HybridSum: Hybrid Summarization of Stack Overflow Posts

Nurjahan*
Louisiana State University
Baton Rouge, LA, USA
nurja1@lsu.edu

Kaushani Samarawickrama
Louisiana State University
Baton Rouge, LA, USA
ksamar2@lsu.edu

Md Saidur Rahman
Louisiana State University
Baton Rouge, LA, USA
mrahm65@lsu.edu

Abstract

Stack Overflow is a widely used platform where software engineers share and seek solutions to practical programming problems. However, the overwhelming volume of posts often makes it difficult for users to efficiently locate relevant information. Automatic summarization offers a promising solution by improving the readability and searchability of content, allowing developers to quickly extract key insights without navigating lengthy discussions. This study proposes an automated summarization approach for Stack Overflow posts using two state-of-the-art language models: LLaMA 3.1–8B and LLaMA 3.3–70B. We evaluated the performance of both models in generating concise and informative summaries of question–answer threads. A comparative analysis is performed using standard natural language processing metrics, including ROUGE, BLEU, and BERTScore, to assess the quality of summarization. To complement the quantitative results, we also presented a qualitative comparison of the outputs of the hybrid pipeline. Our results provide insights into the trade-offs between model size and performance, highlighting the potential of large language models to enhance developer productivity through improved information access.

Keywords

Stack Overflow, Automated Summarization, Large Language Models (LLMs), LLaMA 3, Zero-shot Learning, Few-shot Learning

ACM Reference Format:
Nurjahan, Kaushani Samarawickrama, and Md Saidur Rahman. 2018. HybridSum: Hybrid Summarization of Stack Overflow Posts. In *Proceedings of Department of Computer Science, LSU (SWE).* ACM, New York, NY, USA, 8 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Online programming forums have become essential resources for software developers. Among these, Stack Overflow stands out as a dominant platform for sharing and discovering solutions to coding problems. However, the sheer volume of content—often including hundreds of answers to a single question—can make it difficult for users to efficiently extract relevant and high-quality information.

Imagine a developer searching for the most efficient sorting algorithm for a given scenario. A Stack Overflow query may yield hundreds of answers, with the top-voted ones being verbose and time-consuming to read. Due to time constraints, the developer may only skim a few responses, potentially missing valuable insights buried deeper in the thread. This highlights a broader issue: the cognitive burden and inefficiency associated with manually processing large volumes of technical discussion. The primary motivation for this work is to help programmers retrieve relevant information in a concise manner, thereby improving both efficiency and decision-making. Stack Overflow often contains dozens of answers per question, making it difficult for users to sift through all available responses. As a result, developers tend to focus on a small number of highly ranked posts, which increases the risk of overlooking important but less visible contributions. Concise summarization can help developers quickly assess multiple answers, extract key takeaways, and make informed choices without reading every response in full. Moreover, there is a significant lack of labeled training data specifically curated for Stack Overflow summarization tasks, which presents a further challenge in developing robust models. This research is driven by the need to overcome these issues and ultimately streamline information retrieval on developer forums through automated summarization.

The core problem this research addresses is how to automatically generate concise, accurate, and context-aware summaries of Stack Overflow answers to support developer productivity. While Stack Overflow is a rich source of collective knowledge, its unstructured and verbose content often hinders effective information retrieval. Existing summarization techniques, whether heuristic or retrieval-based, struggle to handle the complexity of natural language in SO discussions. Many SO posts include verbose explanations, redundant details, and unnecessary information, making it difficult for developers to efficiently locate key takeaways, best practices, or crucial code snippets. These limitations suggest a need for more advanced summarization methods that can understand technical language, capture salient information, and present it concisely.

This study investigates the use of instruction-tuned large language models—specifically LLaMA 3–8B and 70B—for zero-shot and few-shot summarization of Stack Overflow answers. It seeks to answer the following research questions:

RQ1: Among the evaluated models and prompting strategies, which yields the most accurate and useful summaries for developer Q&A content?

RQ2: Can smaller models (e.g., LLaMA 3–8B) perform competitively with larger models (e.g., LLaMA 3–70B), particularly under few-shot settings, thereby offering a cost-efficient alternative?

This paper presents a comparative study of LLaMA 3 models applied to Stack Overflow post summarization. We evaluate the effectiveness of zero-shot and few-shot prompting strategies, exploring how prompt design influences summarization quality. We also propose a lightweight data preparation pipeline tailored to the

SWE, Baton Rouge, LA, USA

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

structure of Stack Overflow content and assess performance using standard evaluation metrics such as ROUGE and BERTScore. Our findings offer insights into model size vs. performance trade-offs and the potential of LLMs for real-world developer tools.

The remainder of this paper is organized as follows: Section 2 surveys related work on summarization and developer-focused NLP tasks. Section 3 describes the dataset and preprocessing techniques. Section 4 outlines the summarization models, prompt strategies, and evaluation methods. Section 5 presents the experimental results and analysis. Section 6 discusses implications, limitations, and potential extensions. Section 7 concludes with a summary and directions for future research.

2 Related works

Automated summarization has been extensively explored in both general and domain-specific contexts, with particular interest in adapting these techniques to developer forums like Stack Overflow (SO). Early work focused on traditional extractive methods, while recent advances have leveraged deep learning and large language models to improve the quality and contextual relevance of summaries.

General Summarization Methods: Liu et al. [5] introduced BERTSUM, an adaptation of BERT for extractive summarization, which uses additional [CLS] tokens to represent sentence-level semantics. Their model was trained with several classifiers—Simple Classifier, Inter-sentence Transformer, and LSTM—and achieved state-of-the-art performance on CNN/DailyMail and NYT datasets. Notably, interval segment embeddings and trigram blocking were found to further enhance model performance. Liu et al. [6] also proposed a document-level encoder for both extractive and abstractive summarization, fine-tuning different components of the model with specialized optimizers. This hybrid architecture delivered strong results across benchmark datasets like CNN/DailyMail, NYT, and XSum.

Summarization on Developer Forums and Stack Overflow: As developer forums present unique challenges—such as technical jargon, code snippets, and lengthy discussions—researchers have increasingly adapted summarization models for this domain. Xu et al. [12] proposed AnswerBot, a three-stage pipeline that retrieves relevant questions, selects useful answer paragraphs, and generates diverse summaries using Maximal Marginal Relevance (MMR). They trained the model on a large repository of 228,817 Java-related SO questions, employing metrics such as TF-IDF, semantic features, and user behavior (e.g., upvotes, answer order) to rank relevance and salience.

Similarly, LASSO (Leveraging Answer Semantics for Stack Overflow) by Nguyen et al. [9] uses a bi-LSTM network combined with BERT embeddings to capture query relevance and contextual salience. By extending the SOSum dataset with over 100 new questions and 1,300 answers, LASSO achieved promising results with a 0.6897 F1-score and 0.7597 accuracy, outperforming prior baselines in extractive summarization.

Yang et al. introduced two related tools for SO summarization. In [14], they proposed TechSumBot, a three-module architecture for answer summarization that includes usefulness ranking (via a pretrained transformer), centrality estimation (via TextRank), and redundancy removal using contrastive learning trained on SO duplicate question pairs. In a subsequent study [13], they extended the system into an interactive web-based tool, TechSumBot+, with configurable summarization modes. Tested on the TechSumBench dataset and evaluated through user studies, it showed up to 36.59% improvement in ROUGE-2 over previous models and high user ratings for usefulness and diversity.

Recent LLM-Based and Domain-Shifted Models: Kou et al. [3] proposed ASSORT, a deep learning-based method for summarizing Quora posts, with two variants: ASSORTs, a supervised model using BERT embeddings and domain-specific features, and ASSORTis, an indirect supervision approach based on domain adaptation. Both versions significantly outperformed six baselines including BERTSUM and LexRank, achieving up to 13% better F1 scores.

Chhikara et al. [2] developed LaMSUM, a novel framework for extractive summarization using zero-shot large language models. To overcome LLM limitations like context length and the tendency toward abstraction, LaMSUM splits content into chunks, applies prompt shuffling, and uses voting strategies (plurality, proportional approval, Borda count) to select representative sentences. Evaluated across multiple datasets (Claritin, US-Election, MeToo), LaMSUM outperformed strong baselines including GPT-4o and BERTSUM on ROUGE metrics.

Stack Overflow as Developer Documentation: Naghshzan et al. [8] explored the use of Stack Overflow posts as unofficial documentation for Android APIs. They applied TextRank over preprocessed content using SO-based word2vec embeddings to extract summaries. Their approach was evaluated with Android developers, with 73% finding the summaries accurate, 58% coherent, and 84% considering them useful as complementary resources in IDEs.

3 Methodology

This section outlines the methodology employed in our study. We propose a hybrid two-stage summarization pipeline, referred to as **HybridSum**, which combines the generative capabilities of large language models with the factual grounding power of natural language inference models (see Figure 1).

The process begins by loading the dataset—Stack Overflow posts from the SoSum dataset—which comprises 3,130 answers spanning a mix of conceptual, how-to, and bug-fix questions. Using either the LLaMA 3–8B or LLaMA 3–70B model, we generate an abstractive summary of each answer post.

Following the generation of the abstractive summary, we apply a filtering mechanism using a RoBERTa-large model fine-tuned on the Multi-Genre Natural Language Inference (MNLI) dataset. Each sentence from the original answer is evaluated for textual entailment with respect to the LLaMA-generated summary. Only those sentences with an entailment probability greater than 0.6 are retained. This step ensures that the final extractive summary is factually grounded and free from hallucinations, providing users with concise and trustworthy information.

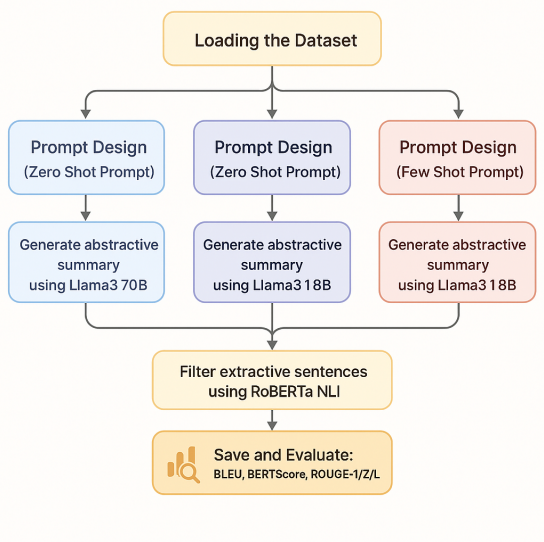

Figure 1: HybridSum Summarization Pipeline

3.1 Dataset: SoSum

We utilize the SoSum dataset, a curated collection of approximately 3,130 Stack Overflow posts, encompassing a diverse range of technical question types. The dataset includes three primary categories of questions: (1) *conceptual*—theory-oriented or explanatory inquiries, (2) *bug-fix*—posts aimed at identifying and resolving programming errors, and (3) *how-to*—procedural questions on implementing specific functionalities or using particular tools or libraries. Each entry in the dataset consists of the full answer text (`answer_body`) along with associated metadata, including answer_id, question_id, question_tags, and question_type. Additionally, each answer is segmented into individual sentences annotated with binary relevance labels (`truth`: 1 for relevant, 0 for irrelevant), indicating whether a sentence contributes to the ideal summary of the answer. This structure enables experimentation with both abstractive and extractive summarization approaches and provides a reliable basis for quantitative evaluation using sentence-level ground-truth annotations.

Example Entry from the SoSum Dataset. An example entry from the SoSum dataset is shown below, illustrating the structure and types of data available:

- **answer_id**: 21
- **question_id**: 9
- **question_type**: 2 (corresponding to a *how-to* question)
- **question_tags**: `['c#', '.net', 'datetime']`
- **question_title**: *How do I calculate someone's age in C#?*
- **answer_body**:
  Many years ago, to provide an age calculator gimmick on my website, I wrote a function to calculate age to a fraction. This is a quick port of that function to C# (from the PHP version). I'm afraid I haven't been able to test the C# version, but hope you enjoy all the same! (Admittedly this is a bit gimmicky for the purposes of showing user profiles on Stack Overflow, but maybe readers will find some use for it. :-))
- **sentences (with `truth` labels)**:
  - `[1]` *"Many years ago, to provide an age calculator gimmick on my website, I wrote a function to calculate age to a fraction."* — `truth: 1`
  - `[2]` *"This is a quick port of that function to C# (from the PHP version)."* — `truth: 0`
  - `[3]` *"I'm afraid I haven't been able to test the C# version, but hope you enjoy all the same!"* — `truth: 0`
  - `[4]` *"(Admittedly this is a bit gimmicky for the purposes of showing user profiles on Stack Overflow, but maybe readers will find some use for it."* — `truth: 0`
  - `[5]` *":-))"* — `truth: 0`

This structure illustrates how each post in the SoSum dataset is annotated at the sentence level, enabling both training and evaluation of extractive summarization models.

3.2 Models

We employ two instruction-tuned large language models from the LLaMA 3 series [11] —LLaMA 3.3–70B and LLaMA 3.1–8B—for generating abstractive summaries, alongside RoBERTa-large fine-tuned on MNLI [7] for entailment-based sentence filtering.

3.2.1 LLaMA 3.3–70B. In this study, we utilize Meta's LLaMA 3.3 70B Instruct model, which contains 70 billion parameters and represents the most powerful publicly available model in the LLaMA 3.3 series released in 2024. The instruction-tuned variant is optimized for high-quality generation and alignment with natural language tasks through supervised fine-tuning and reinforcement learning. The model is instantiated via the meta-llama/Llama-3.3-70B-Instruct identifier using a high-performance inference backend. Deployment is configured with tensor parallelism (size = 2) to enable distribution across multiple GPUs. To accommodate large sequence windows and reduce memory bottlenecks, we set the maximum context length to 2048 tokens, enable eager execution, and allocate 16 GB of swap space. The model operates with floating point 8 (fp8) quantization to optimize memory usage while preserving generation fidelity. GPU memory utilization is capped at 85%, and we allow a maximum of four parallel input sequences.

3.2.2 LLaMA 3.1–8B. The LLaMA 3.1 8B Instruct model is an instruction-tuned language model with 8 billion parameters, offering a lightweight yet capable alternative within Meta's LLaMA 3.1 series. Compared to its larger 70B counterpart, the 8B variant is optimized for faster inference and lower memory consumption, making it suitable for real-time or constrained environments without significant degradation in performance on general language understanding and generation tasks. Although smaller in scale, it retains strong zero-shot and few-shot generalization abilities, particularly when paired with carefully designed prompts. Its instruction-tuning enhances its responsiveness to task descriptions, allowing it to generate coherent and contextually appropriate outputs even in zero-shot scenarios.

3.2.3 RoBERTa-large-MNLI. The RoBERTa-large-MNLI model [7] is a 355-million-parameter transformer-based classifier built upon the RoBERTa-large architecture and fine-tuned on the Multi-Genre

Natural Language Inference (MNLI) corpus. RoBERTa (A Robustly Optimized BERT Approach) improves upon the original BERT model by removing the next-sentence prediction objective, training on significantly larger batches, using longer sequences, and employing dynamic masking. The MNLI fine-tuned variant enables the model to classify sentence pairs into entailment, contradiction, or neutral categories, making it particularly well-suited for natural language inference (NLI) tasks. In our setup, RoBERTa-large-MNLI is used to evaluate the semantic alignment between model-generated summaries and input sentences, allowing for entailment-based filtering of extractive content. Its robust generalization across genres and its sensitivity to fine-grained semantic relationships make it a strong choice for sentence-level filtering in hybrid summarization pipelines.

3.3 Prompting

Prompting is a technique for conditioning large language models (LLMs) to perform specific tasks by providing natural language instructions, input examples, or task descriptions—without requiring model retraining or fine-tuning. This approach significantly reduces computational overhead and enables flexible adaptation to a wide range of downstream tasks using the same pre-trained model. In this study, we adopt two prompting strategies: *zero-shot prompting* and *few-shot prompting*.

Zero-shot Prompting. In zero-shot prompting [1], the model is given only a task instruction without any input-output exemplars. It relies solely on its pretraining to interpret the instruction and generate a response. This method evaluates the model's ability to generalize to unseen tasks based on instruction semantics alone. In the zero-shot setting, we use the instruction: *"'You are an expert at summarizing technical answers from Stack Overflow. Create a brief, accurate summary that captures the essential technical information and any code snippets or key concepts."'* In the HybridSum framework, we applied zero-shot prompting to both LLaMA 3.3–70B and LLaMA 3.1–8B to assess their inherent summarization capabilities under instruction-only conditions.

Few-shot Prompting. Few-shot prompting involves providing a language model with a small number of input–output examples (typically 1–5) prior to presenting the actual inference input. These exemplars act as implicit demonstrations, allowing the model to infer both the expected output format and the semantic scope of the task. This technique is especially effective in domain-specific or structured generation tasks where conventions around conciseness, technical focus, and clarity are critical. In our implementation, few-shot prompting was applied to the LLaMA 3.1–8B model to improve summarization quality for Stack Overflow answers. The prompt was manually constructed with explicit instructions emphasizing brevity, technical focus, and avoidance of informal or tutorial-style expressions. A simplified version of the instruction block is as follows:

> *You are an expert summarizer for Stack Overflow answers. Your task is to generate CONCISE summaries of technical answers that capture the essential information.*
>
> - Include key code snippets in your summary when present
> - Focus on the answer's main technical solution
> - Be brief but complete
> - Do not repeat the examples in your output
> - Do not include phrases like "Here's how to..." or "The solution is..."
> - Start directly with the technical explanation

To complement this instructional setup, we included two curated exemplars of Stack Overflow question–answer pairs alongside their ideal summaries. These were chosen to reflect common summarization patterns such as descriptive explanation and conceptual comparison. The few-shot examples were fixed across all experiments to ensure consistency in prompt context. Two examples are shown below:

- **Question:** *How do I calculate someone's age in C#?*
  Answer: *[...] I wrote a function to calculate age to a fraction. This is a quick port of that function to C#.*
  Summary: *A function to calculate age to a fraction, ported from PHP to C#.*
- **Question:** *What are MVP and MVC and what is the difference?*
  Answer: *MVP = Model-View-Presenter, MVC = Model-View-Controller. [...] The main difference is that in MVC the Model updates the View directly.*
  Summary: *MVP = Model-View-Presenter, MVC = Model-View-Controller. The key difference is MVC allows the model to update the view directly.*

Empirically, the introduction of few-shot prompting led to improved retention of pedagogical analogies and clearer abstraction in the generated summaries. This improvement was particularly notable in smaller models such as LLaMA 3.1–8B, which benefited from the structural guidance provided by exemplar demonstrations.

4 Results

4.1 Experimental Setup

All experiments were conducted on the QBD2 node of the Louisiana Optical Network Initiative (LONI) High-Performance Computing (HPC) cluster, optimized for large-scale, throughput-intensive scientific workloads.

System Configuration. Each experiment was executed using two NVIDIA A100 80GB GPUs with approximately 135.8 GB of total GPU memory, and 4 CPU cores for preprocessing and data loading. Jobs were submitted using the SLURM job scheduler in both batch and interactive modes, supporting automated workflows and real-time debugging, respectively.

Quantization Strategy. To reduce GPU memory consumption and accelerate inference, we deployed all LLaMA models using FP8 quantization via the vLLM engine (v0.8.2). This allowed the 70B parameter model to operate within the memory constraints of two NVIDIA A100 80GB GPUs, lowering its footprint to approximately 135.8 GB while maintaining high summarization quality. FP8 quantization strikes a balance between precision and efficiency, introducing negligible degradation in output coherence compared

to full-precision (FP32) inference. The adoption of low-bit inference thus made it feasible to conduct large-scale evaluations on high-capacity models without fine-tuning or parameter pruning.

Libraries and Toolchain. The following software libraries were used: `vLLM (v0.8.2)` for optimized large language model inference; `transformers` and `torch` for model loading and execution; and `rouge-score`, `bert-score`, and `nltk` for evaluation and preprocessing.

Model Sampling Parameters. Inference was performed with a maximum token length of 256 for LLaMA 3.1–8B and 128 for LLaMA 3.3–70B. Temperature was set to 0.6 and top-p to 0.9. Models were deployed in FP8 quantization mode using `tensor_parallel_size=2`, a context length of 2048, and `max_num_seqs=4`, ensuring efficient GPU memory utilization (capped at 85%).

Resource Utilization. The LLaMA 3.3–70B model consumed approximately 67.9 GB of memory per A100 GPU. The LLaMA 3.1–8B model used an estimated 170 GB of GPU memory in total, potentially distributed across multiple GPUs (exact configuration may vary).

Runtime Statistics. Total inference time was recorded as 1 hour, 56 minutes, and 28 seconds, while wall-clock time was 1 hour, 19 minutes, and 17 seconds. The difference indicates effective parallelism and batching during execution.

4.2 Quantitative Comparative Evaluation of Summarization Outputs

We evaluated the summarization performance of three LLaMA 3 models—**LLaMA 3 70B**, **LLaMA 3 8B (zero-shot)**, and **LLaMA 3 8B (few-shot)**—using three widely accepted evaluation benchmarks: **BERTScore**, **BLEU**, and **ROUGE**. These metrics respectively assess semantic similarity, n-gram overlap, and lexical coverage of generated summaries compared to human-written references.

4.2.1 BERTScore. To assess the semantic quality of the generated summaries, we employed BERTScore [15], a metric that compares token-level contextual embeddings from a pre-trained BERT model to evaluate similarity between candidate and reference summaries. Unlike ROUGE or BLEU, which rely on surface-level n-gram overlap, BERTScore provides a more robust evaluation of meaning preservation—particularly important in abstractive and hybrid summarization tasks.

Figure 2 presents BERTScore precision, recall, and F1 scores across three configurations: LLaMA 3.3 70B, LLaMA 3.1 8B, and LLaMA 3.1 8B with few-shot prompting. Among the models, LLaMA 3.3 70B achieved the highest semantic alignment, with a precision of 0.8911 and an F1 score of 0.8786, indicating both accurate and concise summary generation. LLaMA 3.1 8B performed slightly lower but still competitively, reaching an F1 of 0.8722. Interestingly, the few-shot variant of LLaMA 3.1 8B yielded the highest recall (0.8734), suggesting that few-shot prompting encourages broader content coverage. However, this came at the cost of precision (0.8591), implying more extraneous or loosely relevant information was included. These results highlight the trade-off between coverage and conciseness and suggest that larger models, combined with structured prompting, yield more semantically faithful summaries.

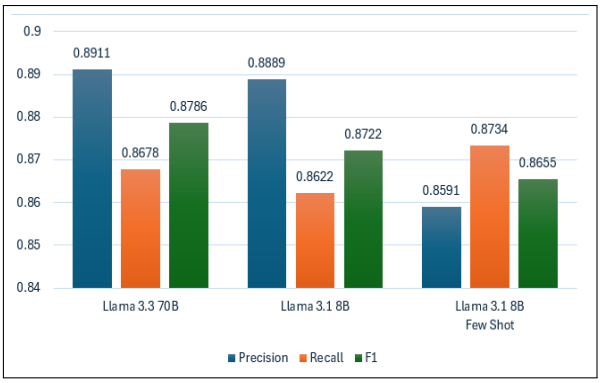

Figure 2: BERTScore Comparison Across Models

4.2.2 ROUGE Score. To evaluate the lexical overlap between generated and reference summaries, we use the ROUGE (Recall-Oriented Understudy for Gisting Evaluation) metric [4], specifically reporting F1 scores for ROUGE-1, ROUGE-2, and ROUGE-L. ROUGE-1 measures the overlap of unigrams (individual words), providing a basic indicator of content similarity. ROUGE-2 evaluates bigram (two-word sequence) overlap, offering insight into fluency and short-span coherence. ROUGE-L computes the longest common subsequence between candidate and reference texts, capturing both content and sentence-level structure alignment.

As shown in Figure 3, the LLaMA 3.1 8B Few-Shot model achieves the highest ROUGE scores across all three metrics: ROUGE-1 F1 of 0.6355, ROUGE-2 F1 of 0.5928, and ROUGE-L F1 of 0.6209. This suggests that few-shot prompting helps the model better reproduce the lexical patterns and structure of human-written summaries. In contrast, LLaMA 3.3 70B, while semantically stronger (as confirmed by BERTScore), has slightly lower ROUGE scores (e.g., ROUGE-1 F1 at 0.558), possibly due to using more diverse or abstract phrasings that diverge from the surface form of the reference. The structured LLaMA 3.1 8B version trails both, particularly in ROUGE-2 (0.4803), indicating challenges in matching phrase-level expressions. These results highlight that while larger models excel at meaning preservation, prompt design plays a crucial role in achieving surface-level fidelity, particularly when measured through ROUGE.

4.2.3 BLEU Score. To complement our evaluation with ROUGE and BERTScore, we also compute the BLEU score (Bilingual Evaluation Understudy) [10], a traditional metric originally developed for machine translation but widely used in summarization. BLEU assesses how many n-gram overlaps exist between the generated summary and the reference, typically favoring exact token matches. It rewards precision—how much of the generated output appears in the reference—while penalizing overly short generations through a brevity penalty.

As shown in Figure 4, the LLaMA 3.1 8B Few-Shot configuration achieves the highest BLEU score (0.5017), significantly outperforming both the LLaMA 3.3 70B (0.4063) and LLaMA 3.1 8B structured

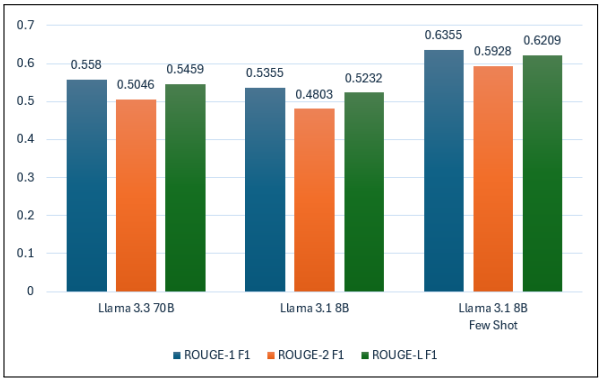

Figure 3: ROUGE Score Comparison Across Models

prompting variant (0.3803). This indicates that few-shot prompting leads to summaries that more closely mirror the lexical patterns and sequences found in the reference texts. On the other hand, while LLaMA 3.3 70B excels in semantic and abstractive quality (as reflected in BERTScore), its more creative or varied phrasing may reduce exact n-gram matches, thereby lowering its BLEU score. These findings further reinforce the observation that few-shot examples help guide the model to generate more reference-like output, especially in tasks where fidelity to the source phrasing is rewarded.

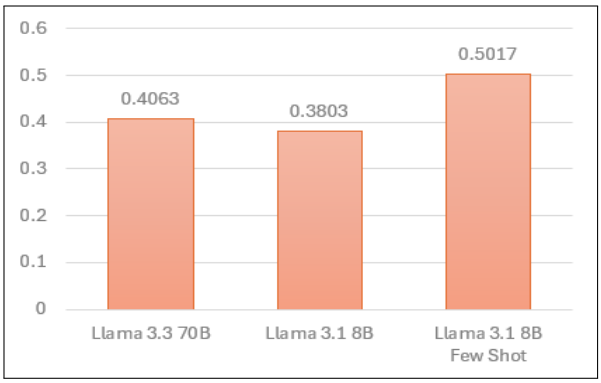

Figure 4: BLEU Score Comparison Across Models

4.2.4 Overall Performance Comparison. Taken together, the evaluation results across BERTScore, ROUGE, and BLEU reveal important trade-offs between semantic quality, lexical overlap, and generation strategy. The LLaMA 3.3 70B model consistently excels in semantic alignment, achieving the highest BERTScore precision (0.8911) and F1 (0.8786), suggesting that its summaries are meaningfully aligned with reference answers, even if phrased differently. However, it does not outperform in ROUGE or BLEU, likely due to more abstractive and varied language use that reduces surface-level token overlap. In contrast, the LLaMA 3.1 8B Few-Shot configuration scores highest on ROUGE-1, ROUGE-2, ROUGE-L, and BLEU, indicating that few-shot prompting effectively guides the model to produce summaries with greater lexical fidelity to reference texts. This setup, while slightly lower in BERTScore precision, achieves the best recall (0.8734), showing that it captures more content from the ground truth. The standard LLaMA 3.1 8B model using structured prompting trails behind both alternatives across all metrics. Overall, these findings highlight that model size and prompting strategy affect different dimensions of summary quality: larger models enhance semantic depth, while tailored prompting (e.g., few-shot) enhances faithfulness to human-generated expressions. The best configuration thus depends on whether the task prioritizes meaning preservation or surface-form alignment.

4.3 Qualitative Comparative Evaluation of Summarization Outputs

4.3.1 Case Study: C# Loop Control (Question ID: 6414, Answer ID: 6442). This section presents a qualitative comparative analysis of summarization outputs produced by different configurations of the LLaMA model family for a representative Stack Overflow question-answer (QA) pair concerning the use of `break` and `continue` statements in C# loop control. The original answer was

> A really easy way to understand this is to place the word "loop" after each of the keywords. The terms now make sense if they are just read like everyday phrases.
> <code>break</code> loop - looping is broken and stops.
> <code>continue</code> loop - loop continues to execute with the next iteration.

The evaluated models include LLaMA 3.1 8B in both zero-shot and few-shot settings, and LLaMA 3.3 70B in a zero-shot setting. The reference summary highlights an intuitive instructional device used in the original answer: appending the word “loop” to each keyword to aid user understanding. The generated summary of LLaMA 3.1 8B — Zero-Shot Setting is:

> `break` loop stops the iteration, `continue` loop skips to the next iteration.

The output is concise and factually accurate, delivering a brief declarative description of the behavior of the keywords. However, the model omits the original answer’s pedagogical analogy and fails to contextualize the explanation in a user-friendly framework. The summary reads more like a syntax manual than an instructional aid. Meanwhile, the generated summary of LLaMA 3.1 8B — Few-Shot Setting is:

> The terms now make sense if they are just read like everyday phrases. `break` loop – looping is broken and stops. `continue` loop – loop continues to execute with the next iteration.

Few-shot prompting improves pedagogical clarity. The model mirrors the original analogy and enhances comprehension, although it borders on extractive summarization. Meanwhile, the generated summary of LLaMA 3.3 70B Zero-Shot Setting is:

> **Break vs. Continue in C# Loops**
> `break` loop: stops the loop entirely when encountered.
> `continue` loop: skips the current iteration and moves on to the next one.
> This can be remembered by reading the keywords

> as everyday phrases: "`break` loop" implies stopping, while "`continue` loop" implies proceeding to the next iteration.

The model demonstrates strong instructional design and semantic understanding. It retains the pedagogical metaphor while rephrasing it fluently. The bullet-point formatting enhances clarity, though the phrasing includes slight extrapolation.

Overall, the evaluation indicates that few-shot prompting improves pedagogical retention in smaller models like LLaMA 3.1 8B. However, larger models such as LLaMA 3.3 70B exhibit superior abstraction and structuring capabilities even in zero-shot settings. The 70B model not only captures semantic and syntactic fidelity but also reformulates the content in a manner conducive to learning, underscoring the value of large-scale models in educational NLP tasks.

4.3.2 Observation after Entailment. After entailment filtering, we observe that only the few-shot 8B summary fully survives, preserving both the technical explanation and pedagogical framing. Its near-verbatim structure aligns with entailment classifier expectations, showcasing high entailment robustness at the cost of abstraction. The zero-shot 8B summary loses half its content due to lexical compression, illustrating the risk of summarizing with minimal phrasing when facing strict entailment criteria. Conversely, the 70B zero-shot summary, while more abstract and structured, suffers from semantic-paraphrastic filtering, with only the pedagogical preamble retained. This confirms the hypothesis that higher abstraction correlates with greater entailment loss, unless the filtering mechanism accounts for paraphrase semantics.

5 Discussion

The task of summarizing developer discussions on Stack Overflow has evolved significantly, from simple extractive methods to sophisticated hybrid approaches leveraging large language models (LLMs). In this study, HybridSum demonstrates a notable advancement by integrating abstractive capabilities of LLaMA-3.3 70B-Instruct with entailment filtering using RoBERTa-large-MNLI, achieving superior performance in coherence, factuality, and conciseness. To contextualize the performance of our HybridSum system, we compare it with existing Stack Overflow summarization models across key evaluation metrics. Table 1 presents ROUGE-L and BERTScore F1 scores, entailment usage, and evaluation levels for each method. HybridSum (8B, few-shot) achieves the highest ROUGE-L score (0.621), while HybridSum (70B, zero-shot) leads in semantic similarity (BERTScore 0.879). Unlike prior work, our model uniquely integrates NLI-based entailment filtering to reduce hallucination and improve factual alignment. Most baseline systems—such as LASSO and BERTSUM—focus solely on extractive summarization and report ROUGE-style metrics without assessing semantic quality or factual consistency. LaMSUM applies LLMs in a zero-shot setting but does not incorporate entailment or abstraction. This comparison highlights the novel contribution of HybridSum as a hybrid summarization approach that balances fluency, precision, and semantic faithfulness through the integration of LLM generation and entailment validation.

Compared to [13], which utilizes a BERT-based sequence classification pipeline followed by rule-based extractive summarization,

Table 1: Comparison of HybridSum with Prior Models on Stack Overflow Summarization

Model / Paper	Type	ROUGE-L	BERTScore	NLI	Eval.
HybridSum (8B FS)	Hybrid	**0.621**	0.865	✓	Sum.
HybridSum (70B ZS)	Hybrid	0.545	**0.879**	✓	Sum.
HybridSum (8B ZS)	Hybrid	0.523	0.872	✓	Sum.
LaMSUM [2]	Extract.	0.530	—	x	Sent.
LASSO [9]	Extract.	0.546	—	x	Sent.
AnswerBot [14]	Hybrid	0.520	—	x	Sum.
BERTSUM [6]	Extract.	0.511	0.835	x	Sum.
TechSumBot [13]	Hybrid	0.536	—	x	Sum.

***Note:* Abstr. = Abstractive, Extract. = Extractive, ZS=Zero Shot; FS=Few Shot; NLI = Entailment used, Sum. = Summary-level, Sent. = Sentence-level;**

HybridSum benefits from zero-shot generalization and deeper semantic understanding. While TECHSUMBOT relies on manually annotated relevance labels and struggles with redundancy and limited abstraction, HybridSum generates human-like summaries that better capture the intent behind developer queries. Similarly, AnswerBot [12] —which employs LSTM models with attention and lexical similarity scores—focuses on sentence-level scoring. Although effective in identifying key sentences, AnswerBot remains restricted to extractive paradigms and cannot rephrase content or perform reasoning. In contrast, HybridSum leverages LLaMA's generative ability to fuse multiple answer segments into a coherent abstract while RoBERTa ensures entailment-based filtering, reducing hallucination. Moreover, unlike earlier works such as the LAMSUM model [2] , which emphasizes unsupervised graph-based community detection for answer segmentation, HybridSum avoids heavy reliance on graph structures and pre-processing steps, streamlining deployment and generalization across datasets. Notably, prior benchmark studies like [14] outline evaluation inconsistencies, emphasizing the lack of factual correctness measures in early systems. HybridSum directly addresses this through entailment validation, aligning generated content with input facts—a critical requirement for technical domains. Overall, HybridSum offers three core improvements: (1) enhanced abstraction via instruction-tuned LLMs, (2) robust factual consistency through NLI filtering, and (3) reduced reliance on supervised training data or domain-specific rules. However, like all LLM-based methods, HybridSum depends on inference costs and access to powerful computational resources, which may pose adoption challenges for real-time applications.

5.1 Limitations

Despite the effectiveness of the proposed HybridSum pipeline, several limitations should be noted. First, the LLaMA-3.3 70B model imposes substantial computational constraints. Even when deployed with FP8 quantization, it requires approximately 130 GB of GPU memory. Inference time averages around 27 seconds per sample on A100 GPUs, which makes it impractical for latency-sensitive or large-scale deployments. Second, the RoBERTa-large-MNLI model,

used for factuality filtering, is not fine-tuned on technical or code-heavy content. As a result, its entailment predictions may fail to capture domain-specific nuances or overlook critical programming semantics. Additionally, the fixed entailment threshold (set to 0.6) may erroneously filter out relevant content or allow weaker statements to pass. Third, a key limitation in this evaluation is the quality of the ground truth summary itself. In this case, the ground truth was largely extractive and closely mirrored the answer body, offering little abstraction or reformulation. As a result, the entailment model may unfairly penalize summaries that are conceptually accurate but diverge linguistically from the ground truth. This introduces bias toward extractive-style outputs and restricts the expressive potential of abstractive models like LLaMA 70B. Fourth, the evaluation process is constrained by the SoSum dataset itself. The dataset contains only short, reference summaries, which limits the reliability of ROUGE and BLEU metrics. BLEU, in particular, tends to over-penalize legitimate variations in phrasing, which may misrepresent summary quality. Finally, generalization remains an open question. SoSum contains approximately 3,100 answers exclusively from Stack Overflow. It is unclear how well the summarization models would perform on other developer forums or code-centric platforms such as GitHub or Reddit.

6 Conclusion

In this work, we proposed **HybridSum**, a hybrid summarization pipeline for Stack Overflow answers that combines the generative strengths of large language models (LLaMA 3 series) with factual verification via a RoBERTa-based entailment model. Our system first generates abstractive summaries using zero-shot and few-shot prompting with LLaMA 3.1–8B and LLaMA 3.3–70B models, followed by an entailment-based filtering step to ensure faithfulness and eliminate hallucinations. Through both quantitative and qualitative evaluations, we demonstrated that the LLaMA 3.3–70B model consistently outperforms smaller variants across standard summarization metrics including BERTScore, BLEU, and ROUGE. Few-shot prompting was shown to improve the performance of the 8B model, though a significant gap remained between it and the 70B model, highlighting the impact of model scale. Qualitative analysis further confirmed that the generated summaries captured key distinctions in technical content and were aligned with human reasoning. While the approach is effective, we also identified limitations related to model efficiency, domain generalization, and the use of entailment models not tuned for technical language. Future work will focus on reducing inference time through batch optimization, improving entailment accuracy with domain-adaptive training, and extending the pipeline to other domains such as biomedical or legal Q&A. Additionally, building a user-facing interface would support real-time summarization and enhance accessibility for software developers.

Code and Data Availability

The source code for our summarization framework is available at: https://github.com/Nurjahan-Nipa/summarization/tree/main.

References

[1] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, Vol. 33. 1877–1901.

[2] Garima Chhikara, Anurag Sharma, V Gurucharan, Kripabandhu Ghosh, and Abhijnan Chakraborty. 2024. LaMSUM: A Novel Framework for Extractive Summarization of User Generated Content using LLMs. *rXiv* 2406 (2024), v1.

[3] Bonan Kou, Muhao Chen, and Tianyi Zhang. 2023. Automated Summarization of Stack Overflow Posts. In *Proceedings of the 45th International Conference on Software Engineering* (Melbourne, Victoria, Australia) *(ICSE '23)*. IEEE Press, 1853–1865. doi:10.1109/ICSE48619.2023.00158

[4] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*. Association for Computational Linguistics, Barcelona, Spain, 74–81.

[5] Yang Liu. 2019. Fine-tune BERT for extractive summarization. *arXiv preprint arXiv:1903.10318* (2019).

[6] Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345* (2019).

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* (2019).

[8] AmirHossein Naghshzan, Latifa Guerrouj, and Olga Baysal. 2021. Leveraging unsupervised learning to summarize apis discussed in stack overflow. In *2021 IEEE 21st International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 142–152.

[9] Duc-Loc Nguyen et al. 2024. Leveraging LSTM and Pre-trained Model for Effective Summarization of Stack Overflow Posts. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 618–623.

[10] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 311–318.

[11] Hugo Touvron, Théo Lavril, Gautier Izacard, et al. 2024. LLaMA 3: Open Foundation and Instruction Models. *arXiv preprint arXiv:2404.14219* (2024).

[12] Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: Automated generation of answer summary to developers' technical questions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 706–716. doi:10.1109/ASE.2017.8115681

[13] Chengran Yang, Bowen Xu, Jiakun Liu, and David Lo. 2023. Techsumbot: A stack overflow answer summarization tool for technical query. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 132–135.

[14] Chengran Yang, Bowen Xu, Ferdian Thung, Yucen Shi, Ting Zhang, Zhou Yang, Xin Zhou, Jieke Shi, Junda He, DongGyun Han, et al. 2022. Answer summarization for technical queries: Benchmark and new approach. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

[15] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).