Pavan Emani

Summary

The article discusses the transition from Text2SQL and Retrieval-Augmented Generation (RAG) to Table-Augmented Generation (TAG) as a more effective method for AI-driven data queries, leveraging the LOTUS framework to integrate AI and database capabilities.

Abstract

The article "Goodbye, Text2SQL: Why Table-Augmented Generation (TAG) is the Future of AI-Driven Data Queries!" explores the limitations of current AI methods like Text2SQL and Retrieval-Augmented Generation (RAG) in handling complex data queries. It argues that TAG, a new approach developed by researchers from Stanford and Berkeley, overcomes these limitations by combining AI's semantic reasoning with the computational power of databases. TAG utilizes a multi-step process involving query synthesis, execution, and answer generation, enhanced by the LOTUS framework, which allows for semantic queries over structured and unstructured data. This integration enables more sophisticated and context-rich responses to user queries, addressing the critical gap in real-world applicability of AI in data analysis.

Opinions

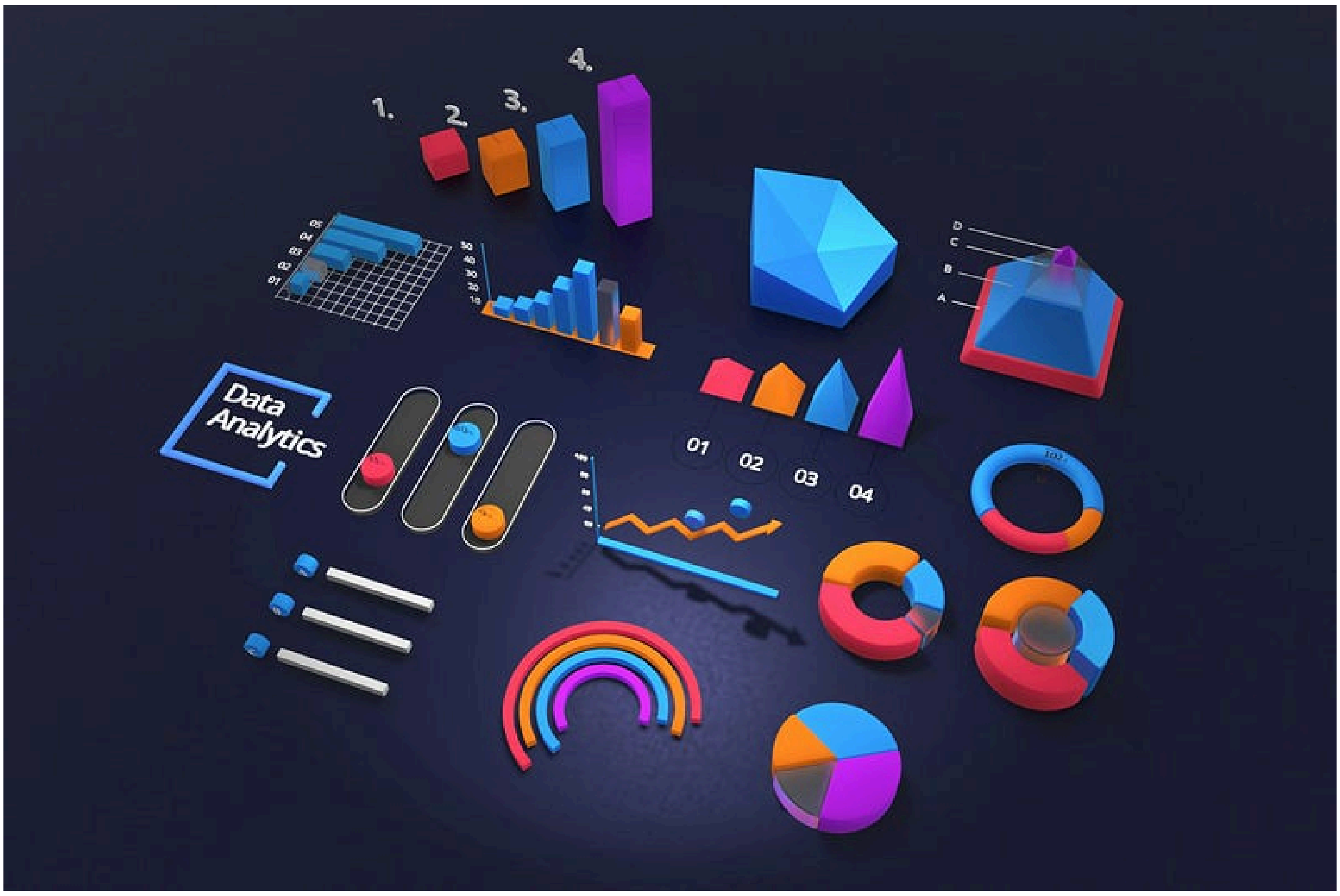
- Text2SQL and RAG methods are inadequate for complex queries, as they either translate queries into SQL or perform simple lookups without capturing real-world complexity.
- The inability of existing methods to effectively combine AI reasoning with database computational power is a major bottleneck for deriving actionable insights from data.
- TAG is presented as a superior alternative, capable of generating complex queries that incorporate multiple data sources and types, and performing advanced operations like sentiment analysis.

query execution.

- The author believes that TAG, powered by LOTUS, represents a significant advancement in AI-driven data querying, offering flexibility, customization,

Goodbye, Text2SQL: Why Table-Augmented Generation (TAG) is the Future of AI-Driven Data Queries!

Exploring the Future of Natural Language Queries with Table-Augmented Generation.



Imagine you’re a business analyst, trying to understand why your company’s sales dropped last quarter. You query your database with a simple natural language question: “Why did sales drop last quarter?” The ideal scenario would be that the AI system instantly provides you with a context-rich, insightful answer — something that ties together all relevant data points, trends, and market insights. However, the reality is far from ideal.

Current AI methods for querying databases, such as Text2SQL and Retrieval-Augmented Generation (RAG), fall significantly short. These models are limited by their design, either only interpreting natural language as SQL queries or relying on simple lookups that fail to capture the complexity of real-world questions.

Why does this matter? Using Natural Language to query SQL databases is the new norm ever since LLMs started capturing the limelight! Businesses today are drowning in data but starving for insights. The inability of existing methods to effectively leverage both AI’s semantic reasoning and databases’ computational power is a major bottleneck in making data truly actionable. It’s clear that we need a new approach — one that can understand and answer the wide range of questions real users want to ask.

But using Natural language in such a scenario comes with challenges:

- **Text2SQL**: This approach is designed to convert natural language questions into SQL queries. While it works well for straightforward questions like “What were the total sales last quarter?” it fails when questions require more complex reasoning or knowledge that is not explicitly stored in the database. For example, a question like “Which customer reviews of product X are positive?” requires sentiment analysis over text data — a capability outside the scope of SQL queries.
- **Retrieval-Augmented Generation (RAG)**: RAG models attempt to use AI to find relevant data records from a database, but they are limited to point lookups and cannot handle complex computations. They often fail to provide accurate

Consider a business scenario where you need to understand trends from customer reviews, sales data, and market sentiment all at once. Text2SQL cannot handle free-text data. And not to forget hallucinations! RAG addresses this to some extent, but its inefficient with large datasets and can provide inaccurate or incomplete answers, especially when it doesn't have the knowledge of target database or it cannot exactly translate the user intent into a functioning SQL!

And so, these approaches leave a large portion of potential user queries unanswered, leading to a critical gap in real-world applicability.

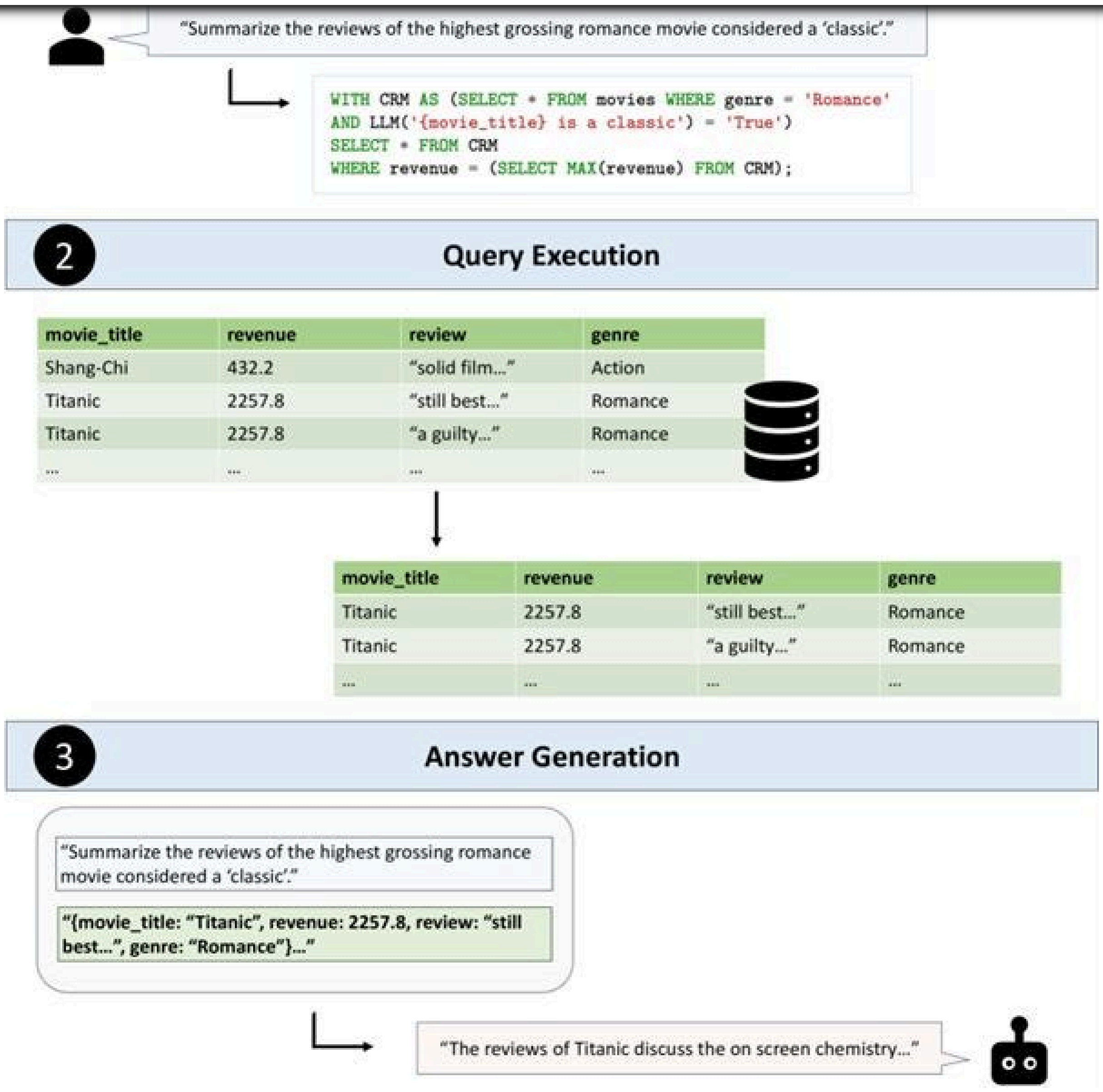
So, what is Table Augmented Generation (TAG) and How it addresses some of these challenges?

Table Augmented Generation (TAG)

TAG is a new augmentation approach that researchers from Stanford and Berkeley are proposing to address the limitations in Text2SQL approach. Here's a link to their paper: https://arxiv.org/abs/2408.14717

Here's how it works:

- **Query Synthesis**: First, the user's natural language request is translated into an executable database query. Unlike Text2SQL, TAG can generate more than just SQL queries; it can synthesize complex queries that combine multiple data sources and types. For example, notice this image that the researchers provided



"Summarize the reviews of the highest grossing romance movie considered a 'classic'."

```
WITH CRM AS (SELECT * FROM movies WHERE genre = 'Romance'
AND LLM('{movie_title} is a classic') = 'True')
SELECT * FROM CRM
WHERE revenue = (SELECT MAX(revenue) FROM CRM);
```

movie_title	revenue	review	genre
Shang-Chi	432.2	"solid film..."	Action
Titanic	2257.8	"still best..."	Romance
Titanic	2257.8	"a guilty..."	Romance
...	...	...	...

movie_title	revenue	review	genre
Titanic	2257.8	"still best..."	Romance
Titanic	2257.8	"a guilty..."	Romance
...	...	...	...

"Summarize the reviews of the highest grossing romance movie considered a 'classic'."

"{movie_title: "Titanic", revenue: 2257.8, review: "still best...", genre: "Romance"}..."

"The reviews of Titanic discuss the on screen chemistry..."

TAG Query Synthesis. Source: https://arxiv.org/abs/2408.14717

Notice how the user query "Summarize the reviews of the highest grossing romance movie considered a **'classic'**" has been translated into:

```
WITH CRM AS (SELECT * FROM movies WHERE genre = 'Romance'
AND LLM('{movie_title} is a classic') = 'True')
SELECT * FROM CRM
WHERE revenue = (SELECT MAX(revenue) FROM CRM);
```

TAG introduced a new LLM call using the line ***LLM('{movie_title} is a classic') = 'True')***. This is the **"Augmentation"** step. The SQL query or more specifically the

2. **Query Execution**: Once the query is synthesized, it is executed against the database. TAG leverages the computational power of databases to efficiently handle large-scale data retrieval and exact computations, which language models struggle to perform.

3. **Answer Generation**: In this final step, the AI model uses the retrieved data to generate a context-rich answer. The model combines world knowledge, semantic reasoning, and domain-specific understanding based on the augmentation in step 1, to produce a comprehensive response to the user’s question.

Another key component that enables TAG to function effectively is the **LOTUS** framework.

LOTUS: The Framework Powering TAG’s Capabilities

As I mentioned above, In order for TAG to work, we need a robust framework that can seamlessly integrate AI capabilities with traditional database systems. This is where **LOTUS** (Leveraging Optimization Techniques for Unifying Semantic Queries) comes into play. LOTUS is designed to bridge the gap between the reasoning power of large language models (LLMs) and the computational strength of databases, enabling more complex and meaningful data queries.

What is LOTUS?

LOTUS is a novel framework that empowers TAG by enabling **semantic queries** over tables containing both structured and unstructured data. It integrates LLMs directly into the database query processing pipeline, combining the best of both worlds — high-performance data management from databases and advanced reasoning and natural language understanding from AI models.

Key Features of LOTUS:

filtering, ranking, and aggregation using natural language processing. For instance, instead of a traditional SQL filter, a LOTUS query might use a language model to determine which rows contain positive sentiment or relevant entities, bringing a whole new level of sophistication to querying.

- **Optimized Query Execution:** LOTUS is built with an **optimized semantic query execution engine** that can handle complex queries more efficiently by batching LLM operations and integrating them into the database's native query processing. This reduces latency and improves performance, making it possible to answer more complex questions quickly.
- **Flexibility and Customization:** The framework allows developers to build custom pipelines that blend traditional SQL operations with advanced AI capabilities. For example, in a financial services use case, LOTUS could enable a query that not only retrieves historical stock data but also analyzes recent news sentiment to provide insights into potential future movements — all in one go.
- **Enabling the TAG Framework:** LOTUS serves as the backbone for implementing the TAG model by supporting multi-step, complex queries that require both database computations and LLM reasoning. It allows the TAG framework to go beyond standard SQL or retrieval-augmented methods, delivering more comprehensive answers that are grounded in both data and external knowledge.

More on LOTUS can be found here: https://github.com/TAG-Research/lotus

There are some excellent examples in the GitHub link on how to use LOTUS for performing semantic joins. I'm planning to try it out soon — stay tuned for more in a future post!

Thanks for reading :)

In Plain English 🚀

Thank you for being a part of the **In Plain English** *community! Before you go:*

- Be sure to **clap** and **follow** the writer 👏
- Follow us: **X** | **LinkedIn** | **YouTube** | **Discord** | **Newsletter**
- Visit our other platforms: **CoFeed** | **Differ**
- More content at **PlainEnglish.io**

Genai Llm Machine Learning AI Sql

Do Not Use LLM or Generative AI For These Use Cases

Choose correct AI techniques for the right use case families

7 min read

Alberto Romero

DeepSeek Is Chinese But Its AI Models Are From Another Planet

OpenAI and the US are in deep trouble

13 min read

Jim Clyde Monge

How To Install And Use DeepSeek R-1 In Your Local PC

Here's a step-by-step guide on how you can run DeepSeek R-1 on your local machine even without internet connection.

7 min read

Kenny Vaneetvelde

Want to Build AI Agents? Tired of LangChain, CrewAI, AutoGen & Other AI Frameworks? Read this!

Frameworks like LangChain, CrewAI, and AutoGen have gained popularity by promising high-level abstractions for building AI systems. Yet…

17 min read