aabhra

Reinventing analytic systems to unify and analyze the world's data, Part 3

Quantitative analysis using ontology-grounded, fine-tuned LLMs

Partha Nageswaran, Founder & CEO, Aabhra Inc
Amit Chaudhari, Technical Lead, Aabhra Inc

Introduction:

Large Language Models (LLMs) can be used as an interface for natural language (NL) queries of any complexity. However, LLMs alone do not understand the full semantics of data used to answer the queries. Quantitative analysis of cross industry data, including time-series data, being expressed in NL and being executed in a timely and accurate manner is still a far cry.

A novel solution to this problem is to ground LLMs using ontologies to influence a more structured and semantically-grounded reasoning approach to perform analysis of any complexity. Combined with a two-step generative process (discussed below) including ontological grounding, fine-tuning and chain of thought approaches, at Aabhra we are incorporating support for NL based enterprise class analysis in our analytics platform that quants, data scientists and analysts are trying on AWS.

LLMs and analytics:

Several approaches to converting NL queries to existing query languages for execution are being researched in academia and the industry. Text to SQL, Retrieval Augmented Generation (RAG), Table Augmented Generation[(1)] (TAG) and other approaches all struggle with converting and executing NL based analytic models accurately, with limited correctness as of now.

Standard challenges include hallucination (production of incorrect or misleading information) and lack of semantic understanding of the NL query and data. These result in generation of inaccurate database or SQL queries, amongst other issues.

To overcome these limitations, efforts such as grounding or RAG (a process of leveraging verifiable source of information to increase the correctness of LLM outputs) and fine-tuning (a technique to adjust a pre-trained model to better suit specific tasks or datasets) are being tried in academic research. Several variations, such as RAG with GraphQL, are also being experimented with, with varying degrees of success, but are ways from being proven to be ready for general quantitative analysis that combines data from across various industries, including time-series data.

While very specific, bespoke analysis of a limited kind can possibly be achieved with some degree of correctness, these approaches are not yet sufficiently correct nor can they handle complex enterprise class analysis. For instance, fine-tuning models specific to domains runs the

risk of causing amnesia or catastrophic forgetting[2] (CF), where the LLMs forget previously learned information while learning new things.

What quants, data scientists and analysts want:

Increasingly, users want to be able to simply express, in NL, complex analytics on any data without having to write code or perform digital janitorial work.

For example, on the lower to medium end of the complexity of analytics, financial analysts want to be able to execute NL queries such as:

- NL Query 1: Probable price movement using Monte Carlo simulations: Compute the probable price movement of opening price of stocks of IBM and VOD based on 20 days of historical data for an observation day that is 100 days ago, using 1000 iterations of Monte Carlo simulation.

- NL Query 2: Correlation analysis, looking for signal opportunities: Calculate the correlation between 100 days of opening price of stocks and the end of day price of bonds for all stocks and bonds issued by IBM, MSFT and ATT.

Adding more complexity, financial analysts want to apply screened universe to queries such as the above ones:

- NL Query 3: Correlation analysis, looking for signal opportunities, with screening: Calculate the correlation between the opening price of stocks and the end of day price of bonds of all stocks and bonds issued by company headquartered in NYC and LN.

From here, they would build models in, for example, a Jupyter NB environment where they compose piece-wise analysis such as the above into larger models and workflows.

What is needed for such NL queries to work:

Approaches such as RAG, TAG and Text to SQL have a tough time understanding the semantics of the queries and certainly of the data that is needed to answer these questions. To try and overcome these limitations, research efforts are considering semantic augmentation in the context of RAG, for instance.

From decades of experiences, we know that semantics are represented in ontologies. Therefore, to correctly interpret any general query of any complexity expressed in NL, at Aabhra we have taken the approach to fine-tune and ground LLMs using ontologies, and optimally use chain-of-thought reasoning in a two-step generation approach.

- **Step 1: NL translation to pure functional query:**

  Rather than go straight to a DB query or SQL, or create domain specific languages (DSL) which are limiting, at Aabhra we have created a pure functional query language (Aabhra QL) that is domain agnostic. By working with separate logical LLM instances, one for converting NL to Aabhra QL using our Sarvam (Sanskrit for “everything”) ontology for grounding and fine-tuning the model, we are able to generate Aabhra QL expressions that accurately reflect NL intent with a high degree of confidence.

- **Step 2: Pure functional query to backend queries:**

  The conversion of the Aabhra QL to DB or SQL queries is then achieved in a second step that leverages output from another logical LLM instance that has been grounded by the same Sarvam ontology. This LLM has been sufficiently fine-tuned to leverage insights about the semantics of the data using their intrinsic genetic properties (DataGenes™) to generate back-end queries and eventually, the right results for the NL queries with a high degree of confidence.

By converting the NL to a pure functional representation and then converting the pure functional representation to backend queries (SQL as of the date of this writing, to begin with), we are well on our way to processing NL queries such as the above on large time-series and non-time-series data. These queries can then be composed to represent analytic models in a Jupyter environment, for instance. *Aabhra support for NL queries will be available in an upcoming release.*

Aabhra allows for the generated query to be tweaked with minimal effort as the query language fosters intent driven analysis without having to code or perform digital janitorial work. In many cases, due to the simplicity of the Aabhra QL, crafting the query directly is almost as easy or sometimes easier than expressing it in NL.

The 1st and 2nd articles in this series (links at the top of this post) explains in reasonable depth how Aabhra QL works and how Aabhra leverages the Sarvam ontology to find, unify and use the right data when executing analytics in semantically accurate ways.

The future of analytics is already here:

To find out more about easily unifying the world's data and performing semantically accurate, intent driven analysis of any complexity using Aabhra on AWS, drop us a note at enquire@aabhra.com.

(1) https://arxiv.org/html/2408.14717v1
(2) https://arxiv.org/abs/2308.08747