Leveraging LLMs for Real-Time Business Insights from Google BigQuery

The integration of large language models (LLMs) with data warehouses like Google BigQuery is revolutionizing how businesses extract valuable insights from their data[1]. This article explores two primary approaches to achieve this: Retrieval Augmented Generation (RAG) and LLM-Generated Code. We'll delve into the mechanics of each approach, analyze their pros and cons, and recommend the best solution for real-time business insights from your BigQuery data.

Understanding Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) enhances the capabilities of LLMs by grounding them in external knowledge sources[2]. Instead of solely relying on the information it was trained on, an LLM using RAG first retrieves relevant information from a designated knowledge base before generating a response. This approach involves identifying and retrieving contextually relevant information from external knowledge sources, such as your BigQuery database, to augment the LLM's knowledge and improve the quality of its responses[3]. The LLM then combines this retrieved knowledge with its internal knowledge to produce a more accurate and comprehensive answer[4].

RAG offers several advantages:

- **Reduced Hallucination:** By grounding the LLM in factual data, RAG minimizes the risk of the model generating incorrect or nonsensical outputs[4].
- **Improved Accuracy:** Access to external knowledge sources ensures that the LLM's responses are based on the most current and relevant information[5].
- **Enhanced Trust:** RAG allows the LLM to cite its sources, enabling users to verify the information and trust the generated responses[4].
- **Efficient Updates:** Integrating new information is relatively easy with RAG, as it doesn't require retraining the entire model[4].
- **Reduced Contradictions:** RAG helps minimize contradictions and inconsistencies in the generated text by fine-tuning or prompt-engineering the LLM to generate text entirely based on the retrieved knowledge[6].
- **Increased Business Control:** RAG redirects the LLM to retrieve relevant information from pre-determined knowledge sources, giving businesses more control over the generated text output[2].

Exploring LLM-Generated Code

While RAG focuses on augmenting the LLM's knowledge with external information, another approach involves leveraging the LLM's ability to generate code. LLM-Generated Code utilizes

the code generation capabilities of LLMs to query and manipulate data in BigQuery[7]. In this approach, the LLM generates Python code based on the user's query, which is then executed to retrieve the required information from BigQuery[8]. The LLM learns the patterns and structures of different programming languages by training on vast amounts of code from various sources, such as GitHub repositories, Stack Overflow, and code snippets from websites[9].

Some benefits of LLM-Generated Code include:

- **Automation:** Automating the code generation process can save developers time and effort[7].
- **Efficiency:** LLMs can generate code quickly, potentially leading to faster query execution and data retrieval[7].
- **Flexibility:** LLMs can generate code in various programming languages, providing flexibility in how you interact with your data[10].

However, it's essential to be aware of the potential limitations of this approach:

- **Code Complexity:** LLMs can sometimes produce complex code blocks that are difficult to understand and troubleshoot[7].
- **Training Data Limitations:** The quality and scope of the training data significantly influence the performance of LLMs, and they may not always generate accurate or efficient code[7].
- **Contextual Understanding:** LLMs may lack sufficient contextual understanding of specific use cases or niches, leading to code that doesn't fully meet the requirements[7].
- **Security Risks:** Security in AI-generated code is a crucial concern. LLMs can sometimes generate code with vulnerabilities, requiring careful review and refinement to ensure security[7].

Real-time Responsiveness with BigQuery

Before diving deeper into the comparison of RAG and LLM-Generated Code, it's crucial to understand how BigQuery facilitates real-time data analysis. BigQuery offers several features that enable real-time responsiveness:

- **Continuous Queries:** BigQuery's continuous queries analyze streaming data in real-time, allowing you to gain immediate insights from your data as it flows in[11].
- **DataFrames Integration:** Continuous queries integrate with BigQuery DataFrames, enabling data scientists to work with real-time data directly within their Python notebooks[12].
- **Streaming APIs:** BigQuery provides streaming APIs for direct ingestion of data, supporting event-driven architectures and real-time data pipelines[11].

These capabilities make BigQuery a suitable platform for building LLM-powered applications that require real-time responsiveness.

Comparing RAG and LLM-Generated Code for BigQuery

When deciding between RAG and LLM-Generated Code for querying business data from BigQuery with real-time responsiveness, consider the following table summarizing the pros and

cons:

Feature	RAG	LLM-Generated Code
Accuracy	Higher accuracy due to grounding in external knowledge [5]	Potential for errors in generated code, leading to inaccurate results [7]
Responsiveness	Can be slower due to the retrieval step [2]	Potentially faster as it directly executes code [7]
Complexity	Simpler to implement and maintain [4]	Requires expertise in code generation and execution [9]
Security	More secure as it relies on trusted knowledge sources [2]	Potential security risks if the generated code is not properly vetted [7]
Cost	Can be more cost-effective as it doesn't require extensive code generation [2]	May incur higher costs due to the need for code execution resources [7]
Maintainability	Easier to maintain as it doesn't involve managing code [4]	Can be more challenging to maintain due to the need for code updates and debugging [7]
Contextual Understanding	Leverages the LLM's ability to understand context and retrieve relevant information [3]	May lack sufficient contextual understanding, leading to less accurate or relevant code [7]

Furthermore, both RAG and LLM-Generated Code face challenges when dealing with complex queries that require converting natural language into SQL. These challenges include:

- **Context Collection:** Gathering and integrating relevant information from various sources within the database[13].
- **Retrieval:** Efficiently retrieving the necessary data from BigQuery based on the user's query[13].
- **SQL Generation:** Accurately translating the user's intent into executable SQL queries[13].

However, RAG offers a significant advantage in this context. By bridging the knowledge gap

between LLMs and enterprise databases, RAG allows businesses to leverage their own data for more relevant and accurate insights[14].

Choosing the Best Solution

For your specific scenario, **RAG appears to be the more suitable approach.** While LLM-Generated Code offers potential speed advantages, RAG provides higher accuracy and security, which are crucial when dealing with business-critical data. RAG's ability to ground the LLM in your BigQuery data ensures that the generated insights are reliable and trustworthy. Moreover, BigQuery's serverless architecture provides consistent performance, making it a reliable platform for real-time LLM-powered applications[1].

Recommended LLM Models for RAG

Several LLM models excel in RAG tasks. Here are a few recommendations:

- **Claude 3.5 Sonnet:** This model demonstrates excellent performance across various RAG tasks and supports contexts up to 200k tokens[16].
- **Gemini 1.5 Flash:** Offers strong performance and supports contexts up to 1M tokens, making it suitable for handling large amounts of data[16].
- **Qwen2–72B-Instruct:** A powerful open-source model with great performance in short and medium context RAG tasks[16].

It's worth noting that open-source LLMs are becoming increasingly powerful and can be a viable alternative to closed-source models for RAG tasks[17]. When selecting an LLM, consider factors like knowledge cut-off dates, context window size, and performance benchmarks to choose the best fit for your specific needs[16].

Conclusion

Integrating LLMs with Google BigQuery opens up new possibilities for real-time business insights. While both RAG and LLM-Generated Code offer unique advantages, RAG's focus on accuracy and grounding makes it the preferred choice for your scenario. By carefully selecting the right LLM model, such as Claude 3.5 Sonnet or Gemini 1.5 Flash, and implementing a robust RAG pipeline, you can unlock the full potential of your BigQuery data and gain a competitive edge in today's data-driven world. RAG, combined with BigQuery's real-time capabilities like continuous queries, enables you to generate accurate and reliable business insights with the responsiveness required in today's dynamic business environment.

Works cited

1. BigQuery performance drives real time decisions | Google Cloud Blog, accessed December 12, 2024, https://cloud.google.com/blog/products/data-analytics/bigquery-performance-drives-real-time-decisions
2. What is RAG? - Retrieval-Augmented Generation AI Explained - AWS, accessed December

12, 2024, https://aws.amazon.com/what-is/retrieval-augmented-generation/
3. What is Retrieval-Augmented Generation(RAG) in LLM and How it works? | by Sahin Ahmed, Data Scientist | Medium, accessed December 12, 2024, https://medium.com/@sahin.samia/what-is-retrieval-augmented-generation-rag-in-llm-and-how-it-works-a8c79e35a172
4. What Is Retrieval-Augmented Generation aka RAG - NVIDIA Blog, accessed December 12, 2024, https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/
5. What is retrieval-augmented generation (RAG)? - IBM Research, accessed December 12, 2024, https://research.ibm.com/blog/retrieval-augmented-generation-RAG
6. What is Retrieval-Augmented Generation (RAG)? | Google Cloud, accessed December 12, 2024, https://cloud.google.com/use-cases/retrieval-augmented-generation
7. LLMs for Code Generation: A summary of the research on quality ..., accessed December 12, 2024, https://www.sonarsource.com/learn/llm-code-generation/
8. How do LLMs generate valid code syntax almost every time? - Reddit, accessed December 12, 2024, https://www.reddit.com/r/learnmachinelearning/comments/18lzfe4/how_do_llms_generate_valid_code_syntax_almost/
9. How does a Large Language Model (LLM) writes Code - AI Verse Info, accessed December 12, 2024, https://aiverseinfo.com/how-llm-writes-code/
10. Large Language Models for Code Generation - Part 1 - Vectara, accessed December 12, 2024, https://www.vectara.com/blog/large-language-models-llms-for-code-generation-part-1
11. Should BigQuery be used for real-time data analytics? : r/googlecloud - Reddit, accessed December 12, 2024, https://www.reddit.com/r/googlecloud/comments/1eb5j5w/should_bigquery_be_used_for_realtime_data/
12. BigQuery continuous queries makes data analysis real-time | Google Cloud Blog, accessed December 12, 2024, https://cloud.google.com/blog/products/data-analytics/bigquery-continuous-queries-makes-data-analysis-real-time
13. Top 4 Challenges using RAG with LLMs to Query Database (Text-to-SQL) and how to solve it. - Wren AI Cloud, accessed December 12, 2024, https://getwren.ai/post/4-key-technical-challenges-using-rag-with-llms-to-query-database-text-to-sql-and-how-to-solve-it
14. Announcing Select AI with Retrieval Augmented Generation (RAG) on Autonomous Database - Oracle Blogs, accessed December 12, 2024, https://blogs.oracle.com/datawarehousing/post/announcing-select-ai-with-rag-on-adb
15. LLMs For Structured Data - Neptune.ai, accessed December 12, 2024, https://neptune.ai/blog/llm-for-structured-data
16. What's the Best LLM to Use for RAG? | by Naman Tripathi - Medium, accessed December 12, 2024, https://medium.com/@naman1011/whats-the-best-llm-to-use-for-rag-476bec1bfa97
17. Best LLMs for RAG: Top Open And Closed Source Models - Galileo, accessed December 12, 2024, https://www.galileo.ai/blog/best-llms-for-rag
18. What are the recommended LLM "backend" for RAG : r/LocalLLaMA - Reddit, accessed December 12, 2024, https://www.reddit.com/r/LocalLLaMA/comments/17oy5q3/what_are_the_recommended_llm_backend_for_rag/