apps is made. LDA is used to find mobile development trends and issues faced

by developers. It is observed that most of the developers post answers only for one platform. Android and iOS are hot platforms for development. Accessing web content and graphics editing makes it hard to answer issues.

Entropy and the number of documents by dominant topic are used as metrics.

In another study Rosen and Shihab (2016), SO questions related to 3 mobile platforms (Android, iOS, and Windows Phone) are studied to find what mobile developers are asking, what issues they face, what are the most difficult questions they ask, and what common issues are across the different platforms they face. Pre-processing and LDA are applied using MALLET. It is observed that mobile related questions are increasing on SO. 38 question types are identified after manually categorizing the results of LDA. The API-related questions, device input, HTML5/Browser, location, and connectivity questions are most difficult to answer. Further research is carried out to find the popular question on each platform, and it is observed that App Distribution, Connectivity, Orientation, Tools, User Interface, and File Operations are the most popular Android questions. App distribution is the most popular iOS question, and Windows Phone developers are more focused on language and toolset. Android developers ask most of the why questions, Windows Phone developers ask most of the how questions, and iOS developers ask the most what questions. Cohen's Kappa and Chi-Square tests are also used to support analysis.

S. Wang, Lo, and Jiang (2013) explore the interaction between developers. SO data are automatically divided into 5 topics by using LDA. Labeling of the category is performed manually because LDA does not label topics into categories. The distribution of questions and developers is studied. Graphs are used to link developers with Q&A they ask and answer. The research framework is divided into data processing, help networks, and text mining modules. The findings of the research are that most developers ask only one question, most developers answer only one question, very few developers are answering many questions, the majority of developers are asking questions, and only a few developers are connected.

The available configuration of Latent Dirichlet Allocation does not work well for Stack Overflow and GitHub corpora. Treude et al. pre-configured corpora and created an optimum parameter set for LDA to get good results from the SO and GitHub. Data related to 8x most popular programming languages were extracted from SO and GitHub. 40x corpora were created for each repository. The Irace 2.3 tool is used to determine the optimum parameters. Random Forest is used to determine the best set of features that can be used to pre-configure the corpora for topic modeling. It is observed that a different set of features is optimal for SO and GitHub due to the difference in contents on both sites. Treude and Wagner (2018, 2019)

The authors Cline et al. (2018) developed a question retrieval system that is trained to retrieve Python posts and display the top 10 results to the user. Three techniques, K-Mean clustering, Topic Modeling, and the ensemble method are used to develop the retrieval system. The results are evaluated by using recall and user feedback. Recall of Non-Negative Matrix Factorization and ensemble method is 67% and K-Mean clustering is 50%.

On CQA sites, the response of users uses a vast vocabulary. It is difficult to retrieve the contents using a simple search. Different responses may be semantically the same but differ in vocabulary. To overcome this, the authors Zolaktaf, Riahi, Shafiei, and Milios (2011) developed a technique called QuestionAnswering Topic Model (QATM) to retrieve and rank data from stack overflow, which is based on statistical topic modeling. 15822 Q&A pairs were selected to train the model. The results of the technique are compared with LDA. The model was able to capture topic dependencies.

In a research study, all questions on stack overflow related to concurrency are collected. Topic modeling and clustering on questions are applied, and questions are divided into eight categories. Furthermore, the popularity of questions is calculated to find the real difficulties of programmers. Correctness and performance are the most discussed topics among concurrency developers. This work will help developers and educators to teach and learn the problem of concurrency development. S. Ahmed and Bagherzadeh (2018)

Software architecture is crucial for linking problem and solution domains in software systems, but traditional graphical tools do not perform in abstract analysis. To address this, Architecture Description Languages (ADLs) have been introduced as a more formal approach. However, based on feedback from practitioners and content analysis of forums and mailing lists, reveals that ADLs also face significant challenges. Authors identified 17 key issues that software engineers encounter and proposed solutions, contributing to the literature and offering guidance for future research Anwar, Bibi, Rana, Kadry, and Afzal (2024).

3.6 Behavior Study of Stack Overflow Users

User behavior and gender-specific studies are reviewed in this section. Modeling user behavior is very important on CQA sites because users are the main contributors. Relevant techniques and tools used to model and study user behavior are given in Tables A-12 and A-13, respectively.

User behavior is modeled in the presence of badges as an incentive to a user based on his activity. It is observed that in the presence of badges, user activity increases. The quality of user action changes as they are near the badge. Anderson, Huttenlocher, Kleinberg, and Leskovec (2013)

Personality traits of SO users are identified by using Linguistic Inquiry and Word Count (LIWC). Topreputed users are more extroverted, and authors of up-voted posts express significantly fewer negative emotions. The distribution of personality traits is also compared using ANOVA and Tukey's HSD test. Changes in personality with time can be researched in the future, and these results can be used to hire good managers. Bazelli, Hindle, and Stroulia (2013)

In this paper Grant and Betts (2013), SO data are analyzed to see the changes in the behavior of users by warding badges. Three different badges are analyzed, and the activity of the users is modeled when they are near to achieving and after achieving badges. It is observed that there is an increase in the activity of the user to achieve the particular badge.

Users are influenced by the badges of other community users and try to earn the same badges. Friends data is not available on SO; the connection between users was determined by posting on the same question. Paired T-test and generalized estimation equation model are used. GEE allows fitting repeated measure Logistic Regression. Halavais, Kwon, Havener, and Striker (2014)

Murgia and Alessandro Murgia (2016) experimented with the identification of duplicate questions and answering questions using a bot. A bot that is specialized to answer 50 git errors was designed and was allowed to answer the question on SO. In the first run (90 days), it emulates human identity and gets 187 reputation points. Its answers were also accepted by users. In the second run, it displayed a bot ID that received huge criticism from the users and was banned after 25 days. It is concluded from this experiment that humans do not like it when they are answered by machines or expect a high-quality answer from machines, and also that it is possible to use bots for the identification of duplicate questions.

In this research Schenk and Lungu (2013), the location of SO users is used to study the knowledge transfer between geographical regions. 20% of users who have 75% of reputation have provided location information in their profile. The Yahoo Geolocation API is used to enrich the original data set with additional geographical information per user. Data is visualized using the Quicksilver tool, which is part of the Moose analysis framework. It is observed that North America and Europe are the main contributors, whereas Asia is a strong importer of information.

Another research study, similarity between user interactions on SO based on the reputation of users, is studied. 3x data dumps of the SO are used in this time-based study. Firstly, characteristics that help to build the reputation of users are identified, and then users are grouped into categories based on reputation score. Finally, the similarity is calculated. Structural Equation Modeling is used to create three models for each year, and evaluation is made using the Comparative Fit Index, Root Mean Square Error of approximation, and Standardized Root Mean Square Residual. After that, a predictive model is created to predict user reputation using CKD, DQM, and PIS variables. Another

technique called the quadratic assignment procedure is used to make stochastic inferences based on reputation similarity and associated factors. T. Sinha, Wei, and Carley (2015)

V. S. Sinha, Mani, and Gupta (2013) studied the various sites of the Stack Exchange network to explore the content created by a limited number of users, longevity of users, earning reputation without participating in Q&A, and diffusion across various sites of SE. It is observed that a limited number of users are the main content creators. Users leave the site if their questions are not answered. Reputation can't be gained without Q&A. 20 to 80% users might have diffused from other forums of SE.

In another paper Treude and Filho (2012), various opportunities and challenges faced by social programmers are discussed. Programmers are collaborating from the beginning, but the evaluation of Web 2.0 has changed the method of collaboration. The SO has changed the search process, and just in time, programmer-like concepts have been introduced. Distributed and collaborative software development is easy now. SO also has an impact on education.

An article Vasilescu, Capiluppi, and Serebrenik (2013) is an extension of the author's previous work Vasilescu, Capiluppi, and Serebrenik (2012). In this study, the findings of the previous paper are reconfirmed, and two more open-source project communities (WordPress and Drupal) users are studied for gender-specific participation and engagement. Gender is recognized automatically based on the user names and location data. A tool for gender identification is available online, but where it is not possible to identify the gender, manual analysis is used to identify the gender of a user. The data of mailing lists were analyzed by using the MLstats tool. Statistical analysis is used to analyze the data; ANOVA, KruskalWallis one-way analysis, t-test, Wilcoxon-Mann-Whitney test, and multiple contrast test procedures are used. A pilot survey was also conducted to evaluate the accuracy of the gender identification process. It is observed that WordPress and Drupal communities are more female-friendly. Females ask more questions, but answers provided by females are close to the number of questions provided by males. Males engage for a longer duration. It is also observed the SO community is not a female-friendly community and females disengage early.

The association between software development and crowdsourcing is studied by exploring GitHub and Stack Overflow. The SO data dump of Aug-12 is studied. GitHub data is downloaded through GHTorrent in MongoDB format. GitHub and SO users are linked by email addresses; as a result, onequarter of users were linked, but only active users were studied in this research. Data were analyzed using ANOVA, Kruskal-Wallis, Wilcoxon-Mann-Whitney test, and Bonferroni correction. The results of the study are: active Github committers ask a few questions and answer more questions; more active SO answerers make more commits on GitHub; and SO activity accelerates GitHub committing. Vasilescu, Filkov, and Serebrenik (2013)

CQA sites are changing knowledge sharing; this hypothesis is analyzed in this research Vasilescu, Serebrenik, Devanbu, and Filkov (2014). R-help mailing list and two StackExchange sites (Stack Overflow and Cross-Validation) are studied. Common users in R-help and SE are identified by identity merging. R developers are categorized into core developers, package maintainers, and users. Zawinski's algorithm is used to get distinct threads from R-help. The findings are given after a user survey and analysis of r-help and SE data. It is observed that the activity of R-help decreases after 2010, developers are more active on SE sites, and R experts answer the question quickly on SE. R-help contributors are more active on SE due to gamification features.

In a research Beeharry and Ganoo (2018), the survey data set of stack overflow (2017) is analyzed to see the trends, job-seeking status, and plans of developers. 525 male and 525 female respondents' data are analyzed. It is observed that developers are moving to big data and cloud platforms. The jobseeking status is predicted by using a random forest model with 87.64% accuracy and it is observed that developers prefer to stay in their current company.

Stanford Social Network Analysis Platform (SNAP) is used to create the network graph of SO tags, and the HITS algorithm is used to find the top 10 experts and learners. Three case studies on programming languages, operating systems, and web development topics were conducted. It is

observed that expert users are experts on multiple topics, i.e., they are top answerers across multiple tags, whereas learners are asking about only one tag/topic. Paul, Tripathi, and Tewari (2018)

SO encourage the users to revise answers to improve the quality of content. But revising answers has a positive and negative impact. To study the behavior of users and the impact of revising answers, this research was conducted. 3.8 million answers were analyzed by the authors. Statistical, quantitative, and qualitative analysis is performed. It is observed that revision activity increases on the badge awarding day, and 25% of users stop revising answers after receiving their first revision badge. The user's behavior is steered by the revision badges. Most of the revisions are rolled back. There is a need to change the badge awarding mechanism to maintain the quality and quantity of revisions. S. Wang, Chen, and Hassan (2018)

In research May, Wachs, and Hannak (2019), the gender difference in the SO community is studied.´ Sample data of equally balanced gender of 20000 SO users with reputations greater than 100 is selected. The gender of the user is predicted by using the genderComputer tool. Interaction between the users is analyzed by using the network analysis Louvain algorithm. Oaxaca-Blinder decomposition is applied to study the success rate of men and women. It is observed that men gain 50% more reputation as compared to women. Men answer more questions and cast more votes. The female asks more questions. The questions of females get more upvotes. The main source of female reputation is questioned they ask. The reputation mechanism should be changed to increase the participation of females in a community.

In their series of articles, Xu et al. studied the career paths and behavior of developers. A linkage between the activity of developers before and after finding a new job is determined. Data from SO and Stack Overflow Careers is used. It is observed that developers tried to gain more reputation before finding a new job. In 2014, after finding new jobs, online contribution decreased by 20.3% L. Xu, Nian, and Cabral (2014). In 2015, a 25% reduction in reputation gaining and an 8% reduction in other activities was observed L. Xu, Nian, and Cabral (n.d.). In 2016, after finding the job, their activity to gain a reputation on SO was reduced by 25%. Whereas, other activities of the developers that are not linked with reputation are also reduced by 7% L. Xu, Nian, and Cabral (2016). In 2018 and 2019, 23.7% and 7.4% less generation in reputation and non-reputation activity is observed L. Xu, Nian, and Cabral (2019); L. Xu, Nian, Cabral, et al. (2018). The authors tested the hypothesis and related the reduction in answers and edits and found that the reduction in answers is due to career concerns and time availability, whereas the reduction in edits is due to time availability. Another reason for the reduction in user activity is a skill mismatch, as the user has to learn new technologies that are required in the job.

Western countries contribute most of the knowledge on SO, whereas eastern countries contribute less and consume more knowledge. To study this mismatch, Oliveira, Muller, Andrade, and Reinecke (2018) performed Value-Sensitive analysis and interviewed SO users from the US, China, and India. SO design was studied by using SO tours and blog posts posted by company employees. Productivity, Niceness, Quality, Reputation, and Ranking are values embedded in SO design. To determine the values that users expect from SO 25x, participants were interviewed. The authors found that SO design is more focused and less social due to which US users are comfortable, and it does not affect their contributions. Asian users are not comfortable with focused and individualist values; rather, Asians are more social, and this hinders them from sharing more on SO.

Sun, Ghosh, Sharma, and Kuttal (2018) studied the flocking and migration behavior of 12578 SO and GitHub users by using Network analysis. The community of users on SO (answerer and commenter of the same question) and GitHub (editor of the same file) was identified by using the Louvain method. Migration behavior was determined by viewing past projects. 7.5% and 8.7% users made flocks on GitHub and SO, respectively. The average migration rate on GitHub is 5% and 2% on SO.

In this research, Ahmed and Srivastava T. Ahmed and Srivastava (2017) studied the behavior of users on stack overflow. The reasons for the decreased activity of high-reputation users are identified

through user surveys. Professionalism and ethical issues are also highlighted in this research. Spatial analysis reveals that users preferred to answer questions that are posted from the same region (i.e., country, state, and city). To find the reasons for fewer activities, authors sent emails to various SO users and, based on the feedback of users, designed a questionnaire. The response helped the authors to categorize the reasons for less activity on SO. The topmost reasons are that users find SO less attractive, duplicate questions, and the quality of questions. The authors observed users answer questions of the same gender most of the time and found answering as a gender-free activity. Many users are posting links to their sites to get traffic.

3.7 Stack Overflow for API Documentation

This sub-section is further divided into five sub-sections, i.e., project documentation, crowd documentation, API misuse, API usage patterns, and deficient documentation. The technique used for API documentation is given in Table A-14, and tools related to API documentation are given in Table A-15.

3.7.1 Stack Overflow for Project Documentation

In this paper Campbell, Zhang, Xu, Hindle, and Miller (2013), deficient areas of project documentation are identified by applying LDA to project documentation and SO questions. The Java SAX parser is used to locate SO questions. ScalaNLP API is used for preprocessing data sets. The CVB0 algorithm implemented in the Stanford Topic Modeling Toolbox is used for LDA.

In another paper, Dagenais and Robillard (2010) researched the creation and evaluation process of documentation. Interviews with the contributors and users are conducted. Documentation is analyzed, and grounded theory is used. SO data is used to find users of projects who were interviewed. The manual process is used to select experts.

3.7.2 Stack Overflow for Crowd Documentation

The research Jiau and Yang (2012) confirmed the inequality of crowd-sourced API documentation. A technique to manage the inequality of documentation based on the object-oriented concept of inheritance is also proposed. Crowd documentation has sufficient coverage for practical usage. Inequality of resource allocation is severe. The Pareto principle is used to measure inequality. Concepts are collected from the API reference documentation, and Q&A are mapped to concepts. Traceability of concept and Q&A is maintained. Mean and standard deviation are used to measure quality.

Joorabchi, English, and Mahdi (2016) proposed that SO data can be used for Educational Data Mining (EDM). The author used Wikipedia as a dictionary and collected all posts that have links to Wikipedia pages. The author develops a tool to get the category and sub-category of Wikipedia articles that were referred to in the question/answer. After that, a graph of Wikipedia categories and SO questions was built using Gephi. The top topic on Wikipedia based on question count was identified, and manual analysis is proposed to use these topics in the classroom.

In another research Parnin and Treude (2011), the authors measure the coverage of API documentation on social media like blogs, SO, and other such sites. It is concluded that 87% of the API documentation is shared on social media. Google is used as a search tool, and the top 10 results for each API method are used for analysis by using coding.

Parnin, Treude, Grammel, and Storey (2012) analyzed SO Q&A to check the API documentation. The API documentation of Android, GWT, and Java is evaluated. API usage data are collected through Google Code Search. Traceability between API methods and SO documentation was maintained. It is observed that SO data cover 87% of documentation, but few classes are heavily used and few are neglected. Furthermore, the documentation process at SO is slow.

In the study L. B. de Souza, Campos, and Maia (2014), SO data are utilized to extract cookbooks for three popular APIs (SWT, STL, and LINQ). How To Do It types of questions are used for the creation of cookbooks. LDA implementation of Mallet is used for the extraction of topics from posts.

Q&A pairs that match a topic are used to organize chapters of the cookbook. Rules defined by authors and Q&A ranking by the crowd are used to select Q&A pairs. Cookbooks are organized on the linked pages. Experimental results show that 63-71% chapters are well defined in meaning, 72-88% recipes are related to the chapter, and 65-75% pairs met properties of appropriateness, self-containment, and reproducibility of snippets.

The knowledge posted on stack overflow is used to create a cookbook because API documentation lacks many examples. The cookbook for three libraries, i.e., SWT, LINQ, and QT, is created automatically. How to do it questions related to API were extracted from the SO data dump by using a rule-based binary classifier. LDA is used to determine the chapters and contents of the chapters. Authors rank the potential posts that will be included in the cookbook by giving a manual threshold value. An online survey was used to check the quality of the cookbook. Souza et al. (2019)

SO is utilized to design the intermediate programming course in Spain. 126 students were used in the experiments. The main source of programming concepts is SO. This experiment gives excellent results with a 22% increase in language understanding, 16% improvement in the correct operation of the system, and 53% better communication & documentation. Lopez-Nores et al. (2019)

Online Q&A communities are vital for knowledge sharing, but sustaining user participation is a challenge, especially among inactive users. To address this, we analyzed the contribution patterns of active Stack Overflow users over the past eleven years, using social cognitive and social exchange theories. Our findings show that knowledge reciprocation and social interaction boost contributions, while knowledge seeking among active users has a negative impact. Peer recognition and repudiation have mixed effects. This study offers insights to encourage greater participation in online Q&A platforms Mustafa, Zhang, and Naveed (2023).

3.7.3 API Misuse on Stack Overflow

Programmers consult SO posts to learn new APIs. But the API code on SO is misused. To check this, the authors T. Zhang, Upadhyaya, Reinhardt, Rajan, and Kim (2018) designed a framework ExampleCheck to extract patterns from 385K Java repositories, analyzed 21K SO posts, and observed that 31% post has API usage violations. Database, crypto, and networking APIs are often misused. Demonstration and architecture of ExampleCheck are given in Reinhardt, Zhang, Mathur, and Kim (2018).

In the research W. Wang and Godfrey (2013), authors studied Android and iOS posts to find the obstacles that developers face while using APIs. A list of APIs was collected from Android and iOS official API documentation. After that, obstacles relevant to a particular API were collected by using regular expressions. Furthermore, topic modeling is applied to each obstacle, and it was observed that several repeated scenarios occur while using APIs.

3.7.4 Studing API Usage Patterns on Stack Overflow

In an article Kavaler et al. (2013), API usage of Android applications and SO Q&A are analyzed to find what programmers want to know and why. The following are studied in this paper: The relationship between classes and the mentions of these classes in SO. Relationship between volume of documentation and number of questions on SO. The number of available answers depends upon the use of a class and the volume of available documentation. Android code is converted to a human-readable format by using APKTool. Negative binomial regression is used. Demand for class documentation from SO grows slower than class usage in code. Class documentation size and relationship with question mentions on SO. Class documentation size has a strong positive relationship with getting the first answer on SO.

In another paper Mastrangelo et al. (2015), uses of the Java unsafe library are given in various JVMs. Java is a safe language, but it allows use of unsafe for various situations. Unsafe API is not documented. SO data is used to answer only one research question to get insight into what people

are discussing about unsafe on Q&A sites. SO posts are parsed using island grammars, and finally, a manual analysis is used to identify exact uses of unsafe APIs.

Stack Overflow post shows code as plain text, making code search difficult because the same method name is used in more than one API / class. In this research Subramanian and Holmes (2013), this problem is addressed by parsing SO posts and developing Abstract Syntax Trees. Oracles of the API are also developed to search methods that belong to APIs. Code snippets from the Android API of size greater than 2 LOC are used in experiments. A snippet analysis framework is developed, and Eclipse Tool is used, which provides the Document Object Model to represent AST. Code snippets are wrapped with class and method declarations before parsing. The proposed approach was compared with SO search, and it is observed that the proposed approach decreases misreported results by 51%.

When faced with unfamiliar APIs, developers often rely on tutorials for basic understanding and Stack Overflow (SO) for task-specific guidance. However, extracting relevant information from these sources can be challenging due to irrelevant content and language differences. This article introduces PLAN, a system that automatically retrieves API knowledge from tutorials and SO using natural language queries. PLAN identifies key instructional and SO Q&A pairs, merging them into comprehensive API knowledge datasets. By leveraging a deep-transfer-metric-learning-based model, PLAN assesses relevance and suggests alternative APIs to bridge the lexical gap between queries and knowledge pairs. Evaluations on Java and Android datasets demonstrate PLAN's effectiveness, outperforming previous methods, as further confirmed by a user study D. Wu, Jing, Zhang, Feng, et al. (2023).

Developers often use API tutorials and Stack Overflow (SO) to learn unfamiliar APIs. Tutorials are divided into fragments, each explaining different aspects of API usage. Identifying relevant fragments is crucial for better API understanding, but existing methods are either labor-intensive or fail to consider relevance effectively. To address this, we propose SO2RT, an approach that automatically links SO posts to tutorial fragments using a semi-supervised transfer learning model. This method reduces the need for manual labeling and enhances API learning. Evaluated on Java and Android datasets, SO2RT outperforms existing techniques and proves effective in a user study, also demonstrating its utility in API recommendation D. Wu, Jing, Zhang, Zhou, and Xu (2023).

Mobile computing has become a huge part of our everyday lives and work, with major companies pouring resources into developing mobile platforms. For developers, finding the right APIs is essential, but the process can often be tedious and time-consuming. To help with this, authors created RAMC (Recommendation APIs for Mobile Computing), an API recommendation system designed to make development easier and more efficient. RAMC uses word embedding techniques and pulls data from Stack Overflow posts and Java core packages to suggest the most relevant APIs. It also provides developers with useful labels, similar questions, and related code examples. To test how well RAMC works, we ran a simulation using a common mobile development issue, and the system effectively recommended the right APIs and relevant tags for the developers Wen, Zhang, Hu, Zhu, and Wang (2024).

3.7.5 Deficient Documentation Identification

In the article by Sushine et al., problems related to API protocol are studied in two parts. First SO data is studied to find various problems related to API protocols. Questions are filtered to select relevant questions only and are divided into 13 topics. This study gives an idea about what is hard to find and why it is hard. Another study was conducted in the lab with programmers. Programmers were given protocol-related tasks and were monitored with think-aloud. The various methods used by programmers to implement the tasks were monitored. After that, open coding of the programmer's thought was performed, and four common categories were identified. It is observed that programmers spent most of their time searching the API documentation. API documentation does not fulfill the information needs of developers, and protocol-specific documentation is required to be maintained. Sushine, Herbsleb, and Aldrich (2015)

In another paper Treude and Robillard (2016), API documentation is augmented by using various machine learning algorithms. SO data related to Java APIs were extracted using SO API. Extracted data were manually annotated by authors, and 1574 meaningful sentences were collected. Stanford NLP parser is used to automatically tag all code elements as a noun. LexRank, Update Summarization, and Knowledge Patterns algorithms were implemented, and these approaches were compared with a supervised insight sentence extractor (SISE). SISE uses different string features. WEKA's StringToWordVector is used to turn the text into a separate feature for a single word, bigrams, and trigrams. Different machine learning algorithms, i.e., K-Nearest Neighbor, Decision Trees, Naive Bayes, Random Forest, and Support Vector Machine algorithms, implemented in WEKA are tested for the classification of development set text. Support vector machines give the best results to balance precision and code coverage at the information gain thresholds of 0.02 and 0.03. To validate the proposed algorithm, a comparative study was performed in which practitioners were asked to rank the sentences extracted by different algorithms. It was analyzed that 70% of sentences extracted by SISE are meaningful, whereas 43% and 35% are meaningful sentences extracted by MMR and LexRank, respectively.

3.8 Developing Recommender System

In this sub-section, various recommender and ranking systems are discussed, which are created using the SO data set. Techniques and tools used to create these recommender systems are given in Table A-16 and Table A-17.

In S. Chang and Pal (2013), a question routing strategy is proposed to get high-value answers. Key factors for the value of the question-answer thread are identified. The number of answers, average score, and number of comments in 1st hour are used to model linear regression to learn the weight of all the explanatory variables. LDA is used on tags to find topics. Random Forest, Sorting, and SVM are used to predict when users will come online. Graphs are used to model the collaboration between users. Precision and recall are used to evaluate the different models. The spectral clustering model outperforms the LDA.

In another paper Stanley and Byrne (2013), an ACT-R-inspired Bayesian probabilistic model is proposed to suggest tags for SO posts. Both the body and title are used to suggest tags. The hypothesis for prediction is based on the tag history of the user and the association between the tag and post (title & body). The Python Natural Language Processing Toolkit is used to tokenize and lemmatize the post. The scaled entropy measure is used to identify stop words. Logistic regression is used to tune the attentional weighting. The proposed model achieved 65% accuracy, which is quite high as compared with previous research.

Maity et al. (2019) proposed a deep learning-based content-cum-user method to recommend tags. This method is based on the contents of the question (title and body) and the relationship between the user and the tag. Node2Vec is used to extract network features. The data set of 0.5 million posts is used for training and testing. The proposed technique is compared with six baseline methods, and a gain of 60.8% and 36.8% in precision and recall over the TagCombine method is achieved.

StackInTheFlow, a recommender system, is developed to recommend the SO posts to developers when there is an error or there is no progress in coding for a long time. Based on the behavior of the developer, StackInTheFlow learns the tags over time and personalizes the display of results for the developer. StackInTheFlow is implemented as an IntelliJ Java IDE. The system generates a search query based on the similarity of keywords extracted from code or error trace. After querying, the top four results are recommended to the developer. StackInTheFlow was evaluated based on the usage log from different users. It is observed that automatic queries gain more attention from users. Greco (2018); Greco, Haden, and Damevski (2018)

Tags in SO posts categorize the posts on various subjects/domains. But, in many cases, tags do not always represent the post and are noisy. A recommender system, EnTagRec, is proposed, which proposes tags to post. This system learns tags from historical software objects and is implemented

using Bayesian (Labeled-LDA) and Frequentist. These models output probability scores for each tag, and the final tag is determined by a weighted sum of probability scores. The system is evaluated on 4x data sets, and EnTagRec achieved 27.3% improvements over the TagCombine method. S. Wang, Lo, Vasilescu, and Serebrenik (2014) EnTagRec++ is an advanced version of EnTagRec; it also uses user information and a set of tags that a user uses. S. Wang, Lo, Vasilescu, and Serebrenik (2018)

An approach based on topic modeling and collaborative voting is used to determine answers to new questions. Topic modeling (LDA) is used to determine the user interest, and collaborative voting is used to determine the expertise of the user. User interest and expertise are stored as a user profile. When a new question is posted, its contents are matched with user profiles, and a ranked list of users is predicted as potential answerers. This technique is compared with TF.IDF, and 21.5% improvement in prediction is reported. Tian, Kochhar, Lim, Zhu, and Lo (2013)

This research focuses on identifying the most important features, gaps, and improvement areas in development tools by analyzing feedback from software engineers on platforms like StackOverflow. By using techniques from Big Data, Data Mining, Deep Learning, and Transformers, the study aims to categorize and evaluate user feedback. The findings will help organizations prioritize feature development and address user concerns, ultimately aiding in the creation of more effective development tools Anwar and Afzal (2024).

Efforts to enhance software developers' learning include integrating GitHub into courses, developing collaborative platforms, and using gamification. These initiatives have shown promise in improving practical learning and collaboration skills. However, traditional books often fall short in covering practical aspects in depth. To address this, we propose an application that enhances the reading experience by using latent Dirichlet allocation (LDA) and a generative pre-trained transformer (GPT) to predict and retrieve relevant real-world issues from Stack Overflow. This tool bridges the gap between theory and practice, and user studies show positive feedback on its effectiveness in improving learning Anwar, Afzal, and Iltaf (2024).

Software developers often use Stack Overflow to find code examples, as it's more efficient than consulting documentation or tutorials. However, the vast amount of information can be overwhelming, making it difficult to find relevant code snippets. To address this, we present Que2Code, a tool that recommends the best code snippets from Stack Overflow based on user queries. It works in two stages: first, it retrieves semantically equivalent questions by generating paraphrases of the query; second, it ranks and recommends the best code snippets from the retrieved posts. Our evaluations on Python and Java datasets, along with a human study, confirm the effectiveness of Que2Code. Authors also released code and data for further research Gao et al. (2023).

In recent years, websites like Facebook and Wikipedia have thrived thanks to user participation and content creation. But with the massive amount of posts and users, it's impossible for administrators or moderators to manually review everything, so scalable solutions are essential. This research explores the growing need for automated tools to manage user-generated content using cutting-edge machine learning and data science techniques. It introduces new ideas for building a recommendation system that helps users and moderators tackle outdated content, share high-quality questions, speed up moderation tasks, and improve the overall quality of the platform Annamoradnejad and Habibi (2023).

3.9 Mining Emotions From Stack Overflow Data

In this sub-section, papers related to sentiment analysis are given. Techniques and tools used for emotion mining are given in Table A-18 and Table A-19, respectively.

In a paper Novielli, Calefato, and Lanubile (2015), various tools for sentiment analysis are listed to check the suitability of tools. Various aspects of sentiment analysis are also given, and polarity is most widely used. Various challenges of applying sentiment analysis are also highlighted. The SentiStrength tool is used for sentiment analysis. The authors have identified that various effective states are expressed in SO data, and it is proposed that sentiment analysis tools are domain-specific; they don't give desired results when used outside the domain.

Existing sentiment analysis tools are not suitable for the SE domain and produce negative results. To overcome this, the authors B. Lin et al. (2018) developed a technique for sentiment analysis for software library recommender systems. The authors extracted 40k sentences from stack overflow and manually labeled them into positive, neutral, and negative. Recursive Neural Network (RNN) was used by the author to compute the sentiment of sentences. The newly developed technique is named Stanford CoreNLP-SO. The proposed technique was applied to three different SE data sets named Stack Overflow discussion, mobile app reviews, and JIRA issue comments. The comparison of the technique was made with four state-of-the-art techniques named SentiStrength, NLTK, Stanford CoreNLP, and SentiStrength-SE. After the extensive training, authors observed the negative results. The results of all the techniques are comparable but are not satisfactory to be used to recommend the APIs.

The purpose of Islam and Zibran (2018) research is to investigate which tool is best for the SE domain and to what extent these tools agree or disagree with each other. JIRA issue comments, Stack Overflow posts, and code review comment data sets were created by authors, and three sentiment analysis tools named SentiStrength-SE, Senti4SD, and EmoTxt were tested. SentiStrength-SE achieved the highest accuracy on the JIRA data set, whereas Senti4SD has high accuracy on the Stack overflow data set. The overall average accuracy of SentiStrength-SE is 65.26% on the code review data set.

In the SE domain, sentiment classifiers trained on a general-purpose dataset have a bias of classifying neutral sentences as negative. A balanced Gold standard dataset consisting of 4423 posts was developed manually. A Semantic model was created by using 20 million stack overflow posts. The authors tried to overcome this by creating an SVM-based classifier using lexicon-based, keyword-based, and semantic features using the R interface. 25x features ranked by information gain were selected. Senti4SD was compared with SentiStrength and SentiStrength-SE. The authors observe 19% improvements in precision for negative and 25% improvements in recall for neutral. Calefato, Lanubile, Maiorano, and Novielli (2018)

In this research H. Yin and Pfahl (2017b), the authors extracted functional requirements from crowdsourcing sites and classified requirements into the Kano model. The proposed approach consists of two steps. In the first step, authors identified and extracted text as a functional or dysfunctional requirement using machine learning, and in the second step, authors applied sentiment analysis on requirements using a dictionary-based method. The data set is taken from the App Store and Stack Overflow. A subset of collected data was manually labeled by the authors into functional and dysfunctional requirements. MaxEnt, Na¨ıve Bayes, Support Vector Machines, and Decision Tree algorithms were used for classification. The authors used a confusion matrix for evaluation and achieved an accuracy of 65% using the proposed approach. Functional predictive value and dysfunctional predictive value were also calculated. MaxEnt and SVM algorithms give the best results. The probability Proportional to Size (PPS) method is used to check the accuracy of the first step by choosing 20% of classified text and manually checking the accuracy. The accuracy of sentiment analysis was 81%.

Detecting the sentiment of a document is an important and valuable task. As sentiment analysis is domain-specific, therefore, a sentiment classifier Senti4SD trained on SO is used in this research to classify the sentiments as per document type and programming language. Data about 8 popular programming languages were collected. 1000 posts for each programming language were randomly selected for sentiment classification. From the classification results, one can observe that most of the questions are negative, most of the answers are neutral, and an equal number of comments and questions are positive in the selected data set. Question sentiments of programming languages follow the pattern of baseline. C++ & C languages have more negative questions as compared to other languages. Ling and Larsen´ (2018)

Sentiment analysis, especially when adapted to specific domains, is an essential tool for understanding user opinions. However, current research in the software engineering (SE) domain often shows inconsistent results across different datasets. To address this issue, this paper introduces a

new approach using transfer learning-based classifiers that are fine-tuned for various SE datasets. The study benchmarks both machine learning and deep learning classifiers, revealing that transfer learning models like GPT and BERT outperform traditional methods. The BERT large model, in particular, achieves an F1-score of 0.89 on the Stack Overflow dataset, exceeding current state-of-the-art tools. This research offers valuable insights and sets the stage for more accurate, domain-specific sentiment analysis tools in software engineering Anwar, Afzal, Al-Shehari, et al. (2024).

Software engineers often share their opinions on social media, and analyzing these opinions through sentiment analysis can provide valuable insights into their feedback on various tools and topics. However, general-purpose sentiment analysis tools, typically trained on movie and review datasets, do not perform well in the software engineering (SE) domain. While efforts have been made to develop SEspecific sentiment analysis tools, existing models struggle with accurately detecting negative and neutral sentiments across different datasets. This study introduces a hybrid approach combining deep learning with a fine-tuned BERT model, creating several variants (Bert-Base, Bert-Large, Bert-LSTM, Bert-GRU, and Bert-CNN) and integrating them using fuzzy logic into a ”Fuzzy Ensemble” model. The model is tested on four benchmark datasets—Stack Overflow, JavaLib, Jira, and Code Review—using various evaluation metrics. When compared with state-of-the-art SE sentiment analysis tools like SentiStrengthSE, Senti4SD, SentiCR, and a fine-tuned GPT model, the Fuzzy Ensemble outperforms them, achieving a maximum F1-score of 0.883 and effectively addressing the limitations of existing tools, especially in predicting neutral sentiments across diverse datasets Anwar, Afzal, Altaf, Kadry, and Kim (2024).

J. Wu, Ye, and Zhou (2021) highlighted that most sentiment analysis tools, typically trained on movie and product review datasets, perform poorly in the software engineering (SE) domain. To address this, the authors fine-tuned the BERT Base model (BERT-FT) using GitHub and Stack Overflow datasets. They compared BERT-FT’s performance with two existing sentiment analysis tools, SentiCR and Senti4SD, and found that BERT-FT outperformed both. For the comparison, BERT-FT was independently fine-tuned on six different datasets to generate the results.

Obaidi, Nagel, Specht, and Klunder (2022) carried out an extensive systematic mapping study to ex-¨ plore and categorize sentiment analysis tools that have been specifically developed for the software engineering domain. Their research aimed to provide a comprehensive overview of existing tools, examining their methodologies, application areas, and effectiveness within the context of software development. This study serves as a valuable resource for understanding the current landscape of sentiment analysis tools in software engineering, highlighting gaps and potential areas for future research.

Batra, Punn, Sonbhadra, and Agarwal (2021) identified significant opportunities for enhancing sentiment analysis tools in the software engineering domain due to their low F1-scores. They trained three BERT variants—BERT-Base, RoBERTa, and ALBERT—on datasets from GitHub, Jira, and Stack Overflow, and introduced a ranking-based ensemble approach of these BERT models. Additionally, they incorporated DistilBERT, a lighter version of BERT, into their approach. Their proposed model demonstrated improvements of 6 to 12% across all datasets. However, the study has limitations, including the fact that no single model performed best across all datasets and the absence of comparisons with existing sentiment analysis tools specific to the SE domain. The highest F1-score achieved by their best model was 0.85.

In the study by Biswas, Karabulut, Pollock, and Vijay-Shanker (2020), the BERT Base model was trained on 55,000 manually annotated Stack Overflow sentences. The performance of this model was evaluated against the authors’ earlier models, RNN4SentiSE and SentiMoji. The single-sentence BERT model achieved an F-measure of 0.87. However, the study did not include comparisons with other sentiment analysis tools specifically designed for the software engineering domain.

Open source software development is a collaborative effort involving developers, organizations, and stakeholders, which can impact the health of projects. Issues within these communities, known as ”community smells,” can indicate organizational and social problems that lead to increased costs

and reduced software quality. To address this, we propose a novel method for detecting community smells by analyzing organizational, social, and emotional factors. Our approach uses a multi-label learning model to identify 10 types of community smells. We employ an ensemble classifier chain (ECC) with genetic programming (GP) to develop optimal detection rules. Tested on 143 open source projects, our method achieved an average F-measure of 93%, outperforming existing techniques. We also examine key community metrics that influence the detection of each smell type Almarimi, Ouni, Chouchen, and Mkaouer (2022).

Social media reviews significantly impact restaurant choices and business success. To analyze the growing volume of reviews, automated Sentiment Analysis (SA) is essential. This study introduces a hybrid approach combining Support Vector Machine (SVM) with Particle Swarm Optimization (PSO) and oversampling techniques to address data imbalance in sentiment prediction. Using data from Jeeran, an Arabic review platform, our method optimizes feature weights and balances review sentiments, improving analysis accuracy Obiedat et al. (2022).

Sentiment analysis offers a way to gauge team mood in software projects by evaluating text-based communication for positive, negative, or neutral sentiments. Despite its advancements, current tools still struggle with issues such as misclassifications and the inability to detect nuances like irony or sarcasm. Moreover, most tools provide analysis only after the fact, making it challenging to make real-time adjustments to communication. This paper proposes a solution for real-time sentiment analysis in software projects and assesses its effectiveness through a study with twelve practitioners. The study examines how integrating real-time sentiment analysis into developers' daily routines could enhance communication and workflow. While a long-term evaluation is needed, preliminary results indicate that real-time analysis holds promise for improving team interactions Schroth, Obaidi, Specht, and Klunder (2022).¨

In recent research, Uddin, Gueh´ enuc, Khomh, and Roy (2022) examined several sentiment analysis´ tools specific to software engineering and found that none of them performed adequately on their own. To address this, they developed an ensemble-based sentiment analysis tool combining five existing tools with RoBERTa. The predictions from these tools are fed into RoBERTa, resulting in an F1-score of 0.805 for the ensemble model, compared to an F1-score of 0.801 for RoBERTa used alone.

Mula, Kumar, Murthy, and Krishna (2022) conducted experiments with different word embeddings, feature selection methods, class balancing techniques, and deep learning classifiers to determine the most effective combination for sentiment analysis in software engineering text. They found that using the SMOTE sampling technique, ANOVA feature selection, and a deep neural network with one dropout layer and two hidden layers significantly enhanced the accuracy of sentiment analysis tasks.

Current sentiment analysis tools primarily focus on text data, overlooking the significance of verbal communication, which is common in developer meetings. To address this, SEnti-Analyzer Herrmann, Obaidi, and Klunder (2022) was developed, a tool that effectively analyzes both text and verbal commu-¨ nication, achieving an accuracy rate of 88% to 90%.

In recent research, B. Wang, Zhang, Du, Gao, and Li (2023) introduced a transformer-based multimodal framework called Prototype Augmented Multimodal Teacher Student Network (PAMD) for sentiment analysis. This framework is particularly effective in situations where limited training data is available, and there are missing modalities. The model was trained on three datasets, two of which are not related to software engineering, while the third is the JavaLib dataset. The model was tested with varying rates of missing data, achieving a maximum accuracy of 0.829 and an F1-Score of 0.579. However, the results for the JavaLib dataset were not reported by the authors.

In a recent study, Mark and Andy Swillus and Zaidman (2023) analyzed software testing-related posts on Stack Overflow, categorizing them as positive or negative using two existing sentiment analysis tools, RoBERTa and SentiCR. The authors then applied grounded theory to further classify these posts into different categories to assess the emotions of software engineers towards software testing. The findings revealed that emotions such as insecurity, despair, and aspiration significantly

influence software engineers' attitudes, particularly regarding the complexity of projects involving testing.

The social dynamics of software projects are gaining more attention in research and practice, with sentiment analysis tools being used to assess a team's mood through text-based communication. These tools rely on pre-labeled datasets from sources like GitHub and Stack Overflow. This paper explores whether the sentiment labels in these datasets align with how potential software team members perceive them. Through an international survey with 94 participants, the study compares participants' perceptions with the pre-labeled data. The findings reveal three key insights: (1) while the median perception aligns with the predefined labels in 62.5% of cases, there is significant variation in individual participants' ratings; (2) no participant completely agrees with the predefined labels; and (3) datasets labeled using guidelines are more accurate than those labeled ad hoc Herrmann, Obaidi, Chazette, and Klunder (2022).¨

4. TECHNIQUES USED BY RESEARCHERS

Different techniques mentioned by researchers in their publications are summarized in Table 3, categorized into 15 distinct groups for clarity and understanding. This categorization not only highlights the popularity of certain methods but also provides insights into the evolving research trends in mining Stack Overflow data.

- Topic Modeling: The dominance of LDA (Latent Dirichlet Allocation) as a topic modeling technique reflects its robustness in uncovering hidden structures within large text datasets. The continued reliance on LDA suggests a need for further exploration of more advanced or hybrid topic modeling techniques that could provide deeper insights or improve computational efficiency.
- Machine Learning: The frequent use of SVM, Logistic Regression, Decision Tree, and Random Forest underscores the importance of these methods in handling structured data and making predictive analyses. The field might benefit from a shift towards more sophisticated ensemble methods or integrating machine learning with deep learning techniques to improve accuracy and adaptability in various contexts.
- Deep Learning: The preference for RNNs among deep learning techniques indicates their effectiveness in sequence prediction tasks. However, the lack of specificity in some papers regarding the exact deep learning methods suggests a potential gap in the reporting of methodological details, which could hinder reproducibility and the advancement of knowledge in the field. Future research could focus on comparing different deep learning architectures or developing new models tailored to the unique challenges of Stack Overflow data.
- Statistical Methods: The widespread use of Correlation points to the reliance on statistical methods for establishing relationships between variables. However, as the complexity of datasets increases, there might be a growing need for more sophisticated statistical techniques, such as causal inference models, to draw more meaningful conclusions.
- Qualitative Methods: The frequent use of surveys, interviews, and open-coding reflects a strong interest in understanding user behavior and motivations. However, the field could benefit from more innovative qualitative methods, such as ethnographic studies or longitudinal analyses, which could provide deeper insights into user engagement and content evolution on Stack Overflow.
- Search Techniques: The prevalence of Google as a search tool highlights the need for efficient and accessible information retrieval methods. Future work could explore the development of domainspecific search engines or enhanced query formulation techniques to improve the relevance and accuracy of search results.
- Recommender Systems: The lack of specificity in some papers regarding the recommender systems used suggests an opportunity for future research to focus on the design and evaluation

of more transparent and interpretable recommender algorithms, which could better serve the diverse needs of Stack Overflow users.

- Network Analysis: The variety of techniques used in network analysis points to the complexity of social interactions and knowledge dissemination on Stack Overflow. There is potential for future studies to leverage advanced network analysis methods, such as dynamic network modeling or multi-layer networks, to capture the evolving nature of user interactions over time.
- Sentiment Analysis: The unspecified sentiment analysis techniques in several publications indicate a gap in methodological transparency. This gap could be addressed by future research that rigorously compares different sentiment analysis approaches, including lexicon-based, machine learning-based, and hybrid methods, to determine their efficacy in the context of Stack Overflow. Clustering: The use of clustering techniques in several studies suggests an ongoing interest in grouping similar content or users. However, there is room for improvement in reporting the specific clustering methods used and exploring more advanced clustering techniques, such as hierarchical or density-based clustering, to uncover deeper patterns in the data.
- Vector Space Model (VSM): The identification of multiple VSM techniques reflects the importance of text representation in Stack Overflow research. Future work could investigate the integration of VSM with deep learning-based embedding techniques to improve the semantic understanding of user-generated content.
- Code Analysis: The various techniques identified for code analysis highlight the technical depth required to understand and improve code-related discussions on Stack Overflow. There is potential for future research to develop more automated and scalable code analysis tools, particularly those that leverage machine learning or deep learning for more accurate and efficient processing.
- Sampling: The mention of different sampling techniques indicates the challenges in dealing with large datasets. Future studies could explore the implications of different sampling strategies on the generalizability of findings and develop guidelines for best practices in sampling from usergenerated content.
- Data Analysis: The widespread use of qualitative, quantitative, and statistical analysis methods suggests a balanced approach to understanding Stack Overflow data. However, as the field evolves, there may be a growing need for more integrated analytical frameworks that combine these approaches to provide a more holistic understanding of the data.
- Other Techniques: The inclusion of rarely used or uncategorized techniques under ”Other” reflects the diversity of approaches in the field. As research progresses, there may be opportunities to standardize or refine these techniques to enhance their applicability and impact.

5. TOOLS USED BY RESEARCHERS

The tools identified in the selected papers for addressing research problems, data preprocessing, data analysis, and other tasks are categorized into seventeen distinct groups, as shown in Table **??**. These categories cluster tools with similar functionalities, enabling researchers to make informed choices when tackling specific challenges. For example, Mallet Toolbox and Gensim emerge as the preferred tools for topic modeling due to their efficiency and adaptability in handling large datasets.

Data preprocessing tools are further subdivided into five categories: Stemmer, Natural Language Processing (NLP), Parser, Code Processing, and HTML Tools. The Snowball Stemmer stands out as the most utilized tool for stemming, while the Eclipse Island parser is frequently chosen for parsing tasks. Stanford NLTK is recognized as a leading NLP toolkit, highlighting its versatility in processing natural language data.

In machine learning, ten different tools and libraries are prevalent across the studies, with Weka being

Table 3. Techniques used for mining stack overflow dataset

Category	Techniques	Techniques	Techniques
Topic Modeling	LDA Allamanis and Sutton (2013); Anwar, Afzal, and Iltaf (2024); Anwar, Bibi, et al. (2024); Arwan et al. (2015); Kichou et al. (2020)	n-gram MovshovitzAttias and Cohen (2013)	link-LDA Movshovitz-Attias and Cohen (2013)
	SVD Movshovitz-Attias et al. (2013)	STM Riahi (2012)	RankSLDA San Pedro and Karatzoglou (2014)
	TF-IDF San Pedro and Karatzoglou (2014)	LSI San Pedro and Karatzoglou (2014)	LDA-R San Pedro and Karatzoglou (2014)
	Labeled-LDA S. Wang et al. (2014)	Topic modeling S. Ahmed and Bagherzadeh (2018)	
Machine Learning	Logistic Regression Anderson, Huttenlocher, Kleinberg, and Leskovec (2012)	SVM Campos and de Almeida Maia (2014)	K-Nearest Neighbors Campos and de Almeida Maia (2014)
	Decision Tree Campos and de Almeida Maia (2014)	Random Forests Campos and de Almeida Maia (2014)	Linear Regression S. Chang and Pal (2013)
	Multilayer Perceptron Campos and de Almeida Maia (2014)	Stochastic GradientBoosted Trees Correa and Sureka (2013)	Bagging meta Classifier Pal et al. (2012)
	AdaBoost Lal et al. (2014)	Naive Bayes Campos and de Almeida Maia (2014)	Conditional Random Field D. Ye, Xing, Foo, et al. (2016)
Deep Learning	Deep Learning Maity et al. (2019)	CNN Anwar et al. (2023); B. Xu, Shirani, Lo, and Alipour (2018)	RNN Anwar et al. (2023); B. Lin et al. (2018)
	Bi-View Hierarchical Neural Network Z. Yao et al. (2018)	CNN+LSTM Anwar et al. (2023)	Transfer Learning Anwar and Afzal (2024); Anwar, Afzal, Al-Shehari, et al. (2024); Anwar, Afzal, Altaf, et al. (2024); Anwar, Afzal, and Iltaf (2024)
Statistical	Correlation Anderson et al. (2012)	ANOVA Bazelli et al. (2013)	Tukey's HSD test Bazelli et al. (2013)
	Generalized Estimating Equation Model Halavais et al. (2014)	AIC Kavaler et al. (2013)	Gaussian Discriminant Analysis Pal et al. (2012)
	Structural Equation Modeling T. Sinha et al. (2015)	Flesch-Kincaid Reading Ease (FKRE) Squire and Funkhouser (2014)	Bayesian probabilistic model Stanley and Byrne (2013)
	Hypothesis Testing L. Xu et al. (2014)	Open card sort S. Ahmed and Bagherzadeh (2018)	

continued on following page

Table 3. Continued

Category	Techniques	Techniques	Techniques
Data Analysis / Methods	Focus group May et al. (2019)	Grounded Theory Anwar, Bibi, et al. (2024); Dagenais and Robillard (2010)	Open-coding Nasehi et al. (2012)
	Interviews Singer et al. (2013)	Survey Singer et al. (2013)	Exploratory Study Y. Wu et al. (2019)
	Systematic Mapping Fontao˜ et al. (2018)	Quantitative Analysis H. Zhang, Wang, Chen, Zou, and Hassan (2019)	Statistical analysis Gujral et al. (2018)
	Trend Analysis M. Ahmad and Cinneide (2019´	Principal Component Analysis Treude and Wagner (2019)	Spatial Analysis T. Ahmed and Srivastava (2017)
Recommender / Ranking	Recommender system Annamoradnejad and Habibi (2023); Anwar and Afzal (2024); Anwar, Afzal, and Iltaf (2024); Ponzanelli (2014); Wen et al. (2024)	Popularity Ranking San Pedro and Karatzoglou (2014)	Collaborative Voting Tian et al. (2013)
Network	Network analysis Bosu et al. (2013)	Graph Partitioning V. Kumar and Pedanekar (2016)	PageRank V. Kumar and Pedanekar (2016)
	LexRank Ponzanelli, Mocci, and Lanza (2015b)	HITS Algorithm Paul et al. (2018)	
Category	**Techniques**	**Techniques**	**Techniques**
Opinion	Sentiment Analysis Anwar, Afzal, Al-Shehari, et al. (2024); Anwar, Afzal, Altaf, et al. (2024); Calefato et al. (2015)		
Clustering Techniques	Spectral clustering S. Chang and Pal (2013)	Jenks Natural Breaks clustering V. Kumar and Pedanekar (2016)	Clustering S. Ahmed and Bagherzadeh (2018)
Vector Space Model (VSM)	VSM Ponzanelli et al. (2016)	Node2Vec Maity et al. (2019)	doc2vector S. Xu et al. (2018)
	word2vector S. Xu et al. (2018)	snippet2vec Y. Ye et al. (2018)	
Code Analysis	Program Slicing T. Zhang et al. (2018)	Code Clone Detection Nishi et al. (2019)	Code description matching Wong et al. (2013)
	Versioning Baltes, Treude, and Diehl (2018)	H-AST Ponzanelli, Mocci, and Lanza (2015a)	
Sampling	Random stratified Beyer et al. (2018)	SMOTE Omondiagbe et al. (2019)	ADASYN Omondiagbe et al. (2019)
	Homogenous Purposive Abdur (2018)	Random Subspace Hassan et al. (2018)	
Other	Genetic Algorithms (GA) Ponzanelli, Mocci, Bacchelli, and Lanza (2014)	User Persona Model (UPM) Riahi (2012)	Dependency Finder Subramanian et al. (2014)
	Simple Knowledge Organization System (SKOS) Voß (2012)	BIDE T. Zhang et al. (2018)	Query Generation Greco (2018)

the most commonly employed for implementing various algorithms, and SciKit-Learn being a favored library for machine learning applications. The Stack Overflow (SO) API is the most widely used tool for data extraction from the SO site, underscoring its utility in research.

Eight tools are highlighted for their role in data analysis, reflecting the diversity of methods used in the field. Five development frameworks are also listed, indicating the tools that support the coding and implementation processes in research projects. Tools that do not fit neatly into the established categories, due to their unique or niche applications, are grouped under the ”Others” category. This categorization not only streamlines tool selection for researchers but also underscores the evolving landscape of research tools and the need for adaptability in future studies.

6. RESEARCH FINDINGS

During the review of papers, we tagged each paper in reference management software. Once the review of all papers is complete, we perform expedition analysis on tags, and as a result, 12 significant topics were extracted based on tags. We labeled these tags and plotted the topic against each year. This analysis helps us to find the year-wise trend of topics as given in Figure 6. In the early years of SO, most of the research is done on reputation-related topics. From 2013 to 2018, machine learning algorithms were extensively used on SO data sets. Most of the gender-specific studies were conducted in 2018. The most dominant research is on reputation, machine learning, and Q&A analysis-related topics.

6.1 Tools Development

Various tools have been developed by researchers. These tools can be used for various purposes and are helping to solve problems. SODA Latorre et al. (2015) is a tool to visualize data. It is helpful to understand discussion trends and analyze discussion topics in different time intervals. Rattigan and Jensen (2010) developed a tool to find statistical dependencies in rational data. A tool for auto-tagging Saha et al. (2013) of posts is developed. Similarly, EnTagRec S. Wang et al. (2014) and EnTagRec++ S. Wang, Lo, et al. (2018) were developed to recommend tags to the users. 27.3% Improvements over the TagCombine method were reported. Simple Knowledge Organization System (SKOS) Voß (2012) was developed by using SO tags and Wiki categories. Folksonomies of tags & categories can be transformed into a concept scheme in SKOS. A tool for document classification S et al. (2013) was developed which reduces classification error 24-41% as compared to existing tools. The efficiency of code retrieval was improved to 51% by using snippet analysis framework Subramanian and Holmes (2013). Wong et al. (2013) developed a tool to document code and improve code readability. To increase the performance of programmers a tool to recommend SO posts in the IDE was developed Greco (2018); Greco et al. (2018). CASE Winston et al. (n.d.) was developed to answer a cloze-style question and this tool has 63.3% better accuracy as compared with the baseline BiGRU method. A ranking system to identify digital heroes Abdur (2018) was developed. This tool can be used to acknowledge the contribution of heroes. Ye et al developed two domain-specific tools. One is software-specific POS tagger D. Ye, Xing, Li, and Kapre (2016) which is 5.8% accurate as compared with Stanford POS tagger. The other tool is Software Specific Named Entity Recognizer (S-NER) D. Ye, Xing, Foo, et al. (2016) for named entity recognition. Sim and Gallardo-Valencia (2013) studied the different elements of SO design and created a tool like SO.

6.2 Finding Experts

The expert finding is a quite mature research area because many techniques to find experts are already developed. The basic goal is to find experts early and retain them in the community by rewarding them. A limited number of users are the main content contributors V. S. Sinha et al. (2013). The work of Movshovitz-Attias et al. (2013) show that experts can be identified based on their first month’s activity in the community. Expert users have many similarities T. Sinha et al. (2015). Reputation-based personality traits are identified by Bazelli et al. (2013).Pal et al. (2012) technique,

Table 4. Summary of tools used for mining stack overflow dataset

Category	Tools		Count	Citations
Topic Modeling	MALLET Toolbox, Stanford TM Toolbox, Gensim, Vowpal Wabbit		20	S. Ahmed and Bagherzadeh (2018); Allamanis and Sutton (2013); Alrashedy (2018); Anwar, Afzal, and Iltaf (2024); Arwan et al. (2015); Barua et al. (2014); Campbell et al. (2013); Cline et al. (2018); L. B. de Souza et al. (2014); "Finding expert users in community question answering" (2012); Gujral et al. (2018); Kichou et al. (2020); Lezina and Kuznetsov (2013); A. Rahman et al. (2019); Riahi (2012); Rosen and Shihab (2016); Saha et al. (2013); Silva et al. (2018); Souza et al. (2019); W. Wang and Godfrey (2013)
Stemmer	Snowball Stemmer, Porter Stemmer		12	S. Ahmed and Bagherzadeh (2018); Allamanis and Sutton (2013); Arwan et al. (2015); Gujral et al. (2018); Hassan et al. (2018); Omondiagbe et al. (2019); Ponzanelli (2014); Ponzanelli, Bavota, Di Penta, et al. (2014); Ponzanelli, Bavota, et al. (2015); Ponzanelli et al. (2016,?); Ponzanelli, Bavota, Penta, et al. (2014); Ponzanelli, Mocci, and Lanza (2015a); A. Rahman et al. (2019); Silva et al. (2018); Souza et al. (2019); Treude and Robillard (2016)
Database	MySQL, SQLite, PostgreSQL, eXist-db, MSSQL Server, MongoDB, MariaDB		20	Abdur (2018); Arwan et al. (2015); Bacchelli et al. (2012); Baltes, Dumani, et al. (2018); Grant and Betts (2013); Hassan et al. (2018); Joorabchi et al. (2016); Ling and Larsen (2018); Morrison and Murphy-Hill´ (2013); Ponzanelli (2012); Ponzanelli et al. (2013a, 2013b); Squire and Funkhouser (2014); Stanley and Byrne (2013); Vasilescu, Filkov, and Serebrenik (2013)
Search	Solr Search Engine, Google, Bing API, Yahoo, Google Code Search, Apache Lucene		15	Bacchelli et al. (2012); L. B. L. de Souza et al. (2014); Parnin et al. (2012); Ponzanelli (2012, 2014); Ponzanelli et al. (2013a, 2013b); Ponzanelli, Bavota, Di Penta, et al. (2014); Ponzanelli, Bavota, et al. (2015); Ponzanelli, Bavota, Penta, et al. (2014); Ponzanelli, Mocci, and Lanza (2015a); M. M. Rahman et al. (2013); Silva et al. (2018); Sim and Gallardo-Valencia (2013); Souza et al. (2019)
NLP	Apache NLTK, ScalaNLP API, Stanford POS Stanford To OpenCalais	OpenNLP, tagger, kenizer,	16	Allamanis and Sutton (2013); Alrashedy (2018); Bajaj et al. (2014); Campbell et al. (2013); Cline et al. (2018); Fontao et al. (2018); Hassan et al. (2018); Omondiagbe˜ et al. (2019); Singh and Shadbolt (2013); Stanley and Byrne (2013); S. Wang et al. (2014); S. Wang, Lo, et al. (2018); Wong et al. (2013); S. Xu et al. (2018); Z. Yao et al. (2018); D. Ye, Xing, Foo, et al. (2016); D. Ye, Xing, Li, and Kapre (2016)
Parser	Eclipse Island ESPRIMA Java SAX Stanford NLP AST Parser, Parboiled2	parser, parser, parser, parser, jSoup,	13	Bacchelli et al. (2012); Campbell et al. (2013); Mastrangelo et al. (2015); Parnin et al. (2012); Ponzanelli (2012); Ponzanelli et al. (2013a, 2013b); Ponzanelli, Mocci, Bacchelli, and Lanza (2014); Ponzanelli, Mocci, Bacchelli, Lanza, and Fullerton (2014); Ponzanelli, Mocci, and Lanza (2015a,?); Subramanian et al. (2014); Treude and Robillard (2016); Z. Yao et al. (2018)

continued on following page

Table 4. Continued

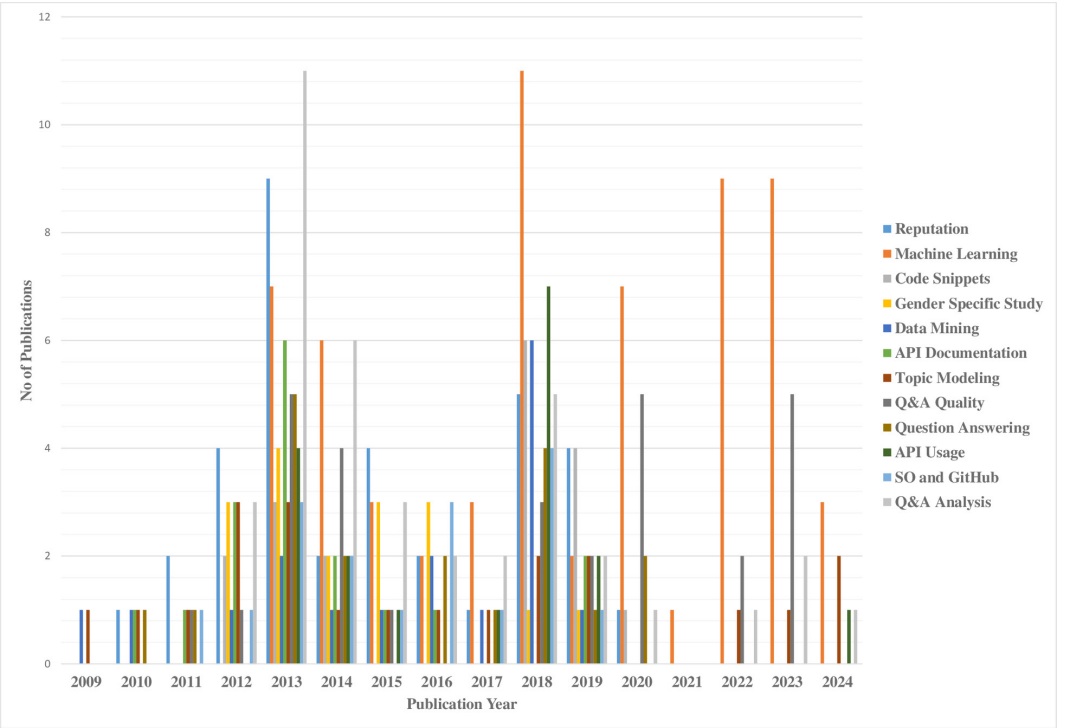
Category	Tools	Count	Citations
Machine Learning	Weka, Adaboost classifier, SVMlight, TensorFlow, LibSVM, Liblinear, JGAP, SciKit-Learn, CRF++ ToolKit, Irace 2.3	25	Alrashedy (2018); Anwar, Afzal, Al-Shehari, et al. (2024); Anwar, Afzal, Altaf, et al. (2024); Asaduzzaman et al. (2013); Calefato et al. (2015); Campos and de Almeida Maia (2014); D. Chang et al. (2013); Cline et al. (2018); Correa and Sureka (2014); L. B. L. de Souza et al. (2014); Gkotsis et al. (2014); Hassan et al. (2018); R. Kumar, Lifshits, and Tomkins (2010); Lal et al. (2014); Lezina and Kuznetsov (2013); Omondiagbe et al. (2019); Pal et al. (2012); Saha et al. (2013); Treude and Robillard (2016); Treude and Wagner (2018, 2019); S. Xu et al. (2018); Z. Yao et al. (2018); D. Ye, Xing, Foo, et al. (2016); Y. Ye et al. (2018)
Network Analysis	Gephi, Python SNAP library	6	Anwar, Bibi, et al. (2024); Bosu et al. (2013); R. Kumar et al. (2010); Lal et al. (2014); Paul et al. (2018); D. Ye, Xing, and Kapre (2016)
Sentiment Analysis	SentiStrength, Senti4SD	4	Calefato et al. (2015); Ling and Larsen (2018); Novielli´ et al. (2015)
API	Youtube API, element- Tree XML API, Yahoo Geolocation API, SO API	10	Abdur (2018); Anwar, Afzal, Al-Shehari, et al. (2024); Anwar, Afzal, Altaf, et al. (2024); Anwar, Afzal, and Iltaf (2024); Greco (2018); Greco et al. (2018); May et al. (2019); Ponzanelli (2014); Ponzanelli, Bavota, Di Penta, et al. (2014); Ponzanelli, Bavota, et al. (2015); Ponzanelli et al. (2016,?); Ponzanelli, Bavota, Penta, et al. (2014); M. M. Rahman et al. (2013); Rosen and Shihab (2016); Schenk and Lungu (2013); Sim and Gallardo-Valencia (2013); Treude and Robillard (2016); Treude and Wagner (2018, 2019)
Code Processing	Eclipse JDT, DUDE, Neo4j, APKTool, SIM, SearchCode, SimHash, SourcererCC, Ninka, git blame, Simian	18	Allamanis and Sutton (2013); Kavaler et al. (2013); Nishi et al. (2019); Ponzanelli (2014); Ponzanelli, Bavota, Di Penta, et al. (2014); Ponzanelli, Bavota, et al. (2015); Ponzanelli, Bavota, Penta, et al. (2014); Ponzanelli, Mocci, and Lanza (2015a); Ragkhitwetsagul et al. (2019); M. M. Rahman et al. (2013); Subramanian et al. (2014); Sun et al. (2018); Wong et al. (2013)
Data Analysis	mlstats tool, Excel, Matlab, pROC package, Quicsilver tool, MAXQDA, WVTool, ANCOVA	13	Calefato et al. (2015); Convertino et al. (2017); Morrison and Murphy-Hill (2013); Pal et al. (2012); Schenk and Lungu (2013); Silva et al. (2018); Tian, Liang, and Babar (2019); Vasilescu et al. (2012); Vasilescu, Capiluppi, and Serebrenik (2013)
Development Tools	t Eclipse, Ionic, Visual Studio, RStudio, IntelliJ, R language, Scala, Python, PHP	17	Abdur (2018); Anwar et al. (2023); Anwar, Bibi, et al. (2024); Beyer et al. (2018); Calefato et al. (2018); Greco (2018); Greco et al. (2018); Ling and Larsen (2018);´ Lopez-Nores et al. (2019); Lord et al. (2018); Morrison´ and Murphy-Hill (2013); Omondiagbe et al. (2019); Ponzanelli (2014); Ponzanelli, Bavota, Di Penta, et al. (2014); Ponzanelli, Bavota, et al. (2015); Ponzanelli, Bavota, Penta, et al. (2014); Subramanian and Holmes (2013); Subramanian et al. (2014); Vasilescu et al. (2014); H. Yin and Pfahl (2017b)
HTML Tools	HTML scrapping, Html cleaner, httunit	3	M. M. Rahman et al. (2013); Souza et al. (2019)
Data Extractor	BigQuery, Wikipedia Miner, GH Rest API, SE Data Explorer, GHTorrent, XML dump importer	13	Abdur (2018); Anwar et al. (2023); Anwar, Afzal, and Iltaf (2024); Bacchelli et al. (2012); Baltes, Dumani, et al. (2018); Nishi et al. (2019); Ponzanelli (2012); Ponzanelli et al. (2013a, 2013b); Singh and Shadbolt (2013); Sun et al. (2018); Vasilescu, Filkov, and Serebrenik (2013)

continued on following page

Table 4. Continued

Category	Tools	Count	Citations
Others	Various Tools (Google2Str, FFM- PEG, RQDA, etc.)	18	Alrashedy (2018); Anwar, Bibi, et al. (2024); Beyer et al. (2018); Correa and Sureka (2014); Diamantopoulos et al. (2019); Fontao et al. (2018); Huna et al. (2016);¯ May et al. (2019); Ponzanelli et al. (2016,?); Ponzanelli, Mocci, and Lanza (2015b); Reinhardt et al. (2018); S et al. (2013); Sun et al. (2018); Vasilescu, Serebrenik, and Van Den Brand (2013); Wilde (2009); D. Ye, Xing, Foo, et al. (2016); T. Zhang et al. (2018)

Figure 6. Temporal analysis of publication topics



identify experts with .94 precision and .85 recall. The most popular and successful technique to find experts is based on the difficulty and quality of content. Like V. Kumar and Pedanekar (2016) mapped the expertise of uses based on the quality of Q&A. Reputation, Z-score, and average duration between up and downvotes are features used to identify experts Hanrahan et al. (2012). In another research Huna et al. (2016), reputation, best answer ratio, Z-score, and posted answers are used to identify experts. These features determine the content quality. In another research, experts from another community, i.e. GitHub are identified and SO questions are routed to them as a result, 7 out of 15 experts answered the questions. One difference between experts and normal users is that experts post answers across multiple tags, whereas learners ask about only one tag/topic Paul et al. (2018). The work of Bhanu and Chandra (2016) identified topical experts and gained 5-16% accuracy over the baseline reputation-based method. A regional-based study of expert users shows that on SO North American and European users are the main contributors of content as Asian users are the importers of information Schenk and Lungu (2013). It is also determined that linking data to

external knowledgebases helps to find experts Singh and Shadbolt (2013). Interviews are also used by Dagenais and Robillard (2010) to find the experts. The main motive behind this exercise is to find API documentation creators.

6.3 Q&A Analysis / Mining

Various researchers studied the unanswered, closed, deleted, duplicate, migrated, and obsolete questions. These questions affect the usefulness of social networks; therefore, strategies should be developed to deal with such questions. S. Chang and Pal (2013) researched the reason for unanswered questions because unanswered questions are increasing. Convertino et al. (2017) named unanswered / late-answered questions as dark side questions and developed a classifier to identify such questions. The false Positive (FP) of the proposed approach is 34.4% whereas, the FP of the na¨ıve approach is 57%. A classifier is developed by Correa and Sureka (2014) to identify which question will be deleted. Similarly, Correa and Sureka (2013) results show that Stochastic Gradient Boosted Trees give the best accuracy in classifying closed questions. Lezina and Kuznetsov (2013) used User, Post & Tag features to predict closed questions at the time of posting. H. Zhang, Wang, Chen, Zou, and Hassan (2019) studied the obsolete answers and found that 58.4% answers are obsolete at the time of posting. Furthermore, 20.5% of obsolete answers are updated. A reproducibility study to find duplicate questions on SO was performed by Silva et al. (2018). The recall rate of both tools is reduced by 20.5% for DupPredictorRep and 28.5% for DupRep. In R. Kumar et al. (2010); Lal et al. (2014) automatically identified the questions that are irrelevant on SO and will be migrated to other sites of the SE network.

Different types of questions are posted on SO like how-to questions, errors & debugging, information requests, etc. Categorization of questions helps to extract relevant questions quickly. Many researchers categorized the SO questions, for example, in Campos and de Almeida Maia (2014) SO questions are categorized and usage of each type of question is explained. Treude, Barzilay, and Storey (2011) identified 10 categories of questions. Beyer et al. (2018) classify Android questions. Barua et al. (2014) identified topics of interest by using LDA for categorization. 80 to 97% accuracy is achieved to classify the question automatically Ponzanelli, Mocci, Bacchelli, and Lanza (2014); Ponzanelli, Mocci, Bacchelli, Lanza, and Fullerton (2014). Rosen and Shihab (2016) categorized mobile app development questions and identified the development trends. Research of Linares-Vasquez et al. (2013) also identifies trends and hot topics in´ mobile development. In Bajaj et al. (2014) temporal trends in mobile development are identified. Ghrairi, Kpodjedo, Barrak, Petrillo, and Khomh (2018) studied VR-related questions and found that JavaScript and

C# is the most popular programming language among VR developers. The work of Allamanis and Sutton (2013) concludes that confusing programming concepts are independent of programming languages and can be identified by analyzing SO Q&A.

Many researchers worked to predict the answer / best answer for a question. J.-W. Lin (2018) gained 8.2% improvement over baseline methods (first, last and longest). In another paper, the model with Q&A text and user metadata features gives the best result for the best answer prediction S. Xu et al. (2018). Gkotsis et al. (2014) evaluated 21 sites to predict the best answer. Decision trees are used for prediction. The technique of Tian et al. (2013) gained 21.5% improvements over TF.IDF to predict the best answer. The features to predict accepted and unaccepted answers are the length of code in the answer, the reputation of the user, similarity of text between Q&A, and the time lag between question & answer Omondiagbe et al. (2019). Winston (n.d.) researched to predict answers for cloze-style questions. The accuracy of CASE-BiGRU-CC + ft on the training set is 44% and the test set is 45%. The response/answer rate of some languages is higher. For example, Lord et al. (2018) observed that the response rate of Python posts is greater than R whereas, the accepted answer rate of R is higher than Python. The effect of editing different segments of posts is studied in G. Li et al. (2015) and concludes that editing has positive and negative effects.

Some researchers created data sets by using SO data dump. The purpose of these data sets is to quickly parse SO posts Ponzanelli, Mocci, and Lanza (2015a), maintain the block-level history of posts Baltes, Dumani, et al. (2018), and develop answer code data set Z. Yao et al. (2018).

Stanley and Byrne (2013) created a system to recommend & clean tags. This system achieved 65% accuracy in predicting and cleaning tags. Another study Gujral et al. (2018), identifies the tags associated with logging questions.

Various studies have been conducted to find Q&A which will gain reputation. Anderson et al identified 27 features and used Logistic Regression to predict which question would gain a reputation. Most of the questions gain reputation in first-hour Anderson et al. (2012). Bosu et al. (2013) proposed guidelines to gain reputation and performed an analysis of hot topics. Ellmann and Schnecke (2018) predict which question will be answered and voted on. Various parameters that increase the quality of Q&A are identified. Another researcher Calefato et al. (2015) proposed that time is the most important factor to increase the chance of answer acceptance, other factors are presentation, effectiveness, and reputation. Squire and Funkhouser (2014) also determine the features of the best posts and conclude that a 40-50% code-to-text ratio is ideal for a good post.

Extraction and utilization of code snippets embedded in SO posts is an active research area. P. Yin, Deng, Chen, Vasilescu, and Neubig (2018) improve the accuracy of code extraction, the proposed method outperforms AcceptOnly, ALL, and Random methods. 7 issues that affect the understandability of SO code snippets are identified by Treude and Robillard (2017). Similarly, Y. Wu et al. (2019) studied the utilization of code snippets and the top barriers for utilization are modification of code, incomprehensible code, and low quality. In another paper, Nishi et al. (2019) identified the top reasons for duplicating code which is an error in executing code and unexpected behavior of code. M. Ahmad and Cinneide (2019)´ study the effect of adding code snippets in classes. 70% classes copied snippets affect code cohesion and in 70% of cases lost cohesion is never gained. Code snippets in posts are also edited and evolve. Baltes, Dumani, et al. (2018) worked on code evaluation and identified that comments trigger code edits. The programming language of code snippets is predicted by Alrashedy (2018) with 91.1% accuracy. Some papers analyzed the code snippets and found various problems in code related to security. For example, A. Rahman et al. (2019) observed that 7.1% posts include insecure code including 9.8% accepted answers. Code injection is the most occurring insecure coding practice. Y. Ye et al. (2018) developed a method to detect insecure code in SO posts, this method outperforms all the existing ones. Tahir et al. (2018), identify popular code smells and anti-patterns discussed on SO. 59% questions on code smells are related to C#, JavaScript, and Java. Toxic code snippets and licensing issues in SO posts are also identified by Ragkhitwetsagul et al. (2019). Fischer et al. (2019) classify secure and insecure cryptography code. The system achieved an AUC-ROC of 0.992.

Papers to predict the quality of SO posts mainly identify different features that should be used to estimate the quality because the reputation of the post alone is not a good estimator of quality. These features include the difficulty of the question L. Xu et al. (2016), complete user profile Ginsca and Popescu (2013), and correlation between question & answer quality Y. Yao et al. (2013). Nasehi et al. (2012) Identify the attributes of good Q&A using grounded theory.

Users also share the web links of other questions and websites in SO posts. Research is conducted to determine the motive of link sharing and frequently shared links. Carlos et al. (2013) observed that official documentation links are most frequently shared. In another paper, D. Ye, Xing, and Kapre (2016) conclude that reference information for problem-solving and reference answers are the topmost motives of URL sharing. Joorabchi et al. (2016) performed the mapping between SO posts and Wikipedia categories to create notes.

6.4 Topic Modeling

Topic modeling techniques are used in various publications. Papers related to topic modeling are categorized into various categories. Here, only those papers are given whose main contribution is achieved by using topic modeling. In Movshovitz-Attias and Cohen (2013) topic modeling is used

to predict comments to the programmer. It will save code documentation time. In another paper Ponzanelli, Mocci, and Lanza (2015b), topic modeling is used to improve LexRank for document summarization. Treude and Wagner (2019) created the optimum parameter set for topic modeling for SO and GitHub corpora. This set outperforms the baseline by 4%. In another paper, Zolaktaf et al. (2011) created a new technique for topic modeling to capture topic dependencies. Coogle (2019) developed the Label-Hierarchy technique to arrange tags in the hierarchy. This technique was compared with Labeled-LDA and 88.47% accuracy is achieved.

6.5 Behavior Study

Users have similar behavior in private & public social network platforms Raj et al. (2011) and interaction between users is a temporal process R. Kumar et al. (2010). Replies on the same post are a measure to determine the connection between users Halavais et al. (2014). The user profile represents the activity and the ability of the user. The basic motive of developers to participate in social networking is to learn and show their work Singer et al. (2013). User profiling is also a measure to judge the behavior of users. Various techniques for user profiling are developed like RankSLDA San Pedro and Karatzoglou (2014) and Segmented topic model "Finding expert users in community question answering" (2012); Riahi (2012).

The Gamification mechanism is an important factor in the success of the community. User activity increases when near achieving badges Grant and Betts (2013). 25% users stop revising answers after receiving their first badge S. Wang, Chen, and Hassan (2018). R-Help users are more active on SO due to gamification Vasilescu et al. (2014). In the research of Vasilescu, Filkov, and Serebrenik (2013), it is determined that active SO answerers make more commits on GitHub and SO activity accelerates GitHub committing. In another research Lotufo et al. (2012), Gamification helps to make the bug-tracking system more effective.

Gender-specific studies are also conducted on SO to judge the behavior and activity of male and female users. Males engage for a longer duration. It is observed that the SO community is not a femalefriendly community and females disengage early Vasilescu et al. (2012); Vasilescu, Capiluppi, and Serebrenik (2013). Female asks more questions and males answer more questions. The current reputation mechanism supports answerers, therefore the reputation mechanism should be changed to increase the participation of females May et al. (2019).

The majority of users on SO are asking questions. S. Wang et al. (2013) studied the asker and answerer trend and interaction between users. It is observed that the majority of users are asking questions and few developers are connected. L. Wu et al. (2015, 2016) divided the users into two types, Type-A askers and Type-B answerers. Type-A users should be twice as large as Type-B users. The design of SO also affects the contributions of users. SO design is more focused and less social, therefore Asian users are contributing less as compared to American and European users Oliveira et al. (2018). Sun et al. (2018) study flocking and migration behavior of users. 7.5% and 8.7% users make a flock on GitHub and SO respectively. The average migration rate on GitHub is 5% and 2% on SO. A flock of two migrates more frequently.

User behavior on SO and the job-seeking trend is also studied by researchers. L. Xu et al. (n.d., 2014, 2016, 2019, 2018) observed a reduction in reputation-gaining activity after finding a job. Beeharry and Ganoo (2018) find programming, database & job seeking trends among the SO community. Developers prefer to stay in their current company. A positive relation between programming knowledge and age is also observed by Morrison and Murphy-Hill (2013).

6.6 API Documentation

Crowd documentation covers practical issues and covers most of the API official documentation Jiau and

Yang (2012). In another research Parnin and Treude (2011), it is observed that social media cover 87% of API documentation. Different APIs are evaluated to check the coverage in another paper

Parnin et al. (2012). Similarly, deficient areas of project documentation are identified in Campbell et al. (2013). Size of class documentation has a positive relationship with getting first answer Kavaler et al. (2013). observe programmers while coding difficult problems. Programmers spend their most of time searching API documentation Sushine et al. (2015). Another paper W. Wang and Godfrey (2013) studies obstacles in using APIs and different reoccurring obstacles are listed.

Subramanian et al. (2014) develop a tool to extract Java and JavaScript code from SO posts and link with API documentation with 97% precision. SO Q&A are used to create cookbooks. Results show that 63-71% chapters are well defined in meaning L. B. de Souza et al. (2014). In another paper Treude and

Robillard (2016), a machine learning algorithm is implemented to extract 70% meaningful sentences from SO. The research of Souza et al. (2019) to create a cookbook achieved 77.91% accuracy and 78% precision. Mastrangelo et al. (2015) studied the SO questions related to the Java unsafe API to analyze what users are discussing about unsafe. Reinhardt et al. (2018); T. Zhang et al. (2018) studied the API misuse patterns and identified that database, crypto, and networking APIs are often misused.

6.7 Recommender System

Various recommender systems are developed to recommend SO posts, code snippets, and tags to programmers to save time. Most of these recommender systems are developed as an Eclipse IDE plugin. The limitations of these systems are that they are developed by using offline data set and their accuracy is low. For example, Seahawk is developed to integrate SO data in IDE to save programmers searching time Bacchelli et al. (2012); Ponzanelli (2012); Ponzanelli et al. (2013a, 2013b). Similarly, in Arwan et al. (2015) code from SO posts is extracted and used to develop code recommender which has 48% average precision and 58% average recall. Another Eclipse IDE plugin, Prompter Ponzanelli (2014); Ponzanelli, Bavota, Di Penta, et al. (2014); Ponzanelli, Bavota, et al. (2015); Ponzanelli, Bavota, Penta, et al. (2014) is developed and evaluated in a user study. The average correctness of Prompter is 44% and its recommendations are volatile. In another paper M. M. Rahman et al. (2013), solutions of programming errors are recommended with 94% accuracy. A recommender system is developed which extracts how-to posts and recommends related posts to programmer 77.14% accuracy during codding L. B. L. de Souza et al. (2014). S. Chang and Pal (2013) develop a system by using machine learning algorithms to route questions to potential answerers. developed a recommender system to recommend SO posts related to video tutorials Ponzanelli et al. (2016,?). Maity et al developed a tag recommender. This system gains 60.8% and 36.8% in precision and recall over the TagCombine method Maity et al. (2019).

6.8 Emotion Mining

In Novielli et al. (2015), Novielli et al used the SentiStrength to determine the polarity of SO posts. But the results are not satisfactory. The authors proposed that suitable tools should be developed for the Sentiment Analysis of SO data. In other research, B. Lin et al. (2018) used Recursive Neural Network (RNN) for sentiment analysis. The results of the technique are comparable with SentiStrength, NLTK, Stanford CoreNLP, and SentiStrength-SE, but the results are not satisfactory. In another paper, SentiStrength-SE,

Senti4SD, and EmoTxt are compared but these struggle to compute negative sentiments Islam and Zibran (2018). A tool named Senti4SD which is trained on SO data set for sentiment analysis was compared with SentiStrength and SentiStrength-SE. The authors observe 19% improvements in precision for negative and 25% improvements in recall for the neutral class as compared with SentiStrength Calefato et al. (2018). H. Yin and Pfahl (2017b), in his work extract functional and dysfunctional requirements. In their work, the accuracy of sentiment analysis was 81%. In another paper, H. Yin and Pfahl (2017a) extract user requirements from SO. The authors observed that 82% of questions are neutral. Ling and Larsen (2018),´ performed sentiment analysis of posts related to

various languages to get the sentiment of programmers of different languages. Results of the complete data set after performing sentiment analysis are compared with each selected programming language.

7. RESEARCH GAP

Based on an extensive review of the literature, several research gaps have been identified. These gaps highlight the limitations of current studies and suggest areas where future research could make significant contributions. Below is a detailed discussion of these gaps, including the reasons for their existence and potential avenues for advancing the field.

7.1 Accuracy in Cookbook Creation From Stack Overflow (SO) Data

Current efforts to create cookbooks from SO data sets exhibit limited accuracy. For instance, while L. B. de Souza et al. (2014) report that 63-71% of cookbook chapters are meaningfully defined, and Treude and Robillard (2016) achieve 70% accuracy in extracting meaningful API documentation, Souza et al. (2019) later increased accuracy to 77.91% using a classifier. However, even this improvement leaves significant room for enhancement. The primary reason for this gap is the complexity of natural language processing (NLP) in diverse contexts within SO posts. Addressing this gap could lead to the development of more reliable tools for creating cookbooks, which in turn would greatly aid developers in finding relevant solutions efficiently.

7.2 Limitations in Auto-Tagging Tools

Auto-tagging tools are currently limited, achieving only 68% accuracy in tagging SO posts Saha et al. (2013). Since auto-tagging significantly increases the likelihood of receiving answers, enhancing these tools' accuracy is crucial. The challenge lies in the nuanced and context-dependent nature of tags in SO posts. Developing more sophisticated tagging algorithms, possibly through advanced machine learning models, could lead to tools that better understand context, thereby improving the efficiency and relevance of the SO platform.

7.3 Gaps in Social Media Coverage of API Documentation

Studies show that social media covers 87% of API documentation Parnin and Treude (2011); Parnin et al. (2012), but gaps remain in identifying and understanding the topics not covered. This gap persists because current tools are not designed to detect these omissions or analyze their causes. Addressing this could lead to the creation of tools that better monitor and evaluate the comprehensiveness of API documentation, providing developers with more complete information and potentially uncovering areas that need more attention.

7.4 Temporal Analysis of Question and Answer Quality

Temporal analysis of the quality of SO posts could reveal trends in how the community's collective knowledge and expertise evolve over time. This analysis is currently underexplored, likely due to the challenge of continuously tracking and evaluating vast amounts of data over extended periods. By developing tools to perform such analyses, researchers could better understand the dynamics of knowledge sharing and how to improve the quality of future contributions.

7.5 Evaluation of Expertise Shapes in Skill Management Tools

The concept of expertise shapes, extracted from SO user profiles, could be further explored to enhance skill management tools Anwar, Bibi, and Ahsan (2013). Although current tools provide some insights into user expertise, they do not fully leverage the depth and breadth of data available from SO. Expanding on this could lead to more accurate and detailed assessments of skills, benefiting both individual developers and organizations seeking to manage talent effectively.

7.6 Enhancing Recommender Systems for Stack Overflow Posts

The accuracy of current SO post recommender systems is relatively low. For example, the SO code recommender achieves only 48% average precision and 58% average recall Arwan et al. (2015), while Prompter's correctness is about 44% Ponzanelli (2014); Ponzanelli, Bavota, Di Penta, et al. (2014); Ponzanelli, Bavota, et al. (2015); Ponzanelli, Bavota, Penta, et al. (2014). Furthermore, many recommender systems operate offline, limiting their real-time effectiveness. Addressing this gap by developing online systems with better search and ranking capabilities could significantly enhance user experience and productivity.

7.7 Maturity of Sentiment Analysis Tools for Software Engineering

Existing sentiment analysis tools in the Software Engineering domain are not sufficiently mature. Tools like SentiStrength-SE and Senti4SD perform well only on specific data sets and lack generalizability Calefato et al. (2018); B. Lin et al. (2018). The root of this gap lies in the diversity of language and context across different SE data sets. Developing domain-specific sentiment analysis tools, possibly by creating tailored dictionaries and leveraging advanced NLP techniques, could vastly improve the accuracy and applicability of these tools.

7.8 Impact of Editing on Stack Overflow Post Quality

SO encourages users to edit Q&A and awards badges for doing so. However, this might incentivize edits that do not necessarily improve, or may even degrade, the quality of posts. This issue has not been extensively studied, likely due to the difficulty of assessing post-edit effects on quality. Analyzing the impact of these edits on post quality and the satisfaction of the original question-asker could lead to insights that refine the platform's editing policies and practices.

7.9 Utilizing Stack Overflow Data for Educational Tools and Course Development

The rich data from SO can be leveraged to create dynamic programming course outlines and educational tools. However, this potential remains largely untapped, possibly due to the challenge of integrating diverse data types into cohesive educational frameworks. Developing systems that utilize tags, question types, and rankings to design courses could revolutionize how programming is taught, making it more responsive to current industry trends.

7.10 Gender-Specific Opinion Mining in Software Engineering

Current sentiment analysis research does not distinguish between the opinions of male and female developers on various tools, topics, and issues. This gap exists likely due to a lack of focus on gender-specific analysis within the SE community. Exploring this area could reveal significant differences in how tools and practices are perceived, potentially leading to more inclusive and effective SE practices.

7.11 Automating Topic Labeling in LDA

Latent Dirichlet Allocation (LDA) is widely used for topic extraction, but it does not automatically assign meaningful labels to the topics. Automating this process using machine learning could enhance the utility of LDA in SE research, making it easier for researchers to quickly identify and categorize relevant topics without extensive manual intervention.

7.12 Expertise-Based Product Evaluation

SO posts can be analyzed to identify domain experts, who can then be ranked based on their reputations and used to evaluate products related to their expertise. This approach remains underexplored, likely due to the challenges in accurately identifying and ranking experts. By developing

methods to automate these processes, organizations could leverage expert opinions to enhance product development and evaluation.

8. CONCLUSION

In this paper, we survey the 214 papers related to Stack Overflow. The goal of the survey is to semiautomatically categorize the papers, and identify the various research gaps. 62 topics are identified by using LDA. These topics are labeled by using the C-Value / NC-Value method. The topics are grouped into eight categories and several sub-categories. A review of relevant papers for each category is given. Along with the review, we identified various research trends, tools, and techniques that are of interest to readers. Findings of review are also given based on the results of the paper. Finally, we identified various research gaps in the existing research and proposed research ideas for future researchers. This paper is helpful for a researcher who is planning to use SO data set in their research. Researchers, who are searching for research ideas will also benefit from this paper.

For practitioners, we emphasized how the improved accuracy of tools like auto-tagging, recommender systems, and sentiment analysis could directly benefit software development processes by enhancing code searchability, providing better recommendations, and offering more accurate sentiment insights. These improvements can lead to more efficient problem-solving, better project management, and an overall increase in productivity within software development teams.

For researchers, we discussed how the identified research gaps and proposed future directions could stimulate new studies that push the boundaries of current methodologies. For instance, addressing the limitations in existing machine learning algorithms for creating cookbooks from Stack Overflow data could lead to the development of more sophisticated tools that bridge the gap between research and practice. Additionally, exploring gender-specific sentiment analysis and the temporal evaluation of expertise on Stack Overflow could open new avenues for research in software engineering education and community dynamics.

A comparative analysis with other platforms, such as GitHub, Reddit, or specialized programming forums, could offer valuable insights. In the future, we plan to explore this direction. A follow-up study could examine how SO's influence compares to or complements these platforms, thereby offering a more rounded view of its impact on the programming ecosystem. Incorporating more advanced analytical techniques or a mixed-methods approach could yield deeper insights into the complexities of user behaviors on SO. Techniques such as machine learning-based behavioral analysis, social network analysis, or longitudinal studies could offer a more nuanced understanding of how users interact with the platform over time. In future research, we plan to explore these advanced methodologies to enhance the depth of our analysis.

CONFLICTS OF INTEREST

We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome.

FUNDING STATEMENT

No funding was received for this work.

PROCESS DATES

This manuscript was initially received for consideration for the journal on 03/05/2024, revisions were received for the manuscript following the double-anonymized peer review on 09/28/2024, the

manuscript was formally accepted on 09/26/2024, and the manuscript was finalized for publication on 10/7/2024.

CORRESPONDING AUTHOR

Correspondence should be addressed to Zeeshan Anwar (Pakistan, zeeshan.phdcse@students.mcs.edu.pk)

ACKNOWLEDGMENT

The author would like to thank Deanship of Scientific Research at Majmaah University for supporting this work under Project Number 1234.

REFERENCES

Abdur, R. (2018). *A conceptual model and web platform for discovering digital heroes from stack overflow developer community forum using a data mining approach* [Unpublished doctoral dissertation]. Moi University, Eldoret, Kenya.

Ahmad, A., Feng, C., Ge, S., & Yousif, A. (2018). A survey on mining stack overflow: Question and answering (q&a) community. *Data Technologies and Applications*, *52*(2), 190–247. DOI: 10.1108/DTA-07-2017-0054

Ahmad, A., Feng, C., Khan, M., Khan, A., Ullah, A., Nazir, S., & Tahir, A. (2020). A systematic literature review on using machine learning algorithms for software requirements identification on stack overflow. *Security and Communication Networks*, *2020*(1), 8830683. DOI: 10.1155/2020/8830683

Ahmad, M., & Cinneide, M. O. (2019). Impact of stack overflow code snippets on software cohesion:´ A preliminary study. In *Proceedings of the 16th international conference on mining software repositories* (pp. 250–254). IEEE Press. /DOI: 10.1109/MSR.2019.00050

Ahmed, S., & Bagherzadeh, M. (2018). 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* (p. 30). ACM. DOI: 10.1145/3239235.3239524

Ahmed, T., & Srivastava, A. (2017). Understanding and evaluating the behavior of technical users. a study of developer interaction at stackoverflow. *Human-centric Computing and Information Sciences*, *7*(1), 8. DOI: 10.1186/s13673-017-0091-8

Akoglu, L., Chau, D. H., Kang, U., Koutra, D., & Faloutsos, C. (2012). Opavion: Mining and visualization in large graphs. In Candan, K. S., Chen, Y., Snodgrass, R. T., Gravano, L., & Fuxman, A. (Eds.), *Proceedings of the 2012 international conference on management of data - sigmod '12* (pp. 717–720). ACM Press. DOI: 10.1145/2213836.2213941

Allamanis, M., & Sutton, C. (2013). Why, when, and what: Analyzing stack overflow questions by topic, type, and code. In *Proceedings of the 10th international working conference on mining software repositories* (pp. 53–56). IEEE Press. DOI: 10.1109/MSR.2013.6624004

Almarimi, N., Ouni, A., Chouchen, M., & Mkaouer, M. W. (2022). Improving the detection of community smells through socio-technical and sentiment analysis. *Journal of Software*.

Alrashedy, K. (2018). *Predicting the programming language of questions and snippets of stack overflow using natural language processing* [Unpublished doctoral dissertation]. University of Victoria.

Anderson, A., Huttenlocher, D., Kleinberg, J., & Leskovec, J. (2012). Discovering value from community activity on focused question answering sites. In Yang, Q., Agarwal, D., & Pei, J. (Eds.), *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining - kdd '12* (p. 850). ACM Press. DOI: 10.1145/2339530.2339665

Anderson, A., Huttenlocher, D., Kleinberg, J., & Leskovec, J. (2013). Steering user behavior with badges. In *Proceedings of the 22nd international conference on world wide web* (pp. 95–106). ACM. DOI: 10.1145/2488388.2488398

Annamoradnejad, I., & Habibi, J. (2023). Automatic moderation of user-generated content. In *Encyclopedia of data science and machine learning* (pp. 1344–1355). IGI Global.

Anwar, Z., & Afzal, H. (2024). Mining crowd sourcing repositories for open innovation in software engineering. *Automated Software Engineering*, *31*(1), 11. DOI: 10.1007/s10515-023-00410-z

Anwar, Z., Afzal, H., Ahsan, A., Iltaf, N., & Maqbool, A. (2023). A novel hybrid cnn-lstm approach for assessing stackoverflow post quality. *Journal of Intelligent Systems*, *32*(1), 20230057. DOI: 10.1515/jisys-2023-0057

Anwar, Z., Afzal, H., Al-Shehari, T., Al-Razgan, M., Alfakih, T., & Nawaz, R. (2024). Mining the opinions of software developers for improved project insights: Harnessing the power of transfer learning. *IEEE Access : Practical Innovations, Open Solutions*, *12*, 65942–65955. DOI: 10.1109/ACCESS.2024.3397211

Anwar, Z., Afzal, H., Altaf, N., Kadry, S., & Kim, J. (2024). Fuzzy ensemble of fined tuned bert models for domain-specific sentiment analysis of software engineering dataset. *PLoS One*, *19*(5), e0300279. DOI: 10.1371/journal.pone.0300279 PMID: 38805433

Anwar, Z., Afzal, H., & Iltaf, N. (2024). Learning beyond books: A hybrid model to learn real-world problems. *Computer Applications in Engineering Education*. DOI: 10.1002/cae.22792

Anwar, Z., Bibi, N., & Ahsan, A. (2013). Expertise based skill management model for effective project resource allocation under stress in software industry of pakistan. In *2013 6th international conference on information management, innovation management and industrial engineering* (Vol. 1, pp. 509–513). DOI: 10.1109/ICIII.2013.6702986

Anwar, Z., Bibi, N., Rana, T., Kadry, S., & Afzal, H. (2024). Collaborative solutions to software architecture challenges faced by it professionals. *International Journal of Human Capital and Information Technology Professionals*, *15*(1), 1–29. DOI: 10.4018/IJHCITP.342839

Arwan, A., Rochimah, S., & Akbar, R. J. (2015). Source code retrieval on stackoverflow using lda. In *2015 3rd international conference on information and communication technology, icoict 2015* (pp. 295–299). IEEE Press. DOI: 10.1109/ICoICT.2015.7231439

Asaduzzaman, M., Mashiyat, A. S., Roy, C. K., & Schneider, K. A. (2013). Answering questions about unanswered questions of stack overflow. In *Proceedings of the 10th working conference on mining software repositories* (pp. 97–100). IEEE Press. DOI: 10.1109/MSR.2013.6624015

Bacchelli, A., Ponzanelli, L., & Lanza, M. (2012). Harnessing stack overflow for the ide. In *2012 third international workshop on recommendation systems for software engineering (rsse)* (pp. 26–30). IEEE Press. DOI: 10.1109/RSSE.2012.6233404

Bajaj, K., Pattabiraman, K., & Mesbah, A. (2014). Mining questions asked by web developers. In *Proceedings of the 11th working conference on mining software repositories - msr 2014* (pp. 112–121). ACM. DOI: 10.1145/2597073.2597083

Baltes, S., Dumani, L., Treude, C., & Diehl, S. (2018). Sotorrent: Reconstructing and analyzing the evolution of stack overflow posts. In *Proceedings of the 15th international conference on mining software repositories* (pp. 319–330). ACM. DOI: 10.1145/3196398.3196430

Baltes, S., Treude, C., & Diehl, S. (2018). Sotorrent: Studying the origin, evolution, and usage of stack overflow code snippets. *arXiv preprint arXiv:1809.02814*.DOI: 10.1007/s10664-012-9231-y

Barua, A., Thomas, S. W., & Hassan, A. E. (2014, June). What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering*, *19*(3), 619–654. DOI: 10.1007/s10664-012-9231-y

Batra, H., Punn, N. S., Sonbhadra, S. K., & Agarwal, S. (2021). Bert-based sentiment analysis: A software engineering perspective. In *International conference on database and expert systems applications* (pp. 138–148). DOI: 10.1007/978-3-030-86472-9_13

Bazelli, B., Hindle, A., & Stroulia, E. (2013). On the personality traits of stackoverflow users. In *IEEE international conference on software maintenance, icsm* (pp. 460–463). IEEE Press. DOI: 10.1109/ICSM.2013.72

Beeharry, Y., & Ganoo, M. (2018). Analysis of data from the survey with developers on stack overflow: A case study. *ADBU Journal of Engineering Technology, 7*(2).

Beyer, S., Macho, C., Pinzger, M., & Di Penta, M. (2018). Automatically classifying posts into question categories on stack overflow. In *Proceedings of the 26th conference on program comprehension* (pp. 211– 221). ACM. DOI: 10.1145/3196321.3196333

Bhanu, M., & Chandra, J. (2016). Exploiting response patterns for identifying topical experts in stackoverflow. In *2016 eleventh international conference on digital information management (icdim)* (pp. 139–144). IEEE Press. DOI: 10.1109/ICDIM.2016.7829790

Biswas, E., Karabulut, M. E., Pollock, L., & Vijay-Shanker, K. (2020). *Achieving reliable sentiment analysis in the software engineering domain using bert. In 2020 IEEE international conference on software maintenance and evolution (icsme)*.

Bosu, A., Corley, C. S., Heaton, D., Chatterji, D., Carver, J. C., & Kraft, N. A. (2013). Building reputation in stackoverflow: An empirical investigation. In *Ieee international working conference on mining software repositories* (pp. 89–92). IEEE Press., DOI: 10.1109/MSR.2013.6624013

Calefato, F., Lanubile, F., Maiorano, F., & Novielli, N. (2018). Sentiment polarity detection for software development. In *2018 IEEE/ACM 40th international conference on software engineering (icse)* (pp. 128–128). IEEE Press.

Calefato, F., Lanubile, F., Marasciulo, M. C., & Novielli, N. (2015). Mining successful answers in stack overflow. In *2015 IEEE/ACM 12th working conference on mining software repositories* (pp. 430–433). IEEE. DOI: 10.1109/MSR.2015.56

Campbell, J. C., Zhang, C., Xu, Z., Hindle, A., & Miller, J. (2013). Deficient documentation detection. In *Proceedings of the 10th working conference on mining software (msr 2013)* (p. 57-60). IEEE Press. DOI: 10.1109/MSR.2013.6624005

Campos, E. C., & de Almeida Maia, M. (2014). Automatic categorization of questions from q&a sites. In *Proceedings of the 29th annual ACM symposium on applied computing - sac '14* (pp. 641–643). ACM. DOI: 10.1145/2554850.2555117

Carlos, G., Cleary, B., & Singer, L. (2013). A study of innovation diffusion through link sharing on stack overflow. In *Proceedings of the 10th working conference on mining software (msr 2013)* (pp. 81–84). ACM.

Chang, D., Schiff, M., & Wu, W. (2013). Eliciting answers on stackoverflow. *Working Paper*.

Chang, S., & Pal, A. (2013). Routing questions for collaborative answering in community question answering. In *Proceedings of the 2013 IEEE/ACM international conference on advances in social networks analysis and mining* (pp. 494–501). IEEE Press. DOI: 10.1145/2492517.2492559

Cline, V., Dharwadkar, A., Goyal, R., Engels, D., Srinivas, R., & Rafiqi, S. (2018). Stack overflow question retrieval system. *SMU Data Science Review*, *1*(2), 1–34.

Convertino, G., Zancanaro, M., Piccardi, T., & Ortega, F. (2017). Toward a mixed-initiative qa system: From studying predictors in stack exchange to building a mixed-initiative tool. *International Journal of Human-Computer Studies*, *99*, 1–20. DOI: 10.1016/j.ijhcs.2016.10.008

Coogle, J. J. (2019). *Applying hierarchical tag-topic models to stack overflow* [Unpublished master's thesis]. Virginia Commonwealth University, Richmond, VA, USA.

Correa, D., & Sureka, A. (2013). Fit or unfit: analysis and prediction of'closed questions' on stack overflow. In *Proceedings of the first acm conference on online social networks* (pp. 201–212). ACM. DOI: 10.1145/2512938.2512954

Correa, D., & Sureka, A. (2014). Chaff from the wheat: Characterization and modeling of deleted questions on stack overflow. In *Proceedings of the 23rd international conference on world wide web* (pp. 631–641). ACM. DOI: 10.1145/2566486.2568036

Dagenais, B., & Robillard, M. P. (2010). Creating and evolving developer documentation: Understanding the decisions of open source contributors. In *ACM international symposium on foundations of software engineering* (pp. 127–136). ACM. DOI: 10.1145/1882291.1882312

de Souza, L. B., Campos, E. C., & Maia, M. D. A. (2014, sep). *On the extraction of cookbooks for apis from the crowd knowledge. In 2014 Brazilian symposium on software engineering*. IEEE. DOI: 10.1109/SBES.2014.15

de Souza, L. B. L., Campos, E. C., & Maia, M. D. A. (2014). Ranking crowd knowledge to assist software development. In *Proceedings of the 22nd international conference on program comprehension* (pp. 72– 82). ACM. Retrieved from http://doi.acm.org.proxy.lib.fsu .edu/10.1145/2597008.2597146 DOI: 10.1145/2597008.2597146

Diamantopoulos, T., Sifaki, M.-I., & Symeonidis, A. L. (2019). Towards mining answer edits to extract evolution patterns in stack overflow. In *Proceedings of the 16th international conference on mining software repositories* (pp. 215–219). IEEE Press. DOI: 10.1109/MSR.2019.00043

Ellmann, M., & Schnecke, M. (2018). Two perspectives on software documentation quality in stack overflow. In *Proceedings of the 4th acm sigsoft international workshop on nlp for software engineering* (pp. 6–9). ACM. DOI: 10.1145/3283812.3283816

Finding expert users in community question answering. (2012). In *Proceedings of the 21st international conference companion on world wide web - www '12 companion* (pp. 791–798). ACM Press. DOI: 10.1145/2187980.2188202

Fischer, F., Xiao, H., Kao, C.-Y., Stachelscheid, Y., Johnson, B., & Razar, D. (2019). Stack overflow considered helpful! deep learning security nudges towards stronger cryptography. In *28th {USENIX} security symposium ({USENIX} security 19)* (pp. 339–356).

Fontao, A., Ábia, B., Wiese, I., Estácio, B., Quinta, M., Santos, R. P., & Dias-Neto, A. C. (2018). Support-´ing governance of mobile application developers from mining and analyzing technical questions in stack overflow. *Journal of Software Engineering Research and Development*, *6*(1), 8. DOI: 10.1186/s40411-018-0052-6

Frantzi, K., Ananiadou, S., & Mima, H. (2000). Automatic recognition of multi-word terms. the cvalue/nc-value method. *International Journal on Digital Libraries*, *3*(2), 115–130. DOI: 10.1007/s007999900023

Fu, J., Li, Y., Zhang, Q., Wu, Q., Ma, R., Huang, X., & Jiang, Y.-G. (2020). Recurrent memory reasoning network for expert finding in community question answering. In *Proceedings of the 13th international conference on web search and data mining* (pp. 187–195). DOI: 10.1145/3336191.3371817

Gao, Z., Xia, X., Lo, D., Grundy, J., Zhang, X., & Xing, Z. (2023). I know what you are searching for: Code snippet recommendation from stack overflow posts. *ACM Transactions on Software Engineering and Methodology*, *32*(3), 1–42. DOI: 10.1145/3550150

Ghrairi, N., Kpodjedo, S., Barrak, A., Petrillo, F., & Khomh, F. (2018). *The state of practice on virtual reality (vr) applications: An exploratory study on github and stack overflow. In 2018 IEEE international conference on software quality, reliability and security (qrs)*. IEEE Press.

Ginsca, A. L., & Popescu, A. (2013). User profiling for answer quality assessment in q&a communities. In *Proceedings of the 2103 workshop on data-driven user behavioral modelling and mining from social media - dubmod '13* (pp. 25–28). ACM Press. Retrieved from https://dl.acm.org/ citation. DOI: 10.1145/2513577.2513579

Gkotsis, G., Stepanyan, K., Pedrinaci, C., Domingue, J., & Liakata, M. (2014). It's all in the content: state of the art best answer prediction based on discretisation of shallow linguistic features. In *Proceedings of the 2014 acm conference on web science (websci)* (pp. 202–210). ACM. DOI: 10.1145/2615569.2615681

Graham, K. (2013). Techmatters: "qiqqa" than you can say reference management: A tool to organize the research process. *LOEX Quarterly*, *40*(1), 3.

Grant, S., & Betts, B. (2013). Encouraging user behaviour with achievements: an empirical study. In *Proceedings of the 10th working conference on mining software repositories* (pp. 65–68). IEEE Press. DOI: 10.1109/MSR.2013.6624007

Graßl, I., & Fraser, G. (2022). Scratch as social network: topic modeling and sentiment analysis in scratch projects. In *Proceedings of the 2022 acm/ieee 44th international conference on software engineering: Software engineering in society* (pp. 143–148). DOI: 10.1145/3510458.3513021

Greco, C. (2018). *A behavior-driven recommendation system for stack overflow posts* [Unpublished master's thesis]. Virginia Commonwealth University.

Greco, C., Haden, T., & Damevski, K. (2018). Stackintheflow: behavior-driven recommendation system for stack overflow posts. In *Proceedings of the 40th international conference on software engineering: Companion proceeedings* (pp. 5–8). ACM. DOI: 10.1145/3183440.3183477

Gujral, H., Sharma, A., Lal, S., Kaur, A., Kumar, A., & Sureka, A. (2018). *Empirical analysis of the logging questions on the stack overflow website. In 2018 conference on software engineering & data sciences (coseds) (in-press)*. Ruzica Piskac.

Halavais, A., Kwon, K. H., Havener, S., & Striker, J. (2014). Badges of friendship: Social influence and badge acquisition on stack overflow. In *Proceedings of the 47th hawaii international international conference on systems science (hicss-47 2014)* (pp. 1607–1615). IEEE. DOI: 10.1109/HICSS.2014.206

Hanrahan, B. V., Convertino, G., & Nelson, L. (2012). Modeling problem difficulty and expertise in stackoverflow. In *Proceedings of the acm 2012 conference on computer supported cooperative work companion* (pp. 91–94). ACM. DOI: 10.1145/2141512.2141550

Hassan, S. A., Das, D., Iqbal, A., Bosu, A., Shahriyar, R., & Ahmed, T. (2018). Soqde: A supervised learning based question difficulty estimation model for stack overflow. In *2018 25th Asia-Pacific software engineering conference (apsec)* (pp. 445–454). IEEE Press. DOI: 10.1109/APSEC.2018.00059

Henning, V., & Reichelt, J. (2008). Mendeley-a last. fm for research? In *2008 IEEE fourth international conference on escience* (pp. 327–328).

Herrmann, M., Obaidi, M., Chazette, L., & Klunder, J. (2022). On the subjectivity of emotions in soft-¨ ware projects: How reliable are pre-labeled data sets for sentiment analysis? *Journal of Systems and Software*, *193*, 111448. DOI: 10.1016/j.jss.2022.111448

Herrmann, M., Obaidi, M., & Klunder, J. (2022). Senti-analyzer: joint sentiment analysis for text-based¨ and verbal communication in software projects. *arXiv preprint arXiv:2206.10993*.DOI: 10.1016/j.jss.2022.111448

Hsieh, J. W. (2020). *Asking questions is easy, asking great questions is hard: Constructing effective stack overflow questions* [Unpublished doctoral dissertation]. Oberlin College.

Huna, A., Srba, I., & Bielikova, M. (2016). Exploiting content quality and question difficulty in cqa reputation systems. In *Proceedings of the 12th international conference and school on advances in network science-volume 9564* (pp. 68–81). Springer. DOI: 10.1007/978-3-319-28361-6_6

Islam, M. R., & Zibran, M. F. (2018). A comparison of software engineering domain specific sentiment analysis tools. In *2018 IEEE 25th international conference on software analysis, evolution and reengineering (saner)* (pp. 487–491). IEEE Press. DOI: 10.1109/SANER.2018.8330245

Jiau, H. C., & Yang, F.-P. (2012). Facing up to the inequality of crowdsourced api documentation. *Software Engineering Notes*, *37*(1), 1–9. DOI: 10.1145/2088883.2088892

Jin, X., & Servant, F. (2019). What edits are done on the highly answered questions in stack overflow?: an empirical study. In *Proceedings of the 16th international conference on mining software repositories* (pp. 225–229). IEEE Press. DOI: 10.1109/MSR.2019.00045

Joorabchi, A., English, M., & Mahdi, A. E. (2016, March). Text mining stackoverflow. *Journal of Enterprise Information Management*, *29*(2), 255–275. DOI: 10.1108/JEIM-11-2014-0109

Kavaler, D., Posnett, D., Gibler, C., Chen, H., Devanbu, P., & Filkov, V. (2013). Using and asking: Apis used in the android market and asked about in stackoverflow. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8238 LNCS*, 405–418. doi: DOI: 10.1007/978-3-319-03260-3 35

Kichou, S., Boussaid, O., & Meziane, A. (2020). Tag's depth-based expert profiling using a topic modeling technique. *International Journal on Semantic Web and Information Systems*, *16*(4), 81–99. DOI: 10.4018/IJSWIS.2020100105

Kumar, R., Lifshits, Y., & Tomkins, A. (2010). Evolution of two-sided markets. In *Proceedings of the third acm international conference on web search and data mining - wsdm '10* (pp. 311–319). ACM Press. DOI: 10.1145/1718487.1718526

Kumar, V., & Pedanekar, N. (2016). Mining shapes of expertise in online social q&a communities. In *Proceedings of the 19th acm conference on computer supported cooperative work and social computing companion* (pp. 317–320). ACM. DOI: 10.1145/2818052.2869096

Lal, S., Correa, D., & Sureka, A. (2014, dec). *Miqs: Characterization and prediction of migrated questions on stackexchange. In 21st asia-pacific software engineering conference*. IEEE. DOI: 10.1109/APSEC.2014.89

Latorre, N., Minelli, R., Mocci, A., Ponzanelli, L., & Lanza, M. (2015, sep). *Soda: the stack overflow dataset almanac. In 2015 ieee 5th workshop on mining unstructured data (mud)*. IEEE. DOI: 10.1109/MUD.2015.7327961

Lezina, G. E., & Kuznetsov, A. M. (2013). Predict closed questions on stackoverflow. In *Ceur workshop proceedings* (Vol. 1031, pp. 10–14). Ruzica Piskac. doi: DOI: 10.1.1.394.5678

Li, G., Zhu, H., Lu, T., Ding, X., & Gu, N. (2015). Is it good to be like wikipedia? Exploring the tradeoffs of introducing collaborative editing model to q&a sites. In *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing* (pp. 1080–1091). ACM. DOI: 10.1145/2675133.2675155

M,, Chen, L., Chen, Y., & Wang, J. (2020). Extracting core answers using the grey wolf optimizer in community question answering. *Applied Soft Computing, 90*, 106125. DOI: 10.1145/3180155.3180195

Lin, J.-W. (2018). Predicting the best answers for questions on stack overflow.

Linares-Vasquez, M., Dit, B., & Poshyvanyk, D. (2013). An exploratory analysis of mobile development´ issues using stack overflow. In *Proceedings of the 10th working conference on mining software repositories* (pp. 93–96). IEEE Press. DOI: 10.1109/MSR.2013.6624014

Ling, L., & Larsen, S. (2018). *Sentiment analysis on stack overflow with respect to document type and programming language* [Unpublished master's thesis]. KTH Royal Institute of Technology, Stockholm, Sweden.

Lopez-Nores, M., Blanco-Fernández, Y., Bravo-Torres, J. F., Pazos-Arias, J. J., Gil-Solla, A., & Ramos-Cabrer, M. (2019). Experiences from placing stack overflow at the core of an intermediate programming course. *Computer Applications in Engineering Education*, *27*(3), 698–707. DOI: 10.1002/cae.22109

Lord, L., Sell, J., Bagirov, F., & Newman, M. (2018). Survival analysis within stack overflow: Python and r. In *2018 4th international conference on big data innovations and applications (innovate-data)* (pp. 51–59). IEEE Press.

Lotufo, R., Passos, L., & Czarnecki, K. (2012, jun). Towards improving bug tracking systems with game mechanisms. In M. Lanza, M. D. Penta, & T. Xi (Eds.), *2012 9th IEEE working conference on mining software repositories (msr)* (pp. 2–11). IEEE. Retrieved from DOI: 10.1109/MSR.2012.6224293

Maity, S. K., Panigrahi, A., Ghosh, S., Banerjee, A., Goyal, P., & Mukherjee, A. (2019). Deeptagrec: A content-cum-user based tag recommendation framework for stack overflow.

Mastrangelo, L., Ponzanelli, L., Mocci, A., Lanza, M., Hauswirth, M., & Nystrom, N. (2015). Use at your own risk: the java unsafe api in the wild. In *Proceedings of the 2015 ACM SIGPLAN international conference on object-oriented programming, systems, languages, and applications - oopsla 2015* (pp. 695–710). ACM Press. DOI: 10.1145/2814270.2814313

May, A., Wachs, J., & Hannak, A. (2019). Gender differences in participation and reward on stack overflow.´. *Empirical Software Engineering*, *24*(4), 1–23. DOI: 10.1007/s10664-019-09685-x

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet. cs. *umass. edu*.

Meldrum, S., Licorish, S. A., & Savarimuthu, B. T. R. (2020). Exploring research interest in stack overflow– a systematic mapping study and quality evaluation. *arXiv preprint arXiv:2010.12282*.

Morrison, P., & Murphy-Hill, E. (2013). Is programming knowledge related to age? an exploration of stack overflow. In *Proceedings of the 10th international working conference on mining software repositories* (pp. 69–72). IEEE. DOI: 10.1109/MSR.2013.6624008

Movshovitz-Attias, D., & Cohen, W. W. W. (2013). Natural language models for predicting programming comments. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 35–40. Retrieved from https://www.aclweb.org/anthology/P13-2007\%5Cn

Movshovitz-Attias, D., Movshovitz-Attias, Y., Steenkiste, P., & Faloutsos, C. (2013). Analysis of the reputation system and user contributions on a question answering website: Stackoverflow. In *Proceedings of the 2013 ieee/acm international conference on advances in social networks analysis and mining* (pp. 886– 893). ACM. DOI: 10.1145/2492517.2500242

Mula, V. K. C., Kumar, L., Murthy, L. B., & Krishna, A. (2022). Software sentiment analysis using deeplearning approach with word-embedding techniques. In *2022 17th conference on computer science and intelligence systems (fedcsis)* (pp. 873–882).

Murgia, A. (2016). Among the machines: Human-bot interaction on social q & a websites. In *Chi extended abstracts on human factors in computing systems* (pp. 1272–1279). ACM. DOI: 10.1145/2851581.2892311

Mustafa, S., Zhang, W., & Naveed, M. M. (2023). What motivates online community contributors to contribute consistently? a case study on stackoverflow netizens. *Current Psychology (New Brunswick, N.J.)*, *42*(13), 10468–10481. DOI: 10.1007/s12144-022-03307-4 PMID: 35789627

Nasehi, S. M., Sillito, J., Maurer, F., & Burns, C. (2012). What makes a good code example?: A study of programming q&a in stackoverflow. In *28th IEEE international conference on software maintenance (icsm)* (pp. 25–34). IEEE Computer Society. DOI: 10.1109/ICSM.2012.6405249

Nishi, M. A., Ciborowska, A., & Damevski, K. (2019). Characterizing duplicate code snippets between stack overflow and tutorials. In *Proceedings of the 16th international conference on mining software repositories* (pp. 240–244). IEEE Press. DOI: 10.1109/MSR.2019.00048

Novielli, N., Calefato, F., & Lanubile, F. (2015). The challenges of sentiment detection in the social programmer ecosystem. In *Proceedings of the 7th international workshop on social software engineering - sse 2015* (pp. 33–40). ACM Press. DOI: 10.1145/2804381.2804387

Obaidi, M., Nagel, L., Specht, A., & Klunder, J. (2022). Sentiment analysis tools in software engineering:¨ A systematic mapping study. *Information and Software Technology*, *151*, 107018. DOI: 10.1016/j.infsof.2022.107018

Obiedat, R., Qaddoura, R., Al-Zoubi, A. M., Al-Qaisi, L., Harfoushi, O., Alrefai, M., & Faris, H. (2022). Sentiment analysis of customers' reviews using a hybrid evolutionary svm-based approach in an imbalanced data distribution. *IEEE Access : Practical Innovations, Open Solutions*, *10*, 22260–22273. DOI: 10.1109/ACCESS.2022.3149482

Oliveira, N., Muller, M., Andrade, N., & Reinecke, K. (2018, November). The exchange in stackexchange: Divergences between stack overflow and its culturally diverse participants. *Proc. ACM Hum.- Comput. Interact., 2*(CSCW), 130:1–130:22. DOI: 10.1145/3274399

Omondiagbe, O. P., Licorish, S. A., & MacDonell, S. G. (2019). Features that predict the acceptability of java and javascript answers on stack overflow. In *Proceedings of the evaluation and assessment on software engineering* (pp. 101–110). ACM. DOI: 10.1145/3319008.3319024

Pal, A., Harper, F. M., & Konstan, J. A. (2012, May). Exploring question selection bias to identify experts and potential experts in community question answering. *ACM Transactions on Information Systems*, *30*(2), 1–28. https://dl.acm.org/citation.cfm?doid=2180868.2180872. DOI: 10.1145/2180868.2180872

Parnin, C., & Treude, C. (2011). Measuring api documentation on the web. In *Proceedings of the 2nd international workshop on web 2.0 for software engineering* (pp. 25–30). ACM. DOI: 10.1145/1984701.1984706

Parnin, C., Treude, C., Grammel, L., & Storey, M.-A. (2012). *Crowd documentation: Exploring the coverage and the dynamics of api discussions on stack overflow*. Georgia Institute of Technology, Tech. Rep.

Paul, S. S., Tripathi, A., & Tewari, R. (2018). Social influence and learning pattern analysis: Case studies in stackoverflow. In *Advances in computer and computational sciences* (pp. 111–121). Springer.

Ponzanelli, L. (2012). *Exploiting crowd knowledge in the ide* [Doctoral dissertation, Universita della` Svizzera Italiana]. Retrieved from http://www.inf.usi.ch/faculty/lanza/Downloads/ Ponz2012a.pdf

Ponzanelli, L. (2014). Holistic recommender systems for software engineering. In *Companion proceedings of the 36th international conference on software engineering - icse companion 2014* (pp. 686–689). ACM. DOI: 10.1145/2591062.2591081

Ponzanelli, L., Bacchelli, A., & Lanza, M. (2013a, mar). *Leveraging crowd knowledge for software comprehension and development. In 17th European conference on software maintenance and reengineering*. IEEE Press. DOI: 10.1109/CSMR.2013.16

Ponzanelli, L., Bacchelli, A., & Lanza, M. (2013b). Seahawk: Stack overflow in the ide. In *Software engineering (icse), 2013 35th international conference on* (pp. 1295–1298). IEEE Press.

Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., & Lanza, M. (2014). Mining stackoverflow to turn the ide into a self-confident programming prompter. In *Proceedings of the 11th working conference on mining software repositories - msr 2014* (pp. 102–111). ACM. Retrieved from https://dl.acm.org/citation.cfm?doid=2597073 .2597077 DOI: 10.1145/2597073.2597077

Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., & Lanza, M. (2015). Prompter. *Empirical Software Engineering*, 1–42. DOI: 10.1007/s10664-015-9397-1

Ponzanelli, L., Bavota, G., Mocci, A., Di Penta, M., Oliveto, R., Hasan, M., & Lanza, M. (2016). Too long; didn't watch!: extracting relevant fragments from software development video tutorials. In *Proceedings of the 38th international conference on software engineering* (pp. 261–272). ACM. DOI: 10.1145/2884781.2884824

Ponzanelli, L., Bavota, G., Mocci, A., Di Penta, M., Oliveto, R., Russo, B., . . . Lanza, M. (2016). Codetube: extracting relevant fragments from software development video tutorials. In *Proceedings of the 38th international conference on software engineering companion - icse '16* (pp. 645–648). ACM Press. DOI: 10.1145/2889160.2889172

Ponzanelli, L., Bavota, G., Penta, M. D., Oliveto, R., & Lanza, M. (2014). *Prompter: A self-confident recommender system. In 2014 ieee international conference on software maintenance and evolution*. IEEE Press. DOI: 10.1109/ICSME.2014.99

Ponzanelli, L., Mocci, A., Bacchelli, A., & Lanza, M. (2014). *Understanding and classifying the quality of technical forum questions. In 4th international conference on quality software*. IEEE Press. DOI: 10.1109/QSIC.2014.27

Ponzanelli, L., Mocci, A., Bacchelli, A., Lanza, M., & Fullerton, D. (2014). Improving low quality stack overflow post detection. In *ieee international conference on software maintenance and evolution* (pp. 541–544). IEEE Press. DOI: 10.1109/ICSME.2014.90

Ponzanelli, L., Mocci, A., & Lanza, M. (2015a). Stormed: Stack overflow ready made data. In *IEEE international working conference on mining software repositories* (Vol. 2015-Augus, pp. 474–477). IEEE Press. DOI: 10.1109/MSR.2015.67

Ponzanelli, L., Mocci, A., & Lanza, M. (2015b). Summarizing complex development artifacts by mining heterogeneous data. In *IEEE international working conference on mining software repositories* (pp. 401–405). IEEE Press. DOI: 10.1109/MSR.2015.49

Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G., & Oliveto, R. (2019). Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*.

Rahman, A., Farhana, E., & Imtiaz, N. (2019). Snakes in paradise?: insecure python-related coding practices in stack overflow. In *Proceedings of the 16th international conference on mining software repositories* (pp. 200–204). IEEE Press. DOI: 10.1109/MSR.2019.00040

Rahman, M. M., Yeasmin, S., & Roy, C. K. (2013, oct). *An ide-based context-aware meta search engine. In 20th working conference on reverse engineering (wcre)*. IEEE Press. DOI: 10.1109/WCRE.2013.6671324

Raj, N., Dey, L., & Gaonkar, B. (2011, aug). Expertise prediction for social network platforms to encourage knowledge sharing. In Boissier, O., Benatallah, B., Papazoglou, M. P., Ras, Z. W., & Hacid, M.-S. (Eds.), *IEEE/WIC/ACM international conferences on web intelligence and intelligent agent technology* (pp. 380–383). IEEE Press. DOI: 10.1109/WI-IAT.2011.93

Rattigan, M. J. H., & Jensen, D. (2010). Leveraging d-separation for relational data sets. In *Proceedings ieee international conference on data mining, icdm* (pp. 989–994). IEEE Press. DOI: 10.1109/ICDM.2010.142

Reinhardt, A., Zhang, T., Mathur, M., & Kim, M. (2018). Augmenting stack overflow with api usage patterns mined from github. In *Proceedings of the 2018 26th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering* (pp. 880–883). ACM. DOI: 10.1145/3236024.3264585

Riahi, F. (2012). *Finding expert users in community question answering services using topic models* [Unpublished doctoral dissertation]. Dalhousie University, Halifax, Nova Scotia.

Rosen, C., & Shihab, E. (2016, June). What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering*, *21*(3), 1192–1223. DOI: 10.1007/s10664-015-9379-3

Roy, P. K., Ahmad, Z., Singh, J. P., & Banerjee, S. (2022). Feature extraction to filter out low-quality answers from social question answering sites. *Journal of the Institution of Electronics and Telecommunication Engineers*, 1–12.

Roy, P. K., & Singh, J. P. (2020). Predicting closed questions on community question answering sites using convolutional neural network. *Neural Computing & Applications*, *32*(14), 10555–10572. DOI: 10.1007/s00521-019-04592-0

Roy, P. K., Singh, J. P., & Banerjee, S. (2022). Is this question going to be closed? answering question closibility on stack exchange. *Journal of Information Science*.

S., Martinez, H., Elming, J., & Johannsen, A. (2013). Using crowdsourcing to get representations based on regular expressions. In *Empirical methods in natural language processing* (pp. 1476–1480). Association for Computational Linguistics. doi: DOI: 10.1.1.593.4529

Saha, A. K., Saha, R. K., & Schneider, K. A. (2013). A discriminative model approach for suggesting tags automatically for stack overflow questions. In *Proceedings of the 10th international working conference on mining software repositories* (pp. 73–76). IEEE Press. DOI: 10.1109/MSR.2013.6624009

San Pedro, J., & Karatzoglou, A. (2014). Question recommendation for collaborative question answering systems with rankslda. In *Proceedings of the 8th acm conference on recommender systems - recsys '14* (pp. 193–200). ACM Press. DOI: 10.1145/2645710.2645736

Schenk, D., & Lungu, M. (2013). Geo-locating the knowledge transfer in stackoverflow. In *Proceedings of the 2013 international workshop on social software engineering* (pp. 21–24). ACM. DOI: 10.1145/2501535.2501540

Schroth, L., Obaidi, M., Specht, A., & Klunder, J. (2022). On the potentials of realtime sentiment anal-¨ ysis on text-based communication in software projects. In *Human-centered software engineering: 9th ifip wg 13.2 international working conference, hcse 2022, eindhoven, the netherlands, august 24–26, 2022, proceedings* (pp. 90–109).

Selleras, R. Q. (2020). *Predictive model: Using text mining for determining factors leading to high-scoring answers in stack overflow* [Unpublished doctoral dissertation]. The George Washington University.

Silva, R. F., Paixao, K., & de Almeida Maia, M. (2018). Duplicate question detection in stack overflow: A reproducibility study. In *2018 IEEE 25th international conference on software analysis, evolution and reengineering (saner)* (pp. 572–581).

Sim, S. E., & Gallardo-Valencia, R. E. (2013). Finding source code on the web for remix and reuse. In Sim, S. E., & Gallardo-Valencia, R. E. (Eds.), *Finding source code on the web for remix and reuse* (pp. 289–330). Springer New York. DOI: 10.1007/978-1-4614-6596-6_16

Singer, L., Figueira Filho, F., Cleary, B., Treude, C., Storey, M.-A., & Schneider, K. (2013). Mutual assessment in the social programmer ecosystem. In *Proceedings of the 2013 conference on computer supported cooperative work - cscw '13* (pp. 103–115). ACM. DOI: 10.1145/2441776.2441791

Singh, P., & Shadbolt, N. (2013). Linked data in crowdsourcing purposive social network. In *Proceedings of the 22nd international conference on world wide web - www '13 companion* (pp. 913–918). ACM Press. DOI: 10.1145/2487788.2488079

Sinha, T., Wei, W., & Carley, K. (2015). Modeling similarity in incentivized interaction: A longitudinal case study of stackoverflow. *Nips 2015 workshop on social and information networks, 29th annual international conference on neural information and processing systems.*

Sinha, V. S., Mani, S., & Gupta, M. (2013). Exploring activeness of users in q&a forums. In *Proceedings of the 10th international working conference on mining software repositories* (pp. 77–80). IEEE Press. DOI: 10.1109/MSR.2013.6624010

Souza, L. B., Campos, E. C., Madeiral, F., Paixao, K., Rocha, A. M., & de Almeida Maia, M. ˜. (2019). Bootstrapping cookbooks for apis from crowd knowledge on stack overflow. *Information and Software Technology*, *111*, 37–49. DOI: 10.1016/j.infsof.2019.03.009

Squire, M., & Funkhouser, C. (2014). "A bit of code": How the stack overflow community creates quality postings. In *Proceedings of the 47th hawaii international conference on system sciences (hicss)* (pp. 1425– 1434). IEEE Press. DOI: 10.1109/HICSS.2014.185

Stanley, C., & Byrne, M. (2013). Predicting tags for stackoverflow posts. In *Proceedings of the the 12th international conference on cognitive modelling iccm 2013* (pp. 414–419). doi: DOI: 10.1.1.399.7835

Subramanian, S., & Holmes, R. (2013). Making sense of online code snippets. In *IEEE international working conference on mining software repositories* (pp. 85–88). IEEE., DOI: 10.1109/MSR.2013.6624012

Subramanian, S., Inozemtseva, L., & Holmes, R. (2014). Live api documentation. In *International conference on software engineering* (pp. 643–652). ACM. DOI: 10.1145/2568225.2568313

Sun, M. M., Ghosh, A., Sharma, R., & Kuttal, S. K. (2018). Birds of a feather flock together? a study of developers' flocking and migration behavior in github and stack overflow. *ArXiv, abs/1810.13062*.

Sushine, J., Herbsleb, J. D., & Aldrich, J. (2015, may). Searching the state space: A qualitative study of api protocol usability. In *IEEE 23rd international conference on program comprehension* (pp. 82–93). IEEE. DOI: 10.1109/ICPC.2015.17

Swillus, M., & Zaidman, A. (2023). Sentiment overflow in the testing stack: Analysing software testing posts on stack overflow. *arXiv preprint arXiv:2302.01037*.DOI: 10.1109/ICPC.2015.17

Tabassum, J., Maddela, M., Xu, W., & Ritter, A. (2020). Code and named entity recognition in stackoverflow. *arXiv preprint arXiv:2005.01634*. DOI: 10.1109/ICPC.2015.17

Tahir, A., Yamashita, A., Licorish, S., Dietrich, J., & Counsell, S. (2018). Can you tell me if it smells?: A study on how developers discuss code smells and anti-patterns in stack overflow. In *Proceedings of the 22nd international conference on evaluation and assessment in software engineering 2018* (pp. 68–78). DOI: 10.1145/3210459.3210466

Tian, F., Liang, P., & Babar, M. A. (2019, March). *How developers discuss architecture smells? an exploratory study on stack overflow. In 2019 IEEE international conference on software architecture (icsa)*. DOI: 10.1109/ICSA.2019.00018

Tian, Y., Kochhar, P. S., Lim, E.-P., Zhu, F., & Lo, D. (2013). Predicting best answerers for new questions: An approach leveraging topic modeling and collaborative voting. In *Workshops at the international conference on social informatics* (pp. 55–68). Springer.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the north American chapter of the association for computational linguistics on human language technology-volume 1* (pp. 173–180). DOI: 10.3115/1073445.1073478

Treude, C., Barzilay, O., & Storey, M.-A. D. (2011). How do programmers ask and answer questions on the web? In Taylor, R. N., Gall, H., & Medvidovic, N. (Eds.), *ICSE* (pp. 804–807). ACM. DOI: 10.1145/1985793.1985907

Treude, C., & Filho, F. F. (2012). Programming in a socially networked world: The evolution of the social programmer. In *Cscw workshop on the future of collaborative software development* (pp. 1–3). Retrieved from https://www.researchgate.net/publication/236647900\Programming\in\ a\Socially\Networked\World\the\ Evolution\of\the\Social\Programmer/ file/9c960518a67b6d04a3.pdf

Treude, C., & Robillard, M. P. (2016). Augmenting api documentation with insights from stack overflow. In *Proceedings of the 38th international conference on software engineering - icse '16* (pp. 392–403). ACM Press. DOI: 10.1145/2884781.2884800

Treude, C., & Robillard, M. P. (2017). Understanding stack overflow code fragments. In *Software maintenance and evolution (icsme), 2017 ieee international conference on* (pp. 509–513). DOI: 10.1109/ICSME.2017.24

Treude, C., & Wagner, M. (2018). Per-corpus configuration of topic modelling for github and stack overflow collections. *arXiv preprint arXiv:1804.04749*.DOI: 10.1109/ICSME.2017.24

Treude, C., & Wagner, M. (2019). Predicting good configurations for github and stack overflow topic models. In *Proceedings of the 16th international conference on mining software repositories* (pp. 84–95). DOI: 10.1109/MSR.2019.00022

Uddin, G., Guéhénuc, Y.-G., Khomh, F., & Roy, C. K. (2022). An empirical study of the effectiveness´ of an ensemble of stand-alone sentiment detection tools for software engineering datasets. *ACM Transactions on Software Engineering and Methodology*, *31*(3), 1–38. DOI: 10.1145/3491211

Uddin, G., & Khomh, F. (2019). Automatic mining of opinions expressed about apis in stack overflow. *IEEE Transactions on Software Engineering*. DOI: 10.1109/TSE.2019.2900245

Vasilescu, B., Capiluppi, A., & Serebrenik, A. (2012, dec). Gender, representation and online participation: A quantitative study of stackoverflow. In *International conference on social informatics* (pp. 332–338). IEEE. DOI: 10.1109/SocialInformatics.2012.81

Vasilescu, B., Capiluppi, A., & Serebrenik, A. (2013). Gender, representation and online participation: A quantitative study. *Interacting with Computers*, *26*(5), 488–511. DOI: 10.1093/iwc/iwt047

Vasilescu, B., Filkov, V., & Serebrenik, A. (2013). Stackoverflow and github: Associations between software development and crowdsourced knowledge. In *International conference on social computing* (pp. 188–195). IEEE. DOI: 10.1109/SocialCom.2013.35

Vasilescu, B., Serebrenik, A., Devanbu, P., & Filkov, V. (2014). How social q&a sites are changing knowledge sharing in open source software communities. In *Proceedings of the 17th acm conference on computer supported cooperative work & social computing* (pp. 342–354). DOI: 10.1145/2531602.2531659

Vasilescu, B., Serebrenik, A., & Van Den Brand, M. G. J. (2013). The babel of software development: Linguistic diversity in open source. In *Proceedings of the 5th international conference on social informatics* (*Vol. 8238* LNCS, pp. 391–404). Springer. DOI: 10.1007/978-3-319-03260-3_34

Venkataramani, R., Gupta, A., Asadullah, A., Muddu, B., & Bhat, V. (2013). Discovery of technical expertise from open source code repositories. In *Proceedings of the 22nd international conference on world wide web - www '13 companion* (pp. 97–98). ACM Press. DOI: 10.1145/2487788.2487832

Voß, J. (2012). Linking folksonomies to knowledge organization systems. In *Communications in computer and information science* (Vol. 343 CCIS, pp. 89–97). Springer. doi: DOI: 10.1007/978-3 -642-35233-1 9

Wang, B., Zhang, X., Du, K., Gao, C., & Li, L. (2023). *Multimodal sentiment analysis under modality deficiency with prototype-augmentation in software engineering. In 2023 IEEE international conference on software analysis, evolution and reengineering (saner).*

Wang, S., Chen, T.-H. P., & Hassan, A. E. (2018). How do users revise answers on technical q&a websites? a case study on stack overflow. *IEEE Transactions on Software Engineering*.

Wang, S., Lo, D., & Jiang, L. (2013). An empirical study on developer interactions in stackoverflow. In *Proceedings of the 28th annual acm symposium on applied computing* (pp. 1019–1024). ACM. DOI: 10.1145/2480362.2480557

Wang, S., Lo, D., Vasilescu, B., & Serebrenik, A. (2014). Entagrec: An enhanced tag recommendation system for software information sites. In *Software maintenance and evolution (icsme), 2014 IEEE international conference on* (pp. 291–300).

Wang, S., Lo, D., Vasilescu, B., & Serebrenik, A. (2018). Entagrec++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering*, *23*(2), 800–832. DOI: 10.1007/s10664-017-9533-1

Wang, W., & Godfrey, M. W. (2013, may). *Detecting api usage obstacles: A study of ios and android developer questions. In 10th working conference on mining software repositories (msr).* IEEE. DOI: 10.1109/MSR.2013.6624006

Wen, W., Zhang, B., Hu, Y., Zhu, X., & Wang, Z. (2024). An api recommendation method for querying mobile computing problems. *International Journal of Cognitive Informatics and Natural Intelligence*, *18*(1), 1–16. DOI: 10.4018/IJCINI.344422

Wilde, E. (2009). Feeds as query result serializations. *arXiv preprint arXiv:0911.2193*.DOI: 10.4018/IJCINI.344422

Winston, E. (n.d.). Case-qa: Context and syntax embeddings for question answering on stack overflow.

Winston, E., Dhingra, B., Mazaitis, K., Neubig, G., & Cohen, W. W. (n.d.). Answering cloze-style software questions using stack overflow.

Wong, E., Yang, J., & Tan, L. (2013, nov). *Autocomment: Mining question and answer sites for automatic comment generation. In 28th IEEE/ACM international conference on automated software engineering (ase).* IEEE. DOI: 10.1109/ASE.2013.6693113

Wu, D., Jing, X.-Y., Zhang, H., Feng, Y., Chen, H., Zhou, Y., & Xu, B. (2023). Retrieving api knowledge from tutorials and stack overflow based on natural language queries. *ACM Transactions on Software Engineering and Methodology*, *32*(5), 1–36. DOI: 10.1145/3565799

Wu, D., Jing, X.-Y., Zhang, H., Zhou, Y., & Xu, B. (2023). Leveraging stack overflow to detect relevant tutorial fragments of apis. *Empirical Software Engineering*, *28*(1), 12. DOI: 10.1007/s10664-022-10235-1

Wu, J., Ye, C., & Zhou, H. (2021). *Bert for sentiment classification in software engineering. In 2021 international conference on service science (icss)*.

Wu, L., Baggio, J. A., & Janssen, M. A. (2015, September). The dynamics of collaborative knowledge production. *PLoS One*, *11*(3), e0149151. DOI: 10.1371/journal.pone.0149151 PMID: 26934733

Wu, L., Baggio, J. A., & Janssen, M. A. (2016). The role of diverse strategies in sustainable knowledge production. *PLoS One*, *11*(3), e0149151. DOI: 10.1371/journal.pone.0149151 PMID: 26934733

Wu, Y., Wang, S., Bezemer, C.-P., & Inoue, K. (2019). How do developers utilize source code from stack overflow? *Empirical Software Engineering*, *24*(2), 637–673. DOI: 10.1007/s10664-018-9634-5

Xu, B., Shirani, A., Lo, D., & Alipour, M. A. (2018). Prediction of relatedness in stack overflow: Deep learning vs. svm: A reproducibility study. In *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement* (pp. 21:1–21:10). ACM. DOI: 10.1145/3239235.3240503

Xu, L., Nian, T., & Cabral, L. (2014). What makes geeks tick? a study of stack overflow careers.

Xu, L., Nian, T., & Cabral, L. (2016). What makes geeks tick ? A study of stack overflow careers.

Xu, L., Nian, T., & Cabral, L. (2019). What makes geeks tick? a study of stack overflow careers. *Management Science*.

Xu, L., Nian, T., & Cabral, L. (2018). *What makes geeks tick? a study of stack overflow careers* (Tech. Rep.).

Xu, S., Bennett, A., Hoogeveen, D., Lau, J. H., & Baldwin, T. (2018, November). Preferred answer selection in stack overflow: Better text representations ... and metadata, metadata, metadata. In *Proceedings of the 2018 EMNLP workshop w-NUT: The 4th workshop on noisy user-generated text* (pp. 137–147). Association for Computational Linguistics. DOI: 10.18653/v1/W18-6119

Yao, Y., Tong, H., Xie, T., Akoglu, L., Xu, F., & Lu, J. (2013). Want a good answer? ask a good question first! *ArXiv e-prints*. Retrieved from http://arxiv.org/abs/1311.6876DOI: 10.18653/v1/W18-6119

Yao, Z., Weld, D. S., Chen, W.-P., & Sun, H. (2018). Staqc: A systematically mined question-code dataset from stack overflow.

Ye, D., Xing, Z., Foo, C. Y., Ang, Z. Q., Li, J., & Kapre, N. (2016, mar). *Software-specific named entity recognition in software engineering social content. In 2016 IEEE 23rd international conference on software analysis, evolution, and reengineering (saner)*. IEEE. DOI: 10.1109/SANER.2016.10

Ye, D., Xing, Z., & Kapre, N. (2016, April). The structure and dynamics of knowledge network in domainspecific q&a sites: A case study of stack overflow. *Empirical Software Engineering*, 1–32. DOI: 10.1007/s10664-016-9430-z

Ye, D., Xing, Z., Li, J., & Kapre, N. (2016). Software-specific part-of-speech tagging. In *Proceedings of the 31st annual acm symposium on applied computing - sac '16* (pp. 1378–1385). ACM Press. DOI: 10.1145/2851613.2851772

Ye, Y., Hou, S., Chen, L., Li, X., Zhao, L., Xu, S., & Xiong, Q. (2018). Icsd: An automatic system for insecure code snippet detection in stack overflow over heterogeneous information network. In *Proceedings of the 34th annual computer security applications conference* (pp. 542–552). DOI: 10.1145/3274694.3274742

Yin, H., & Pfahl, D. (2017a). A method to transform automatically extracted product features into inputs for kano-like models. In *International conference on product-focused software process improvement* (pp. 237–254). Springer. DOI: 10.1007/978-3-319-69926-4_17

Yin, H., & Pfahl, D. (2017b). A preliminary study on the suitability of stack overflow for open innovation in requirements engineering. In *Proceedings of the 3rd international conference on communication and information processing* (pp. 45–49). ACM. DOI: 10.1145/3162957.3162965

Yin, P., Deng, B., Chen, E., Vasilescu, B., & Neubig, G. (2018). Learning to mine parallel natural language/source code corpora from stack overflow. In *Proceedings of the 40th international conference on software engineering: Companion proceeedings* (pp. 388–389). ACM. DOI: 10.1145/3183440.3195021

Zhang, H., Wang, S., Chen, T.-H. P., & Hassan, A. E. (2019). Does the hiding mechanism for stack overflow comments work well? no! *arXiv preprint arXiv:1904.00946*.DOI: 10.1145/3183440.3195021

Zhang, H., Wang, S., Chen, T.-H. P., Zou, Y., & Hassan, A. E. (2019). An empirical study of obsolete answers on stack overflow. *IEEE Transactions on Software Engineering*.

Zhang, T., Upadhyaya, G., Reinhardt, A., Rajan, H., & Kim, M. (2018). Are code examples on an online q&a forum reliable?: a study of api misuse on stack overflow. In *2018 IEEE/ACM 40th international conference on software engineering (icse)* (pp. 886–896). IEEE Press.

Zolaktaf, Z., Riahi, F., Shafiei, M., & Milios, E. (2011). Modeling community question-answering archives. *Cs.Dal.Ca*, 1–5. Retrieved from https://web.cs.dal.ca/˜eem/cvWeb/pubs/2011 -Zolaktaf-NIPS.pdf

Xiaochun Cheng won full scholarship for all his university education, received the BEng Degree in Computer Engineering with first class degree at national best (at the time) Computer Science Department, PhD in Computer Science with distinction at national best (at the time) Computer Science Department as the Youngest PhD awardee of the university at the time. Attended work based training and completed Executive MBA. One project was funded with 16 million Euro budget. Contributed for five times best conference paper awards so far. 6 papers were in the top 1% of the academic field by Data from Essential Science Indicators. Won 3 times national competitions. Won a national award for research. Two solutions achieved national best results and were adopted nationally.