

*LLM-Based Semantic Retrieval and
Synthetic Response Generation for
Longitudinal Survey Data*

Albert Weerman, Chirag Patel

Paper No: 2026-001

**CESR-SCHAEFFER
WORKING PAPER SERIES**

The Working Papers in this series have not undergone peer review or been edited by USC. The series is intended to make results of CESR and Schaeffer Center research widely available, in preliminary form, to encourage discussion and input from the research community before publication in a formal, peer-reviewed journal. CESR-Schaeffer working papers can be cited without permission of the author so long as the source is clearly referred to as a CESR-Schaeffer working paper.

LLM-Based Semantic Retrieval and Synthetic Response Generation for Longitudinal Survey Data

Albert Weerman¹ and Chirag Patel¹

¹Center for Social and Economic Research, University of Southern California

July 2025

Abstract

Background: The Understanding America Study (UAS) has accumulated over 700 surveys across 11 years, offering a rich but complex resource for researchers. Accessing relevant information can be challenging without deep familiarity. This paper introduces an AI system that uses natural language processing to match user questions with past survey items and generate data-driven synthetic responses, making longitudinal data more accessible, interactive, and scalable for research.

Methods: The system uses natural language processing to interpret user-posed survey questions and identify semantically similar items within the Understanding America Study database. Once matched, it aggregates historical responses and generates a synthetic response based on actual survey data. The output includes the generated response, sampling weights, and metadata in a downloadable format, allowing integration with respondent-level longitudinal data for further analysis. In this AI system, we use multiple large language models (LLMs) and provide a comparative analysis of the models' effectiveness in leveraging the UAS survey data for generating responses.

Results: We are evaluating the AI system on performance and system quality metrics using the LLM-as-a-judge evaluation strategy. From the results of the evaluation, we observed the system accurately matched user questions to relevant survey items and produced synthetic responses consistent with historical data patterns, enabling seamless integration with respondent-level analyses.

Conclusion: This approach demonstrates how AI can enhance access to complex longitudinal survey data, enabling faster, more intuitive exploration and supporting scalable, respondent-level research across time. Smaller, more optimized models like Gemma can deliver both efficiency and reliability in RAG workflows, offering a practical balance of speed, quality, and cost-effectiveness.

1 Introduction

Accessing relevant information from large-scale longitudinal survey data can be a significant challenge for researchers, particularly when datasets span hundreds of surveys and thousands of variables collected over many years. The Understanding America Study (UAS), with over 700 surveys conducted across more than a decade, is one such valuable but complex resource. To make this data more accessible and usable, we present an AI-driven system that allows users to pose natural language questions and receive synthetic responses based on aggregated historical data. By leveraging advanced natural language processing, the system identifies the most relevant survey items, generates statistically grounded answers, and provides downloadable data outputs linked to respondent profiles. This tool offers a scalable, user-friendly interface for exploring longitudinal data, supporting both hypothesis generation and integration with ongoing research [1].

2 Dataset

We are using the dataset from the UAS study, specifically all the categorical questions that have ever been asked in the panel. The UAS is a panel of households at the University of Southern California (USC) of approximately 14,700 respondents, growing to 20,000 by end of 2025 representing the entire United States. Participants use computers, tablets and/or smartphones to anonymously answer UAS surveys on a wide range of topics. These include everything from current events, such as elections or disaster response, to health, aging and finances. The UAS is representative of the U.S. population since panel members are recruited using Address-Based Sampling methods, so that everyone with a postal address has a chance of being invited to join the panel. Tablets are provided as needed to enable those without access to participate [2].

3 Methodology

In our approach, we adapt the Retrieval-Augmented Generation (RAG) framework to interpret user queries, retrieve the most semantically relevant historical survey questions, and generate context-aware responses. RAG is a hybrid methodology that combines the strengths of information retrieval and language generation. Rather than relying solely on a pre-trained language model, RAG supplements generation with external knowledge sources, ensuring responses are both contextually rich and grounded in factual information. At its core, RAG operates in two stages. First, it uses embeddings to represent queries and documents, enabling efficient retrieval of relevant pieces of information. Second, the retrieved content is passed into a generative model, which synthesizes a coherent response tailored to user’s query.

The decision to use the RAG framework was driven by both practical and performance-related considerations. Fine-tuning requires substantial amounts

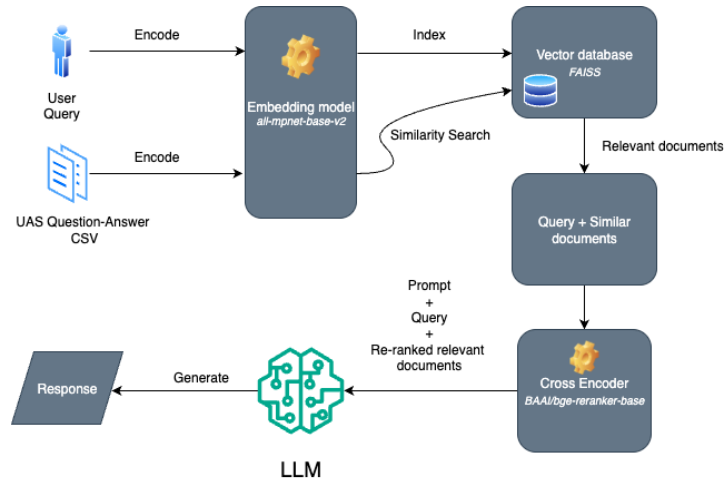


Figure 1: RAG System

of task-specific data and computational resources to adjust the parameters of a large pre-trained model. This process is both costly and inflexible: any change in the knowledge base such as newly available information requires repeated retraining, which is impractical at large scale. Similarly, full model retraining is even more resource-intensive, demanding significant hardware. Therefore, RAG offers a more scalable and flexible solution for tasks where knowledge evolves, computational budgets are constrained, and response accountability is essential.

3.1 Data Pre-processing

In order to prepare the corpus of data that will act as an external data source for the RAG system, we have two SQL tables with the UAS survey data and responses. The first SQL table includes the question id, question name and survey name along with the response options for that particular question. The second table includes the responses received for each question for a given survey selected by the respondent. Using this data, we compute the total number of responses for each question for a given survey and then calculate the count for each answer options which is represented by a single row in the second table. Using this information, we prepared a CSV file that included one row for each unique combination of the question id and the survey id. The file also includes a column with response options and the corresponding percentage indicating how many respondents selected that option as the answer, and another column indicating the total number of responses received for that particular question. This CSV is our question-answer dataset which is fed to the AI (RAG) system to generate responses.

3.2 Semantic Embedding Generation and Vector Index Construction

The input dataset consists of question-answer (QA) pairs created by consolidating information from previous surveys. For semantic search, each QA pair is encoded into a high-dimensional vector using the Sentence-Transformer model (all-mpnet-base-V2). This embedding model was chosen for its strong performance on semantic similarity tasks. The resulting embeddings were stored in a FAISS (Facebook AI Similarity Search) vector index to support efficient retrieval and re-ranking.

3.3 Retrieval and Filtering Strategy

When a user question is submitted, it is first embedded using the same Sentence-Transformer and then to improve retrieval quality and reduce semantic noise, we applied a distance-based threshold to the top - k results retrieved via FAISS. This step ensures that only query-document pairs with sufficiently high semantic alignment are considered. All results with a score less than 1.0 were retained for re-ranking. This threshold reduces semantically distant matches. Although dense embeddings effectively capture broad semantic relationships, these scores are purely based on vector distance, not on deep semantic meaning or context. They can miss subtle lexical distinctions critical for accurate question matching. To address this limitation, a re-ranking mechanism was incorporated to refine the initial set of retrieved candidates. In this stage, pairwise similarity scores between user query and each retrieved question are computed using a cross-encoder model. We have used BAAI/bge-reranker-base as cross-encoder model. This cross-encoder leverages attention mechanisms to jointly encode the query and candidate, allowing it to assess both the semantic relevance and lexical similarity including phrase level similarity. Out all the candidates, at most five candidates with highest re-ranker score are selected. To further improve the effectiveness of retrieved context, we addressed the "Lost in the Middle" issue - where language models tends to under-attend to information in the middle of long inputs by integrating a LongContextReader component [3]. This module reorganizes the most relevant portions of the context before it is passed to the language model, ensuring information is not obscured due to positional biases. This step was applied post re-ranking and prior to generation, improving consistency and accuracy of model outputs.

3.4 Response Generation via Augmented Prompting

Once retrieval is complete, the top-ranked results are incorporated into a structured prompt along with the user query. This step is a bridge between retrieval and generation, allowing the language model to generate the output based on the evidence.

3.4.1 Augmentation Strategy

To enhance generation quality and to make sure the output is grounded to retrieved information, the input to LLM is constructed using a standardized prompt template that consists of:

- Context: Retrieved QA pairs
- Question: Original user query
- Instructions: Clear task-specific instructions that describe the context, question, model's role, and expected output.
- Output Format: A specification of the desired output structure, in this case JSON format
- Example: An example comprising of user question and expected output format to guide the models behavior.

This enhanced prompt structure enables the model to better understand the requirements of the task, improve the faithfulness to the context, and adhere to the output constraints. A strict and consistent output format helps to smoother post-processing and evaluation.

3.4.2 Model Integration and Generation setup

The augmented prompt is subsequently provided to a large language model (LLM) for response generation. This process is orchestrated through the LangChain framework, which integrates document retrieval, prompt construction, and model inference within a unified pipeline. Due to its modular design, LangChain enables the creation of custom prompt templates with clearly delineated sections, as outlined above. At runtime, all prompt components are dynamically embedded, thereby supporting flexible and adaptive query handling. In addition, LangChain facilitates seamless integration with locally hosted LLMs via Ollama, enabling consistent execution across different models and simplifying comparative evaluation.

For open-source models, Ollama serves as the runtime environment, providing a lightweight and efficient infrastructure for deploying LLMs on both GPU and CPU hardware. The decision to employ Ollama is motivated by several advantages over cloud-based or subscription-dependent API services. Most importantly, it offers cost savings, since open source models can be executed offline without incurring usage fees. It also ensures greater data privacy and control, since inference is performed locally and sensitive information never leaves the host environment, thereby aligning with institutional privacy and compliance requirements. In addition, local execution can reduce latency as it removes network overhead, which is particularly beneficial during iterative development involving prompt engineering and pipeline refinement. Finally, reproducibility and version stability are enhanced, given that local model instances remain fixed over time, unlike cloud APIs which may be updated without notice. Taken

together, these characteristics position Ollama as a robust, privacy-preserving, and developer-friendly solution for the present system.

3.5 Models

In the described RAG pipeline, we are using open-source LLMs to generate responses. We have used the following open-source models:

- Llama 3.1 (8B) - Meta
- Gemma 3 (4B) - Google
- Qwen 3 (8B) - Alibaba Cloud
- Mistral (7B) - Mistral AI

The choice to employ lightweight open-source LLMs is primarily motivated by their optimization for local deployment through Ollama. Such models are well-suited for execution on personal or edge devices, where a lightweight framework facilitates both system management and operational efficiency. This configuration gives researchers greater control over model behavior, eliminates dependence on external API services, and ensures enhanced data privacy while simultaneously reducing costs.

4 Test Dataset

To evaluate and compare the performance of the four LLMs within the RAG system, a manually curated set of 20 test queries was constructed (Table 1 on page 7). Each query was designed such that the information required to generate an appropriate response was fully contained within the external knowledge corpus of the system, namely the UAS survey data. This design ensured that all queries were answerable without recourse to parametric memory of the models, thus isolating performance within the retrieval-augmented framework.

5 Performance Evaluation

In order to evaluate the complete pipeline, we have prepared a test dataset of 20 questions which are randomly selected from the UAS survey data and rephrased to ensure the exact question is not being used to test the system. Rephrasing the questions also enables us to simulate real world use-case where in the users wont be searching for the exact survey question. The dataset includes question and ground-truth i.e. Expected output. To better understand the performance of the AI system, we conducted a comparative evaluation by swapping out the language model used during the generation stage. The goal is to assess how different small open source LLMs influence the quality, relevance, and accuracy of the final output when provided with the same retrieved context.

ID	Query
Q01	Are you worried about losing your insurance?
Q02	Are you worried of Monkeypox?
Q03	How many people have digestive problems?
Q04	Do you have any issues in taking medicines because of memory or health related issues?
Q05	Do you talk to family or friends about your finances?
Q06	Have you looked into how to think critically?
Q07	How is your mental and emotional health?
Q08	How many alcoholic drinks do you have each week?
Q09	How many cards do you have?
Q10	How many checking accounts are in your name?
Q11	How many debit cards are in your name?
Q12	How much time spent on Social Media?
Q13	How often are you agitated?
Q14	How often are you irritated?
Q15	How often did your family discuss financial matters around you?
Q16	How often do you travel by airplane?
Q17	How satisfied were you with your job you in the past?
Q18	If new housing units are built in the neighborhood, how would your rent change?
Q19	What are people’s opinions on wearing masks?
Q20	What is helpdesk rating?

Table 1: Test queries used for evaluating the RAG system.

To comprehensively evaluate the performance of the system across different models, we use diverse set of metrics that asses system quality and performance. These metrics not only enable us to benchmark which models perform best, but also help us understand the behavior of different components in our RAG system, such as the retriever, generator and their interaction. In addition to testing the performance of components, to ensure the system performs well in practical deployments, it is important to include time based performance metrics like latency, Time Taken to Generate First Token and Tokens/Sec.

5.1 Latency-Related Metrics

These metrics capture how quickly the AI system generated response which is very critical in real-world applications.

5.1.1 System Latency

This measures the total time taken by the system to generate the response from the time it received the user query. This encompasses the entire pipeline, including query processing, document retrieval, and LLM generation. Lower latency

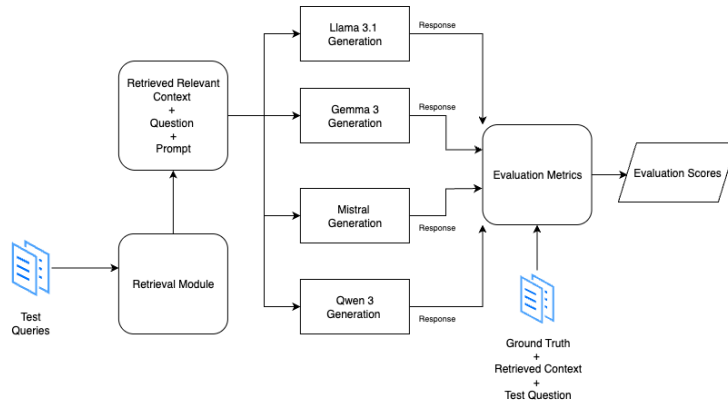


Figure 2: Evaluation Framework

is preferred, but there is often trade-off between latency and answer quality, as more complex retrieval or larger LLMs may take more time to generate higher-quality responses.

5.1.2 Time To First Token (TTFT)

This measures the time taken by the LLM to output the first token in the response which is crucial if the system is streaming the output. TTFT focuses specifically on the model’s initial responsiveness, unlike system latency, which measures the time taken by the complete pipeline.

5.1.3 Tokens Generated Per Second

This metric is used for evaluating generation speed, to track how fast a model produces its output. This is useful when comparing different models on their generation efficiency. It is calculated as the number of tokens generated divided by the generation time excluding retrieval. This throughput reflects the efficiency of the LLM during text generation.

5.2 System-Level Metrics

5.2.1 Context Recall

To evaluate how well the retrieved documents support answer generation, we use context recall. Measures whether the retrieved documents actually contain the information necessary to answer the user’s query. Higher recall indicates that the retrieval component successfully retrieves supporting knowledge for downstream generation.

$$ContextRecall = \frac{\text{Number of documents in the answer supported by the retrieved context}}{\text{Total number of documents in the answer}}$$

5.2.2 Faithfulness

Faithfulness measures how accurately the generated answer reflects the information included in the retrieved context. A faithful answer should not hallucinate facts or introduce new and unsupported claims beyond what is present in the retrieved context. If faithfulness is high, it indicates that the LLM is not hallucinating or generating a response that is not supported by the context provided. This metric is important for our RAG system because an unfaithful LLM response can mislead users even though the retrieved context is accurate.

5.2.3 Answer Similarity

Measures how similar the generated answer is to the ground truth (reference answer). This metric uses embeddings to get vector representations of both the reference answer and the answer generated by the LLM and then calculates the cosine similarity between these vectors to see how similar they are. By comparing embeddings of generated answers and ground-truth answers, this metric captures whether the model produces answers that are meaningfully aligned with the expected response. High similarity indicates that the system's answer conveys the intended meaning of the ground-truth answer.

5.2.4 Answer Relevancy

This metric is used to assess how relevant the generated answer is to the user's question. The purpose of this paper is mainly to evaluate whether the generated response appropriately answers the question asked without digressing from the key requirements of the question. There could be a scenario where the response generated by the LLM includes keywords that are present in the actual response, however, it might not be relevant to the question asked by the user, in such situation, answer relevancy becomes a critical metric in evaluation the performance of the system as a whole.

5.2.5 Answer Correctness

This measures whether the content of the generated answer is factually correct. This particular metric evaluates is different from consistency as it not only evaluates the closeness in meaning or wording of generated answer to reference answer but also evaluates if the generated answer is correct with respect to the questions and reference answer. This metric measures that the system provides useful and accurate information to the user.

5.3 Evaluation Strategy

In order to evaluate the performance of our RAG system across different LLMs, we are using an LLM-as-a-judge strategy. This strategy involves using an LLM to not only generate answers, but also to grade and assess the quality of those answers across various dimensions mentioned above. We are using two different

approaches in evaluation - using our own custom evaluation prompt with the 4 large language models used in the comparative analyses and another approach is using an already available library like RAGAS to evaluate the system based on the prompts in RAGAS itself. From both the approaches we see that custom evaluation using our own prompt helps the model better evaluate the nuances of the task and give results which more accurately reflect the results generated by the system. The test dataset which includes 15 queries along with the ground-truth answers is used as the input to our evaluation. For each query in our test dataset, the retriever component of the RAG system outputs the relevant documents (question-answer) pairs, the relevant documents along with the query are passed to the generator LLM which in our case would be four different LLMs that we are comparing. The response generated by LLM forms the final component that is needed for evaluation. So, the test query, context retrieved by the retriever, ground-truth answer and generated answer completes our test data for the evaluation system.

5.3.1 LLM-as-a-judge using custom prompt

In this approach, the evaluator LLM is provided with a structure prompt that include:

- Question
- Retrieved Context
- Generated Answer
- Reference Answer

The prompt is carefully designed to mimic human judgment and provide a consistent score across the test data set. Along with the above data, we also define the following:

- Rubrics that includes granularity of scoring scale i.e (0.0-5.0)
- Instructions and definitions of each metric that it will evaluate the data on.
- If needed, examples with explanation.

This custom judge setup enables us to fine-tune the evaluation based on the domain and handle cases where a general-purpose evaluator would miss details. We are evaluating each LLM's response against the 4 LLMs including itself. Next, for a given LLM system under test, we average the scores for each metric evaluated by the 4 judge LLMs to compare the performance of the LLMs in the RAG system.

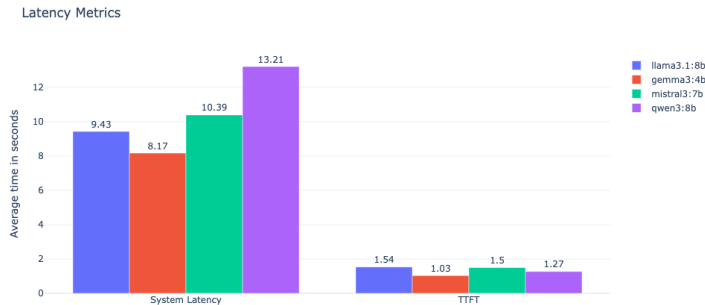


Figure 3: Performance Results

5.3.2 LLM-as-a-judge using Ragas

In this approach, we evaluate each LLM with a fixed evaluator LLM i.e llama3.1:8b using Ragas which is a library used to evaluate LLM based applications. For evaluation of each LLM, we pass the below data to Ragas evaluation function along with a list of metrics on which we want to evaluate the LLM.

- Question
- Retrieved Context
- Generated Answer
- Reference Answer

Along with this data, we also pass llama3.1:8b as the evaluator LLM, the reasoning behind using llama3.1 as the evaluator LLM is mainly because it pairs strong instruction following with long-context support ideal for rubric-driven evaluation [4]. For each query in the test dataset we evaluate the RAG system with each LLM using Ragas own system prompts and the evaluator LLM. For each metric, Ragas outputs a score between 0-1.

6 Results

We conducted a comparative evaluation of four mentioned LLMS within the RAG framework, focusing on three core latency metrics: system latency, time-to-first-token (TTFT), and tokens per second. The results of the evaluation are captured in Figures 3 and 4.

6.1 System Latency

In terms of end-to-end system latency, four models ranked as follows:

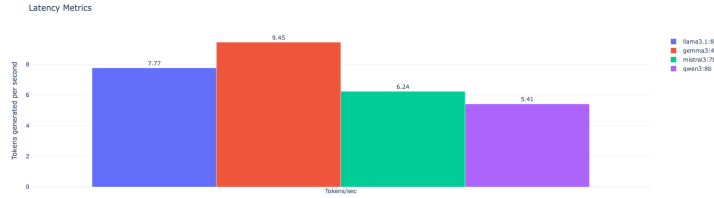


Figure 4: Performance Results

1. Gemma 3 4B
2. Llama 3.1 8B
3. Mistral 7B
4. Qwen 3 8B

Gemma achieved the lowest overall latency, attributed to its smaller parameter size (4B) and efficient inference pipeline, which minimize computational overhead. Llama followed, demonstrating a balance between model size and optimization, while Mistral performed moderately. Qwen exhibited the highest latency, reflecting heavier computational demands and slower generation throughput.

6.2 Time To First Token (TTFT)

For TTFT, which measures how quickly a models begins generating, the rankings are as follows:

1. Gemma 3 4B
2. Qwen 3 8B
3. Mistral 7B
4. Llama 3.1 8B

Here, Gemma again led with the minimal startup delay, underscoring its responsiveness. Qwen outperformed Llama and Mistral. However, this result requires careful interpretation as this speed is an artifact of Qwen’s default output behavior. The model produces `¡think¡/think¡` tag before beginning its actual response, since TTFT is measured as the time until first token is produced, the emission of this tag is registered as an early response. Llama, in contrast exhibited the slowest TTFT, likely due to heavier model initialization and context processing requirements.

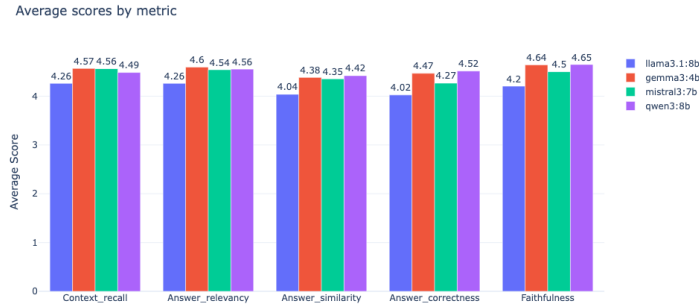


Figure 5: Evaluation Results - Custom LLM-as-a-judge

6.3 Tokens Per Second

When analyzing throughput, measured in tokens per second, the models ranked as:

1. Gemma 3 4B
2. Llama 3.1 8B
3. Mistral 7B
4. Qwen 3 8B

Gemma again dominated, delivering the highest token generation rate. Llama ranked second, maintaining solid throughput once generation began, while mistral lagged slightly behind. Qwen performed the worst, generating tokens at the slowest rate and compounding it high system latency. For generation quality evaluation, we also conducted a comparative evaluation of four large language models in the generation phase of our RAG system.

6.4 Evaluation Methods

6.4.1 Model-as-a-judge Evaluation

Each of the four candidate models were used in turn as a judge, scoring the responses of all models on a scale of 0-5. Scores were averaged across judges. This approach enabled us to capture relative model preferences and robustness under different evaluators.

6.4.2 RAGAS-based Evaluation

In parallel, we also used the RAGAS evaluation framework with Llama 3.1 (8B) as a judge model. We selected Llama 3.1 8B for this role because it is instruction-tuned, making it well suited to follow evaluation prompts consistently. RAGAS

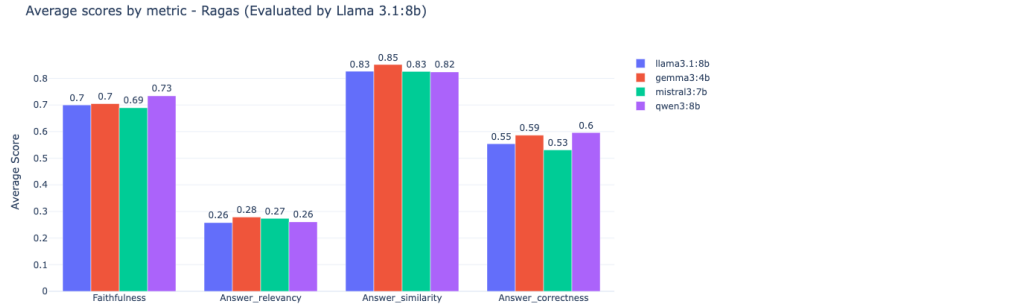


Figure 6: Evaluation Results - RAGAS

provides an automatic evaluation of generative responses against retrieved context, scoring each metric on a continuous scale. This provided an independent and standardized perspective on model performance.

6.5 Comparative Performance

The two evaluation approaches produced largely consistent trends, though with some variation in absolute score ranges due to the different scales used.

- Gemma 3 (4B) and Qwen 3 (8B) consistently outperformed the other models across most metrics.
- Gemma 3 showed particular strength in context recall and faithfulness, suggesting that it was especially effective at grounding its responses in the retrieved context and maintaining factual consistency.
- Qwen 3 performed comparably to Gemma overall, it excelled on answer correctness, with responses more frequently aligned with the expected or reference answer.
- The differences between Gemma and Qwen were marginal. In practice, their performance levels for our use case can be regarded as nearly equivalent.
- Given this near-parity, system performance metrics were considered as tie-breaker. The model with lower latency was selected for deployment, balancing quality with efficiency.
- Llama 3.1 (8B) and Mistral (7B) achieved comparable results to each other, but consistently trailed behind Gemma and Qwen.
- Their strengths were in answer similarity and relevancy, where they produced coherent and semantically similar responses. However, they underperformed on faithfulness and correctness.

- Overall, both can be regarded as solid baseline generators, but less reliable for contexts requiring high factual precision.

7 Conclusion

Our comparative evaluation of four LLMs within the RAG framework highlights clear trade-offs between efficiency and generative quality. Across latency measures—system latency, time-to-first-token, and tokens per second—Gemma 3 (4B) consistently achieved the best performance, benefiting from its smaller parameter size and optimized inference pipeline. Qwen 3 (8B), while slower overall, distinguished itself with strong correctness scores in generation, often matching Gemma’s quality.

Taken together, Gemma and Qwen emerged as the top-performing models, with only marginal differences in quality. Given their near parity, we prioritized system performance as the deciding factor, selecting Gemma for deployment due to its lower latency and stronger throughput. Llama 3.1 (8B) and Mistral (7B) produced coherent outputs and remain viable as baseline models, but they trailed behind in both responsiveness and factual precision.

Overall, the findings suggest that smaller, more optimized models like Gemma can deliver both efficiency and reliability in RAG workflows, offering a practical balance of speed, quality, and cost-effectiveness.

References

- [1] Grimmer J Roberts M E Stewart B M. “Machine Learning for Social Science: An Agnostic Approach.” In: *Annual Review of Political Science* Vol. 24:395-419 (2021). DOI: <https://doi.org/10.1146/annurev-polisci-053119-015921>.
- [2] *Understanding America Study*. URL: <https://uasdata.usc.edu>.
- [3] Nelson F. Liu Kevin Lin John Hewitt Ashwin Paranjape Michele Bevilacqua Fabio Petroni Percy Liang. “Lost in the Middle: How Language Models Use Long Contexts”. In: *Transactions of the Association for Computational Linguistics* 12 (2024). DOI: <https://aclanthology.org/2024.tacl-1.9/>.
- [4] Zhiyu Ma Aaron Grattafiori Abhimanyu Dubey Abhinav Jauhri... In: (2024). arXiv: 2407.21783 [cs.AI]. URL: <https://arxiv.org/abs/2407.21783>.