Measuring the Effectiveness of Large Language Model for Answer Generation: A Case of Quantum Software Engineering

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

Safia Kanwal
safia.kanwal@aru.ac.uk
Faculty of Creative and Digital Arts and Sciences, Anglia Ruskin University
Peterborough, United Kingdom

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

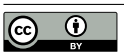
Arif Ali Khan
arif.khan@oulu.fi
M3S Empirical Software Engineering Research Unit, University of Oulu
Oulu, Finland

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

Abstract

Quantum Software Engineering (QSE) is an emerging field garnering interest among quantum researchers, developers, and tech giants for developing software to realize quantum computing's potential. Quantum developers frequently refer to Question and Answer (Q&A) platforms to address QSE challenges. However, the average response time to developers' questions on Q&A platforms is not immediate and can take days, leading to frustration. To address this delay, this study proposes an alternative approach for generating accurate answers for QSE-related topics using Large Language Models (LLMs). The automated LLM-based pipeline uses a few-shot prompting technique to examine whether a medium-sized LLM can be guided to generate domain-specific answers aligned with expert expectations. To evaluate the quality of these responses, we measure their semantic similarity against validated, human-authored answers. The experiments were conducted on a dataset of 263 quantum computing-related Q&A pairs, showing that the LLM achieved an average semantic similarity score of 64% compared to developers' answers. Moreover, a detailed analysis of low similarity scores was conducted to find potential reasons using the LLMs-as-Jury approach. The results demonstrate that only 4% cases are possibly hallucinated or provided incorrect information. These results highlight the promise of LLMs in assisting developers with contextually relevant information while underscoring their current limitations in addressing highly technical and domain-specific challenges. The proposed approach can serve as a stepping stone for enhanced developer experiences by integrating LLM with the Q&A platforms, providing immediate responses. Additionally, the approach can provide software developers with opportunities to refine their responses by analysing the LLM-generated output.

*Corresponding author



SAC '26, Thessaloniki, Greece

ACM ISBN 979-8-4007-2294-3/2026/03
https://doi.org/10.1145/3748522.3779916

CCS Concepts

• **Hardware → Quantum computation**.

Keywords

Quantum Software Engineering, Question Answering, Large Language Model, DeepSeek, LLMs-as-Jury

ACM Reference Format:
Nek Dil Khan, Safia Kanwal, Javed Ali Khan, Arif Ali Khan, and Muhammad Azeem Akbar. 2026. Measuring the Effectiveness of Large Language Model for Answer Generation: A Case of Quantum Software Engineering . In *The 41st ACM/SIGAPP Symposium on Applied Computing (SAC '26), March 23–27, 2026, Thessaloniki, Greece.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3748522.3779916

1 Introduction

Quantum technology has been receiving global attention due to its advantages across industries, including cybersecurity [26], finance [15], communication [33], and aerospace [31]. This shift became possible due to quantum computing(QC). QC operates on 0 and 1, qubits can exist in superposition, representing both states simultaneously, allowing faster complex calculations than classical computers [10]. Despite their advantages over classical computing, quantum software development faces challenges in utilizing its potential [24, 35]. Oyeniran et al. [25] suggest QC is transforming various domains through high-speed complex tasks but faces challenges like high cost and lack of expertise. Traditional software engineering cannot address quantum systems' unique challenges, as their complexity differs from classical computing [28]. As a result, QSE has emerged as a distinct field of study, focusing on the methodologies, tools, and techniques necessary to build software systems capable of leveraging quantum computers [24]. QSE remains in early stages, requiring effective solutions for developers' technical challenges.

Considering this growing research interest, quantum and software engineering researchers have proposed various approaches for QSE activities. Yue et al. [34] provide thoughts on gathering requirements for software projects. Ahmad et al. [1] proposed a reference architecture approach for quantum computing as a service. Khan et

al. conducted a study on quantum programming to identify trends, frameworks, and challenges [20]. De la Barrera et al. [11] proposed a mapping study for quantum software testing to identify defects and opportunities. Similarly, Tech giants, including Google[1], Microsoft[2], and IBM[3], are investing in quantum software and hardware for their industrial applications. However, to fully utilize the capabilities of QC, novel quantum software methodologies are needed, which remain a challenge. To address this, quantum developers need to be equipped with the necessary skills, processes, tools, and knowledge of challenges with effective quantum development. For this purpose, Q&A platforms provide useful information to the quantum developers on common challenges [17]. Software practitioners use Q&A forums, like Stack Overflow (SO), to discuss and address challenges to software development [2] [8]. For example, Khan et al. [19] used Q&A to extract development methodologies challenges; De Dieu et al. [8] used it for extracting software architecture information; and El-Aoun et al. [21] used it for detecting quantum developers' challenges.

Furthermore, based on our knowledge, a key challenge with the existing Q&A platforms is the lack of immediate, domain-specific support when developers face technical issues, particularly with emerging technologies, like QSE and blockchain. Bhat et al. [4] found that 30% of questions on Q&A platforms thave a response time of more than a day. This motivates us to examine LLM performance in automatically generating instant answers to practitioners ' questions, as an alternative to expert-driven answers on Q&A platforms, especially for emerging technologies lacking human expertise. Recently, LLMs have shown success in automating software engineering tasks [16]. Kabir et al. [18] investigated ChatGPT-generated answers to SO questions through linguistic and manual analysis. Similarly, De Silva et al. [7] evaluated ChatGPT and LlaMA-2 in addressing SO questions and identified a significant decline in developers' SO activities since LLMs' release.

Complementary to Kabir et al. [18] and De Silva et al. [7] approaches, we explore fine-tuning medium-sized LLMs' potential, specifically the DeepSeek 32B model, to identify its reliability in generating developers' answers in SO by developing an automated pipeline for QSE, as it is open source with reasoning capabilities. Rather than relying on large, expensive models, we develop a more cost-effective approach using a few-shot prompting technique to guide LLMs in generating contextually appropriate answers. The LLM-based pipeline was evaluated on 263 quantum computing-related Q&A pairs, achieving an average semantic similarity score of 64% compared to human expert answers. These results demonstrate LLMs potential as a tool for quantum software developers by providing faster responses to their technical questions. Our study contributes to developing an efficient method for knowledge sharing in quantum software engineering, offering an alternative to traditional Q&A platforms addressing domain-specific challenges.

The paper is structured as follows: Section 2 reviews related work on QSE. Section 3 outlines the proposed research methodology. Section 4 describes the proposed experimental setup. Section 5 explores the key findings. Finally, Section 6 concludes the paper.

[1]https://quantumai.google/quantumcomputer accessed on June 9, 2025
[2]https://www.microsoft.com/en-us/research/research-area/quantum-computing/ accessed on June 9, 2025
[3]https://www.ibm.com/quantum accessed on June 9, 2025

2 Related Work

In this section, we elaborate on the state-of-the-art approaches.

2.1 Q&A mining for QSE

QSE is an emerging field where researchers mine developer Q&A forums to understand the frequent challenges. El Aoun et al analyzed over 3,100 Q&A posts and 43,000 GitHub issues related to QSE [21], highlighting several QSE-related challenges through qualitative coding and topic modeling. Khan et al. [20] analyzed 6,935 Q&A platform posts for quantum software insights, identifying 20 prevalent topics. Popular topics included physical theories and computing foundations, while challenging topics covered practical issues, and algorithm tuning. They found Qiskit as the most adopted framework [20]. Husain et al. [17] analyzed 2,829 QSE-related questions, developing a taxonomy of quantum challenges including Tooling issues, Theoretical questions, Learning barriers, Conceptual misunderstandings, Runtime errors, and API usage problems. Their GPT-validated DL classifiers achieved 89% accuracy in categorizing posts [17]. Aktar et al. [3] use Q&A forums to identify architecture decision-making for quantum software development. Similarly, Upadhyay et al. [29] analyzed quantum software evolution practices by examining 21000 GitHub repositories.

2.2 LLM Utilization in Software Repositories

LLMs like ChatGPT have influenced how developers seek and share knowledge [5]. Since ChatGPT's release, debate has emerged about AI assistants replacing Q&A forums like Stack Overflow. ChatGPT offers instant help, but accuracy concerns exist [7]. Kabir et al. [18] evaluated Stack Overflow questions answered by ChatGPT versus human responses. They found 52% of ChatGPT answers contained incorrect information, with 77% being more verbose. Users still preferred ChatGPT 35% of the time due to comprehensiveness [18]. In addition, Hasan et al. explored GPT-3's ability to answer software questions compared to Stack Overflow [13]. GPT-3 answers were 66% shorter, with higher lexical overlap (35%), suggesting better topic focus. The AI showed 25% more positive sentiment indicating politeness [13]. However, practitioners sometimes prefer human answers, and LLM reliance might reduce Stack Overflow community engagement over time.

Furthermore, researchers have examined LLMs' impact on Q&A communities. Da Silva et al. [7] studied ChatGPT and LLaMA on SO questions, finding LLMs rivaled human answers in some areas but didn't consistently outperform them. Their work confirmed decreased posting activity as developers obtained instant AI solutions. Helic and Santos analyzed Q&A post evolution post-ChatGPT [14], finding a decline in contributions, while remaining questions became longer and more complex. ChatGPT filtered simpler programming problems, which users now solve privately with AI, leaving harder problems for the public forum [14]. This means post-2023 Q&A data may differ qualitatively due to AI's filtering effect. Molavi et al. proposed an LLM framework for personalized answer generation using Stack Exchange data [23]. The study found that few-shot prompting with examples of accepted answers improved the response personalization. This approach shows how mining Q&A interactions can inform LLM outputs, potentially benefiting niche domains like quantum computing.

Table 1: Comparison with State-of-the-Art Approaches

Study	Methodology Used	Contributions
[18]	Empirical evaluation of ChatGPT vs. human answers on 517 SO programming questions (manual analysis, linguistic analysis, user study)	Identified high incidence of incorrect (52%) and verbose answers by ChatGPT; highlighted that users appreciated ChatGPT's clarity despite errors
[9]	Comparative analysis of 270 privacy-related questions: Stack Overflow accepted answers vs. ChatGPT answers	Demonstrated ChatGPT's answers were almost as accurate as human answers on privacy questions (71% vs 74% correctness), with ChatGPT matching the accepted solution in 56% of cases
[32]	Experiments comparing ChatGPT vs. SO answers for compiler-error questions.	Showed that an LLM can equal or outperform Stack Overflow solutions for compiler errors when given well-crafted prompts
[22]	User study (between-subjects): one group used ChatGPT, another used SO to solve programming tasks	ChatGPT improved productivity for tasks, providing better code for algorithmic and API questions, while Stack Overflow excelled at debugging help. showed evidence of platform's strengths
[23]	Framework for LLM- Driven personalized answer generation using StackExchange data (0-shot vs. few-shot prompts with past accepted answers)	Showed that including examples of a user's preferred answers greatly improves LLM personalization, LLM responses aligned much more closely with individual needs when few-shot learning was used used
Our Proposed Approach	Automated LLM-based answer pipeline for QSE: uses few-shot prompting with domain examples; evaluates outputs against expert answers (semantic similarity metrics)- Used LLM-as-a-Jury to find the reason of failure/low semantic similaity with the ground truth	provides immediate, context-rich answers to QC questions, addressing slow forum response times. Demonstrates an average of 64% similarity to human answers, offering a blueprint for integrating LLMs into Q&A platforms for niche domains. Only 14% have similarity less than threshold, LLMs-as-a-Jury demonstrate that only 4% of the original answers hallucinate, the main reason for the rest of the answers to be under the threshold is their style difference from the ground truth answers.

2.3 Comparison With The State-Of-The-Art

Recent studies have examined LLMs impact on Software engineering Q&A. Kabir et al. [18] analyzed ChatGPT's Stack Overflow answers using mixed methods (517 Q&A pairs comparison, linguistic analysis, user study), highlighting articulation vs. accuracy trade-off in LLM responses [18]. Studies have compared LLM and human-generated answers on software topics. Delile et al. [9] evaluated ChatGPT's performance on privacy-related programming questions using 270 Stack Overflow privacy questions. ChatGPT achieved 56% correct answers and 71% overall accuracy versus 74% for human answers, approaching human-level performance but showing lower reliability. Both human and LLM answers achieved an accuracy of under 75% [9]. Furthermore, Widjojo and Treude [32] studied LLMs ability to solve programming errors compared to SO, finding modern LLMs often match accepted answers. However, LLM answers quality depends on prompt design, suggesting effective use requires prompt engineering expertise [32]. Moreover, Liu et al. [22] found that programmers using ChatGPT completed algorithmic and library tasks faster with better while Stack Overflow excelled at debugging due to human experts' error detection. This shows LLMs are effective for defined problems, while human expertise remains valuable for complex debugging. The study was limited by its controlled setting with students [22]. Molavi et al. [23] developed personalised LLM answer generation in education using Stack Exchange data to fine-tune prompts. Their experiments demonstrated that few-shot prompting enhanced personalisation of LLM responses through metrics and human evaluation, though testing was limited [23]. The proposed approach complements the existing approach in evaluating the performance of LLMs in generating answers to developers' questions on Q&A platforms. However, in the proposed approach, unlike previous methods, we experimented with comparatively medium-sized LLMs, specifically the DeepSeek-r1: 32B model, which is less costly and was deployed for answer generation in an emerging domain, namely QSE. In the proposed approach, we developed several prompts using a few-shot learning method to create a comprehensive answer-generating prompt based on QSE question patterns, rather than relying on user prompt skills. Additionally, the proposed approach provides a comprehensive automated pipeline for generating developers' answers and semantically comparing them with expert-generated answers. Furthermore, we used the LLMs-as-Jury approach to investigate the root cause of developers' answers with low semantic values. Table 1 compares the proposed approach with existing work.

3 Proposed Research Methodology

This section elaborates on the research questions that guide the proposed methodology. We then describe the approach used to answer these questions, including the experimental setup, dataset, evaluation methods, and the use of LLMs in generating answers for QSE. The proposed methodology focuses on leveraging a fine-tuned LLM to generate accurate responses to QSE queries and evaluate their performance compared to human-generated answers.

3.1 Research Questions

This study aims to explore the potential of LLMS as an alternative source for generating immediate and timely answers in the emerging field of QSE. By assessing the effectiveness of LLMs and comparing their performance to that of human expert responses, we seek to provide insights into the applicability of AI-driven solutions in addressing technical challenges within QSE. We developed the following research questions to validate our proposed methodology.

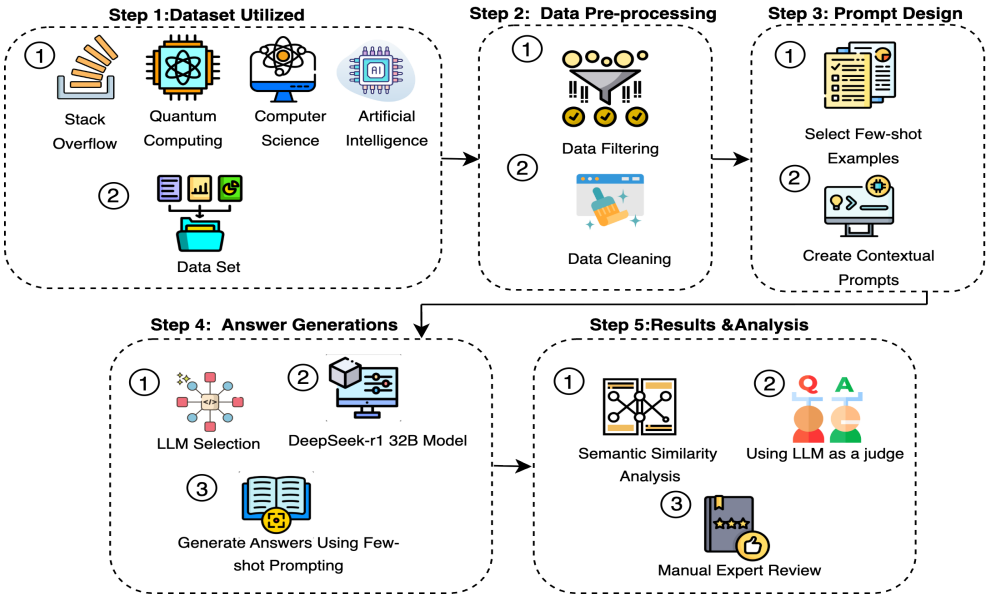


Figure 1: Proposed Research Method Diagram.

RQ1: Can LLMs be used as an alternative source to generate answers to emerging software engineering domains (QSE)?

RQ2: What is the performance of LLM-generated answers compared to human experts' generated answers?

To address RQ1, we aim to investigate whether LLMs, specifically open-source models such as DeepSeek, can provide contextually relevant and accurate answers to QSE-related questions. This involves assessing the model's ability to generate high-quality answers by utilising a specially curated dataset of quantum computing questions and answers. For RQ2, we compare the performance of LLM-generated answers against human-generated answers by evaluating their semantic similarity and overall quality. This comparison is vital to understanding current LLM capabilities in QSE.

3.2 Research Method

The proposed research methodology for answering the research questions comprised several key phases, which are illustrated in Figure 1. For the proposed approach, in step 1, we utilised a previously developed dataset about QSE developers' discussions on various Q&A platforms [17] to evaluate the performance of a medium-sized LLM for answer generation on a relatively emerging domain. In step 2, we applied several preprocessing steps to the developers' discussions to present them more effectively for the LLM. The next step in the proposed approach is the systematic development of a prompt to generate better answers for developers in response to their questions. In the next phase, the proposed approach utilises a DeepSeek 32B model with the optimized few-shot prompt to generate the developers' answers and obtain a response in a more structured JSON format for further analysis. In the final step, semantic similarity is identified for the LLM-generated and expert-generated answers using the all-MiniLM-L6-v2 model embeddings. Additionally, to evaluate the low similarity score between the expert and LLM-generated answers, we employed the LLM-as-Jury approach by designing a prompt and experimenting with ChatGPT, Gemini, and Mistral AI LLMs to validate the possible reasons for the low similarity score. Additionally, the human experts evaluated and validated the answers and outputs generated by the LLMs. Below, we elaborated on each methodological step in detail.

4 Experimental Setup

In this study, we present an automated pipeline for generating QSE-related answers using few-shot prompting with a medium-sized Large Language Model (LLM), specifically the **DeepSeek-r1:32b** model[12], deployed via the **Ollama framework**.

Ollama[4] is an open-source, modular framework designed to run LLMs locally in an efficient and privacy-preserving manner. It enables seamless offline integration of various models without requiring persistent cloud access, making it particularly suitable for academic research and other applications that require offline processing. To the best of our knowledge, this is among the first works in software engineering research to systematically deploy an *offline LLM pipeline* tailored for a domain as specialized as Quantum Software Engineering (QSE). Moreover, we selected the **DeepSeek 32B-r1** model over comparable open-source alternatives for four reasons: (1) its parameter size offers a favourable balance between computational feasibility and reasoning performance, (2) it supports instruction-following behavior suitable for technical Q&A generation, (3) it can be efficiently executed on local high-memory machines using Ollama without requiring large-scale distributed systems and (4) recent evaluations indicate that DeepSeek-R1 and its distilled variants rank among the strongest open-source models for reasoning-intensive tasks, including mathematics, coding and

[4]https://ollama.com

Algorithm 1: QSE Experiment Workflow (Model: DeepSeek-r1: 32B)

Input : QSE CSV (2,832 rows); prompt template "1"; few-shot exemplars = 2; **Model** = *DeepSeek-r1: 32B.*
Output : `outputs/qse_predictions.jsonl`, `outputs/experiment_manifest.json`.

Data Preparation & Preprocessing
RAW ← LoadCSV(`data/qse_raw.csv`)
D_1 ← Filter(RAW, HasAnswer) `// 1,492 rows`
D_2 ← Filter(D_1, answer_id = accepted_answer_id) `// 1,147 rows`
D_3 ← Filter(D_2, **not** ContainsURLorAnchor) `// 265 rows`
CLEANED ← Map(D_3, StripHTMLandNormalize) `// 265 rows`
EXEMPLARS ← SelectTopN(CLEANED, 2)
EVAL_SET ← CLEANED \ EXEMPLARS `// 263 rows`
Experimental Execution (Offline)
MODEL ← LoadModel(" `DeepSeekr1:32b`") `// Ollama runtime, no external calls`
PROMPT ← LoadPromptTemplate("1")
PREDICTIONS ← ∅
foreach *ex* ∈ *EVAL_SET* **do**
  CTX ← {`QUESTION_TEXT`: ex.question_text; `CONTEXT_IF_PROVIDED`: ""; `EXEMPLARS`: FormatFewShot(EXEMPLARS)}
  input ← RenderPrompt(PROMPT, CTX)
  out ← Generate(MODEL, input) `// Run DeepSeek:r1 32B`
  out ← EnsureJSONFields(out, ["answer"])
  ans ← NormalizeWhitespace(RemoveURLs(out.answer))
  Append(PREDICTIONS, {qid: ex.question_id; reference: ex.accepted_answer_text; generated: ans; prompt})
Result Generation & Storage
SaveJSONL(PREDICTIONS, "outputs/qse_predictions.jsonl")

logical problem-solving [12]. The algorithmic presentation of the experiment is shown in Algorithm 1.

4.1 Data Collection

This study uses the publicly available dataset of QSE discussions introduced in the study [17] by systematically extracting Q&A content from four Stack Exchange forums. The dataset contains the following features: **QuestionId:** The original Stack Exchange question identifier; **QuestionTitle:** The title or subject line of the question; **QuestionBody:** The main content or description of the question. **QuestionTags:** Tags used to categorize the question; **QuestionDate:** The date the question was posted; **AcceptedAnswerId:** The Id of the answer accepted by the question asker; **AnswerId:** The Id of the specific answer under evaluation; **AnswerBody:** The full content of the answer post; **AnswerDate:** The date the answer was posted. As this study focuses on comparing the LLM's compatibility with humans, we used only features related to questions and answers.

4.2 Data Pre-Processing

The initial input dataset comprised **2,832** Q&A records, developers' discussion related to QSE. Many questions don't have answers; we removed 1,340 such rows, yielding 1,492 instances with answers. To ensure that each instance contained a community-validated solution, we then restricted the corpus to entries where the answer_id matched the accepted_answer_id, resulting in 1,147 high-confidence Q&A pairs. Finally, because the proposed experiments utilise an offline LLM and therefore cannot resolve external resources at inference time, we excluded any rows that contained external references (e.g., URLs or href links), resulting in a modelling dataset of 265 Q&A pairs. Of these, two exemplars were reserved for few-shot prompting during development, leaving 263 Q&A pairs for training/evaluation in the main experiments. After finalizing the subset of questions for the experiments, the data is cleaned by removing the HTML tags and incidental markup, standardizing whitespace, and ensuring well-formed plain text inputs. This sequence improved textual consistency, reduced spurious tokens, and mitigated formatting artifacts that could bias the tokenizer or distract the model from core semantics.

4.3 Few-Shot Learning

This study adopts a few-shot strategy to guide answer generation with the offline DeepSeek-r1:32b model. In this technique, a couple of examples are embedded in the prompt to guide the model for in-context learning. It shows promising results for straight-forwards tasks and has quickly become a standard prompting technique now [6]. From the preprocessed corpus (Section 4.2), two Q&A instances were reserved as exemplars on the basis of their length and excluded from evaluation. These examples were embedded alongside each query in a consistent format, helping the model infer the expected structure, style, and level of detail for generating appropriate answers. The selected examples are presented in the Github[5]. The prompt used to generate answers is given in the following box 4.5.

4.4 Answer Generation

At the core of the pipeline is the interaction with the *DeepSeek-r1:32b model*, which generates an answer based on the provided prompts. For each target question, we construct a prompt by combining three key contextual elements: the *question title*, the *question body*, and associated *tags*. These components are integrated into a unified prompt using a structured template, as shown in 4.5. We adopted a few-shot prompting approach, inserting two exemplar Q&A pairs before each target question to guide the model in generating domain-relevant responses. The model is instructed to return answers in JSON format for structured parsing. However, we observed that the model's compliance with this format requirement was inconsistent. While it successfully returned valid JSON in many cases, it occasionally failed to adhere to the specified structure. Also, DeepSeek embeds its reasoning inside a <think> tag. So for the final answers, a post-processing routine is adopted to remove these <think> elements and extract the actual answer field, ensuring consistency and comparability during evaluation.

[5]https://github.com/SafiaK/Quantum_Q-A_Using_LLM/blob/main/few_shot_examples.json

4.5 Result and Evaluation

Semantic Similarity: To evaluate the quality of generated answers, semantic similarity scores are computed between the original accepted answers and the cleaned responses. For this, cosine similarity on sentence embeddings is employed rather than simple lexical overlap metrics, as this approach captures semantic meaning even when answers use different wording or phrasing. This approach is consistent with established research on answer similarity , which has been shown to be an effective metric for evaluating the quality of model-generated answers [27].

$$\text{cosine\_similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{1}$$

Where A and B represent the embedding vectors of the original and generated answers, respectively. The resulting similarity score ranges from 0 to 1, where higher values indicate greater semantic alignment between the answers. The experiments show that the similarity score ranges between 0.269 and 0.885, with a median of 0.667 and an average score of 0.644. The overall performance suggests that while the model is effective at generating answers. However, there are challenges with accuracy in more complex cases.

Analysis: This section presents a detailed analysis of the experimental outcomes obtained using the *DeepSeek-r1:32b* model on the finalized QSE evaluation set. Out of 263 cases, 3 failed to produce results, so we excluded them for further analysis. Across **260** question–answer (Q&A) instances, semantic similarity between model outputs and community-accepted references was predominantly mid-range: **36** cases (**13.6%**) were low (< 0.5), **200** (**76.9%**) were medium (0.5–0.8), and **24** (**9.2%**) were high (≥ 0.8). These statistics show that more than 86% of responses were reliable and aligned with the human-generated answers. To identify the reasons for the low score, we further investigated these cases to improve the system by mitigating these reasons in the future. In this study, these instances are labelled as failed cases. The observed failures could be attributed to either factual inaccuracies in the generated response or significant stylistic and linguistic divergence from human-authored answers, despite the content being contextually appropriate and technically reliable. To find the reason, we employed the "LLMs-as-Jury" strategy. In this technique, a handful of different LLMs are used, and their opinion are ensembled in a voting mechanism. Prior research has consistently shown that LLM-as-jury outperform single-model judgment, exhibits reduced intra-model bias, and is closely aligned with human judgment [30].

As there were minimal cases for analysis, the online large-sized LLMs were used as judges. In the first step for analysis, a diagnostic prompt was carefully crafted to ask LLM to figure out the reason for failure in low similarity cases, as detailed in Prompt 4.5. Three models were used for this evaluation: GPT (via the ChatGPT interface[6]), Gemini (via Google AI Studio[7]), and Mistral (via its official interface[8]). Each model was given a structured file containing only the failed cases, along with relevant metadata (question title, body, tags, human-authored answer, LLM-generated answer, and similarity score). The models were then instructed to complete a new column stating the likely reason for each observed failure. Each of the three LLMs independently reviewed the failed cases and provided its decision for each case. After collecting the responses, the results were consolidated by identifying cases where at least two models agreed on the reason. The following Table 2 shows the result. The results show that only 33.3% of 13.6% cases are the ones which actually hallucinate or provide incorrect responses, which is almost 4% of the total data.

Prompt for Answer Generation on QSE Using DeepSeek-r1 32B model

Answer-Generation Prompt:

```
You are an expert quantum computing researcher and
     educator with deep knowledge in quantum
     mechanics, quantum algorithms, and quantum
     information theory. Your task is to provide
     comprehensive, accurate, and well-structured
     answers to quantum computing questions.

## Instructions:
1. Analyze the given question carefully,
     considering both the explicit question and
     any implicit context.
2. Provide a detailed, technically accurate answer
     that addresses the core question.
3. Use clear explanations and avoid unnecessary
     jargon unless technical precision is required.
4. Structure your response logically with clear
     sections when appropriate.
5. Include relevant quantum computing concepts,
     principles, and examples.
6. If the question involves mathematical concepts,
     explain them in accessible terms.
7. Consider practical implications and the current
     state of quantum computing technology.
8. Ground your answer strictly in the provided
     information (question and optional context).
     Do not rely on external links or resources
     and do not include citations or URLs.

## Response Format:
Please respond with a JSON object containing your
     answer in the following format:
{
  "answer": "Your comprehensive answer here"
}

## Task:
Now, please answer the following question using
     the same format and approach.

Question:
{{QUESTION_TEXT}}

(Optional) Context:
{{CONTEXT_IF_PROVIDED}}
```

[6]https://chatgpt.com/
[7]https://aistudio.google.com/
[8]https://chat.mistral.ai/chat

Low-Similarity Investigation Prompt

```
Prompt:
We are experimenting with a Q&A platform where developers
      post quantum computing questions and practitioners
      respond. Because human replies can be delayed, we
      are integrating a specialized Large Language Model
      (LLM) to provide quick answers. In our experiment,
      some cases achieved comparatively low similarity
      between the accepted human answer and the
      LLM-generated answer; we refer to these as "failed
      instances."

We assume two possible reasons for low similarity:
1) Reason A  Factually wrong or irrelevant: The generated
      answer contains factual errors or addresses a
      different issue than the human answer.
2) Reason B  Semantically/stylistically different but
      correct: The generated answer is not wrong, but the
      language, terminology, structure, or tone differs
      substantially from the human answer, reducing
      measured similarity.

Task:
For each failed instance (record), assign exactly one
      reason label:
- "Reason A" if the generated answer is factually wrong
      or irrelevant.
- "Reason B" if the generated answer is essentially
      correct but semantically/stylistically different.

Output format (per record):
{
  "record_id": "<ID>",
  "reason": "Reason A" | "Reason B",
  "brief_justification": "One-sentence rationale
      referencing key mismatch."
}

Reference distribution (observed in our analysis):
- Reason B (semantically/stylistically different but
      correct): 88.9%
- Reason A (factually wrong or irrelevant): 11.1%
```

5 Discussion

5.1 Findings and interpretations

The proposed approach presents the preliminary investigation and evaluation of medium-sized LLM performance in processing 263 Q&A pairs. Notably, 86% of the generated responses achieved a semantic similarity above 0.5, indicating that the few-shot prompting setup produced reliable outputs for most developers' questions. Within this range, 76.0% of responses were in the mid-range (0.5–0.8), 9.1% scored high (≥ 0.8), and 14.8% were low (< 0.5), with mean and median similarity values of 0.637 and 0.667, respectively. To better understand these low-similarity cases, an *LLM-as-jury* analysis was conducted using three independent LLMs that jointly reviewed and labelled the responses. Consensus was reached for all examined cases, revealing that most low scores resulted from stylistic or structural differences rather than factual errors. Only a small portion of the total dataset represented truly incorrect or irrelevant content. The results show that, as an answer to RQ1, we can use a medium-sized LLM-based automated pipeline as an alternative to provide timely and reliable answers to developers' questions on Q&A platforms. This research finding complements the findings of Da Silva et al. [8], which show that LLMs perform comparatively better when addressing developers' questions related to programming tasks on Stake Overflow. Similarly, based on the cosine value and the LLM-as-Jury analysis approach, considering the given limitations and the dataset at hand, the reliability is quite promising, as it provides a satisfactory answer more than 95% of the time, in response to RQ2. Taken together, these findings demonstrate that *DeepSeek-r1:32b*, employed in an offline, cost-efficient, and customizable configuration, frequently captures the right concepts (as reflected in the mid-range distribution).

Table 2: Consensus Summary of Failure Reasons

Failure Reason	Percentage
Semantically correct, stylistically different	66.7%
Factually incorrect or irrelevant	33.3%

5.2 Implications of the Proposed Approach

The findings show LLMs have significant potential to enhance answer generation in Q&A platforms by serving as intelligent assistants that complement human expertise. For developers, LLMs can be integrated into QSE support systems to provide timely, relevant responses to routine inquiries, reducing waiting times and aiding users with foundational challenges. When guided, LLMs can provide concise answers that align with community norms through clarity, brevity, and code-focused explanations. Implementing post-processing to eliminate verbosity, reasoning traces, or structured wrappers ensures outputs remain natural and valuable. Additionally, evaluation expert feedback or comparison with community-accepted answers can help refine these models to reflect domain-specific terminology and reasoning patterns found in quantum computing frameworks like Qiskit and Q. This integration enables developers to use LLMs as collaborative partners for routine questions while human experts focus on complex, novel topics, enhancing the efficiency and responsiveness of QSE knowledge-sharing workflows.

5.3 Threats to Validity

While this study provides valuable insights, several threats to the validity of the findings must be considered. The dataset consists of 263 Q&A pairs related to QSE developers' discussions, limiting the applicability. Although the dataset covers diverse categories, it needs testing by extending to emerging domains, like blockchain and DevOps, through recent Q&A platform data. The proposed approach, used DeepSeek-r1:32b, a medium-sized LLM, limits exploration of other medium-sized LLMs,like Qwen2.5-32B, Gemma 2 27B, and Mistral 22B. These will be explored to compare their performances in answer generation. We used a medium-sized LLM, while utilising LLMs, including ChatGPT, Gemini, and Mistral, as LLM-as-Jury to validate low-similarity cases, whereas literature uses ChatGPT and Llama for answer generation. Importantly, since our error analysis revealed that a small but non-negligible portion of the LLM-generated answers (approximately 4% of the full dataset) contained factual inaccuracies, this limitation reinforces that medium-sized LLMs should be viewed as assistive rather then authoritative tools for addressing highly technical QSE queries.

6 Conclusion

This study examines the potential of medium-sized LLMs to generate precise and contextually relevant responses to QSE queries. The findings suggest that LLMs, particularly the DeepSeek-r1:32B model, can provide timely answers to the developers' QSE questions on Q&A platforms, with **85%** of the 263 evaluated items achieving a semantic similarity above **0.5**. To investigate the remaining **14%** of low-scoring cases, an LLM-as-Jury approach was used with two possible reasons identified through manual evaluation, employing a two-out-of-three consensus rule. The analysis showed that most instances were valid in content but differed from human references in phrasing, tone, or structure, with only 4% representing unreliable answers where LLM didn't perform well. Overall, medium-sized LLMs with few-shot prompting are an effective alternative for generating timely answers to developers' questions on Q&A platforms for emerging domains. The results demonstrate that medium-sized LLms can practically support QSE developers by providing immediate, context aware assistance in situations where expert responses may be delayed.However, the proposed approach needs to be evaluated on a relatively larger dataset and its performance assessed with QSE experts.

Replication Package: The replication package is publicly available at the following GitHub repository:

https://github.com/SafiaK/Quantum_Q-A_Using_LLM

References

[1] Aakash Ahmad, Ahmed B Altamimi, and Jamal Aqib. 2024. A reference architecture for quantum computing as a service. *Journal of King Saud University-Computer and Information Sciences* 36, 6 (2024), 102094.

[2] 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.

[3] Mst Shamima Aktar, Peng Liang, Muhammad Waseem, Amjed Tahir, Aakash Ahmad, Beiqi Zhang, and Zengyang Li. 2025. Architecture decisions in quantum software systems: An empirical study on Stack Exchange and GitHub. *Information and Software Technology* 177 (2025), 107587.

[4] Vasudev Bhat, Adheesh Gokhale, Ravi Jadhav, Jagat Pudipeddi, and Leman Akoglu. 2014. Min (e) d your tags: Analysis of question response time in stackoverflow. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*. ieee, 328–335.

[5] Sardar Bonabi, Sarah Bana, Vijay Gurbaxani, and Tingting Nian. 2025. Beyond Code: The Multidimensional Impacts of Large Language Models in Software Development. *arXiv preprint arXiv:2506.22704* (2025).

[6] Kaiyan Chang, Songcheng Xu, Chenglong Wang, Yingfeng Luo, Xiaoqian Liu, Tong Xiao, and Jingbo Zhu. 2024. Efficient Prompting Methods for Large Language Models: A Survey. arXiv:2404.01077 [cs.CL] https://arxiv.org/abs/2404.01077

[7] Leuson Da Silva, Jordan Samhi, and Foutse Khomh. 2025. LLMs and Stack Overflow discussions: Reliability, impact, and challenges. *Journal of Systems and Software* (2025), 112541.

[8] Musengamana Jean de Dieu, Peng Liang, and Mojtaba Shahin. 2025. How Do OSS Developers Reuse Architectural Solutions from Q&A sites: An Empirical Study. *IEEE Transactions on Software Engineering* (2025).

[9] Zack Delile, Sean Radel, Joe Godinez, Garrett Engstrom, Theo Brucker, Kenzie Young, and Sepideh Ghanavati. 2023. Evaluating privacy questions from stack overflow: Can chatgpt compete?. In *2023 IEEE 31st international requirements engineering conference workshops (REW)*. IEEE, 239–244.

[10] Richard Fitzgerald. 2000. What really gives a quantum computer its power? *Physics Today* 53, 1 (2000), 20–22.

[11] Antonio García de la Barrera, Ignacio García-Rodríguez de Guzmán, Macario Polo, and Mario Piattini. 2023. Quantum software testing: State of the art. *Journal of Software: Evolution and Process* 35, 4 (2023), e2419.

[12] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[13] Mohammed Mehedi Hasan, Mahady Hasan, Mamun Bin Ibne Reaz, and Jannat Un Nayeem Iqra. 2024. An exploratory analysis of Community-based Question-Answering Platforms and GPT-3-driven Generative AI: Is it the end of online community-based learning? *arXiv preprint arXiv:2409.17473* (2024).

[14] Denis Helic and Tiago Santos. 2025. Stack Overflow Is Not Dead Yet: Crowd Answers Still Matter. *arXiv preprint arXiv:2509.05879* (2025).

[15] Dylan Herman, Cody Googin, Xiaoyuan Liu, Yue Sun, Alexey Galda, Ilya Safro, Marco Pistoia, and Yuri Alexeev. 2023. Quantum computing for finance. *Nature Reviews Physics* 5, 8 (2023), 450–465.

[16] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2023. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* (2023).

[17] Mobashir Husain, Muhammad Sohail Khan, Javed Ali Khan, Nek Dil Khan, Arif 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 *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*. 1742–1755.

[18] Samia Kabir, David N Udo-Imeh, Bonan Kou, and Tianyi Zhang. 2024. Is stack overflow obsolete? an empirical study of the characteristics of chatgpt answers to stack overflow questions. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. 1–17.

[19] 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.

[20] 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).

[21] 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.

[22] Jinrun Liu, Xinyu Tang, Linlin Li, Panpan Chen, and Yepang Liu. 2023. Which is a better programming assistant? A comparative study between chatgpt and stack overflow. *arXiv preprint arXiv:2308.13851* (2023).

[23] Mohammadreza Molavi, Mohammadreza Tavakoli, Mohammad Moein, Abdolali Faraji, and Gábor Kismihók. 2025. LLM-Driven Personalized Answer Generation and Evaluation. In *International Conference on Artificial Intelligence in Education*. Springer, 187–194.

[24] Juan Manuel Murillo, Jose Garcia-Alonso, Enrique Moguel, Johanna Barzen, Frank Leymann, Shaukat Ali, Tao Yue, Paolo Arcaini, Ricardo Pérez-Castillo, Ignacio García-Rodríguez de Guzmán, et al. 2025. Quantum software engineering: Roadmap and challenges ahead. *ACM Transactions on Software Engineering and Methodology* 34, 5 (2025), 1–48.

[25] CO Oyeniran, Adebunmi Okechukwu Adewusi, Adams Gbolahan Adeleke, Lucy Anthony Akwawa, and Chidimma Francisca Azubuko. 2023. Advancements in quantum computing and their implications for software development. *Computer Science & IT Research Journal* 4, 3 (2023), 577–593.

[26] Romil Rawat, Rajesh Kumar Chakrawarti, Sanjaya Kumar Sarangi, Jaideep Patel, Vivek Bhardwaj, Anjali Rawat, and Hitesh Rawat. 2023. *Quantum Computing in Cybersecurity*. John Wiley & Sons.

[27] Julian Risch, Timo Möller, Julian Gutsch, and Malte Pietsch. 2021. Semantic answer similarity for evaluating question answering models. *arXiv preprint arXiv:2108.06130* (2021).

[28] Paola Spoletini. 2023. Towards quantum requirements engineering. In *2023 IEEE 31st International Requirements Engineering Conference Workshops (REW)*. IEEE, 371–374.

[29] Krishna Upadhyay, Vinaik Chhetri, AB Siddique, and Umar Farooq. 2025. Analyzing the Evolution and Maintenance of Quantum Software Repositories. *arXiv preprint arXiv:2501.06894* (2025).

[30] Pat Verga, Ameet Singh Rawat, Asli Tokgöz, Ashish Sabharwal, and Sanjiv Kumar. 2024. Replacing judges with juries: Evaluating LLM generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796* (2024).

[31] Rhonda Walthall and Sunil Dixit. 2022. *Impact of quantum computing in aerospace*. SAE International.

[32] Patricia Widjojo and Christoph Treude. 2023. Addressing compiler errors: Stack overflow or large language models? *arXiv preprint arXiv:2307.10793* (2023).

[33] Zebo Yang, Maede Zolanvari, and Raj Jain. 2023. A survey of important issues in quantum computing and communications. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1059–1094.

[34] 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.

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