June 2024

Introduction to Architecting Solutions With Large Language Models

Martin Higgins

Introduction

Large language models (LLMs) such as OpenAI's GPT, Anthropic's Claude, and Google's Gemini have emerged as a powerful new form of artificial intelligence (AI) with many potential applications. LLMs are complex neural networks trained on a massive volume of text data to predict sequences of words that follow patterns. Despite the apparent simplicity of the approach, emergent behavior allows LLMs to be exceptionally capable, rivaling human intelligence in many tasks.

Insurers are keen to understand how they can leverage LLMs to create solutions that enhance productivity and decision quality across the enterprise. This report provides guidance for insurers on best practices for architecting LLM solutions based on insights from industry experts.

Who Should Read This Report: This report is intended for technology leaders interested in building solutions using LLMs. It is assumed that the reader has already selected an LLM vendor and identified specific use cases for their organization.

Key Findings

- **Retrieval-augmented generation (RAG)** involves building a search/retrieval preprocessor to supply the LLM with relevant knowledge and context. It is the dominant approach for solving business use cases with LLMs.
- **Vector searching**, where objects are converted into high-dimensional "fingerprint" vectors that can be compared for similarity, provides an efficient way to surface the most pertinent information for LLMs in complex use cases.
- **LLMs are expected to become integrated** into development stacks and business applications, allowing rapid delivery of AI-powered solutions by nontechnical staff and profoundly impacting the nature of work.

Background

It is currently neither practical nor necessary for insurance companies to build LLMs. Instead, insurers leverage prebuilt LLMs from vendors such as Anthropic, Google, and OpenAI to support their use cases. The challenge is providing the LLM with the relevant information necessary to support the use case.

This report discusses the next steps after technology selection and use case identification. It provides guidance on how to design, construct, and enhance solutions using LLMs to address the identified use cases effectively. The process of evaluating and selecting an LLM vendor, as well as the methodology for identifying and prioritizing use cases, is covered in a separate Datos Insights' report: Insurer Guide to Enterprise LLM Solutions: Selection Factors and Leading Providers.

Architecting Solutions With LLMs

Deploying LLMs as a chatbot can support some tasks, but many enterprise use cases will require integrating LLMs into existing solutions. LLMs can be accessed from other applications via APIs. Calls to the API can then be made to leverage the power of LLMs for things like underwriting and claims assessment, customer service, and other activities. APIs are generally straightforward to use, requiring only a standard developer skillset. The use of APIs is similar to using the chat interface interactively, with prompts playing a similar role interactively and via the API.

Interacting with LLMs requires what is termed "prompt engineering" to structure the input text to get the most effective results. All LLMs have a transient memory, called a "context window," which stores all information from the conversation history to base its responses on. The larger context windows supported by advanced models allow more contextual accuracy.

Training a Model

LLMs can do a variety of tasks out of the box with minimal additional training, even tasks requiring insurance knowledge. However, for more sophisticated use cases, training and knowledge augmentation is often necessary. This can take a few forms:

- **Prompt engineering:** Prompts are knowledge and instructions for the LLM that encapsulate the information required to perform the required task. Some vendors allow for prompts to be administered centrally so that every LLM session is created with the same base set of prompts.

- **In-context learning:** This is prompt engineering that goes beyond standard prompting. For example, providing an underwriting guide as part of a prompt request will allow the LLM to evaluate a submission in light of that guide. However, it is important to note that the LLM retains no knowledge of the prompts after the session is completed.

- **Fine-tuning:** Fine-tuning (or model-tuning) is a more sophisticated technique where additional training can be layered on top of the base (foundational) model. However, fine-tuning, in general, is best for teaching behaviors, not knowledge. An example might be an agent-facing chatbot where the model needs to respond to requests in very specific ways. Fine-tuning is more labor-intensive than the other options and should be considered when other options can't provide the required results.

Working With Structured Inputs and Output

Typically, end users work with LLMs using natural language text as input and output. This approach has obvious limitations when integrating LLMs into other business systems that don't understand natural language. In these situations, it is more useful to communicate with the LLM in structured formats like JSON or XML, both of which are well supported by leading models.

For simple documents, additional metadata is not necessary for the LLM to understand the format when provided with a sample. However, providing the LLM with the XSD or JSON will allow most LLMs to map and generate documents. Prompting may be needed where mapping isn't clear.

Multimodality

Newer LLMs can also process non-text formats as part of the context. This is called "multimodality." With multimodal LLMs, it is possible to analyze documents, images, and even audio. For example, photos of property, vehicles, and equipment can be uploaded along with text-based documents like PDFs for analysis and processing. Using these capabilities, LLMs can extract information from documents like submissions, loss runs, and inspection reports, even when that document format has not been previously encountered.

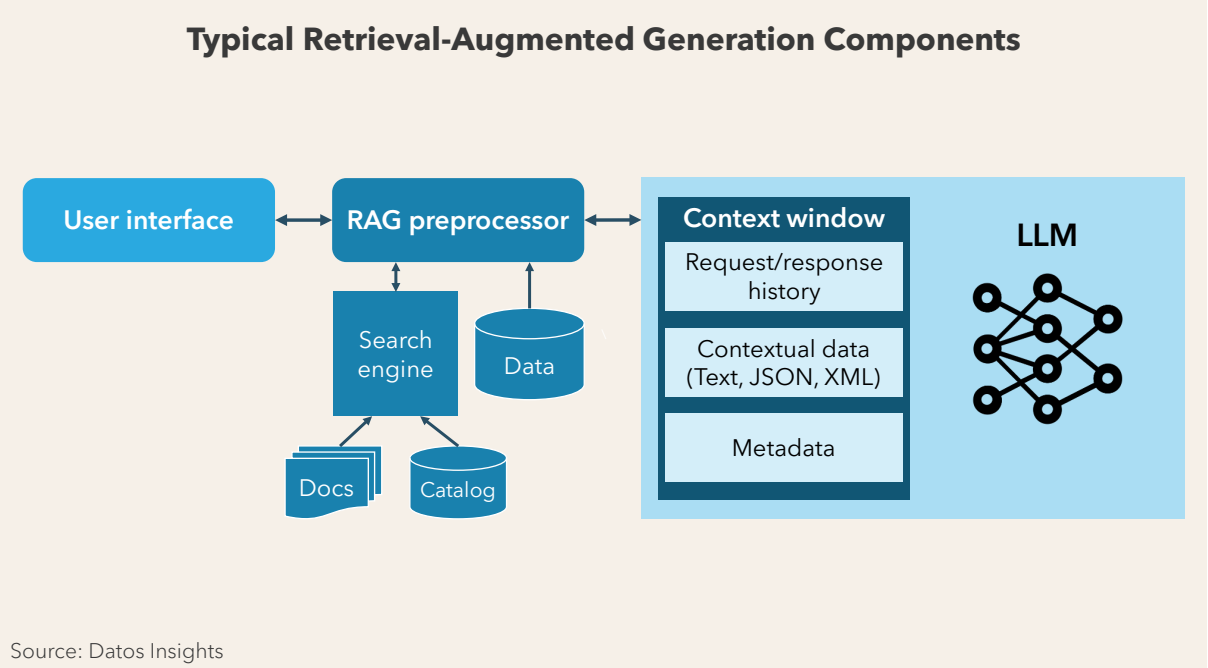
RAG

The dominant approach used to solve support business use cases is RAG. At a high level, it involves building a search/retrieval preprocessor upstream of the LLM, which is responsible for supplying it with knowledge, context, and supplementary information

relevant to the use case. In some cases, the search/retrieval process is straightforward; in other cases, it may require more sophisticated technology like vector databases.

For example, to support an underwriting workbench use case, the preprocessor provides the LLM with third-party data, submission documents, loss-run history, and relevant underwriting guides, allowing the LLM to evaluate a submission with this information. Figure 1 depicts this example use case.

Figure 1: Typical Retrieval-Augmented Generation Components



Source: Datos Insights

The flow of the solution above would be something like the following:

- Email submission is pushed from a submission mailbox to the preprocessor.
- The preprocessor feeds the submission documents to the LLM along with a JSON schema and requests (via a prompt) that the LLM extract the required information from the documents into a JSON document. Prompts can request combining data from multiple loss runs, etc. The prompts also tell the LLM to identify and highlight any documents that could not be read.
- The preprocessor stores JSON in the workbench database.

- The preprocessor interrogates the JSON to determine what supplementary information is required to underwrite and appropriate API calls to fetch it. Depending on the line of business and classes, this could be address validations, vehicle and driver lookups, D&B, social media posts, etc.

- The preprocessor feeds supplementary information, underwriting, and appetite guides to the LLM and, via prompts, asks the LLM to evaluate the submission in the context of these, generate a submission summary, and provide a comprehensive list of all areas of the submission that require underwriter review. The LLM can also be asked to provide an appetite score (based on the appetite guide), an urgency score (based on analysis of the email sentiment), and even a "propensity to bind" score if given the right context information. A scoring methodology can be trained using a prompt technique called "few shot" prompting. The LLM can also be prompted for rationale in scoring to allow the underwriter to validate. This additional information is returned to the user interface in structured JSON format, again generated by the LLM.

- The workbench uses the JSON to prioritize the submission for underwriter review. If the submission falls outside of the appetite, the LLM can be used to generate an email response to the agent explaining why the submission was declined. This can be sent automatically or sent for review to an underwriting assistant who can send it.

- The underwriter opens the submission from their prioritized worklist. They view the information extracted from the submission as well as the summary and analysis provided by the LLM. The underwriter can converse with the LLM via chat since the LLM already has all the submissions and analyses still in context. The underwriter can also interact more formally with the LLM via the workbench user interface, which can issue additional prompts to the LLM behind the scenes to perform common tasks.

It is important to note that in the above, the LLM is invoked with a combination of system-generated prompts and ad hoc, user-generated prompts.

Vector Searching in RAG

An important component of RAG is the retrieval system responsible for finding the information necessary to allow an LLM to support complex use cases. In some cases, such as the above, retrieval may be straightforward. However, in other use cases, the search requirements demand more sophisticated techniques.

One technology with wide adoption in these situations is vector searching. A vector is like a unique fingerprint that captures the key characteristics of an object. With an insurance

claim, for example, the vector might represent the type of incident, the severity of damages, the location, and other relevant factors as numerical values that encode their meaning across hundreds of dimensions.

Once the claims are converted into vectors, users can search for claims that are mathematically similar to a given claim or query. Similarity is measured by how close vectors are to each other in high-dimensional vector space. For example, the words "peril" and "hazard" may have similar vector representations because they are often used in the context of claim processing. Vector similarity search provides an efficient and effective way to surface the most pertinent information for the LLM, ultimately enabling it to tackle complex insurance use cases with a high degree of specificity and nuance.

Implementing an RAG with vector search requires a combination of technologies for storing, indexing, searching, and retrieving relevant information to augment the language model's responses, such as the following:

- A scalable storage system to hold the knowledge base of information from which the model will draw, such as insurance claims data, policy documents, and other relevant sources—this could be a database, data lake, or other storage solution
- A vector search indexing and retrieval system to find the most relevant information for a given query efficiently—popular options include Elasticsearch, Azure AI Search (formerly Cognitive Search), Vespa, and Faiss
- An LLM to generate responses based on the retrieved information

By leveraging a combination of these technologies, it is possible to build an RAG system capable of generating highly relevant and accurate responses to complex insurance queries.

Conclusion

LLMs are a uniquely accessible form of AI in that they require very little expertise to see impressive results. With proper prompting, usable outputs emerge even from free models. Commercial offerings deliver greater accuracy, use case customization, scale, and speed.

Retrieval-augmented generation allows supplying domain documents for better context. Selecting the optimal LLM model involves balancing factors like computing needs, capabilities, and bias mitigation.

Over time, Datos Insights expects LLMs to become integrated into development stacks and business applications, allowing rapid delivery of AI-powered solutions by nontechnical staff. The democratization of AI will have profound impacts on solution development and the nature of work across many industries. Insurers need to begin skill-building and experimentation now to prepare.

About Datos Insights

Datos Insights is an advisory firm providing mission-critical insights on technology, regulations, strategy, and operations to hundreds of banks, insurers, payments providers, and investment firms—as well as the technology and service providers that support them. Comprising former senior technology, strategy, and operations executives as well as experienced researchers and consultants, our experts provide actionable advice to our client base, leveraging deep insights developed via our extensive network of clients and other industry contacts.

Contact

Research, consulting, and events:

sales@aite-novarica.com

Press inquiries:

pr@aite-novarica.com

All other inquiries:

info@aite-novarica.com

Global headquarters:

6 Liberty Square #2779
Boston, MA 02109

www.aite-novarica.com

Author information

Martin Higgins

mhiggins@datos-insights.com

Contributing Authors

Paul Mattern

pmattern@datos-insights.com

Caitlin Simmons

csimmons@datos-insights.com