

Reproducibility Study: RAG-based Financial Q&A Agent with Gemini (Vertex AI)

HU TIANYU
Student ID: 1155249527

February 22, 2026

Abstract

This report documents a reproducibility study of a Retrieval-Augmented Generation (RAG) agent for financial question answering, originally implemented using Google’s Gemini model via Vertex AI. The target claim was a 100% accuracy on a small set of eight factual and financial questions about five major companies. The reproduction was conducted in Google Colab, using a manual RAG pipeline to circumvent dependency issues. The baseline accuracy of 100% was exactly reproduced. A controlled modification was then applied: increasing the LLM temperature from 0.1 to 0.7 while keeping all other components unchanged. The modification resulted in no measurable drop in accuracy, but introduced slight variability in response phrasing. A debug diary details encountered challenges—such as missing LangChain modules and authentication errors—and their resolutions. The study concludes that the core functionality is robust and reproducible, and recommends low temperatures (0.1–0.3) for factual tasks to minimize unnecessary variation.

1 Project Summary

1.1 Project Overview

The project under investigation is a **RAG-based Document Question-Answering Agent** that combines a retrieval component (FAISS over financial documents) with a generative model (Gemini via Vertex AI). The agent is designed to answer questions about companies using a small in-memory knowledge base. This type of system exemplifies an *agentic* workflow because it involves multiple steps: query embedding, similarity search, context assembly, and LLM-based answer generation.

1.2 Original Source and Target Claim

The original implementation was adapted from an instructor-provided repository (shorturl.at/yx112), which demonstrated a simple RAG pipeline for financial Q&A. The claimed performance, as stated in the accompanying documentation, was **100% accuracy** on a set of eight questions covering facts about Apple, Microsoft, Amazon, Tesla, and NVIDIA, when using the Gemini model with temperature 0.1.

1.3 Scope of Reproduction

In line with reproducibility challenge guidelines, this study focuses on a single, well-defined target: the accuracy on those eight questions. No attempt was made to reproduce larger benchmarks or other tasks, as the original material did not provide them. The reproduction

was considered successful if the achieved accuracy matched the claimed 100% within a reasonable tolerance ($\pm 10\%$ as per common practice).

1.4 Modification Chosen

To explore the sensitivity of the agent to stochastic variation, a simple modification was introduced: **increase the temperature parameter of the Gemini model from 0.1 to 0.7**. All other components—retrieval method, number of retrieved chunks, prompt template, evaluation metric—remained identical. The goal was to measure whether higher temperature would degrade factual accuracy or introduce meaningful variability in outputs.

2 Setup Notes

2.1 Environment

All experiments were conducted in a Google Colab notebook (free tier) with the following specifications:

- **Python version:** 3.10.12
- **Hardware:** Intel Xeon CPU @ 2.20GHz, 12.7 GB RAM, optional T4 GPU (not required for inference)
- **Operating System:** Ubuntu 20.04.6 LTS (containerized)

2.2 Dependencies and Installation

The following Python packages were installed via pip (exact versions captured from the Colab environment):

Listing 1: Installation commands

```
!pip install -q \
    langchain-google-vertexai==1.0.0 \
    langchain-community==0.0.10 \
    langchain-text-splitters==0.0.1 \
    faiss-cpu==1.7.4 \
    sentence-transformers==2.2.2 \
    google-cloud-aiplatform==1.38.0
```

All packages were successfully installed without conflicts. The notebook also required authentication for Google Cloud services, which was handled via `google.colab.auth`.

2.3 Vertex AI Configuration

To use Gemini models through Vertex AI, the following configuration was applied:

- **Project ID:** `linen-age-483607-m5`
- **Location:** `us-central1`
- **Model name:** `gemini-2.5-flash` (Note: This is a model alias; the actual deployed version may vary.)
- **Generation parameters (baseline):**
`temperature=0.1, max_output_tokens=1024, top_p=0.95, top_k=40`

The Vertex AI Python client was initialized after authentication using `vertexai.init()`.

2.4 Dataset Construction

A synthetic financial dataset was created manually to cover five well-known companies. The content was written into a text file `financial_data.txt`. An excerpt is shown below:

Listing 2: Sample of the dataset

```
Company: Apple Inc. (AAPL)
Founded: April 1, 1976
Founders: Steve Jobs , Steve Wozniak , Ronald Wayne
CEO: Tim Cook (as of 2025)
Headquarters: Cupertino , California
Revenue 2024: $394.3 billion
...
...
```

The document was split into 5 chunks using `RecursiveCharacterTextSplitter` with `chunk_size=500` and `chunk_overlap=50`. The chunks were then embedded using the sentence transformer model `all-MiniLM-L6-v2` and stored in a FAISS vector store.

2.5 Evaluation Protocol

For each question, the top $k = 3$ chunks were retrieved and inserted into a fixed prompt template:

Listing 3: Prompt template

```
prompt = f"""You -are -a -helpful -financial -assistant . -Use -the -following
context -to -answer -the -question -at -the -end .
If -the -answer -is -not -in -the -context , -say -"I -don 't -have -enough
information -to -answer -that ."
```

Context :

```
{context}
```

Question : -{question}

Answer : """

The LLM response was captured and compared to the expected answer via a case-insensitive substring match. A question was marked correct if the expected string appeared anywhere in the generated answer. This lenient criterion is appropriate for factual verification where paraphrasing is acceptable.

3 Reproduction Target and Metric Definition

3.1 Target Statement

The original work claimed that the described RAG agent, using Gemini with temperature 0.1, achieves **100% accuracy** on the eight predefined questions. This claim was treated as the reproduction target. No additional metrics (e.g., F1, exact match) were provided in the original material, so the substring match was adopted as the closest proxy.

3.2 Test Cases

Table 1 lists the eight questions together with their expected answers and a category label.

Table 1: Test questions and expected answers

Question	Expected Answer	Category
Who is the CEO of Apple?	Tim Cook	factual
What was Microsoft’s revenue in 2024?	\$245.1 billion	financial
When was Amazon founded?	July 5, 1994	factual
What is NVIDIA’s market cap?	\$2.8 trillion	financial
Who founded Tesla?	Martin Eberhard and Marc Tarpenning	factual
What is Apple’s main product?	iPhone	product
How many employees does Amazon have?	1,525,000	factual
What is Microsoft’s stock symbol?	MSFT	financial

4 Reproduction Results (Baseline, Temperature = 0.1)

4.1 Quantitative Results

The baseline evaluation was performed exactly as described in the previous section. All eight questions were answered correctly, yielding a perfect accuracy of 100%. Detailed per-question results, including the exact answer generated and response time, are presented in Table 2. The average response time was 0.62 seconds, with minimal variance.

Table 2: Detailed baseline results (temperature = 0.1)

Question	Generated Answer	Correct	Time (s)
Who is the CEO of Apple?	Tim Cook	Yes	0.70
What was Microsoft’s revenue in 2024?	Microsoft’s revenue in 2024 was \$245.1 billion.	Yes	0.67
When was Amazon founded?	Amazon was founded on July 5, 1994.	Yes	0.60
What is NVIDIA’s market cap?	NVIDIA’s market cap is \$2.8 trillion (as of 2025).	Yes	0.62
Who founded Tesla?	Martin Eberhard and Marc Tarpenning founded Tesla.	Yes	0.56
What is Apple’s main product?	Apple’s main products are iPhone, Mac, iPad, Services, and Wearables.	Yes	0.59
How many employees does Amazon have?	Amazon has 1,525,000 employees.	Yes	0.56
What is Microsoft’s stock symbol?	Microsoft’s stock symbol is MSFT.	Yes	0.65

4.2 Comparison with Claimed Target

The reproduced accuracy (100%) matches the claimed target exactly. Therefore, the baseline result is **fully reproducible** under the conditions described. The slight variations in phrasing (e.g., inclusion of “as of 2025” for NVIDIA’s market cap) do not affect correctness and are within expected behaviour of generative models.

4.3 Observations on Response Quality

All answers were concise, directly derived from the retrieved context, and contained no hallucinations. The retrieval consistently returned relevant chunks: for each question, at least one of the three retrieved chunks contained the exact factual string. This confirms that the embedding model and FAISS index are appropriately configured for this domain.

5 Modification and Results (Temperature = 0.7)

5.1 Rationale for Modification

Temperature controls the randomness of token sampling. Lower temperatures (e.g., 0.1) make the model more deterministic and focused, which is desirable for factual tasks. Higher temperatures (e.g., 0.7) introduce more diversity and creativity, but may increase the risk of off-topic or hallucinated answers. The chosen modification examines whether a moderate increase to 0.7 would degrade accuracy on this simple task, or merely produce paraphrased but still correct answers.

5.2 Modified Model Configuration

A new LLM instance was created with identical parameters except temperature = 0.7. The exact code is:

```
llm_modified = VertexAI(  
    model_name="gemini-2.5-flash",  
    project=PROJECT_ID,  
    location=LOCATION,  
    temperature=0.7,  
    max_output_tokens=1024,  
    top_p=0.95,  
    top_k=40  
)
```

5.3 Quantitative Results after Modification

The modified agent was evaluated on the same eight questions. Table 3 shows the results. Accuracy remained 100%, with an average response time of 0.63 seconds (a negligible increase of 0.01 s).

Table 3: Detailed modified results (temperature = 0.7)

Question	Generated Answer	Correct	Time (s)
Who is the CEO of Apple?	Tim Cook	Yes	0.91
What was Microsoft’s revenue in 2024?	Microsoft’s revenue in 2024 was 245.1 billion .	Yes	0.64
When was Amazon founded?	Amazon was founded on July 5, 1994.	Yes	0.57
What is NVIDIA’s market cap?	NVIDIA’s market cap is 2.8 trillion (as of 2025).	Yes	0.65
Who founded Tesla?	Martin Eberhard and Marc Tarpenning founded Tesla. (Elon Musk joined later).	Yes	0.57
What is Apple’s main product?	Apple’s main products are iPhone, Mac, iPad, Services, and Wearables.	Yes	0.60
How many employees does Amazon have?	Amazon has 1,525,000 employees.	Yes	0.52
What is Microsoft’s stock symbol?	MSFT	Yes	0.57

5.4 Qualitative Differences

While all answers remained factually correct, the higher temperature introduced subtle stylistic changes:

- For the Tesla founders question, the modified answer added an unsolicited parenthetical: “(Elon Musk joined later)”. This extra information, while not requested, is factually accurate and derived from the context.
- The answer to Microsoft’s stock symbol was simply “MSFT” instead of the fuller sentence “Microsoft’s stock symbol is MSFT.”.
- Other answers were nearly identical to the baseline, indicating that the model remains anchored to the retrieved context even at higher temperature.

No hallucinations or factual errors were observed. The retrieval step appears to constrain the generation sufficiently, preventing the model from straying outside the provided context.

5.5 Impact Summary

The modification had **no measurable impact on accuracy** and only a negligible effect on response time. However, the increased stochasticity can be seen in the minor phrasing variations. For applications requiring strict adherence to a canonical form (e.g., exact ticker symbols), a lower temperature is still advisable to avoid unnecessary abbreviation or elaboration.

6 Debug Diary

During the reproduction process, several obstacles were encountered and resolved. This diary documents them for future users.

6.1 Issue 1: Missing langchain.chains Module

Symptom

Attempting to import `RetrievalQA` from `langchain.chains` raised a `ModuleNotFoundError`.

Root Cause

The installed version of LangChain (community edition) does not include the `chains` subpackage by default, or it may have been moved to a different distribution. The exact cause is unclear, but it prevented the use of the convenient `RetrievalQA` wrapper.

Resolution

A manual RAG function was implemented (see Listing 4), which performs similarity search, context assembly, and prompt invocation directly. This approach not only resolved the import error but also provided greater transparency and control over the pipeline.

Listing 4: Manual RAG function

```
def rag_query(question: str, llm, vectorstore, k: int = 3):
    docs = vectorstore.similarity_search(question, k=k)
    context = "\n\n".join([doc.page_content for doc in docs])
    prompt = f"..."
    response = llm.invoke(prompt)
    return {"result": response, "source_documents": docs}
```

6.2 Issue 2: Vertex AI Authentication Failure

Symptom

When first invoking the Gemini model, an authentication error was raised, indicating that the client had not been properly authenticated.

Root Cause

In Google Colab, explicit user authentication and project initialization are required. The notebook initially lacked these steps.

Resolution

The following code was added at the beginning of the notebook:

```
from google.colab import auth
auth.authenticate_user()
import vertexai
vertexai.init(project=PROJECT_ID, location=LOCATION)
```

After re-running, the API calls succeeded.

6.3 Issue 3: Deprecation Warning for HuggingFaceEmbeddings

Symptom

When creating the embeddings, a deprecation warning appeared: “`HuggingFaceEmbeddings` is deprecated and will be removed in a future version.”

Root Cause

The `langchain-community` package has moved embedding classes to a separate package (`langchain-huggingface`). The warning is harmless in the short term.

Resolution

The warning was ignored, as the code continued to work correctly. For long-term maintenance, users are advised to install `langchain-huggingface` and update the import accordingly.

6.4 Issue 4: Truncated Answers

Symptom

In preliminary tests, some answers were cut off mid-sentence, especially for longer responses.

Root Cause

The default `max_output_tokens` for Vertex AI is relatively low (e.g., 256). The prompt and context sometimes exceeded this limit.

Resolution

The parameter was explicitly set to `max_output_tokens=1024` in the LLM constructor. This provided sufficient headroom for all answers in the test set.

7 Conclusions and Recommendations

7.1 Reproducibility Assessment

The baseline claim of 100% accuracy on the eight-question financial dataset is **fully reproducible**. The agent, when implemented as described, consistently produces correct answers with low latency. The manual RAG implementation proved robust and easy to debug.

7.2 Impact of Temperature Modification

Increasing the temperature from 0.1 to 0.7 did not degrade accuracy on this simple task. However, it introduced minor stylistic variations. For factual Q&A where precise wording matters (e.g., stock symbols, numerical formats), a lower temperature remains preferable. The retrieval component effectively anchors the generation, mitigating the risk of hallucination even at higher temperatures.

7.3 What Worked / What Did Not Work

- **Worked well:** FAISS retrieval, sentence-transformer embeddings, Vertex AI Gemini API, manual RAG loop.
- **Did not work initially:** `langchain.chains` import; resolved by manual implementation.

7.4 Key Lessons Learned

1. Relying on high-level LangChain wrappers can lead to brittle imports; a manual implementation is often simpler and more transparent.

2. Always include explicit authentication steps when using cloud services in ephemeral environments like Colab.
3. For factual tasks, keep temperature low (0.1–0.3) to ensure consistent phrasing.
4. Caching embeddings reduces startup time in repeated runs.
5. Document all parameters (model name, temperature, max tokens) to ensure exact reproducibility.

7.5 Recommendations for Future Users

1. If you encounter import errors with LangChain, consider writing a custom RAG function as shown in this report.
2. Use a persistent vector store to avoid recomputing embeddings each session.
3. When comparing models, record the exact generation parameters and prompt templates.
4. For production systems, implement answer validation (e.g., regex checks for expected formats) to catch occasional deviations.
5. If using a different LLM (e.g., DeepSeek), document the change and treat results as not directly comparable to the original.

GitHub Repository and Presentation

All code, including the final notebook and this report, is available at:

<https://github.com/icecakety/FTEC5660/tree/main/homeworks/hw2>