Evaluating Large Language Models for Budget-Related Question Answering

Shivam Singh Rawat
School of Information Sciences
University of Illinois at Urbana-Champaign
Champaign, USA
shivambitid007@gmail.com

Abstract

This study explores the potential of Large Language Models (LLMs) in analyzing U.S. federal budget documents to enhance accessibility and comprehension for policymakers and researchers. We evaluate four LLMs—Llama 3.2 (3B), Mistral (7B), Gemma 2 (2B), and DeepSeek-R1 (1.5B)—on their ability to answer budget-related queries using a structured dataset from 2023 to 2025. Model performance is assessed through a similarity search approach leveraging Sentence Transformers, alongside ROUGE and BLEU scores to quantify accuracy, relevance, and fluency. A 50-question benchmark ensures a consistent evaluation framework, with preliminary results indicating Llama 3.2's superior semantic alignment (0.799 average similarity score). Findings highlight the strengths and limitations of LLMs in budget analysis, offering insights for improving financial data retrieval and interpretation.

1 Literature Review

The evaluation of Large Language Models (LLMs) in Retrieval-Augmented Generation (RAG) systems has gained increasing attention due to their potential to enhance factual accuracy and mitigate hallucinations. This literature review synthesizes recent research on the accuracy of five LLMs—Mistral:7b, Gemma2:2b, LLaMA3.2:3b, DeepSeek-R1:1.5B, and GPT-3.5—in the RAG framework. The discussion focuses on retrieval quality, response accuracy, and hallucination mitigation strategies.

In the research paper "Enhancing LLM Factual Accuracy with RAG to Counter Hallucinations: A Case Study on Domain-Specific Queries in Private Knowledge-Bases" by Li et al. (2024), they use a form of RAG to help improve the factual accuracy of LLM's. They created a Retrieval Augmented Generation (RAG) pipeline that uses several models including a retriever model and a generator model. The retriever model retrieves the most relevant information from the dataset and returns it as context for later generation. The generator model takes the returned information and adds it to the QA prompts as context to generate answers. They also use a Mixedbread.ai embedding model. An embedding model is a model that calculates vector embeddings for words or phrases. This is important, because the smaller the size of embedding, the better it will perform. So they chose this model for it's compact size and high performance. They also use a BgeRerank reranker model, which employs the BAAI/bge-reranker-large model. This model assesses pairs of queries and documents, assigning relevance scores that reflect each document's alignment with the query's intent. BgeRerank reorders the initial broad set of documents, prioritizing the top-N (top-5 out of 10 documents retrieved in our experiments) most pertinent ones to the user's query. For the core generation model, they use LLAMA-2. They chose this model because it is an open source model that allows for continuous training and experiments. They also state that it has demonstrated impressive performance across a wide range of natural language processing benchmarks. Pre-trained on 2 Trillion tokens and further fine-tuned on 100k human annotation data, LLaMA-2 can capture linguistic patterns and domain knowledge, enabling it to generate highly fluent and coherent text.

In the research paper "LRP4RAG: Detecting Hallucinations in Retrieval-Augmented Generation via Layer-wise Relevance Propagation" by (Hu et al., 2024), they highlight the persistent challenge of hallucinations in Retrieval-Augmented Generation (RAG) systems, despite RAG's role as a key method for reducing inaccuracies in large language models (LLMs). The authors propose a novel approach named LRP4RAG, which utilizes Layer-wise Relevance Propagation (LRP) to detect hallucinations by evaluating the relevance between inputs and outputs in the RAG generator. This method aims to address the gap in understanding and detecting the conditions under which hallucinations occur in RAG systems. LRP4RAG leverages a back-propagation technique to analyze the correlation between prompt inputs and LLM-generated responses, processing the relevance matrix through classifiers to predict the presence of hallucinations. The approach is evaluated using the RAGTruth dataset, demonstrating superior performance over existing baselines. This research not only introduces a new method for detecting hallucinations but also underscores the importance of analyzing internal LLM states to improve the reliability of RAG systems.

In the research paper "Honest AI: Fine-Tuning "Small" Language Models to Say "I Don't Know", and Reducing Hallucination in RAG" by Chen et al. (2024), they tackle the issue of hallucination in Large Language Models (LLMs) by proposing a fine-tuning strategy named "Honest AI." This method trains smaller LLMs to accurately respond with "I don't know" when faced with uncertain information, enhancing their reliability in accuracy-sensitive applications. The authors also explore and evaluate several Retrieval-Augmented Generation (RAG) approaches, including integrating search engine and knowledge graph results, and find that combining RAG with fine-tuning yields the best performance in the CRAG benchmark. Their hybrid approach not only improves accuracy but also effectively reduces hallucination by enabling models to better discern and admit uncertainty. This research underscores the importance of fine-tuning in conjunction with RAG to significantly enhance LLM performance, particularly in generating truthful and reliable responses. Although I won't be fine-tuning the LLMs, effective prompt engineering can ensure they respond with "I Don't Know" when asked questions beyond the scope of the U.S. Budget (2023-2025).

In "A Brief Survey of Vector Databases" by (Xie et al., 2023), the authors provide a comprehensive overview of vector databases, which are increasingly relevant in the context of RAG systems. The paper highlights the challenges of managing high-dimensional data and how vector databases offer efficient solutions for storing and retrieving vector embeddings, which are crucial for tasks like similarity search. The authors discuss various similarity search algorithms (e.g., K-Means, Locality Sensitive Hashing, Hierarchical Navigable Small Worlds) and similarity metrics (e.g., Euclidean Distance, Dot Product, Cosine Similarity) that are essential for optimizing retrieval processes in vector databases. These techniques are directly applicable to RAG systems, where the quality of retrieval directly impacts the accuracy of generated responses. The paper also compares popular vector database products like Pinecone, Chroma, and Milvus, emphasizing their scalability, reliability, and compatibility, which are critical factors for integrating them into RAG pipelines. This survey underscores the importance of vector databases in improving the efficiency and precision of retrieval processes, which is vital for improving the performance of LLMs in RAG frameworks. In my project I'll be using Pinecone as my vector database.

Additionally, I drew insights from the Learning semantic similarity for very short texts (De Boom et al., 2015) work on semantic similarity for very short texts, which explores the use of distributed word representations to capture semantic relationships in contexts with limited word overlap. This research informs our understanding of how embedding models can enhance semantic similarity in budget-related queries, particularly when dealing with concise or fragmented text.

2 Descriptive Statistics of the Data

Table 1: Budget Overview

Metric	Value
Number of documents	3
Total pages	528 pages
Number of unique departments in the budgets	21
DPT OF AGRICULTURE	2
DPT OF COMMERCE	3
DPT OF DEFENCE	2
DPT OF HEALTH AND HUMAN SERVICES	2
DPT OF HOMELAND SECURITY	1

The dataset consists of U.S. federal budget documents from 2023 to 2025. These are comprehensive government reports detailing financial allocations, revenue sources, and policy priorities. The dataset is publicly available on the government website. The budget documents comprise a total of 528 pages (188 in 2025, 184 in 2024, and 156 in 2023). Table 1 provides a comprehensive overview of the dataset, including both the descriptive statistics and the frequency of department references, ensuring a clear understanding of the dataset's scope and focus. . Additionally, Table 1 includes a breakdown of the frequency of references to specific departments, such as the Department of Agriculture, Department of Commerce, and Department of Homeland Security.

Table 3 lists 10 sample queries related to the U.S. federal budget for 2023. The queries cover a wide range of topics, including funding allocations, policy measures, and investments across various departments. These queries are designed to evaluate the effectiveness of Large Language Models (LLMs) in retrieving and analyzing budget-related information.

3 Preprocessing and Transformation Steps

The preprocessing and transformation of the U.S. federal budget documents (2023–2025) involved several key steps to prepare the data for efficient storage and retrieval using Pinecone, a vector database. Below is a detailed explanation of the steps:

3.1 Loading the Documents

The budget documents were loaded using the `PyPDFLoader` from the `langchain_community.document_loaders` library. This loader extracts text content from PDF files, making it accessible for further processing.

Three budget documents were loaded:

- `budget_fy2025.pdf`
- `BUDGET-2023-BUD.pdf`
- `BUDGET-2024-BUD.pdf`

3.2 Text Splitting

To handle the large size of the budget documents, the text was split into smaller, manageable chunks using the `RecursiveCharacterTextSplitter` from the `langchain` library. This splitter ensures that the text is divided into meaningful fragments while preserving context.

The following parameters were used:

- `chunk_size=1000`: Each chunk contains up to 1000 characters.
- `chunk_overlap=100`: A 100-character overlap between chunks ensures continuity and prevents loss of context at chunk boundaries.

3.3 Generating Embeddings

The text fragments were converted into vector embeddings using the `SentenceTransformerEmbeddings` model (`all-mpnet-base-v2`). This model generates high-quality embeddings that capture the semantic meaning of the text, enabling efficient similarity search and retrieval.

4 Model parameters, and evaluation results

This study evaluates the performance of four Large Language Models (LLMs) in answering budget-related questions using similarity search. The models analyzed include Llama 3.2 with 3 billion parameters, Mistral with 7 billion parameters, Gemma 2 with 2 billion parameters, and DeepSeek-R1 with 1.5 billion parameters. To assess their effectiveness, similarity scores were computed for ten budget-related queries, measuring the semantic alignment between model responses and retrieved documents. Llama 3.2 achieved the highest average similarity score of 0.799, followed by Mistral at 0.742, while DeepSeek-R1 and Gemma 2 demonstrated moderate performance with scores of 0.670 and 0.661, respectively. These preliminary results indicate that Llama 3.2 and Mistral show strong potential for budget analysis tasks, while DeepSeek-R1 and Gemma 2 may require further optimization to enhance their performance, particularly in handling complex financial and policy-related queries. Table 4, showcases all the results I achieved during the first set of tests.

Table 2: LLM Models and Their Parameters

Model	Parameters
Llama 3.2	3B
Mistral	7B
Gemma 2	2B
DeepSeek-R1	1.5B

References

Xinxi Chen, Li Wang, Wei Wu, Qi Tang, and Yiyao Liu. 2024. Honest ai: Fine-tuning" small" language models to say" i don't know", and reducing hallucination in rag. *arXiv preprint arXiv:2410.09699*.

Cedric De Boom, Steven Van Canneyt, Steven Bohez, Thomas Demeester, and Bart Dhoedt. 2015. Learning semantic similarity for very short texts. In *2015* *ieee international conference on data mining workshop (icdmw)*, pages 1229–1234. IEEE.

Haichuan Hu, Yuhan Sun, and Quanjun Zhang. 2024. Lrp4rag: Detecting hallucinations in retrieval-augmented generation via layer-wise relevance propagation. *arXiv preprint arXiv:2408.15533*.

Jiarui Li, Ye Yuan, and Zehua Zhang. 2024. Enhancing llm factual accuracy with rag to counter hallucinations: A case study on domain-specific queries in private knowledge-bases. *arXiv preprint arXiv:2403.10446*.

Xingrui Xie, Han Liu, Wenzhe Hou, and Hongbin Huang. 2023. A brief survey of vector databases. In *2023 9th International Conference on Big Data and Information Analytics (BigDIA)*, pages 364–371. IEEE.

Table 3: List of Queries

S.no	**Queries**
1	How much discretionary funding does the President's 2023 Budget request for USDA?
2	What steps does the 2023 Budget take to address wildfires?
3	How does the 2023 Budget strengthen the Nation's supply chains?
4	How does the 2023 Budget address the impacts of climate change and extreme weather?
5	How does the 2023 Budget advance U.S. leadership in emerging technologies?
6	What measures does the 2023 Budget propose to strengthen deterrence in the Indo-Pacific region?
7	How does the 2023 Budget address cybersecurity threats?
8	How does the 2023 Budget invest in both scientific research and pandemic preparedness?
9	What investments does the 2023 Budget make to end the HIV/AIDS epidemic?
10	How does the 2023 Budget improve pay and workforce policies for TSA employees?

Table 4: Similarity Scores Across LLMs

Query No.	**Llama Similarity**	**Gemma Similarity**	**Mistral Similarity**	**Deepseek Similarity**
Query 1	0.9472	0.7864	0.8199	0.6683
Query 2	0.7818	0.5828	0.7119	0.7439
Query 3	0.9377	0.8198	0.5992	0.4956
Query 4	0.6681	0.5813	0.6517	0.4657
Query 5	0.5404	0.4357	0.7411	0.4729
Query 6	0.8796	0.6559	0.8997	0.7370
Query 7	0.7051	0.8129	0.7233	0.6774
Query 8	0.7890	0.5704	0.8095	0.6806
Query 9	0.9401	0.5752	0.5997	0.5090
Query 10	0.7981	0.7866	0.8599	0.7446
Average	0.7987	0.6607	0.7416	0.6696