Exploring Developers Discussion Forums for Quantum Software Engineering: A Fine-Grained Classification Approach Using Large Language Model (ChatGPT)

Mobashir Husain
Muhammad Sohail Khan
mobashir70ustb@gmail.com
sohail.khan@uetmardan.edu.pk
Department of Computer Software Engineering, University of Engineering and Technology
Marden, Pakistan

Javed Ali Khan
j.a.khan@herts.ac.uk
Department of Computer Science, University of Hertfordshire
Hatfield, United Kingdom

Nek Dil Khan
nekdilkhan@emails.bjut.edu.cn
Faculty of Information Technology, Beijing University of Technology
Beijing, China

Arif Ali Khan
Arif.khan@oulu.fi
M3S Empirical Software Engineering Research Unit, University of Oulu
Oulu, Finland

Muhammad Azeem Akbar
Azeem.akbar@lut.fi
Department of Software Engineering, LUT University, Finland
Lappeenranta, Finland

Abstract

Quantum Software Engineering (QSE) has recently emerged as a potential research and development direction frequently practiced by many tech joints. However, quantum developers face challenges in optimizing quantum computing and QSE concepts. Quantum developers use the Stack Overflow (SO) platform to report and discuss quantum-related challenges. Also, quantum practitioners use specialized quantum tags to label quantum-related posts in developers' forums. However, these quantum tags referred to more technical quantum aspects than the developer posts. Therefore, categorizing quantum practitioners' questions based on quantum concepts can help quantum developers better identify frequently occurring challenges to QSE. For this purpose, we conducted qualitative and quantitative studies to classify quantum developers' questions into various frequently occurring quantum-related challenges. We extracted 2829 developers' questions from various Q&A platforms using queries and filters based on quantum-related tags. Next, the developers' posts on the Q&A forums were critically analyzed to identify frequently discussed quantum-related challenges and develop a novel grounded theory. The frequent quantum developer challenges identified by analyzing practitioners' discussions in the Q&A forums include Tooling, Theoretical, Learning, Conceptual, Errors, and API Usage. Moreover, using content analysis and grounded theory, the developers' discussions were annotated with commonly reported quantum challenges to develop a ground truth and a novel dataset. A Large Language model (ChatGPT) was used to validate the human annotation and overcome disagreements. Finally, various fine-tuned Deep and Machine learning (D&ML) classifiers automatically classify developer discussions into commonly reported quantum challenges. Additionally, to improve the classification results, we utilized textual data augmentation approaches, such as random deletion, swapping, and insertion with the D&ML classifiers. We obtained average accuracies of 89%, 86%, 84%, 84%, and 80% with FNN, CNN, LSTM, GRU, and RNN classifiers, respectively. This helps quantum researchers and vendors propose solutions and tools to frequently occurring issues for quantum developers.

CCS Concepts

• **Software and its engineering → Software libraries and repositories**.

Keywords

Quantum Software Engineering, Repository mining, developer forums, Stack overflow, machine and deep learning, Natural language processing.

ACM Reference Format:
Mobashir Husain, Muhammad Sohail Khan, Javed Ali Khan, Nek Dil Khan, Arif Ali Khan, and Muhammad Azeem Akbar. 2025. Exploring Developers Discussion Forums for Quantum Software Engineering: A Fine-Grained Classification Approach Using Large Language Model (ChatGPT). In *Companion Proceedings of the 33rd ACM Symposium on the Foundations of Software Engineering (FSE '25), June 23–27, 2025, Trondheim, Norway*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3696630.3731625

FSE Companion '25, Trondheim, Norway

ACM ISBN 979-8-4007-1276-0/25/06
https://doi.org/10.1145/3696630.3731625

1 Introduction

Quantum computing leverages quantum mechanics principles to describe the behavior of particles at the atomic and subatomic levels. Unlike classical computers, which use bits (0s and 1s), quantum computers use quantum bits or "qubits" to process information [35]. Superposition allows qubits to exist in a combination of 0

and 1 states simultaneously, increasing the computational power of quantum systems [37]. Google [7], IBM [18], and Rigetti [38] have demonstrated the ability to create qubits in controlled environments, enabling practical quantum computing. This provides opportunities to solve complex computational problems requiring parallel computation, which is challenging for classical computers. It can resolve complex issues in big data analytics, financial systems, national security, chemistry, cryptography, health analytics and medicine [36].

Quantum software engineering processes must be aligned with hardware advancements to leverage quantum computing. Developing methodologies, processes, tools, and algorithms is crucial for quantum computing domains [49]. Quantum programming languages like Q# [41], Scaffold [21], and Qiskit [19] exist for applications. Microsoft, Google, and IBM have developed cloud platforms that support quantum software development. IBM's platform enables the execution of applications on quantum systems. Developers create quantum software for domains such as chemistry and machine learning [33]. Given the importance of applications, software engineering processes are required for large projects. The QSE was coined to meet the demands for large applications [49]. Initial approaches for quantum requirements engineering [48], quantum software architecture [2], Quantum Development [3], and Quantum Testing [4] are emerging. Developers use social media and Q&A forums to extract information and address challenges [24]. Researchers have analyzed Q&A platform discussions to tackle challenges [33] [22]. These forums provide resources, communities, and problem-solving. QSE are valuable for sharing experiences in less-resourced areas. They enable collaboration, updates on trends, and collective problem-solving. This motivated the exploration of Q&A platform discussions to identify quantum categories such as Tooling and Conceptual, helping developers find information on frequent challenges.

In this study, we evaluate developer discussions on QSE across Q&A platforms like Stack Overflow (SO), Quantum Computing Stack Exchange (QCSE), Computer Science Stack Exchange (CSSE), and Artificial Intelligence Stack Exchange (AISE), categorizing them into challenges faced by software practitioners. This aids quantum developers in finding solutions to specific challenges on these platforms, organized by tags such as qiskit, qcl, qutip, qubit, and tensorflow-quantum. We extracted 2829 developers' questions using quantum-related tags, following El-Aoun et al.'s method [33]. The posts were analyzed to identify frequently discussed quantum challenges and develop a novel grounded theory, cross-validated with those by El-Aoun et al. [33] using topic modeling. Frequent challenges include Tooling, Theoretical, Learning, Conceptual, Errors, and API Usage. Through content analysis and grounded theory, developers' discussions were annotated with common quantum challenges to create a ground truth dataset. ChatGPT validated human annotations and resolved disagreements. Various fine-tuned Deep and Machine learning (D&ML) classifiers were used to automatically classify developer discussions into common quantum challenges. The key contributions of this study are as follows.

- Developed a novel dataset to classify developers' discussions into various frequently reported challenges related to QSE.
- A novel ground truth for quantum-related classification approaches, which, according to our knowledge, is the first annotated dataset to date.
- ChatGPT is utilized to overcome the coders' disagreements to ensure the annotation's validity.
- D&ML algorithms are fine-tuned by identifying hyperparameters to improve accuracy.
- A data augmentation approach is used to improve the performance of the D&ML classifiers by approximately 20%.
- D&ML classifiers are tested and validated on a dataset curated from multiple discussion Q&A forums to ensure the generalizability of the proposed approach.

The paper is organized as follows: Section 2 provides a related work. Section 3 outlines the research methodology. Section 4 details the classification and analysis of challenges faced in QSE. Section 5 explores automated methods for classifying developer discussions. Section 6 discusses the insights derived from developer exchanges on platforms like Stack Exchange. Section 7 concludes the paper and discusses possible future directions.

2 RELATED WORK

This section reviews the pertinent literature on quantum computing, QSE, and mining software repositories.

2.1 Software Quantum-Based Approaches

This subsection reviews the literature on software quantum computing approaches and techniques for quantum software development. Gill et al.[15] provided a systematic review of Quantum Computing (QC), which offers computational advantages over classical computing by leveraging quantum mechanical principles. They highlighted QC's potential to address complex problems in drug design, data science, clean energy, finance, industrial chemical development, secure communications, and quantum chemistry. Akbar et al. [3] proposed a perspective on Quantum Software Engineering (QSE), outlining its lifecycle stages and providing a framework for quantum necessity engineering, software implementation, design, testing, and maintenance. El-Aoun et al.[33] explored challenging aspects of QSE, examining quantum code theory and the gap between quantum and classical computing. Vietz et al. [47] analyzed tools supporting quantum application development, providing categorization and analysis to aid developers in selecting suitable tools. Haghparast et al.[16] explored quantum software engineering challenges, mapped research challenges to the quantum computing workflow model, and identified directions for software engineering research.

2.2 Mining Q&A repositors

We elaborate on the approaches that mine Q&A repositories to enhance developers' understanding. Beyer et al. [9] proposed a method to classify developer questions in SO forums into seven categories. They gathered data from SO forums on Android developers' questions and classified them using ML classifiers and expressions. Treude et al. [44] studied how programmers ask and answer questions on SO, analyzing discussions to categorize question types and assess answers. Their findings show the utility of Q&A sites for code reviews and conceptual questions. Iftikhar et al.[20]

proposed a deep learning approach for predicting correct answers in SO discussions. They extracted metadata and question/answer combinations, applied NLP techniques, and used keyword ranking for feature vectors. Their ensemble deep learning model surpassed state-of-the-art methods, improving the accuracy, precision, recall, and f-measure by 1.72%, 24.96%, 6.57%, and 16.62%, respectively. Zhu et al. [50] studied SO discussions, noting active participation in Q&A. They found a strong link between question comments and the response time. Similarly, [25] analyzed Q&A forum discussions to gain insights into software development methods. El-Aoun et al.[33] analyzed Q&A platforms to uncover challenges faced by quantum developers. Khan et al. [5] proposed an automated approach for extracting requirement-related information from forums, using ML and NLP to categorize discussions. Beyer et al. manually categorized 450 SO posts on Android app development [11]. They found 'How to?' and 'What is the problem?' were common, with issues related to the User Interface and core elements. Their study revealed correlations between the problem and question types.

2.3 Comparison with Existing Literature

The study complements mining software repositories and analyzing developer discussions to improve practitioner understanding. Inspired by El-Aoun et al. and Beyer et al. [9], our approach differs from previous studies. El-Aoun et al.[33] used topic modelling on Q&A forums to identify challenges, while Bayer et al. [9] classified developers' comments related to Android development into fine-grained categories. We critically analyzed QSE discussions from multiple forums to develop a taxonomy, validating it against El-Aoun et al.[33]'s findings. We fine-tuned ML and DL classifiers to classify QSE discussions into challenges like tooling, theory, and API usage. Unlike Beyer et al. [9], we aimed for a generalizable approach using multiple QSE forums. We employed ChatGPT to validate coder disagreements and automate annotation. With limited QSE datasets compared to Android discussions, we used augmentation to improve classifier performance. Table 1 compares existing approaches.

3 PROPOSED RESEARCH METHODOLOGY

In this section, we first elaborate on the research questions to answer the proposed research methodology. Next, we describe the proposed research approach, which includes the research dataset, novel grounded theory, content analysis, and various classifiers used to classify developer discussions into frequently reported challenge types.

3.1 Research Questions

This study aims to develop an automated methodology for systematically classifying developer discussions into identified challenges using fine-tuning D&ML algorithms. The proposed approach will empower quantum developers with structured resources, enabling them to find meaningful information about the quantum challenges confronting them. We developed the following research questions to validate the proposed methodology:

RQ1: What do quantum developers discuss about various challenges in different developers' discussion forums?

RQ2: What frequent QSE challenges can be identified from the developer's discussion?

RQ3: Can ChatGPT work as an annotator and negotiator in developing a ground truth for D&ML classifiers?

RQ4: How do various fine-tuned D&ML algorithms perform in classifying developers' discussions into various QSE challenges?

In pursuit of RQ-1, we aim to analyze developer questions in various forums, including SO, QCSE, CSSE, and AISE, to identify common patterns in how quantum developers express concerns or challenges regarding QSE. This will result in a novel grounded theory to identify frequently occurring challenges related to QSE. RQ2 utilizes the grounded theory and content analysis approach to develop a ground truth for RQ4 by manually annotating developers' discussions into various frequently occurring challenges. For RQ3, we seek to determine whether ChatGPT can annotate developers' discussions about QSE compared to human annotators and explore its use as a negotiator to resolve conflicts between annotators. This aims to overcome the manual complexity of annotating developer datasets for classification tasks. For RQ4, we aim to evaluate and compare various D&ML algorithms by exploring different feature engineering and natural language approaches to fine-tune these algorithms to better classify developers' comments into SQE challenges identified through RQ1.

3.2 Research method

The proposed research methodology for classifying developers' discussions from various social media platforms into QSE challenges comprises four main phases, shown in Figure 1; each methodological step is elaborated below.

3.2.1 Research Data Gathering and Curation. For the proposed approach, a research dataset on QSE is developed by systematically collecting developers' discussions from SO, QCSE, CSSE, and AISE forums due to their importance in software engineering literature [1], [9]. Recently, El Aoun et al. have explored these platforms for identifying quantum-related challenges using topic modeling [33]. These Q&A platforms prove an important source for improving various software engineering activities [1]. As of June 2024, SO had over 23 million, AISE had 73,124, QCSE had 23,740, and CSSE had 140,047 registered users. The SO repository, with over 24 million questions and 35 million answers, is the largest question-and-answer platform for software developers. We focused on collecting developers' discussions about quantum computing, programming languages, algorithms, and tools from these platforms.

We navigated the SO, AISE, QCSE, and CSSE forums using tailored queries to extract developer discussions relevant to QSE. For SO, queries targeted questions tagged with identifiers like post-quantum-cryptography, q#, quantum-computing, qiskit, qcl, qutip, qubit, and TensorFlow quantum, based on El-Aoun et al.[33]. These queries retrieved accepted and non-accepted answers, ensuring a comprehensive view of QSE discussions. Using SEDE, data from August 2008 to August 2024 was extracted. SO had 627 questions with 282 accepted answers. For QCSE, CSSE, and AISE, queries focused on tags like programming, classical computing, and q#. QCSE had 1806 questions with 500 accepted answers, CSSE had 379

Table 1: Summary of Methodologies for Automated Developer Question Classification

Ref.	Algorithms Used	Proposed Methodology	Dataset Used and Size	Key Contributions	QC/QSE?
Beyer et al. [9]	RF, SVM	Automated classification via taxonomy harmonization (7 categories), regex/ML.	1,000 Android SO posts	Improved question search/browsing for developers.	No
Vietz et al. [47]	-	Quantum dev tools taxonomy + comparison framework.	-	Structured quantum tech selection for developers.	Yes
Khan et al. [22]	LDA	Mixed-methods analysis (13,903 posts) with topic modeling.	13,903 SO/SESE/PMSE posts	Identified common dev approaches/challenges.	No
El Aoun et al. [33]	Topic Modeling	QSE challenge identification via Stack Exchange/GitHub.	Stack Exchange + GitHub	Highlighted QSE-specific challenges.	Yes
Proposed Work	**FNN, CNN, LSTM, GRU, RNN**	**Classification of 2,829 quantum dev questions + ChatGPT validation + data augmentation.**	2,829 Q&A posts	QSE challenge taxonomy + LLM validation + enhanced classification.	**Yes**

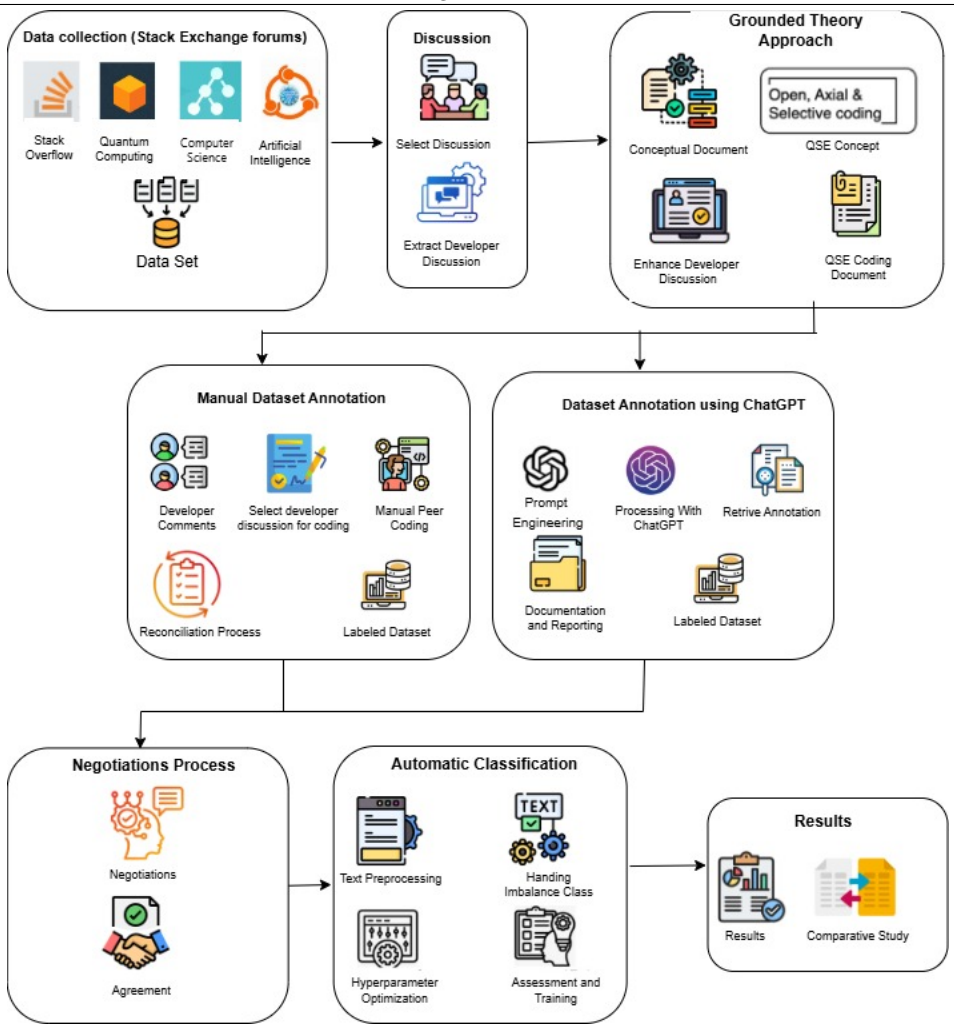

Figure 1: Overview of the proposed research methodology process.

questions with 124 accepted answers, and AISE had 17 questions with 4 accepted answers, as shown in Table 2.

3.2.2 Grounded theory approach . In the previous step, developers' QSE discussions were collected from various Q&A forums as raw data. To make this data parusable, we manually selected a sample of 300 developers' discussions to critically analyze it and identify frequently occurring challenges related to QSE and developed a grounded theory for QSE. For this purpose, we employed the corbins-grounded theory framework [40] to develop a coding guideline, which works as an input for preparing an annotated dataset for D&ML algorithms. Grounded theory constructs a theoretical framework based on recurring evidence from the QSE dataset. The

Table 2: Dataset for Machine Learning Experiments

Topic	Total Questions	Tag Set	Data Source
Stack Overflow	627	post-quantum cryptography, q#, quantum-computing, qiskit, qcl, qutip, qubit, TensorFlow-quantum	Stack Exchange
Quantum Computing	1806	programming, classical computing, q#, qiskit, cirq, ibm-q-experience, machine-learning, qutip	Stack Exchange
Computer Science	379	quantum-computing	Stack Exchange
Artificial Intelligence	17	quantum-computing	Stack Exchange
Total	**2829**	-	-

novel ground Theory for QSE involves frequently mentioned challenges quantum practitioners face in Q&A forums. The frequently reported challenges identified are Tooling, Theoretical, Learning, Conceptual, Errors, and API Usage. The identified quantum concepts were verified with challenges identified by El-Aoun et al.[33] using a topic modeling approach. Compared to El-Aoun et al.[33], we discarded three challenges: discrepancy, review, and API change, due to insufficient evidence in developers' discussions. Some concepts occur repetitively; they are merged into frequently occurring challenges. For example, discussions about unexpected results overlap with error troubleshooting and are represented with the error quantum concept. Discussions related to "API change" are grouped in the conceptual category. The "Review" challenge type can be intertwined with the SQE "Conceptual" or "API usage" categories. The quantum grounded theory process was developed iteratively with discussions by the first and fourth authors of the paper. The document helps annotators label the raw developer discussion for D&ML classifiers and minimize discrepancies among coders when annotating the dataset for D&ML learning experiments.

3.2.3 Analyzing content manually. After developing grounded theory, raw developer discussions from Q&A platforms must be annotated to make a purusable dataset for D&ML classifiers. Using the content analysis approach [34] and grounded theory document, developers' discussions are manually annotated. For this purpose, 2829 developers' discussions on QSE collected from Q&A platforms were added in a Microsoft Excel document to be annotated by the first and fourth authors of the paper using grounded theory document and content analysis approach. Moreover, content analysis is challenging and time-consuming due to human annotator involvement [27, 28]. Therefore, to possibly automate and improve the manual annotation process, the same comments were annotated using ChatGPT to compare its performance with human annotators. Annotation forms from human coders were merged to identify interceding agreements and cohesion kappa to assess ChatGPT's applicability in annotating QSE-related comments. To resolve conflicts, another round was conducted between human coders and ChatGPT to agree on comment annotations. The detailed process is described in section 4.2.

3.2.4 Classifying Developer Discussions using D&ML classifiers). After annotating QSE-related developer discussions, we assessed existing fine-tuned D&ML classifiers in categorizing these discussions into QSE-related challenge types identified via a grounded theory approach. We performed the following: A preprocessing pipeline removed special characters, URLs, and symbols using regular expressions. NLP techniques, like stemming and removing stop words, enhanced readability for D&ML algorithms. Feature engineering approaches, such as CountVectorizer and TFIDF, were employed to evaluate their effectiveness in classifying discussions. For deep learning classifiers, various feature values were experimented with and fine-tuned for improved results. Data-balancing methods, like under-sampling and over-sampling, addressed data imbalance across challenges. Data augmentation enhanced classifier performance, including random word deletion with a 20% retention probability, random word swapping up to two times, and random insertion of synonyms. A cross-validation approach trained and validated D&ML classifiers. Also, D&ML model performance was evaluated using the metrics of precision, recall, F1-score, ROC, and learning curves.

4 PROCESSING DEVELOPERS DISCUSSION

This section outlines the process of identifying the various challenges that quantum developers face while developing quantum-related applications. In addition, we shed light on how to annotate developer discussions collected from various Q&A platforms into the challenges identified previously using the grounded theory approach. Below, we elaborated on the process.

4.1 Statements of Developers

Organizing information on these Q&A platforms can help developers identify relevant information promptly. QSE is a new area where quantum developers frequently face challenges [33]. Therefore, we analyze developers' discussions to classify them into frequent challenges that quantum developers face, including tooling, theory, learning, concepts, errors, and API usage. The related quantum codes identified during the annotation emerged as concepts, resulting in a novel quantum theory for software developers. Concepts in the quantum coding guideline were identified by their consistent presence in discussions and relevance to the research approach. If a quantum concept appears less frequently in discussions, it merges into a related quantum concept. For example, the "API change" concept can merge into the "Conceptual category," aligning with the definition where developers ask about the API working. Below, we discuss the identified quantum concepts.

4.1.1 Conceptual. The code "Conceptual" is assigned to developers' discussions on Q&A platforms focusing on understanding the background, limitations, and concepts of quantum programming APIs. Analysis shows that quantum developers discuss quantum computing history or debate the feasibility of certain concepts. Researchers categorize discussions into conceptual categories if questions contain terms like why, is possible [11], what [39], how/why [6], and understanding concepts [9]. The definition for categorizing discussions aligns with approaches by Beyer and Pinzger [11], Rosen and Shihab [39], Allamanis and Sutton [6], and Beyer et al. [9]. An example is in Row 2 of Table 3.

4.1.2 Theoretical. The code "Theoretical" is assigned to Q&A platform discussions on quantum programs, algorithms, and principles. Developers may inquire about the mathematical foundations of

Table 3: Quantum Concepts with Examples from Q&A Forums

Concept	Developers' Discussion Examples
Conceptual	Quantum Computing and Encryption Breaking: "I read that Quantum Computers can break most encryption... How? I get lost at quantum bits being 1, 0, or both. Can someone explain in plain English without the math?"
Theoretical	NP-hard Cryptography: "Are there public key algorithms based on NP-complete problems rather than factorization/discrete logs, in case quantum computing becomes practical?"
Tooling	QASM Simulation Issues: "Using IBM Quantum Experience: Dragging gates is slow, no Toffoli gate. Need QASM editor/documentation. Found these links but unclear which QASM version IBM uses..."
Learning	Book Request: "Looking for a book covering Shor's algorithm, McEliece cryptosystem, Lattice-based crypto, and Discrete logarithms. Any recommendations?"
Error	Q# Build Problem: "Getting error QS1001: Microsoft.Quantum.Canon.dll not found. Missing .nuspec file. Nuget restore didn't help. Solutions?"
API Usage	Break Equivalent in Q#: "How to exit a loop when condition met? In C#: if(i==3) break. What's the Q# equivalent?"

quantum mechanics or seek explanations for quantum algorithms. For instance, a developer might ask how superposition or entanglement affects quantum algorithm design, as shown in Table 3, row 3. It discusses the theoretical implications of practical quantum computing on public-key cryptographic algorithms and complexity theory. El-Aoun et al.[33] identified that developers often discuss theoretical problems related to quantum computing in Q&A discussions using a topic modelling approach.

4.1.3 Tooling. This category pertains to developer discussions on tools and software usage in quantum development. The code "tooling" is assigned to developers' discussions on Q&A platforms that discuss possible quantum tools, seek expertise, or help with quantum-related tools. Quantum computing is an emerging discipline where vendors develop new tools to solve quantum-related problems. During annotation, quantum developers often seek guidance on using specific software tools or frameworks for quantum development. An example is shown in Table 3, Row 4. This category aligns with the El-Aoun et al. [33] approach, which identified "Tooling" as a frequently occurring quantum challenge developers discuss in Q&A forums.

4.1.4 Learning. In the developer discussions dataset where quantum practitioners seek resources about quantum computing from the Q&A community, we assigned the code "learning." This category includes requests for learning resources, tutorials, and references for quantum computing. Developers often seek recommendations for books, online courses, or other resources to enhance their understanding of quantum programming languages and algorithms. For instance, a quantum developer inquires about the best resources for learning Q# or mastering Grover's search algorithm, as shown in Row 5 of Table 3. The "Learning" category for the quantum classification taxonomy aligns with Beyer et al. [9] and El-Aoun et al. [33] "Learning" category identified for android-related topics and quantum challenges using topic modelling, respectively. Also, Allamanis and Sutton [6] identified learning a language/Technology category with different purposes and intentions.

4.1.5 Errors. Quantum computing is a new research and development area with limited expertise in quantum programming compared to traditional paradigms. We identified that quantum developers often seek help resolving errors while developing quantum applications. We assigned the code "errors" to developers' posts on Q&A platforms where quantum practitioners ask for solutions to errors. During manual annotation, we found that quantum developers often encounter challenges related to quantum gates, qubit decoherence, and compatibility issues with quantum hardware. For example, a practitioner asked for help debugging quantum circuits or understanding error messages from a quantum simulator, as shown in row 6 of Table 3. This category aligns with Beyer et al.[9], El-Aoun et al. [33] "Errors", and Beyer and Pinzger [11] "Error and Exception Handling" category.

4.1.6 API Usage. As quantum computing evolves, numerous APIs have been developed for various functionalities that quantum developers can utilize. The code "API usage" is assigned to developers' discussions in the Q&A seeking help in efficiently implementing a particular Quantum-related API. During manual annotation, it was noted that discussions focused on using APIs for quantum programming tasks. Developers seek guidance on integrating quantum computing platforms and services into their projects using APIs. For example, a developer may inquire about best practices for interfacing with the IBM Quantum Experience API or accessing the Rigetti Forest API for quantum program execution, as shown in Table 3, row 7. This category aligns with those of El-Aoun et al.[33] and Beyer et al.[9] API usage category, Allamanis and Sutton's [6] "Do not work" category, and Beyer and Pinzger's [11] "How to" category.

4.2 Developers Statement Labeling

To annotate developer discussions from Q&A platforms, we used content analysis [34] and quantum grounded theory, examining each developer's question to identify their quantum challenge category. The goal is to develop a truth set for training and testing the fine-tuned D&ML algorithms to categorize developer questions into quantum categories automatically. The annotation process is described below:

4.2.1 Annotation of Developer Discussions. Annotation was conducted in two steps. First, the first and fourth authors individually annotated 2829 developers' discussions from Q&A forums. Second, ChatGPT annotated the same dataset to compare its performance with human coders. Human annotators were given a coding guideline and form with developer comments. Coders assessed the title and content to determine the quantum category: tooling, theoretical, learning, conceptual, errors, or API Usage. Table 4 shows an overview of the coding process. Developers' questions were grouped by platform, including answers for clarity. The annotation process was iterative, as shown in Figure 1. The average annotation time was 44 working hours. The coders start and resume annotation at any time. After individual annotation, the results were combined to analyze intercoder disagreements. The intercoding agreement was 90.81%, with Cohen's kappa at 44.97%, indicating substantial

agreement. Conflicts were resolved through discussion, with a senior software engineering expert (third author) intervening for persistent disagreements. To validate the annotation process, we used ChatGPT to annotate developers' discussions into quantum categories and compared its performance with human annotators. We trained ChatGPT with definitions of quantum challenge types from novel quantum-grounded theory using the existing prompt. Figure 2 shows the detailed annotation process. We supplied each developer's discussion to ChatGPT and identified its possible quantum challenge type with a rationale. We recorded the number of discussions where manual annotators and ChatGPT agreed and disagreed. Table 5 shows detailed annotation by human coders and ChatGPT, agreeing on 2547 discussions for the same quantum challenge category and disagreeing on 282, resulting in potential conflicts. The Cohen's kappa was 0.46, indicating substantial agreement between human coders and ChatGPT on the Cohen's kappa scale.

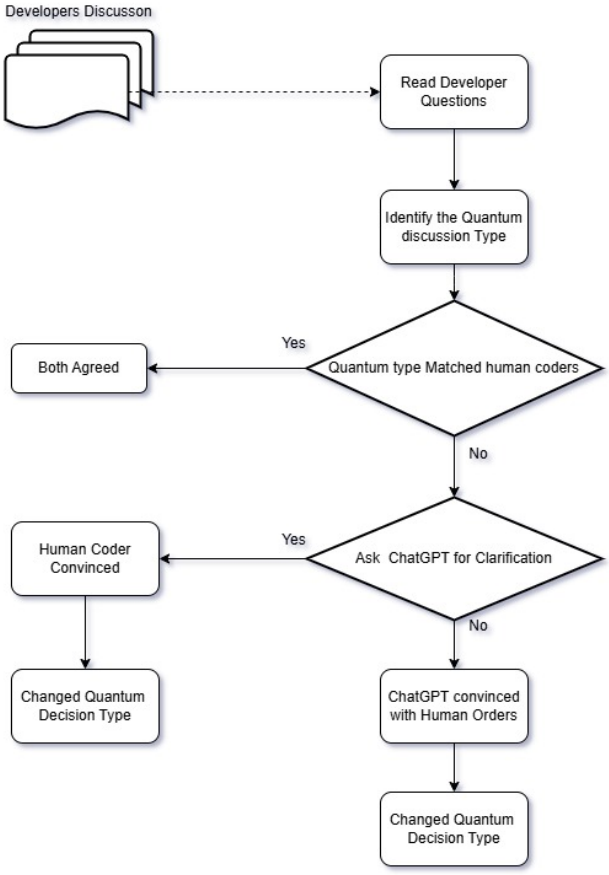

Figure 2: Annotation and negotiation process with ChatGPT.

A negotiation process resolved conflicts between human coders and ChatGPT. For conflicting developer discussions on the Q&A forum, a second round with ChatGPT was initiated. A generic query asked for elaboration on selected quantum categories. This discussion continued until a consensus was reached. Among 282 conflicting discussions, ChatGPT agreed with human annotators on 173 questions after reasoning. Conversely, human coders agreed with ChatGPT on 109 developer feedback after receiving justification. This process curated a conflict-free dataset for quantum software engineering, as shown in Table 6. This step enhances the labelled dataset's quality by validating human annotation with ChatGPT, improving the effectiveness of subsequent D&ML classifiers in identifying quantum developer challenge types.

5 Automated classifications of developer discussions

In the literature, mining developers' feedback on Q&A platforms significantly contributes to software evolution[1]. Researchers have mined Q&A platforms, particularly SO, for various purposes, such as developers' opinions about APIs [45], mining quantum programming [23], identifying challenges for software methodologies and popular topics [22], popular programming language developers [8], and bug severity prediction [42]. Manually identifying useful information for software developers and vendors is cumbersome and resource-intensive [1]. Complementary to these approaches, we mined developer discussions from Q&A platforms to identify frequently mentioned quantum challenges, aiming to provide opportunities for quantum vendors to focus on key areas developers struggle with or discuss most frequently. Below, we elaborate on D&ML experiments and their subsequent steps.

5.1 Experimental setup

For the D&ML experiments, we used text preprocessing and feature engineering to evaluate classifiers and their effectiveness in identifying developer discussions in Stack Exchange forums. We selected D&ML algorithms proven effective with textual data [25] [27], specifically RNN, GRU, LSTM, CNN, FNN, and MLP. These algorithms were used to classify developer questions from Q&A forums and compare their results on classifying discussions into various quantum types. Experiments were conducted in Python, utilizing libraries and tools suited for D&ML tasks to ensure robust results.

5.2 Preprocessing

For the D&ML experiments, cleaning the input data is considered pivotal for improved classification results [17]. We implemented a series of preprocessing steps, starting with removing HTML tags from the discussions to ensure text was free of unwanted formatting. We then filtered out URLs from the Q&A questions to eliminate unnecessary information. To standardize the format, we transformed the entire question text into lowercase to ensure consistency. Subsequently, we removed brackets, punctuation, alphanumeric characters, and other special symbols that could introduce noise into the data. To further enhance D&ML algorithm performance, we employed lemmatization to reduce words to their root forms. This step is crucial for maintaining semantic consistency and improving the model's accuracy. Additionally, we removed stop words from the developers' discussions as they did not add helpful meaning to the classification task.

Table 4: Summary of Annotated Developer Discussions on Quantum Computing

Q_Title	Q_description	Q_Type
Quantum computing & encryption break	I read that Quantum Computers can break most encryption in minutes. How? I get lost at quantum bits being 1, 0, or both. Can someone explain in plain English without the math?	Conceptual
Qubit density to Bloch vector	Given a 2x2 density matrix, how to compute the Bloch sphere point? State $\|0\rangle$-$\|1\rangle$ has matrix [[0.5,-0.5],[-0.5,0.5]] and should be on X axis. Matrix [[0.5,0],[0,0.5]] should be at origin.	Theoretical
What is Quantum Computing?	In physics, particles in multiple states simultaneously. In computing, bits as 1, 0, both, or NULL? Applications to processors, programming, security? Any practical implementations?	Learning
floor/ceil in QCL	What do QCL's floor() and ceil() operators do exactly? They seem related to math operations but need clarification.	Tooling
QuTiP integration failure	Getting errors solving Lindblad equation. Increasing nsteps doesn't help. Changing time values (np.linspace) has no effect. Need solution.	Errors
Can Forest crack crypto?	Can Rigetti Forest break public-key crypto (e.g., Bitcoin) in reasonable time? If yes, show solution using pyQuil.	API Usage

Table 5: Details of Human and ChatGPT Annotations

Quantum Category	Human Freq	ChatGPT Freq
Tooling	600	611
Theoretical	414	446
Learning	159	158
Conceptual	606	597
Errors	803	809
API Usage	247	208
Total Number of Questions: 2829 (Human), 2829 (ChatGPT)		
Disagreement Analysis: There are 282 disagreements between human coders and ChatGPT. The number of learning experiences aligns with this count. In each category, there are disagreements where the human coders, for example, incorporate the learning category where ChatGPT utilizes tools. Overall, disagreements exist across various categories.		

Table 6: Finalized Conflict-Free Annotated Quantum Dataset

Quantum Category	Number of Developers' Discussions
Tooling	596
Theoretical	415
Learning	166
Conceptual	610
Errors	815
API Usage	227
Total	**2829**

5.3 Feature Engineering

For the proposed approach, we used feature engineering to enhance D&ML models' performance in analyzing developers' discussions from stack-exchange forums. We identified effective textual features for short-text analysis in similar technical domains [31] [5] [14]. Techniques like term frequency-inverse document frequency (TF-IDF) and N-grams were utilized. TF-IDF measures a word's importance in a document relative to a corpus, highlighting significant terms in developer discussions. This method identified frequently appearing keywords in Stack Exchange posts, offering insights into relevant topics. N-grams capture important phrases and linguistic patterns in discussions, enabling the analysis of meaningful expressions. The TfidfVectorizer() and CountVectorizer() from scikit-learn were used for efficient text preprocessing and feature extraction. We also experimented with character N-grams to capture subtle variations, enriching the feature space. The Label Encoder from Scikit-Learn prepared textual data by converting categorical labels into numerical representations, facilitating model learning from diverse questions. We optimized deep learning algorithms by adjusting hyperparameters to evaluate classifiers' effectiveness in categorizing discussions into QSE challenges, as shown in Table 7. We tested multiple hyperparameters, including various 'output dim' values in the embedding layer, achieving the highest precision at 100. Adam and RMSProp optimizers were used to assess their effect on accuracy, with Adam performing better in recognizing issue types. Adam's superior performance may be due to its effectiveness with smaller training examples, achieving faster convergence [29]. The dataset is scarce in categories like Learning, with only 166 discussions, making Adam suitable for smaller instances. To address overfitting, we used regularization with a dropout layer. Adam also handles noisy data better than RMSProp [29], making it ideal for dealing with noisy data from social networking platforms.

Table 7: Hyperparameters Values Used to Fine-Tune Classifiers

Hyperparameter	Value
Input_dim	Max_features
Output_dim	512
Units	128
Return_sequences	True
Dropout rate	0.5
Activation (Dense)	'relu' (intermediate layer), 'softmax' (output layer)
Loss	'categorical_crossentropy'
Optimizer	Adam
Learning rate	0.0005
Metrics	['accuracy']

5.4 Data augmentation

QSE is an emerging research domain that is gaining interest from quantum vendors and researchers. Few resources exist for quantum developers to discuss SQE-related issues in quantum applications. On Q&A platforms, unlike Android development or software methodologies, we face challenges with limited experimental data. We used various D&ML classifiers to classify developers' discussions into quantum-related challenges/concepts. Still, we faced performance issues, with maximum accuracies of 51% and 54% using CNN classifiers on the original dataset (details in the results section). Increasing data instances per quantum type can enhance D&ML classifier performance, as deep learning classifiers perform

better with larger datasets. We adopted data augmentation, artificially increasing the dataset size through transformations while preserving labels or semantics. We used noising-based augmentation [32] to increase developer instances per quantum category. A Python function was developed to randomly delete, swap, and insert words in each developer discussion. It deletes 20% of words, swaps two words twice, and inserts two synonyms randomly. Figure 3 illustrates augmentation applied to developer discussions to increase the dataset size. Table 3, row 2, is used in Figure 3 to demonstrate the augmentation process. Textual data augmentation can overcome data limitations in training and validating D&ML classifiers, aiding emerging domains like QSE and IoT with limited developer and end-user data.

5.5 Data Imbalance

In supervised D&ML, imbalanced datasets are considered a critical technical challenge [46] [28]. Data imbalance reflects the unequal distribution of annotation classes within a dataset. The proposed dataset is somewhat imbalanced, as shown in Table 6. When annotating developer discussions, most (815, 28.80%) were categorized as Tooling-related, while only 5.86% (166) were identified as learning-related. Training D&ML classifiers on an imbalanced textual dataset would lead to bias towards the majority class, potentially disregarding minority classes. To address this, we employed two widely used approaches in software [25]: oversampling and undersampling. These techniques aim to balance the dataset, improving D&ML model performance and enabling more accurate predictions of minority data. Oversampling achieves balanced class distribution by randomly repeating minority class examples [12], while undersampling eliminates majority class samples [30]. We found that ML classifiers utilizing oversampling consistently outperformed those utilizing undersampling, as shown in Figure 4. This may be due to the potential loss of critical textual information when using the under-sampling approach [43].

5.6 Assessment and Training

To train and validate the supervised D&ML algorithms, we used a 10-fold cross-validation approach on the textual dataset. Nine folds were used for training and one for validation. The process was repeated 10 times by rotating the folds. Cross-validation checks model performance with limited data and can be used as a resampling method. Each fold had a similar proportion of labels per class. We assessed classifier effectiveness by computing and summarizing average results from the ten-fold cross-validation. We used precision (P), recall (R), and F1-score measures to evaluate and compare the performance of supervised D&ML algorithms. The precision (P) and recall (R) were computed using the following formulas:

$$P_k = \frac{TP_k}{TP_k + FP_k} \tag{1}$$

$$R_k = \frac{TP_k}{TP_k + FN_k} \tag{2}$$

P is determined by dividing the number of accurately classified developers' discussions into quantum types (true positives) by the entire number of comments classified. This metric measures the precision of the model in correctly identifying relevant instances. R measures the model's capacity to identify relevant instances from the total number present. It entails computing the actual positive rate of false negatives. The F1-score combines Precision and Recall, providing a unified measure representing the equilibrium between the classifier's precision and recall.

6 Experimental Results

In this study, we employed three methods–Train-Test Split, feature Engineering, and Data Augmentation–to optimize D&ML classifier performance in classifying Stack Exchange developer discussions into quantum categories. We aim to identify the best configuration for D&ML classifiers to improve the classification of quantum types. We experimented with different training and validation approaches. Using the train-split method, Feedforward Neural Networks (FNN), Convolutional Neural Networks (CNN), and Long Short-Term Memory networks (LSTM) exhibited the highest accuracy rates among the classifiers tested, achieving 45%, 51%, and 46%, respectively. However, the precision, recall, and F1 values obtained for classifying developers discussion into various quantum categories were comparatively low, i.e., the highest F1 values obtained for conceptual, theoretical, learning, tolling, errors, API usage is 0.42, 0.42, 0.15, 0.47,0.77, and 0.22 with MLP, LSTM, CNN, LSTM, CNN, and CNN classifiers, respectively. Similarly, with kfold cross-validation, which is a stable and popular method in software engineering research [17] [26], FNN, CNN, and LSTM classifiers showed accuracy increases of 50%, 54%, and 49%, respectively, with a slightly improving the accuracy compared to the split-train approach. However, the highest F1 values for classifying QSE-related discussion into various quantum types are still low, i.e., 0.46, 0.36, 0.39, 0.47, 0.77, and 0.23 with CNN, FNN, FNN, FNN, CNN, and LSTM classifiers, respectively, for conceptual, theoretical, learning, tolling, errors, and API usage quantum discussion types. One possible reason for getting comparatively low accuracy and F1 values with train-split and kfold is less annotation data. One way is to increase data instances in the annotated developer discussion dataset. However, as SQE is emerging with limited developer discussions on Q&A platforms, data augmentation approaches are necessary to expand the dataset and assess D&ML classifiers' performance artificially. Moreover, we didn't report these values in Table 8, as the values were comparatively low.

By employing a data augmentation approach, the FNN, with its remarkable ability to detect and categorize incorrect discussions with high reliability, continues to excel, particularly in the Errors category. It achieves a precision of 96%, recall of 97%, and an F1-score of 97%, underscoring its exceptional performance and the impact of enriched training datasets on accuracy and resilience. CNN performs better in the Tooling category with an F1 Score of 89%, highlighting gains in model accuracy through augmented learning scenarios, where CNNs leverage spatial scales and additional data dimensions. The LSTM and GRU networks have shown significant improvements in the Tooling category, benefiting from data augmentation. Both models demonstrated enhanced accuracy, with F1 scores reaching 87% and 89%, reflecting their capability to adapt and learn from augmented data. It underscores robust adaptability in handling complex datasets. These outcomes confirm the efficacy of Data Augmentation in improving performance

Original Developer discussion sample	Are there public key cryptographic algorithms that are provably NP-hard to defeat
Tokenization of the developer discussion	["Are", "there", "public", "key", "cryptographic", "algorithms", "that", "are", "provably", "NP-hard", "to", "defeat"]
Random Deletion	["Are", "there", "public", "key", "algorithms", "provably", "NP-hard", "to", "defeat"]
Random Swap, i.e., Public and Key, provably and NP-hard	["Are", "there", "key", "public", "algorithms", "NP-hard", "provably", "to", "defeat"]
Random insertion, i.e., Algorithms become procedures, and defeat becomes overcome.	["Are", "there", "key", "public", "procedures", "algorithms", "NP-hard", "provably", "to", "overcome", "defeat"]
Augmented Developer Discussion	Are there key public procedures algorithms NP-hard provably to overcome defeat

Figure 3: Example of data augmentation with the developer's discussion to artificially increase the dataset size.

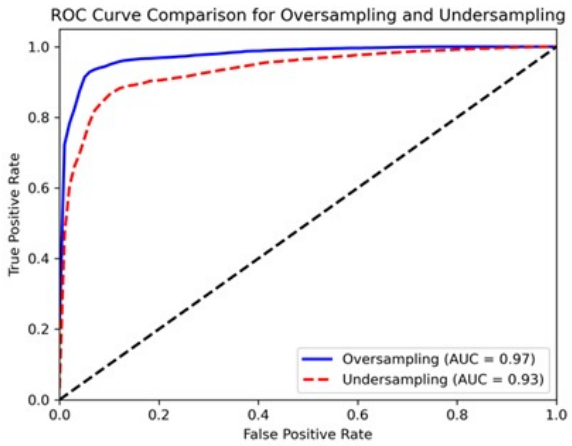

Figure 4: ROC curves showing performances of oversampling and undersampling for FNN model

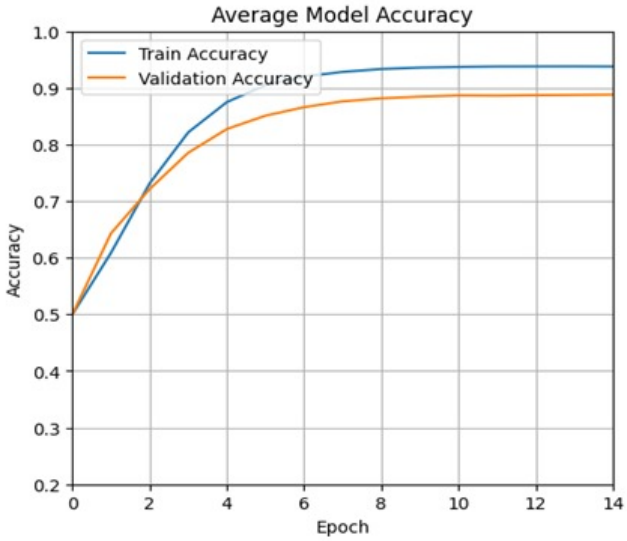

Figure 5: Training and validation Accuracy for FNN

metrics across classifiers. By expanding the training data, these strategies enhance accuracy, adaptability, and robustness against diverse datasets. Such improvements are crucial for developing more reliable and efficient machine learning models. The experimental setup evaluated classifier effectiveness, focusing on improving accuracy, precision, recall, and F1 scores, as summarized in Table 8 for the data augmentation approach.

Furthermore, we examined the baseline configurations of the best classifiers, FNN and CNN. However, we depicted the FNN configurations. As a central part of this analysis, we examined the learning curves of the training datasets to determine how size affects the classification accuracy. Figure 5 shows the average accuracy of training the FNN classifier. The FNN classifier was considered the best-performing classifier for classifying developer discussions into various categories based on its training efficiency and accuracy.

Figure 6 shows the receiver operating characteristic (ROC) curves of the FNN classifier. ROC curves compare the actual positive rate with the false positive rate to determine whether the classifier is sensitive or specific. By examining the accuracy metrics and the ROC curve analysis of the FNN classifier, we can understand how well it classifies text within developers' discussions, which is crucial for analysing inputs from digital platforms. This evaluation is vital for analyzing developers generated content on digital platforms, ensuring accurate and reliable classification of input data.

Additionally, the Area Under the Curve (AUC) for the ROC curves of the FNN classifier highlights its superior classification performance in organizing developers' discussions from online forums into distinct thematic categories. With AUC values near the ideal mark, as depicted in Figure 6, each category demonstrated nearly perfect separation. This is evidenced by a high True Positive Rate (TPR) and a low False Positive Rate (FPR), with the curves approaching the upper-left corner of the plot. The model achieved a macro-average AUC of approximately 1.00, indicating their consistent and precise capability to classify all categories accurately.

These metrics underscore the proficiency of the FNN models in effectively recognizing each discussion category while minimizing misclassifications, thereby confirming their robust performance in

Table 8: Performance of ML and DL Classifiers with Data Augmentation Approaches

QSE Concept	Classifier	Precision	Recall	F1-score
Conceptual	FNN	0.83	0.75	0.78
	CNN	0.91	0.45	0.61
	RNN	0.74	0.70	0.70
	GRU	0.78	0.75	0.74
	LSTM	0.83	0.67	0.74
	MLP	0.62	0.56	0.57
Theoretical	FNN	0.77	0.82	0.77
	CNN	0.76	0.78	0.74
	RNN	0.70	0.66	0.63
	GRU	0.75	0.73	0.70
	LSTM	0.66	0.87	0.75
	MLP	0.56	0.57	0.54
Learning	FNN	0.93	0.90	0.92
	CNN	0.90	0.87	0.88
	RNN	0.92	0.79	0.84
	GRU	0.85	0.83	0.83
	LSTM	0.90	0.83	0.86
	MLP	0.75	0.54	0.62
Tooling	FNN	0.94	0.94	0.94
	CNN	0.88	0.91	0.89
	RNN	0.83	0.86	0.84
	GRU	0.90	0.88	0.89
	LSTM	0.89	0.87	0.87
	MLP	0.64	0.68	0.65
Errors	FNN	0.96	0.97	0.97
	CNN	0.93	0.95	0.94
	RNN	0.92	0.94	0.93
	GRU	0.93	0.94	0.94
	LSTM	0.93	0.95	0.94
	MLP	0.81	0.87	0.84
API Usage	FNN	0.93	0.89	0.91
	CNN	0.93	0.80	0.86
	RNN	0.82	0.73	0.77
	GRU	0.83	0.80	0.81
	LSTM	0.86	0.78	0.81
	MLP	0.59	0.44	0.49
Best Average Accuracy				
FNN				89
CNN				86
RNN				80
GRU				84
LSTM				84
MLP				67

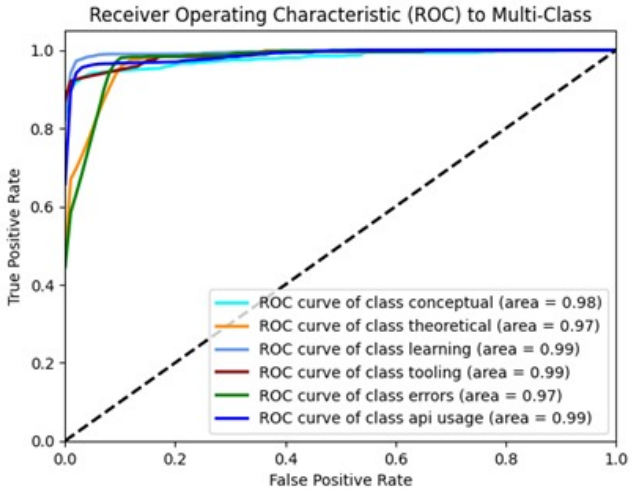

Figure 6: ROC curves for FNN

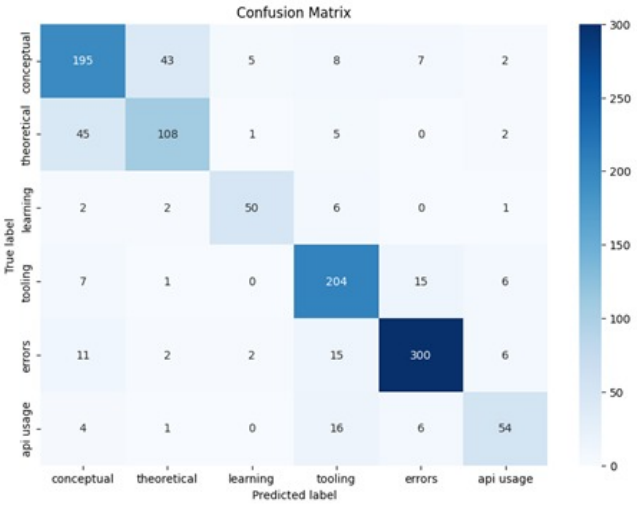

Figure 7: Confusion Matrix for FNN(a) and CNN(b).

multiclass classification within the context of the developer forums. Figure 7 elaborates on the confusion matrix for the FNN classifier. Where the column class label shows the classifiers' predicted class values, and the row labels show the actual classification class values. A confusion matrix evaluates the performance of various D&ML classifiers' abilities to classify developers' discussions on Q&A forums to various QSE challenges.

6.1 Comparative Study with the State of the Art

This section compares the proposed approach with the existing methods regarding scope, performance and generalization. An overview of the comparison is shown in Table 9. Beyer et al.[9] use ML algorithms, including RF and SVM, to categorize Android-related discussions on SO. Their approach achieved an average 90% accuracy, and their approach is limited to classifying developers' posts into Android-related challenges. El-Aoun et al.[33] used topic modeling to analyze Stack Exchange and GitHub conversations to identify QSE-related challenges. The proposed approach is complementary to the El-Aoun et al.[33] approach by focusing on QSE. However, we are classifying developers' discussions into frequently occurring challenges, unlike their approach, which identifies frequently occurring challenges using topic modelling. Wang Edmund [13] uses token-based clone identification to mine Q&A sites for automated comment production. In addition, Beyer et al. [10] conducted a study improving SVM and RF application to analyze Stack Overflow Q&A topics. This study demonstrates comparatively better precision and recall by examining model tuning intricacies and

Table 9: Comparative Study with the State of the Art

Reference Papers	Algorithms Used	Highest Precision, Accuracy, Recall	Dataset Details	Data Size
Beyer et al.[9]	Random Forest, SVM	Precision 90%, Accuracy 90%, Recall 90%	Android-related SO posts	1,000 posts
El-Aoun et al.[33]	Automated Topic Modeling	N/A	Stack Exchange forums, GitHub issues	1,755 issues/posts
Wong Edmund [13]	Token-based clone detection	High accuracy	Android and Java Q&A posts	132,767 mappings
Beyer et al. [10]	SVM, Random Forest	Precision 88%, Recall 87%	Stack Overflow Q&A site	500 posts
A. Ali Khan et al.[22]	LDA Topic Modeling	N/A	Stack Overflow, SESE, PMSE forums	13,903 posts
Our Study	**FNN, CNN, LSTM, GRU, RNN**	**Precision 90%+, Accuracy 84-89%, Recall 90%+**	**Multiple Q&A forums (SO, QCSE, CSSE, AISE)**	**2,429 discussions**

addressing difficulties in managing intricate data structures in a selected dataset of 500 posts. Khan et al. [5] investigate applying NLP and LDA techniques to extract software development models' knowledge and challenges from the stake exchange platform. Compared to the state-of-the-art approaches, the proposed approach analyzed developers' comments from SO, QCSE, CSSE, and AISE for quantum development, aiming to classify practitioners' comments to frequently occurring quantum development challenges. Using preprocessing, feature engineering, data augmentation and advanced D&ML algorithms, including FNN, CNN, LSTM, GRU, and RNN, we achieved precision and accuracy levels from 84% to 89%. The proposed approach can help quantum practitioners categorise Q&A posts based on the frequently identified challenges, providing easy access to related posts and solutions from other developers.

7 Discussions

The insights obtained from developer interactions on diverse social media platforms are powerful tools for software and requirements engineers. This section will delve into our research findings, as elaborated below.

7.1 Rationale for Quantum Software Engineering

Quantum computing presents a transformative approach to computational problem-solving beyond the capabilities of classical computers, justifying the integration of quantum mechanics into software engineering. Quantum software engineering uses characteristics such as superposition and entanglement to change software development. This rationale stems from its superior capacity to address complex problems. Quantum software engineering efficiently addresses optimization, cryptography, machine learning, and simulations using quantum bits (qubits) and algorithms. This section highlights the potential of quantum computing to revolutionize software development and establish specialized quantum algorithms. The approach improved knowledge using D&ML classifiers to automatically classify quantum developers' discussions into common challenges. It helps developers find solutions in the Q&A community without memorizing quantum-specific tags while helping vendors oversee frequent challenges to develop tools and libraries.

7.2 Comparative Analysis of Developer Discussions

Our exploration examines developers' discussions on stake exchange platforms to uncover themes and trends. By analyzing these interactions, we gain insights into the challenges in QSE, enhancing developers' and vendors' understanding of technical issues. This reveals developers' approaches to technology, programming paradigms, and problem solving. We used D&ML algorithms and NLP techniques to classify discussions into Tooling, Theoretical, Learning, Conceptual, Errors, and API Usage. This classification reveals the underlying sentiments and themes and discerns emotional tones. By identifying these patterns, we describe the current developer discourse, enhance our understanding of key issues, and aid in the development of effective programming tools and methodologies.

7.3 Implications of Findings for Software Development Practices

A study of online developer forums revealed software developers' key challenges, with implications for QSE practices. By analyzing discussions on Tooling, Theoretical Issues, Learning Resources, Conceptual Understanding, Errors, and API Usage, we identified areas affecting QSE development. Discussions on tooling inefficiencies highlight the gaps between resources and developer needs, suggesting opportunities for more intuitive tools, especially for quantum computing projects. The theoretical challenges emphasize the need for continuous education and accessible resources to keep pace with advancements. Issues with conceptual understanding and API usage indicate that better documentation and standards can enhance productivity in the field. Improved documentation clarity would aid quicker API adoption. The nature of the errors and their resolution highlights the potential of knowledge-sharing platforms. Addressing these areas can enhance QSE development, leading to faster cycles and more robust applications.

7.4 Insights from Quantitative Analysis of Developer Discussion

The proposed study using D&ML algorithms to classify developer discussions yielded significant findings. Analysis shows effectiveness variability across models, with LSTM exhibiting better accuracy in 'Errors' category, suggesting patterns well-captured by recurrent neural architectures. The 'API Usage' category showed lower performance across models, indicating complex discussions with subtle language that complicated classification. These findings reveal D&ML models' capabilities and limitations for processing natural language data. Model performance improved during feature engineering and data augmentation stages, particularly in 'Conceptual' and 'Learning' categories, suggesting promising text classification potential. Enhancing features and expanding datasets through augmentation can improve context understanding in discussions. This analysis highlights the importance of model

selection in automated text-analysis systems. Choosing appropriate techniques based on data features and classification needs allows data scientists and developers to optimize system performance and decision-making.

7.5 Threats to Validity

While processing and analyzing developer discussions, we encountered several potential threats to the validity of the proposed methodology and findings. Acknowledging these challenges is crucial to ensuring the credibility and reliability of the research outcomes.

7.5.1 Researcher Bias. In the proposed study, the potential for researcher bias is significant, as the first, third and fourth authors and an AI model, such as ChatGPT, labelled and analyzed developer discussions to maintain neutrality and objectivity; our inherent attitudes and previous experiences subtly affected the analysis. This bias could skew the data's performance or emphasize specific findings. Acknowledging the presence of such biases is crucial to ensuring the integrity and reliability of our research findings.

7.5.2 Data Preprocessing Challenges . Data preprocessing is a critical step in the proposed approach, fraught with challenges, such as addressing noisy data, removing biases, and fixing semantic opacity in developer discussions. These issues can significantly impact the quality and interpretability of data, potentially affecting subsequent analyses. Therefore, we must implement rich data preprocessing techniques to mitigate these issues, thereby enhancing the validity of our findings by ensuring that the data fed into our analytical models is as accurate and clean as possible.

7.5.3 Algorithm Selection and Fine Tuning. The selection and fine-tuning of D&ML algorithms to examine developer discussions are fraught with several challenges. The proposed research's effectiveness and applicability depend heavily on choosing appropriate algorithms and optimally tuning them to handle the specific nuances of our data. This process requires deep expertise and careful experimentation, as imperfect algorithm selection or insufficient tuning can lead to misleading outcomes or decreased model effectiveness.

7.5.4 Data Quality Concerns. Data quality is paramount for ensuring the reliability of the machine-learning models used in the study. High-quality data training and evaluation are imperative to produce valid and generalizable results. Data noise, inherent biases, and semantic ambiguities within developer discussions must be addressed rigorously. These factors must be considered to maintain the trustworthiness of our research outputs, as the models may learn from flawed data, leading to unreliable predictions and insights.

7.5.5 Ever-Evolving Landscape of Online Forums. The vibrant and ever-changing landscape of online forums and developer communities presents additional challenges for maintaining the scalability and generalizability of the proposed approach. These platforms have evolved rapidly with new models and continually emerging topics. Thus, our methodology requires ongoing adjustments and updates to remain relevant and effective in capturing the current state of developer discussions across various forums.

8 Conclusions and future work

This paper presents a novel automated method for gathering, categorizing, and evaluating developer conversations on stack-exchange platforms. These conversations cover subjects related to software development, including inquiries, choices, and problems, with an emphasis on QSE. The proposed solution utilizes fine-tuned D&ML techniques to extract and categorize developer discussions from QSE discussion threads. This model was created and tested using cross-validation and comparison with human-AI-labeled data, demonstrating accuracy in recognizing and categorizing developers' discussions related to QSE into various frequently occurring challenges. This improves the accessibility of essential knowledge for quantum developers. The approach can be a powerful tool for quantum developers, enabling them to access relevant discussions efficiently without manual data search and preprocessing to extract useful information based on challenges related to QSE.

Future studies will focus on key areas to expand and enhance the automated categorization model's capabilities. We plan to explore other forums and discussion platforms popular in the QSE community to collect more data. We aim to provide a more intuitive interface and toolchain that classifies developers' discussions into frequently occurring challenges on the run. These efforts aim to develop robust collaborative relationships with field professionals, maintaining the model's relevance and effectiveness in addressing practical barriers in QSE. Moreover, we plan to extend the proposed approach to other emerging technologies' discussions on Q&A forums, such as blockchain, Large language models, deep and transfer learning. Additionally, we aim to develop a toolset for the proposed approach by implementing the best-performing D&ML classifiers and demonstrating their applicability for quantum software developers.

Authorship contributions: M.H. and J.A.K. developed the method, detailed investigation, and manuscript writing and curated the research data set; M.S.K., N.D.K., A.A.K., and M.A.A. revised the methodology and supervised and revised the manuscript writing (revised draft). All authors have read and agreed to the published version of the manuscript.

Funding: The authors have not disclosed any funding.

References

[1] Arshad Ahmad, Chong Feng, Shi Ge, and Abdallah Yousif. 2018. A survey on mining stack overflow: question and answering (Q&A) community. *Data Technologies and Applications* 52, 2 (2018), 190–247.

[2] Aakash Ahmad, Arif Ali Khan, Muhammad Waseem, Mahdi Fahmideh, and Tommi Mikkonen. 2022. Towards process centered architecting for quantum software systems. In *2022 IEEE international conference on quantum software (QSW)*. IEEE, 26–31.

[3] Muhammad Azeem Akbar, Arif Ali Khan, and Saima Rafi. 2023. A systematic decision-making framework for tackling quantum software engineering challenges. *Automated Software Engineering* 30, 2 (2023), 22.

[4] Shaukat Ali and Tao Yue. 2023. Quantum software testing: A brief introduction. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 332–333.

[5] Javed Ali Khan, Lin Liu, and Lijie Wen. 2020. Requirements knowledge acquisition from online user forums. *Iet Software* 14, 3 (2020), 242–253.

[6] Miltiadis Allamanis and Charles Sutton. 2013. Why, when, and what: analyzing stack overflow questions by topic, type, and code. In *2013 10th Working conference on mining software repositories (MSR)*. IEEE, 53–56.

[7] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. 2019. Quantum supremacy using a programmable superconducting processor. *Nature* 574, 7779 (2019), 505–510.

[8] Juan F Baquero, Jorge E Camargo, Felipe Restrepo-Calle, Jairo H Aponte, and Fabio A González. 2017. Predicting the programming language: Extracting knowledge from stack overflow posts. In *Advances in Computing: 12th Colombian Conference, CCC 2017, Cali, Colombia, September 19-22, 2017, Proceedings 12*. Springer, 199–210.

[9] Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger. 2020. What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. *Empirical Software Engineering* 25 (2020), 2258–2301.

[10] Stefanie Beyer, Christian Macho, Martin Pinzger, and Massimiliano Di Penta. 2018. Automatically classifying posts into question categories on stack overflow. In *Proceedings of the 26th Conference on Program Comprehension*. 211–221.

[11] Stefanie Beyer and Martin Pinzger. 2014. A manual categorization of android app development issues on stack overflow. In *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE, 531–535.

[12] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[13] Wong Edmund. 2014. *Mining question and answer sites for automatic comment generation*. Master's thesis. University of Waterloo.

[14] Eman Fatima, Hira Kanwal, Javed Ali Khan, and Nek Dil Khan. 2024. An exploratory and automated study of sarcasm detection and classification in app stores using fine-tuned deep learning classifiers. *Automated Software Engineering* 31, 2 (2024), 69.

[15] Sukhpal Singh Gill, Adarsh Kumar, Harvinder Singh, Manmeet Singh, Kamalpreet Kaur, Muhammad Usman, and Rajkumar Buyya. 2022. Quantum computing: A taxonomy, systematic review and future directions. *Software: Practice and Experience* 52, 1 (2022), 66–114.

[16] Majid Haghparast, Tommi Mikkonen, Jukka K Nurminen, and Vlad Stirbu. 2023. Quantum Software Engineering Challenges from Developers' Perspective: Mapping Research Challenges to the Proposed Workflow Model. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 2. IEEE, 173–176.

[17] Shoaib Hassan, Qianmu Li, Khursheed Aurangzeb, Affan Yasin, Javed Ali Khan, and Muhammad Shahid Anwar. 2024. A systematic mapping to investigate the application of machine learning techniques in requirement engineering activities. *CAAI Transactions on Intelligence Technology* 9, 6 (2024), 1412–1434.

[18] IBM Quantum. 2023. IBM Quantum Computing Roadmap. https://www.ibm.com/quantum/blog/ibm-quantum-roadmap Accessed: 2025-03-28.

[19] IBM Quantum. 2024. Qiskit | IBM Quantum Computing. https://www.ibm.com/quantum/qiskit Accessed: 2025-03-28.

[20] Hafiz Umar Iftikhar, Aqeel Ur Rehman, Olga A Kalugina, Qasim Umer, and Haris Ali Khan. 2021. Deep Learning-Based Correct Answer Prediction for Developer Forums. *IEEE Access* 9 (2021), 128166–128177.

[21] Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T Chong, and Margaret Martonosi. 2015. ScaffCC: Scalable compilation and analysis of quantum programs. *Parallel Comput.* 45 (2015), 2–17.

[22] Arif Ali Khan, Javed Ali Khan, Muhammad Azeem Akbar, Peng Zhou, and Mahdi Fahmideh. 2024. Insights into software development approaches: mining Q &A repositories. *Empirical Software Engineering* 29, 1 (2024), 8.

[23] Arif Ali Khan, Boshuai Ye, Muhammad Azeem Akbar, Javed Ali Khan, Davoud Mougouei, and Xinyuan Ma. 2025. Mining Q&A Platforms for Empirical Evidence on Quantum Software Programming. *arXiv preprint arXiv:2503.05240* (2025).

[24] Javed Ali Khan, Yuchen Xie, Lin Liu, and Lijie Wen. 2019. Analysis of requirements-related arguments in user forums. In *2019 IEEE 27th international requirements engineering conference (RE)*. IEEE, 63–74.

[25] Javed Ali Khan, Affan Yasin, Rubia Fatima, Danish Vasan, Arif Ali Khan, and Abdul Wahid Khan. 2022. Valuating requirements arguments in the online user's forum for requirements decision-making: the CrowdRE-VArg framework. *Software: Practice and Experience* 52, 12 (2022), 2537–2573.

[26] Nek Dil Khan, Javed Ali Khan, Jianqiang Li, Tahir Ullah, Ayed Alwadain, Affan Yasin, and Qing Zhao. 2024. How do crowd-users express their opinions against software applications in social media? A fine-grained classification approach. *IEEE Access* (2024).

[27] Nek Dil Khan, Javed Ali Khan, Jianqiang Li, Tahir Ullah, and Qing Zhao. 2024. Mining software insights: uncovering the frequently occurring issues in low-rating software applications. *PeerJ Computer Science* 10 (2024), e2115.

[28] Nek Dil Khan, Javed Ali Khan, Jianqiang Li, Tahir Ullah, and Qing Zhao. 2025. Leveraging Large Language Model ChatGPT for enhanced understanding of end-user emotions in social media feedbacks. *Expert Systems with Applications* 261 (2025), 125524.

[29] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[30] Sotiris Kotsiantis, Dimitris Kanellopoulos, Panayiotis Pintelas, et al. 2006. Handling imbalanced datasets: A review. *GESTS international transactions on computer science and engineering* 30, 1 (2006), 25–36.

[31] Zijad Kurtanović and Walid Maalej. 2018. On user rationale in software engineering. *Requirements Engineering* 23 (2018), 357–379.

[32] Bohan Li, Yutai Hou, and Wanxiang Che. 2022. Data augmentation approaches in natural language processing: A survey. *Ai Open* 3 (2022), 71–90.

[33] Heng Li, Foutse Khomh, Moses Openja, et al. 2021. Understanding quantum software engineering challenges an empirical study on stack exchange forums and github issues. In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 343–354.

[34] Kimberly A Neuendorf. 2017. *The content analysis guidebook*. sage.

[35] Michael A Nielsen and Isaac L Chuang. 2010. *Quantum computation and quantum information*. Cambridge university press.

[36] Mario Piattini, Guido Peterssen, Ricardo Pérez-Castillo, Jose Luis Hevia, Manuel A Serrano, Guillermo Hernández, Ignacio García Rodríguez De Guzmán, Claudio Andrés Paradela, Macario Polo, Ezequiel Murina, et al. 2020. The Talavera Manifesto for quantum software engineering and programming.. In *QANSWER*. 1–5.

[37] John Preskill. 1998. Reliable quantum computers. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454, 1969 (1998), 385–410. doi:10.1098/rspa.1998.0167

[38] Rigetti Computing. 2020. Why Quantum. https://www.rigetti.com/why-quantum Accessed: 2025-03-28.

[39] Christoffer Rosen and Emad Shihab. 2016. What are mobile developers asking about? a large scale study using stack overflow. *Empirical Software Engineering* 21 (2016), 1192–1223.

[40] Anselm Strauss and Juliet Corbin. 1998. Basics of qualitative research techniques. (1998).

[41] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q# enabling scalable quantum computing and development with a high-level dsl. In *Proceedings of the real world domain specific languages workshop 2018*. 1–10.

[42] Youshuai Tan, Sijie Xu, Zhaowei Wang, Tao Zhang, Zhou Xu, and Xiapu Luo. 2020. Bug severity prediction using question-and-answer pairs from stack overflow. *Journal of Systems and Software* 165 (2020), 110567.

[43] James Tizard, Hechen Wang, Lydia Yohannes, and Kelly Blincoe. 2019. Can a conversation paint a picture? mining requirements in software forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 17–27.

[44] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. How do programmers ask and answer questions on the web?(nier track). In *Proceedings of the 33rd international conference on software engineering*. 804–807.

[45] Gias Uddin and Foutse Khomh. 2019. Automatic mining of opinions expressed about apis in stack overflow. *IEEE Transactions on Software Engineering* 47, 3 (2019), 522–559.

[46] Tahir Ullah, Javed Ali Khan, Nek Dil Khan, Affan Yasin, and Hasna Arshad. 2023. Exploring and mining rationale information for low-rating software applications. *Soft Computing* n.a. (2023), 1–26.

[47] Daniel Vietz, Johanna Barzen, Frank Leymann, and Karoline Wild. 2021. On decision support for quantum application developers: categorization, comparison, and analysis of existing technologies. In *International Conference on Computational Science*. Springer, 127–141.

[48] Tao Yue, Shaukat Ali, and Paolo Arcaini. 2023. Towards quantum software requirements engineering. In *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, Vol. 2. IEEE, 161–164.

[49] Jianjun Zhao. 2020. Quantum software engineering: Landscapes and horizons. *arXiv preprint arXiv:2007.07047* (2020).

[50] Wenhan Zhu, Haoxiang Zhang, Ahmed E Hassan, and Michael W Godfrey. 2022. An empirical study of question discussions on Stack Overflow. *Empirical Software Engineering* 27, 6 (2022), 148.