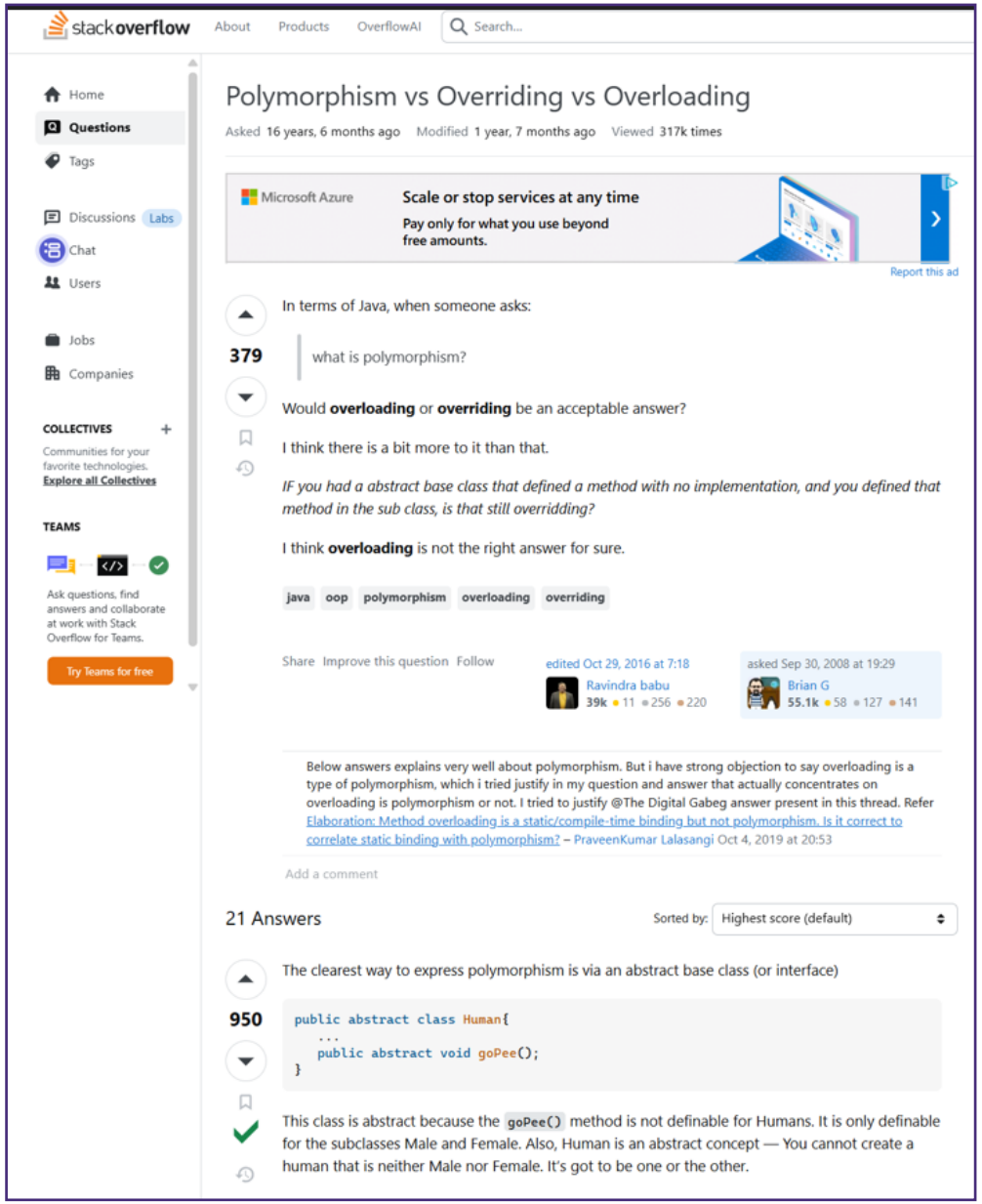
HybridSum: Hybrid Summarization of Stack Overflow Posts

Presented By:

Nurjahan
Kaushani Samarawickrama
Md Saidur Rahman
Division of Computer Science and Engineering, LSU
CSC 7135 — Spring 2025
Date: 21 Apr, 2025

Problem Statement

- **Information Overload**: Average SO answer post contains 300+ words with mixed content (code, explanations, debates).
- **No Domain-Specific Solution**: General NLP models fail to handle SO's unique structure (code blocks, error messages).
- **Time Consuming:** Finding essential information in SO posts can be time-consuming.
- **User Need:** Users demand for tool support to quickly navigate online posts.
- **Example**: *"Difference between Polymorphism vs Overriding vs Overloading?"* → 21 answers (avg. 5min read each ~> 105 mins).



Our Solution

- Hybrid Two-Stage Pipeline:
  - LLaMA3-70B/8B: Generate abstractive draft (captures intent).
  - RoBERTa-NLI: Extract grounded sentences matching the draft (ensures accuracy).

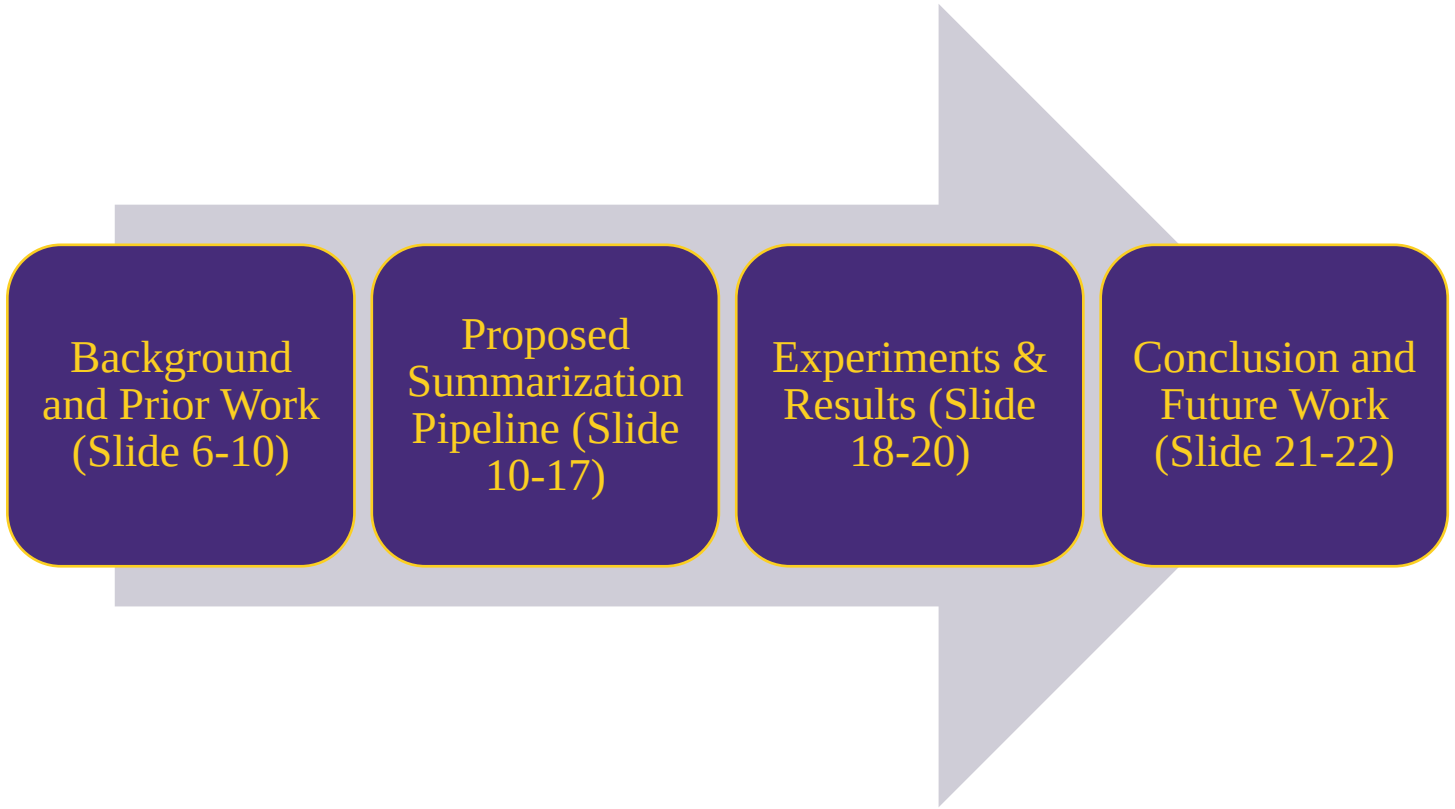
- Quality Trade-offs:
  - *Abstractive Summaries* (LLaMA3) risk technical hallucinations. Several studies have shown that factual inconsistency occurs in up to 30% of abstractive summaries [1].
  - *Extractive Summaries* (BERT) miss key insights by rigid sentence selection.

- Key Innovations:
  - First to combine LLM abstraction with NLI-based extraction for technical Q&A.
  - Resource-Aware: Few-shot LLaMA-8B vs. zero-shot LLaMA-70B trade-off analysis.
  - Evaluated the performance of each model using BERTScore/ Rouge/ Bleu score for comparison.

Motivation and Importance

- **Why This Matters**
  - **Time Savings**: Developers spend 23% of workday reading SO (2023 Stack Overflow survey).
  - **Accuracy Critical**: 68% of developers report being misled by incorrect summaries (ASSORTIS study).
- **Scalable Knowledge Curation**:
  - Processes 100+ posts/min vs. human 5min/post.
  - Adaptable to GitHub Issues, API docs.
- Faithfulness is critical for developer tools (bugs, security, logic).
- Our method ensures factual alignment between source and summary.

Roadmap


Background and Prior Work (Slide 6-10)
Proposed Summarization Pipeline (Slide 10-17)
Experiments & Results (Slide 18-20)
Conclusion and Future Work (Slide 21-22)

LSU College of Engineering

Background

- Summarization Paradigms
  - Abstractive Summarization:
    - Generates new sentences to convey the gist of the source.
    - Challenge in AS is that distorted or fabricated text
  - Extractive Summarization: Selects key sentences directly from the original content. Ensures higher factual consistency.
- Faithfulness and Hallucination
  - LLMs (e.g., LLaMA, GPT) can hallucinate, i.e., generate text not supported by the source.
  - Ensuring faithful summaries is critical for technical domains like Stack Overflow.
- Large Language Models (LLMs)
  - LLaMA 3.1 8B and LLaMA 3.3 70B are the state-of-the-art instruction-tuned LLM.
  - Uses few-shot prompting to generalize from examples without fine-tuning.
- Natural Language Inference (NLI)
  - Task: Given (premise, hypothesis), predict if the hypothesis is: Entailed, Contradicted, Neutral
  - Used RoBERTa-large-MNLI to verify that sentences are entailed by the LLM's output.

LSU College of Engineering

Definition

- **Zero-shot Prompting**
  - A technique where a LLM is instructed to perform a task **without being provided any prior examples** or demonstrations
- **Few-shot Prompting**
  - A technique where the model is given **a few examples** of input-output pairs in the prompt.
  - Used to guide LLaMA 3.1 to generate structured and relevant summaries **without fine-tuning**.
- **Entailment (in NLI)**
  - A hypothesis is entailed by a premise if it must be true given the premise.
  - A sentence from the original post is kept only if it is entailed by the LLaMA-generated summary.
- **Instruction-tuned**
  - A Llama model trained to follow instructions via prompt formatting (e.g., system + user roles).
  - LLaMA 3.1 Instruct responds better to system-level cues like “You are an expert summarizer”
- **Evaluation Metrics:**
  - ROUGE: Measures n-gram overlap between generated and reference summaries.
  - BERTScore: Uses contextual embeddings to compare similarity between generated and reference texts.
  - BLEU: Measures n-gram precision (used in translation and summarization).

LSU College of Engineering

Related Works

Paper	Model	Performance Metrics
Kou et al. [1]	ASSORTs / ASSORTis: Supervised and indirect-supervised models using BERT + domain-specific features	Precision: **0.73**, Recall: **0.69**, F1: **0.71** (SOSum dataset)
Liu et al. [2]	**BERTSUM**: Document-level BERT encoder with inter-sentence transformer layers, for both extractive and abstractive summarization	ROUGE-1: **43.25**, ROUGE-2: **20.24**, ROUGE-L: **39.63** (CNN/DailyMail dataset)
Nguyen et al. [3]	**LASSO**: BERT + Bi-LSTM capturing contextual salience and query relevance; introduced SOSum+ dataset	Precision: **0.7050**, Recall: **0.6750**, F1: **0.6897**, Accuracy: **0.7597** (SOSum+)
Xu et al. [4]	**AnswerBot**: Three-stage pipeline — relevant question retrieval, paragraph selection, summary generation	Human study: Relevance, Usefulness, Diversity scores (no exact values reported)
Yang et al. [5]	**TechSumBot**: Query-focused summarization with Usefulness Ranking, Centrality Estimation, Redundancy Removal	ROUGE-1 ↑ **+10.8%**, ROUGE-2 ↑ **+36.6%**, ROUGE-L ↑ **+17.5%** over best baseline

LSU College of Engineering

Research Gap

While prior works (e.g., ASSORTs, LASSO, TechSumBot) have explored BERT-based summarization using handcrafted features, Bi-LSTM, or unsupervised ranking:

- They **do not utilize instruction-tuned LLMs** like LLaMA for abstractive summarization.
- They **lack mechanisms to ensure factual consistency**, such as **entailment filtering**.
- Most approaches focus on extractive or pipeline-based methods with **limited abstraction**.
- **No comparison across model scales** (8B vs. 70B) or efficient deployment strategies (e.g., FP8 with vLLM) is provided.

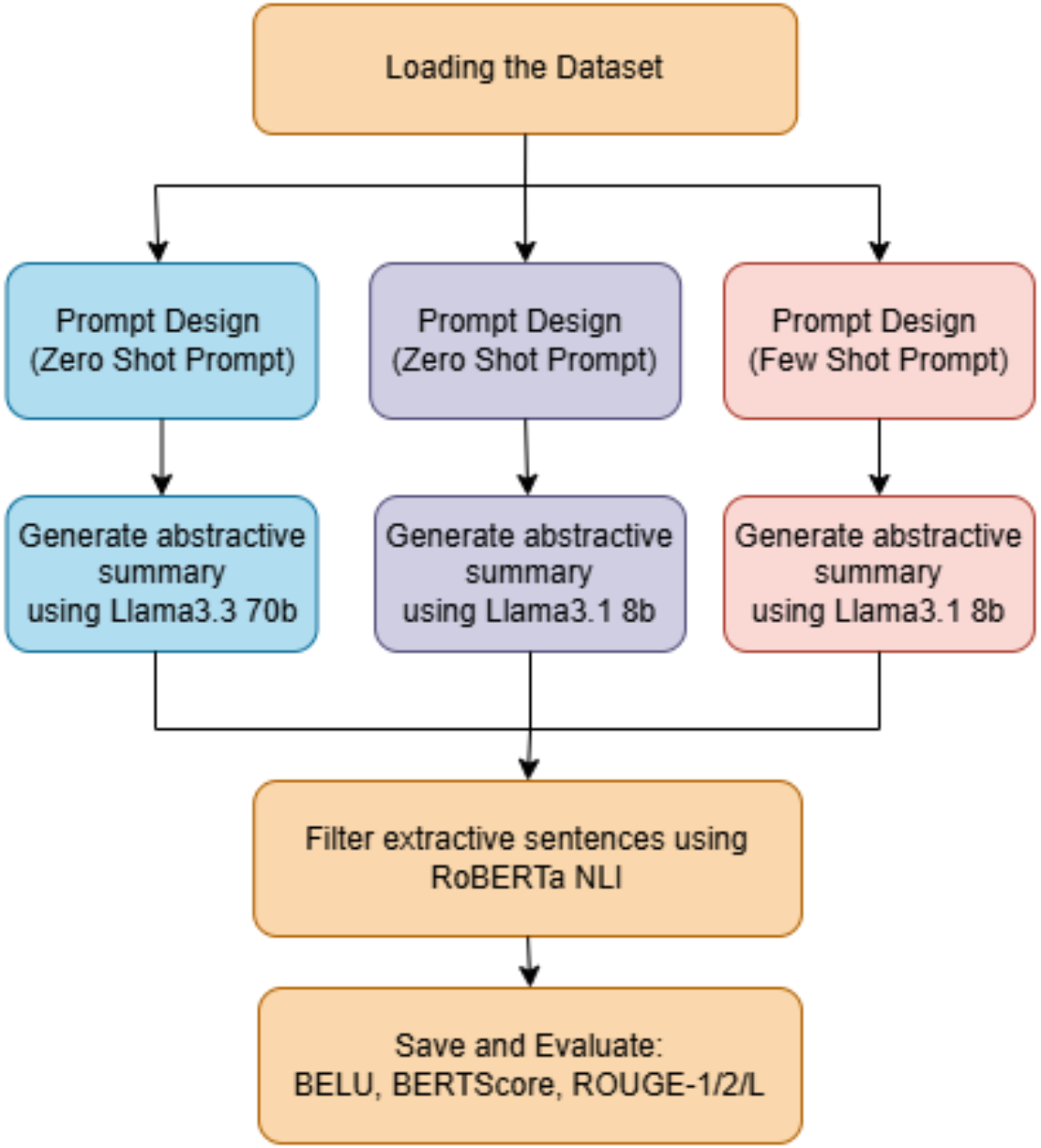
LSU College of Engineering

Research Question

- RQ1: Which model performs best among the two models (Llama3.1 8b or Llama 3.3 70b) evaluated for post summarization?
- RQ2: Can we achieve similar performance using smaller models compared to larger models like LLaMA-3.3-70B, and if so, how can this be accomplished?

LSU College of Engineering

Main Workflow



LSU College of Engineering

Dataset Description

- 3130 Stack Overflow posts: 785 SO questions,
  - 254 *how-to* questions,
  - 322 *conceptual* questions
  - 209 *bug-fixing* questions
- **Source**: SoSum (Stack Overflow Summarization)

Column Name	Description
answer_id	Unique identifier for each answer
answer_body	Full answer text written by a user
question_id	Identifier linking to the corresponding question
question_type	Type of question (e.g., conceptual, implementation, etc.)
question_tags	List of relevant tags (e.g., ['python', 'list'])
question_title	Short title or headline of the question
sentences	List of dictionaries with sentence and optional truth label

{ "answer_id": 12345,
"answer_body": "This is an example answer body.",
"question_id": 67890,
"question_type": 1,
"question_tags": ["Java"],
"question_title": "How to use JSON in Python?",
"sentences": [{
"sentence": "You can use the json module to parse JSON data.",
"truth": 1},
{
"sentence": "It's part of the standard library.",
"truth": 0 }]}

LSU College of Engineering

Prompt Design: Zero Shot (70 B, 8B)

```
sampling_params = SamplingParams(
    temperature=0.6,
     top_p=0.9,
     max_tokens=128 (70B) /256 (8B) )
```

```
messages = [
    {
        "role": "system",
        "content": "You are an expert technical summarizer. Your task is to generate a brief and accurate summary of the provided Stack Overflow answer, based on the question context." }]
```

```
llm = LLM(
        model=llm_model_id,
        tensor_parallel_size=2,
        max_model_len=2048,
        gpu_memory_utilization=0.85,
        quantization="fp8” )
llm_model_id = "meta-llama/Llama-3.3-70B-Instruct” /
llm_model_id = "meta-llama/Llama-3.1-8B-Instruct"
```

Prompt Design: Few Shot (1/2/3)

```
messages = [
    {
        "role": "system",
        "content": "You are an expert technical summarizer. Your task is to generate a brief and accurate summary of the provided Stack Overflow answer, based on the question context and few-shot examples."    },
  {       "role": "user",
          "content": few_shot_prompt}
]
```

few_shot_examples = """### *Example 1*

Question: Why use def main()?

Answer: "What does <code>if __name__=='__main__'</code> mean?" and "why do people use <code>def main()</code>?" The usual answer is: to prevent code from being executed when the module is imported. But more importantly, using `def main()` allows better organization, reusability, and testability of your script.

Summary: Using `def main()` allows the function's functionality to be reused, tested, and avoids execution on import.

- *Example 2*
- *Example 3*

"""

LSU College of Engineering

Entailment Filtering Logic

- Input:
  - Abstractive Summary ← Generated by LLaMA
  - Candidate Sentences ← From original Stack Overflow answer
- Check Entailment: For each sentence S in the original answer:
  - Use RoBERTa-large-MNLI to check if the LLaMA summary entails S.
- Threshold:
  - Keep sentence S only if entailment probability > 0.6

- Example
  - Summary: "def main() helps structure Python code.“
  - Sentence: "Using main() makes code modular." -> EntailedSentence:
  - "Python was invented in 1989.“ -> Not entailed
- **Why Use Entailment?**
  - Filters **hallucinations** in LLM outputs
  - Ensures each sentence in the extractive summary is **grounded in the source**

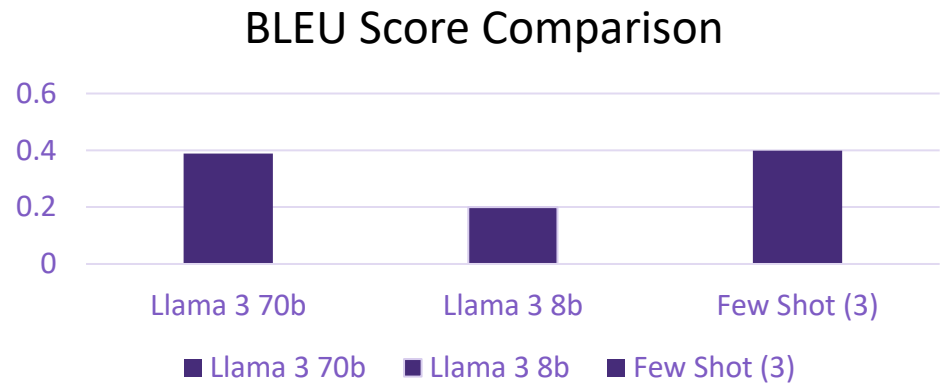
Experimental Results: Setup

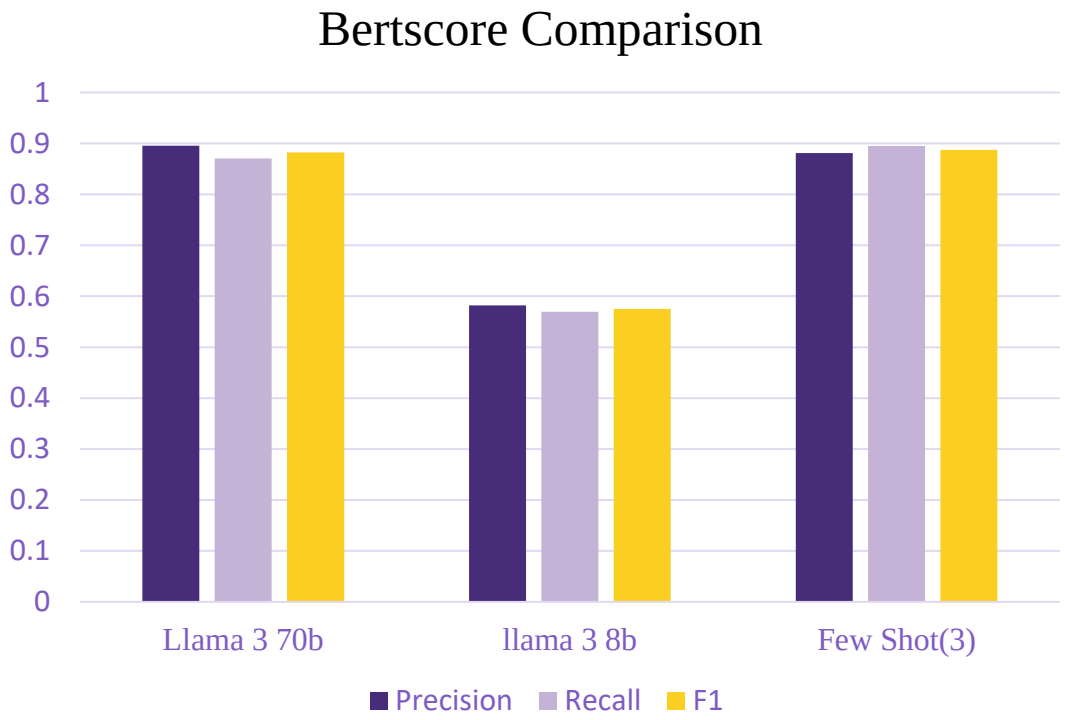
- **Environment**
  - **System**: LONI HPC (QBD2)
  - **GPUs**: 2× A100 80GB
  - **CPU**: 4 cores
  - **Runtime**: SLURM batch or interactive shell
  - **Libraries**:
    vLLM (v0.8.2), transformers, torch, rouge_score, bert_score, nltk

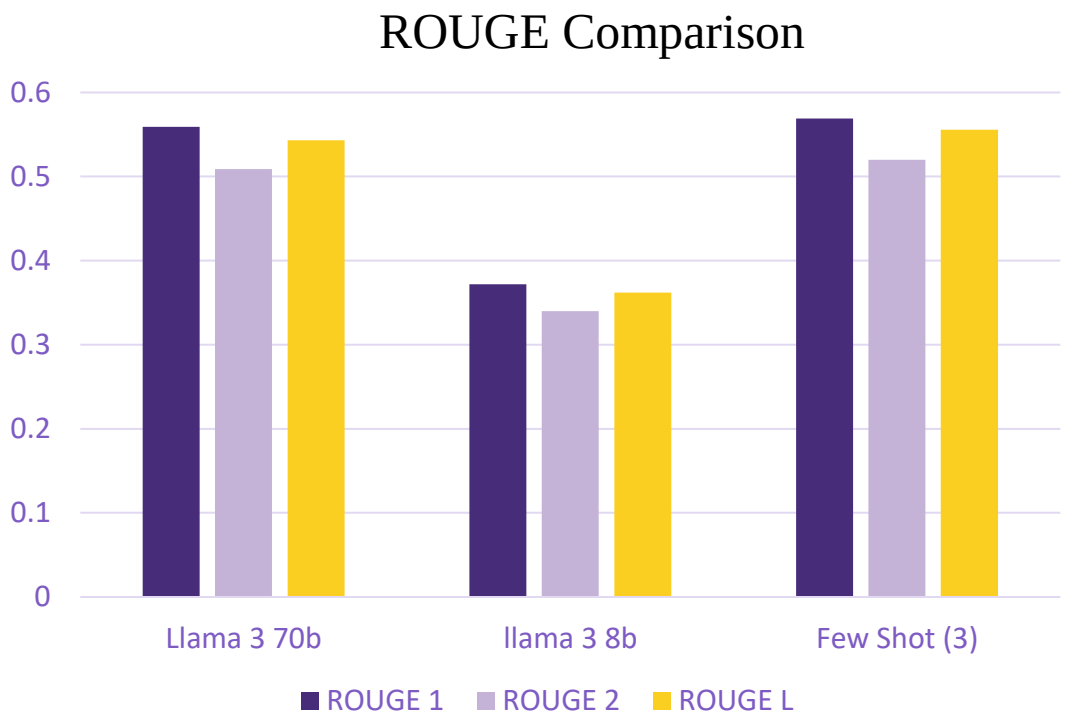
- **Sampling Settings: Llama**
    - max_tokens = 256 (8b), 128 (70b)
    - temperature = 0.6
    - tensor_parallel_size = 2
    - quantization = fp8

- **Resource Used**
  - Total GPU memory used ≈ 67.9 GB × 2 = 135.8 GB across 2 A100 GPUs. (70b)
  - GPU memory used: nearly ~170 GB (8B)
- **Runtime statistics**
  - Total inference time: 1h 56m 28s
  - Total wall-clock time of 1 h19 m 17 s (8b)
  - Total wall-clock time of 2 hours, 1 minute, and 37 seconds (few shot)

Experimental Results: Performance Metric







Qualitative Examples

- Question
  - What is the difference between visibility:hidden and display:none?
- Original Answer
  - <code>display:none</code> means that the tag in question will not appear on the page at all (although you can still interact with it through the dom). There will be no space allocated for it between the other tags. <code>visibility:hidden</code> means that unlike <code>display:none</code>, the tag is not visible, but space is allocated for it on the page. The tag is rendered, it just isn't seen on the page. For example: Replacing <code>[style-tag-value]</code> with <code>display:none</code> results in: Replacing <code>[style-tag-value]</code> with <code>visibility:hidden</code> results in:
- Abstractive Summary (LLaMA 70B)
  - The key difference between `visibility:hidden` and `display:none` is that `visibility:hidden` hides an element while preserving its space in the layout, whereas `display:none` removes the element from the layout altogether.
- Abstractive Summary (LLaMA 8B)
  - The difference between `visibility:hidden` and `display:none` lies in how they affect an element's presence on the page. `display:none` removes the element from the page layout, while `visibility:hidden` hides the element from view but still reserves space for it on the page.
- Extractive Summary (via RoBERTa entailment)
  - <code>display:none</code> removes the element from the layout flow. <code>visibility:hidden</code> hides it but leaves the space.
- Ground Truth
  - <code>display:none</code> removes the element from the layout flow. <code>visibility:hidden</code> hides it but leaves the space.

LSU College of Engineering

Conclusion

- **Summary of Contributions**
  - Proposed a hybrid summarization pipeline combining:
    - Abstractive summaries from LLaMA 3.3 70B
    - Extractive filtering using RoBERTa-large-MNLI entailment
  - Used few-shot prompting and instruction-tuned LLMs for high-quality generation
  - Ensured faithfulness by filtering hallucinations via NLI
  - Evaluated using ROUGE, BERTScore, and BLEU across ~200 Stack Overflow questions

- **Key Results**
  - LLaMA 70B significantly outperformed 8B in summary quality (e.g., BERTScore F1: 0.88 vs. 0.56)
  - Get similar performance by 8b using few shot prompting.
  - Entailment filtering eliminated unsupported claims and increased factual precision
  - Achieved high memory efficiency using FP8 quantization in vLLM

- Code & Dataset:
  - https://github.com/Nurjahan-Nipa/summarization

Limitation and Future Work

- **Model Constraints**:LLaMA-3.3 70B requires ~130GB GPU memory, even with FP8. Inference still takes ~27 seconds per sample on A100s.
- **Faithfulness Filtering:** RoBERTa-large-MNLI isn't fine-tuned for technical content, so entailment judgments may miss nuanced or code-heavy relevance.
- Thresholding is binary (>0.6), which may filter out valid but concise responses or keep marginal ones.
- **Evaluation Metrics:** ROUGE/BLEU rely on limited ground truth from the SoSum dataset. BLEU in particular penalizes stylistic variation.
- **Dataset Generalization:** SoSum contains only ~3K Stack Overflow answers. Generalization to other domains (e.g., GitHub, Reddit) is untested.

- **Dynamic Entailment**:
  - Use confidence-aware or adaptive thresholds based on sentence length or domain.
- **Fine-Tuning & Adaptation**:
  - Fine-tune NLI on Stack Overflow entailment examples or train LLaMA on Q&A-style data.
- **Cross-Domain Expansion**:
  - Apply the method to domains like biomedical QA (*e.g.*, *PubMedQA*), legal summaries, or multilingual technical forums.
- **Real time UI**
  - **Interactive Interface**:
    Build a real-time UI in **Streamlit or Gradio** to upload threads, view LLM summaries, and justify entailment-based extractions.

LSU College of Engineering

References

1. Bonan Kou, Muhao Chen, and Tianyi Zhang. 2023. Automated Summariza-tion of Stack Overflow Posts. In Proceedings of the 45th International Conferenceon Software Engineering (Melbourne, Victoria, Australia) (ICSE ’23). IEEE Press,1853–1865. doi:10.1109/ICSE48619.2023.00158
2. Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders.arXiv preprint arXiv:1908.08345 (2019).
3. Duc-Loc Nguyen et al . 2024. Leveraging LSTM and Pre-trained Model for EffectiveSummarization of Stack Overflow Posts. In 2024 IEEE International Conference onSoftware Maintenance and Evolution (ICSME). IEEE, 618–623.
4. Bowen Xu, Zhenchang Xing, Xin Xia, and David Lo. 2017. AnswerBot: Automatedgeneration of answer summary to developers’ technical questions. In 2017 32ndIEEE/ACM International Conference on Automated Software Engineering (ASE).706–716. doi:10.1109/ASE.2017.8115681
5. Chengran Yang, Bowen Xu, Ferdian Thung, Yucen Shi, Ting Zhang, Zhou Yang,Xin Zhou, Jieke Shi, Junda He, DongGyun Han, et al. 2022. Answer summarizationfor technical queries: Benchmark and new approach. In Proceedings of the 37thIEEE/ACM International Conference on Automated Software Engineering. 1–13.

LSU | College of Engineering

Thank You!
Any Suggestions?