SingleStore

Demystifying AI With the ICC Framework

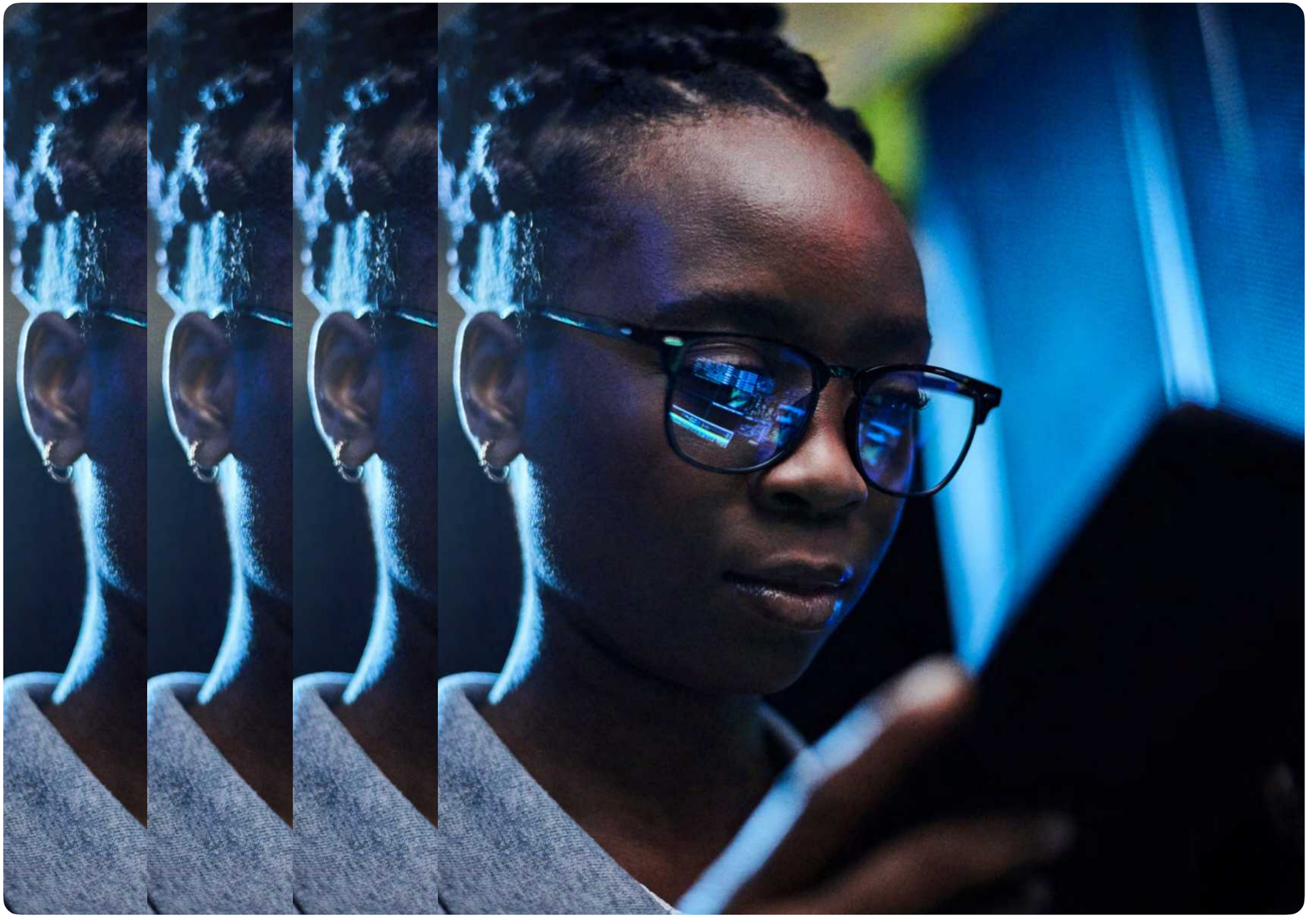
Contents

Chapter 1

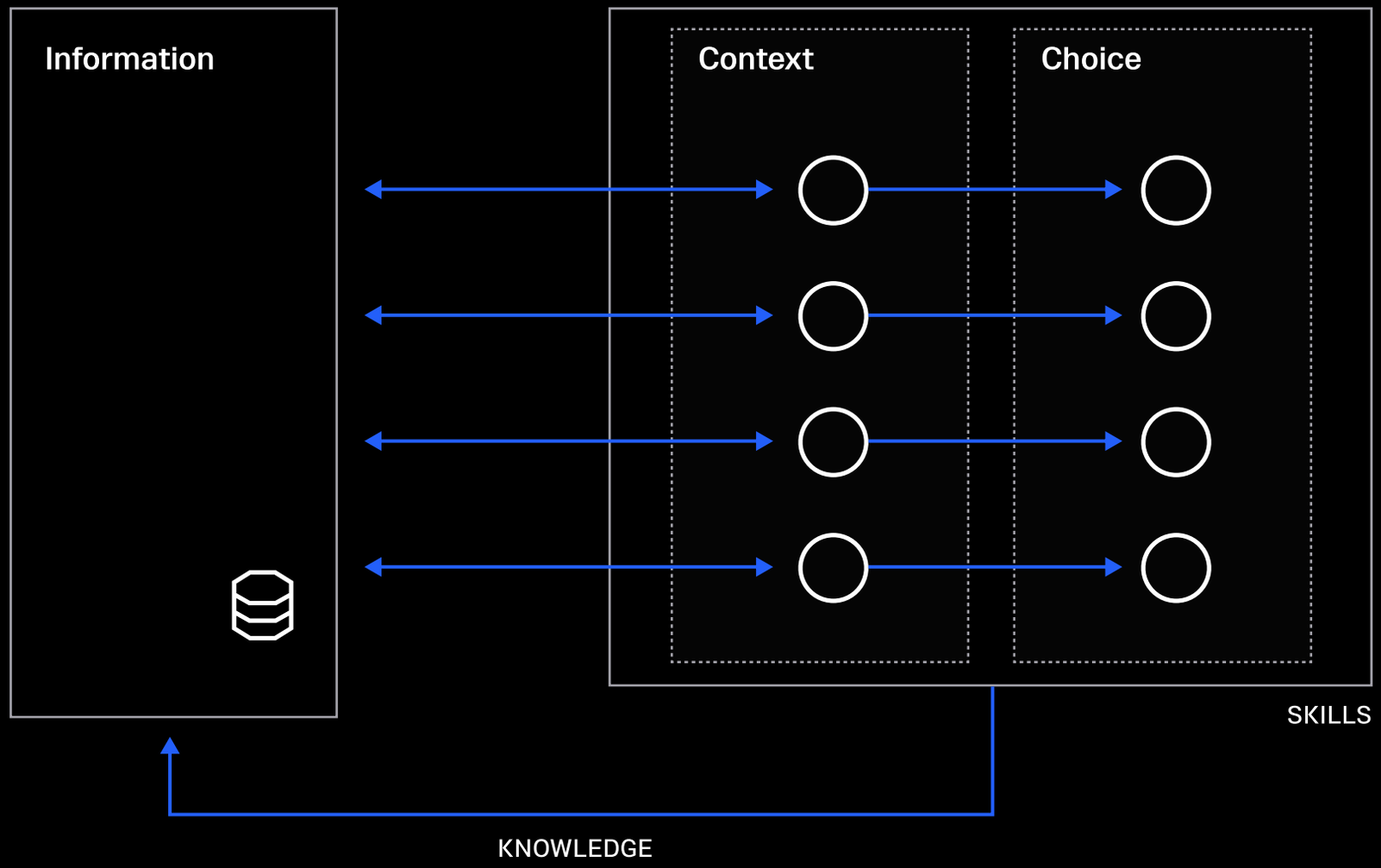
The Trinity of Intelligence (ToI) Framework

In our quest to understand the complex world of Artificial Intelligence (AI), we must first grapple with the concept of intelligence itself. This is true not just for individuals, but also for organizations.

The Trinity of Intelligence Framework, consisting of Information, Context and Choices, offers a lens through which we can examine and comprehend AI.



The Three Pillars of the ToI Framework



Information

Information forms the first pillar of the triad. As intricate intelligent systems, both humans and organizations amass vast quantities of data from inception.

Over time, this information becomes so extensive that we begin to identify patterns and create connections within our memory banks. Yet despite its systematic organization and vast data storage, a physical library lacks intelligence — it cannot acquire new data or connect existing data points based on subject similarity.

In contrast, AI can digitally store and connect data points based on similarity, known as parameters, and the proximity between these points, known as weights. As humans, we collect more data points and connect them together in our memories, creating knowledge. Within Large Language Models (LLMs), the higher the number of parameters, the more “knowledgeable” the LLM is considered due to the higher rate of accuracy in predicting the next word.

It is because of the number of parameters that we currently gauge how much bigger or better a model is versus other models. For example, Meta's Llama has a 70B version, which denotes this particular model was trained on 70 billion data points. It also has a 7B and a 13B version of the same model.

One might ask: why not make extremely large models that are nearly dead accurate in predicting the next data point?

The answer is related to how the prediction is made by learning and mimicking the patterns of data connection within data sets. At its very base, computers use matrix multiplication to train and infer large models. The larger the number of parameters, the longer it takes to do the training.

It turns out that Graphical Processing Units (GPUs) are better at parallel processing, so more and more companies use these to train LLMs.

However, GPUs are relatively scarce and more expensive than the traditional Central Processing Units (CPUs), which consequently means building LLMs is a very expensive and time-consuming endeavor. According to some publicly available data, OpenAI spent over $100 million and several years to build GPT 3.5 based on 175 billion parameters[1].

One of the key things to note about LLMs is they are trained on data that is frozen in time. For example, ChatGPT was trained on publicly available data up to September 2021. So how do you make an LLM aware of your data — especially data in companies that are not publicly available, and the incessant streams of data produced every millisecond?

To effectively use LLMs for private and newly generated data, organizations have two options: fine-tune LLMs and/or use a pattern called Retrieval Augmented Generation (RAG) to provide contextual information to the models at the time a query is asked.

This brings us to the next pillar of the ToI framework.

[1] https://www.wired.com/story/openai-ceo-sam-altman-the-age-of-giant-ai-models-is-already-over/

Context

The second pillar, Context, is essential for informed decision making. We are constantly bombarded with data, but it's our ability to connect this incoming data with existing information in real time that makes it valuable — and more importantly, actionable.

For example, we may not pay attention to all the sounds, smells and visuals while walking in a crowd, but a sudden moment with a familiar food smell connecting to a memory of my favorite childhood dish may send me looking for the source to act on this new information.

Context is critical for making decisions, but it also shapes how we store information and connect it to other data points in our memory banks.

LLMs are similar to this because when you provide the right context, they generate far more meaningful responses than a query that lacks context. When context is missing, an LLM often generates a response that is syntactically correct but lacks accuracy or even meaning. This concept is called “hallucinations,” and when it comes to corporate decision making, either not getting the right context or the inability to get context in real time could be disastrous for making decisions.

Choice

When information and context come together, we make choices that lead to action. Repeated action leads to building skills, which is often referred to as muscle memory.

For example, using my memory or information I have stored about a basketball game, I can act and notice a pattern of outcomes. A good outcome helps us store that information with context, so there is more incentive to repeat those actions based on patterns. Over time, this becomes our skill.

In the world of LLMs, the most common action they perform is to generate the next word (and the word after) based on the data patterns and connections it has been trained on. When you train the model on a different dataset — for example, all publicly available paintings or pictures — the model predicts the next data point and the one after to mimic the pattern from the original data set. This is how some models have the skills to generate content, while others have the ability to generate images, or fill an image with additional objects — even creating videos and audio based on instructions.

To illustrate the ToI Framework's potential, consider a fictional multinational corporation: Acme Bank.

At the information stage, the platform processes copious amounts of financial, operational and customer data, identifying subtle correlations and nascent trends. Next, as the bank grapples with the nuances of globalization, the platform's contextual layer tailors insights into regional markets' unique preferences and regulatory environments.

Ultimately, the final touchstone of choice emerges as Acme Bank leverages AI-powered recommendations, automating logistics, elucidating marketing endeavors and enhancing its financial outlook. All three pillars — Information, Context and Choice — play integral roles in the bank’s transformative journey, underscoring the ToI Framework's relevance in guiding businesses through the AI landscape.

In embracing the foundational pillars of Information, Context and Choice, the ToI Framework delineates a multidimensional path for translating AI potential into tangible outcomes. Seizing the opportunities that lie at the intersection of these pillars,

organizations can unravel the enigmatic tapestry of artificial intelligence and devise strategies that harmonize technology and human intelligence. As the framework illuminates the pathway toward intelligent, context-aware decision making, it foreshadows the evolution of businesses into resilient, adaptive and competitive entities that wield AI as both compass and catalyst.

Now, let's look at an LLM. An LLM is a model (typically a binary file, if you are curious) that has stored all data points (parameters) and connected the similar ones with weights. When asked to answer a question about the data points, the LLM tries to answer the question by predicting the next word (it is a language model, after all) by mimicking the pattern in the data connections. As humans, we have been trying to build Machine Learning (ML) models like this for a very long time. More recently, OpenAI added some additional steps that made a quantum leap in the use of these models.

First, they used an extremely large dataset for training. ChatGPT 3.5 was trained on 175 billion parameters. Second, they used a mechanism called Reinforcement Learning with Human Feedback (RLHF) to further refine the responses generated by the models. This process is also called alignment, as it "teaches" the model what is considered a good response and what is not. The use of these mechanisms led to a major breakthrough in this new kind of model called transformers.

By doing this, LLMs have developed a few key skills that can be applied to a wide range of use cases. These skills include Generation, Summarization, Translation and Analysis.

Generation
This is essentially the completion of words and practically any data based on the data it has been trained on. This means we can use LLMs for generating content, articles, images, sound, videos and, very importantly, code. With code generation, we now use LLMs to build applications.

Summarization
This can be used for taking a large amount of data and condensing it to a few data points, without losing any meaning to the entire data.

Translation
On the other hand, translation can be used to translate words from one language to another — or from words into images, videos or vice versa.

Analysis
This uses an LLM to analyze patterns in data. This means instead of using an LLM like a knowledge database, we also use it for reasoning. For example, I can give a bunch of text and numbers to an LLM and ask it to do a sentiment analysis, or find insights into a numbers dataset (you can see a demo of this with the OpenAI plugin for SingleStore).

LLMs and Information Management

LLMs are revolutionizing the field of artificial intelligence, providing unprecedented capabilities in understanding, generating and responding to natural language.

At the core of their success lies their proficiency in information management, a vital aspect of thriving in the ever-changing digital landscape.

Information management encompasses the processes through which organizations acquire, organize, store and disseminate data. Effective information management plays a critical role in fostering innovation, improving decision making and optimizing operations. LLMs are poised to usher in a new era of information management by automating tedious processes, identifying hidden insights and augmenting human expertise.

One of the most powerful aspects of LLMs lies in their ability to process massive amounts of data rapidly, wherein their skill to identify patterns and discover meaning comes into play. Businesses can leverage this ability for various tasks, like monitoring and analyzing competitors' social media posts or scouring the web for real-time market trends. In doing so, organizations not only save time and resources but also uncover hidden opportunities and potential threats.

However, what is critical is how we manage the transactional information that is generated at an increasingly fast pace — and over a period of time, the data moves from transactional databases to data lakes and data warehouses for archival and point-in-time batch analytics.

With LLMs, companies now need to consider having a catalog of data across the organization and the ability to look up that data using hybrid searches. As the volume and variety of data grows, it also becomes increasingly important to make sure the patterns (analytics) that emerge from the data are continuously fed and updated in a real-time contextual data source. Not having the ability to do this in real time puts a company in a serious handicap of not being able to take full advantage of LLMs.

It is when we combine the myriad of fast and slow-moving data in real time with additional information or context that LLMs become truly powerful, forming a key pillar of AI strategy that makes organizations either thrive or flounder.

LLMs and Contextual Understanding

The ability to generate coherent and contextually appropriate text has changed the way we think about AI-generated content.

However, understanding how these models harness contextual information to generate such content requires a deep dive into the intricate workings of LLMs — and their interaction with the ToI framework.

Context plays a pivotal role in human communication and decision making. To truly appreciate the capabilities of LLMs, we must first comprehend the mechanisms through which they process contextual information. The crux of any LLM lies in its training process, where it gleans patterns and relationships from vast amounts of textual data. As it ingests and processes the language patterns during training, the model internalizes contextual relationships, which it later uses to generate contextually relevant responses.

To illustrate the power of contextual understanding in LLMs, consider a simple example: A user asks an LLM, "What is the capital of the United States?" Their response is more likely to be "Washington, D.C." than a random city like "Chicago" because the LLM has learned the contextual relationship between capitals and countries. Furthermore, if a user asks about an event that happened in Washington, D.C., LLMs can recognize the need for a response that is specific to that city, using its stored contextual knowledge to tailor its answer accordingly.

One of the most remarkable applications of LLMs in contextual understanding is sentiment analysis. By parsing textual data and discerning contextual cues, LLMs can identify and classify emotions within a passage or a piece of text. This ability is particularly beneficial for companies looking to gauge customer sentiment, evaluate feedback or assess employee morale, all through the power of AI-driven contextual analysis.

Challenges remain, however. One of the critical concerns in the implementation of LLMs is their inability to discern the veracity and age of information. Because LLMs are trained on vast data sources from a point in time, they may inadvertently learn false or misleading contextual relationships — or worse, have outdated information.

Inaccurate information can lead to inappropriate or misleading content generation, necessitating a more prominent human presence to monitor and guide these systems, ensuring both accuracy and ethical appropriateness. This adds both cost and risk.

An effective way to avoid and mitigate this risk is by using a pattern called Retrieval Augmented Generation (RAG), or in-context learning. In this pattern, the user query never goes directly to an LLM that has been trained on data frozen in the past.

Instead, the query is intercepted and first sent to the contextual store previously referenced to see if any new information exists about that query. Here, a combination of keyword and semantic match comes into play. The contextual data returns the freshest and most similar data to the query that came in, rewriting the prompt to the LLMs by adding a lot more context. With more context and new information, the LLM is now able to use the same skills on this curated data to avoid hallucinations and stay within the company usage guardrails.

LLMs and Decision Making

The crux of decision making rests on the ability to process and analyze vast quantities of information to arrive at conclusions aligned with predefined goals and objectives.

However, with the exponential growth in data and the increasing complexity of organizational structures and challenges, sole reliance on human decision making has become inefficient and insufficient. This is where LLMs can play a pivotal role.

LLMs — fitted with an arsenal of contextual understanding and information management capabilities — have the potential to sift through large volumes of data to detect hidden patterns and generate insights that help organizations make more informed decisions. By leveraging LLMs to identify correlations and draw upon contextual information, organizational decision makers gain better visibility into the factors affecting their respective domains. In essence, LLMs can complement human expertise by providing data-driven recommendations to drive business outcomes.

Consider for instance a financial institution looking to optimize its investment portfolio. By training an LLM on historical financial data, market trends and investor behavior, the organization can develop a decision-making model capable of generating investment recommendations that cater to specific risk tolerances and investment horizons. This not only saves time and effort, but also helps minimize human biases and errors that may otherwise affect the decision-making process. The ability to make decisions and execute actions based on responses from LLMs — or an ensemble of LLMs — is what makes an app or a service agentic. The power of LLMs as decision-making tools is further underscored when integrated into existing technological infrastructures. Platforms like customer relationship management systems, supply chain management tools or collaborative software suites can be made even more valuable with the addition of LLM-generated insights. This symbiosis effectively amplifies the capabilities of businesses, empowering decision makers to navigate the increasingly data rich and complex environments they operate in.

How organizations take advantage of LLMs to build intelligent and agentic applications will depend on how quickly they can curate vast amounts of data, feeding that data to an ensemble of LLMs to make quick decisions. We are in the early days of generative AI applications, and even today, there are some incredible use cases that are emerging, which we will look at next.

Chapter 2

Generative AI Use Cases for Enterprises

As we saw earlier based on the Generation, Summarization, Translation and Analysis skill set, we can already do a number of things that were previously harder, more time consuming and expensive to accomplish.

When these applications become agentic, we now also open a spectrum of use cases that did not exist earlier.

First, let's look at some real use cases based on conversations with practitioners from companies who have either already implemented and are in production, or are in the process of building.

Data stories

An application that looks at fast-changing data in a database and generates engaging news stories with images and visuals to get people to act on the information instead of only showing a dashboard; A convenience store chain that uses SingleStoreDB realized that there were patterns in selling insightful data for other store managers to close more sales. They used to send charts and graphs to the store managers by email, and almost none of the store managers paid attention to or acted on those insights. Using LLMs, the store chain switched to generating engaging stories based on day-to-day insights that were highly personalized for each store.

For example, one day, some store managers woke up to a headline story in their email that said Brazilian bread sold out at another store two blocks away, adding that the store could not only restock but also put them in a more visible location to double the sales for the product in the next two days. This had a remarkable impact on the entire chain and, consequently, the sales of a number of different products.

Agentic bots

A prime example of LLMs' use is the ability to converse with prospects and customers in natural language. This concept is not entirely new, but companies are now using LLMs and APIs to make these chatbots even more efficient by adding actions as a skill for these virtual agents. These chatbots are designed to understand the nuances of human language, addressing customer concerns and inquiries effectively. The natural language processing capabilities of LLMs enable chatbots to comprehend complex queries, provide relevant responses and act on user input with minimal friction. With the added capability of executing actions, these LLM-powered chatbots can now be used as specialized bots for specific domains and tasks.

Customer experience and support

With 24/7 availability, LLM-powered chatbots cater to customers across different time zones, reducing waiting times and delivering timely support. These chatbots can also open and close support tickets on behalf of users, provide proactive status reports and assign issues to the right product owner in large organizations to keep the customers happy.

These chatbots can also learn from customer interactions and historical data, allowing them to continuously improve their responses and become more contextually aware of customer needs. This means as they gather more information, chatbots are better equipped to predict customer intent and provide personalized recommendations.

Content localization and translation

Another powerful use case of LLMs is in the localization of content and interactions. Having access to a vast array of linguistic and cultural knowledge, an LLM-driven chatbot can process multiple languages, allowing businesses to engage with customers in their preferred dialect. This removes communication barriers and fosters a more inclusive and satisfying customer experience.

Companies like Google, Microsoft and OpenAI are investing heavily in developing LLM-driven applications that extend far beyond mere chatbot experiences. AI-powered assistants like Siri, Alexa and Google Assistant leverage LLM capabilities to understand user intent and preferences, growing smarter the more they interact with users. From scheduling appointments to providing personalized product recommendations, AI assistants can enhance customer experiences, increasing brand perception and customer satisfaction rates.

HR bot

An internal chatbot that uses company data to answer questions related to HR policies, open HR tickets and provide status on existing requests and service tickets.

Legal and ethics bot

An internal bot connected through Slack that answers questions related to contracts and trademarks before employees take any action.

Sales tech

LLMs can be used to mimic the actions of a Business Development Representative (BDR). These bots have been contextualized using product data and previously recorded conversations between prospects and BDRs. Based on the existing information and new context (what the prospect may be telling the bot), a customer conversation with the goal of signing up or a sale can be initiated between the bot and the prospects. There are some companies, like air.ai, that have also built a voice-based interface so the bot can make phone calls and talk to humans like a BDR.

Marketing tech

Build custom emails and content by matching customer attributes and behavior to product capabilities and requirements. At SingleStore, we go one step further and query our growth database in real time using natural language. For example, we can ask questions like, “Show me the last five users who signed up in the last hour, and the activities they did in our app.” Based on the result, we can then craft a custom message and show it to specific users through in-app notifications.

Financial tech

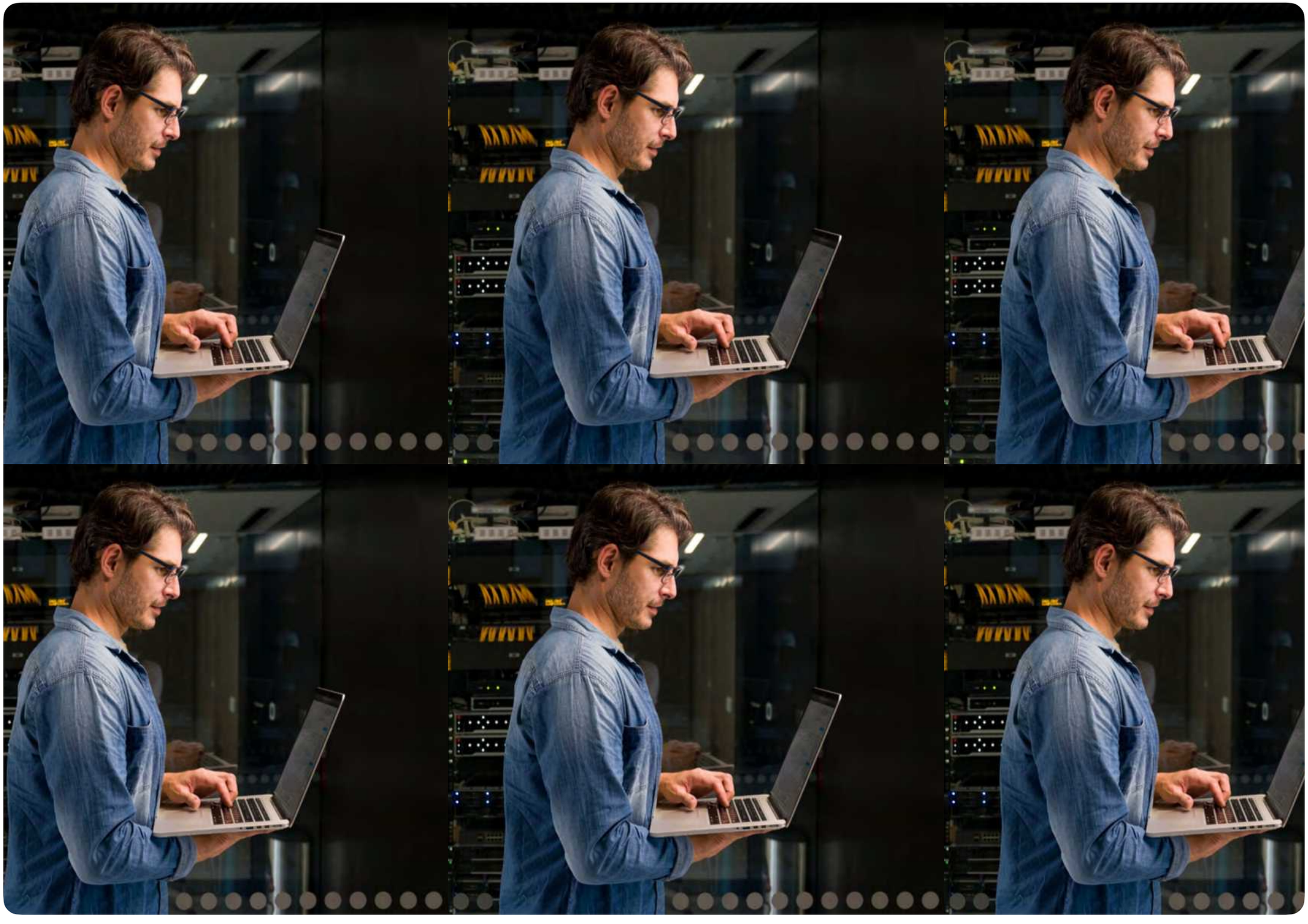
As one can imagine, there are a few different use cases that are emerging within financial tech that rely on LLMs. In terms of knowledge search and management, we see a number of organizations using LLMs to summarize and ask questions about their corpus of data — especially PDFs and internal documents stored in different formats. Another company we work with mimics humans to audit and annotate documents based on the heuristics of accounting practices. In addition, a very large investment management firm is using LLMs to combine their existing corpus of data with real-time stock information to provide guidance and recommendations to agents that manage large personal investment portfolios.

Chapter 3

Decoding AI Terminology and the AI Landscape

The intricate tapestry of AI relies on a myriad of technical terms and concepts, which serve as the building blocks for developing a deeper understanding of the field. To unlock the true potential of AI in the context of the ToI framework, it is essential first to decipher some key terminology and their interwoven connections: LLMs, Weights, Biases, Parameters, Vectors and RAG.

LLMs, like OpenAI's GPT-4, have emerged as essential components in the realm of AI, providing extensive computational capabilities that range from natural language understanding to various digital applications. LLMs serve as the core around which other AI concepts and technologies revolve in the ToI framework. They allow AI systems to process, contextualize and generate human-like language, contributing to the pillars of Information, Context and Choice. An LLM is a generative AI model that has been trained on a very large dataset and has “knowledge” about the pattern of connections between data points so it can predict and generate words, sentences and other data points by predicting the next data points based on user queries.



Tokens

An LLM usually takes an input of words, images, etc., and outputs words, images, audio, etc. The smallest unit of meaning for the words, images, etc., is referred to as a token. For example, one token in OpenAI is roughly equivalent to four English characters.

LLM landscape

There are primarily two big categories of LLMs — proprietary and commercial models, and open-source models.

Weights and biases

In an LLM, the weight refers to the importance of the connection between two data points. Biases, on the other hand, are small adjustments that a model can make to predict or generate based on the query.

Open-source LLMs

A large number of models are now also available for companies to be able to use commercially. The most notable one is Llama 2, which was released by Meta (formerly Facebook). Llama 2 currently has three different versions of the model: 13B, 32B and 70B. Other notable models that were released recently include Orca by Microsoft and Gorilla by students at the University of California, Berkley, who are specifically trained to generate and use APIs.

You can browse, search and download open-source models from a number of different sites, but the one that probably has the largest collection is called Hugging Face.

Proprietary and commercial models

The first one in this category is OpenAI. OpenAI provides several models both for the creation of embeddings (vectors) and for the generation of different kinds of data points — for example, GPT for content, Dalle for images and Whisper for audio. At the time of writing this, OpenAI provides a publicly available chatbot called ChatGPT that allows users to interact with the model in a conversational manner and a set of APIs that can be called by other software programs and libraries.

It is important for an enterprise looking to have a private version of OpenAI or chatbot that is running on a private network and is aware of company data, to know that OpenAI is only available on Microsoft Azure. Currently, there is no way you can run OpenAI on a private network and data if you are on either Google Cloud Platform (GCP) or Amazon Web Services (AWS).

Google offers its own LLM called PaLM2, which can be used to run on your own private network. In addition to PaLM, Google also offers the flexibility to use other LLMs that can be fine-tuned and exposed as APIs that are reachable to services running outside Google through its feature called Model Garden. This means that one could take both PaLM2 and other open-source models available in Model Garden, fine-tune them and run on GCP, but access them over the internet through APIs.

Similar to GCP, AWS has its own model called Titan, and its own set of AI services called AWS Bedrock (similar to Vertex AI). However at the time of writing, Bedrock is not publicly available. AWS also offers another service called SageMaker that is used for building and running your own model. These could be ML models that your company may have built on your own data and are different from the LLMs.

A few other notable commercial models offered by other companies include Anthropic's Claude 2, Cohere, Replicate and Stability AI (Stable Diffusion).

LLM training

Training a model typically refers to feeding it data to understand the pattern among the data connections to make predictions. Re-training an LLM would require having extremely large data sets and a lot of computing power to do matrix multiplications. GPUs and Tensor Processing Units (TPUs) are typically used for these purposes since they are much more efficient in parallel processing. These computing resources are also more expensive than the traditional CPUs.

LLM fine-tuning

Fine-tuning usually refers to adding additional data points or parameters to an existing model to make it "behave" in a certain way. For example, if you wanted an existing LLM to only answer as a pirate (or in a specific manner), you would need to fine-tune the model. Fine-tuning requires using libraries to feed an existing LLM with two different parameters: example query and expected response. Once you feed a large enough dataset, the LLMs are then saved as a fine-tuned model that is used specifically for a purpose. Google and Microsoft both have some low and no-code solutions to fine-tune an existing LLM.

Retrieval Augmented Generation (RAG)

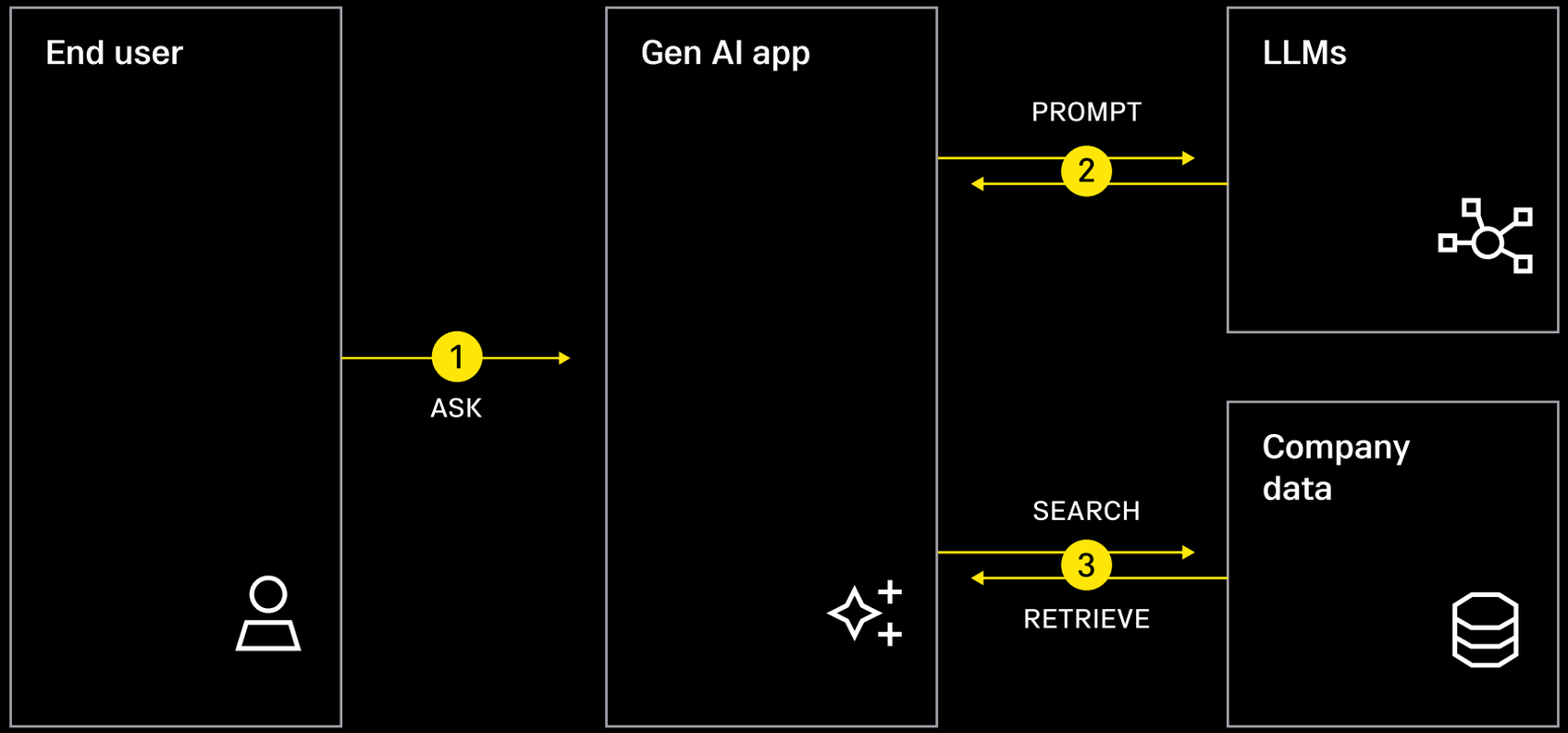
If you are looking to make an LLM aware of your own data, there are typically three things you can do:

1. Retrain the LLM
2. Fine-tune the LLM
3. RAG

RAG is a pattern that intercepts the user query before it goes to an LLM, enriches it with more contextual information retrieved from corporate data and generates a response from the LLM. As an example, imagine a user asks ChatGPT-4, "Who won the gold medal in ice skating at the Winter Olympics in 2022?" Since the model was trained on publicly available data until September 2021, the LLM has no knowledge of that information. With RAG, you can first collect the information from a data source like Wikipedia, convert the data into vectors, do a semantic search on the vectors to find the data most similar to the user query and send that over to the LLM to say, "Here is the user query and here is some data I found in our database that is more similar to the query. Can you now use your skills to answer the user query?"

The following diagram shows the pattern visually.

RAG Pattern


End user
Gen AI app
LLMs
PROMPT
2
1
ASK
Company data
SEARCH
3
RETRIEVE

What should you look for when choosing an LLM for your needs?

Price
How much does it cost to send and receive the tokens?

Context window
How many tokens can be sent to the model in one API call? For example, OpenAI now allows sending up to 32,000 tokens, and Claude2 allows for 100,000 tokens at a time for processing. This means that you could send a very large amount of contextual information to LLMs with large token limits to make them aware of your data in real time.

Ability to run LLMs locally
Are the models you are considering available to run locally, or on your company's private network? This is important if you are building apps that have to be completely "air-gapped" and secure due to data privacy and security issues.

Vectors, embeddings and vector stores

To do a semantic search on data, the data needs to first be converted into a vector representation. One can think of vectors as a numerical representation of a location — like the latitude and longitude of a data point in a virtual, imaginary location, for example [42.343434, 35.3423433].

Now imagine if the number of dimensions is not just latitude and longitude, but it consists of hundreds or thousands of dimensions. In that case, the vector representation can look something like this: [34.343434, 44.34343, 65.343434343, 13.343434, 45.465464, 64.56565, 34.343433] and so on.

This numerical representation is also referred to as embeddings and can be generated against the input of words, documents and large amounts of data by calling embedding model APIs from different generative AI companies like OpenAI and Google. Keep in mind that commercial models used to convert data into vectors and embeddings cost money for each API call.

Semantic search

When an embedding model converts words, images, etc., into vectors, it uses the data it has been trained on to put objects closer to each other that are similar in meaning. For example, when the words "boy" and "prince" are converted to embeddings, they are closer to each other and the word "king." When a semantic search is conducted, the distance between objects (in this case, words) is calculated, and this is how in a semantic search result, we can find out that "husky" is similar to the words "canine" and "animal" — even if these two words were not mentioned in the user search. In the world of semantic search, there are a couple of well-known algorithms that are used to measure similarity: cosine similarity and Euclidean distance.

It is important to take into consideration your existing data, and what kind of queries need to be performed at query time when choosing a vector database.

Vector stores

When objects (words, images, etc.) are converted into vectors or embeddings, they typically need to be stored in a special database that stores the vectors efficiently and searches them using the provided algorithms. Broadly, there are three types of vector stores: vector libraries, vector-only databases and contextual databases.

Vector libraries

These are programming language-based libraries that are open source and can be used by anyone to store the vectors in memory. The most common ones are FAISS by Meta, ANNOY by Spotify and NLSB (supported by Google). However, these libraries store the vectors in memory, which becomes very expensive as the data changes quickly and the size of the data to search goes into terabytes.

Vector-only databases

There are several new vector databases that have emerged that are specialized for storing vectors and doing Approximate Neighbor Searches (ANN). Some examples include Pinecone, Millvus, Weaviate, ChromaDB and Supabase. However, these are not traditional databases, so they lack enterprise-grade features like disaster recovery and extensive metadata filter-based searches. Most importantly, they get very expensive since they lack incremental updates to the data.

Contextual databases

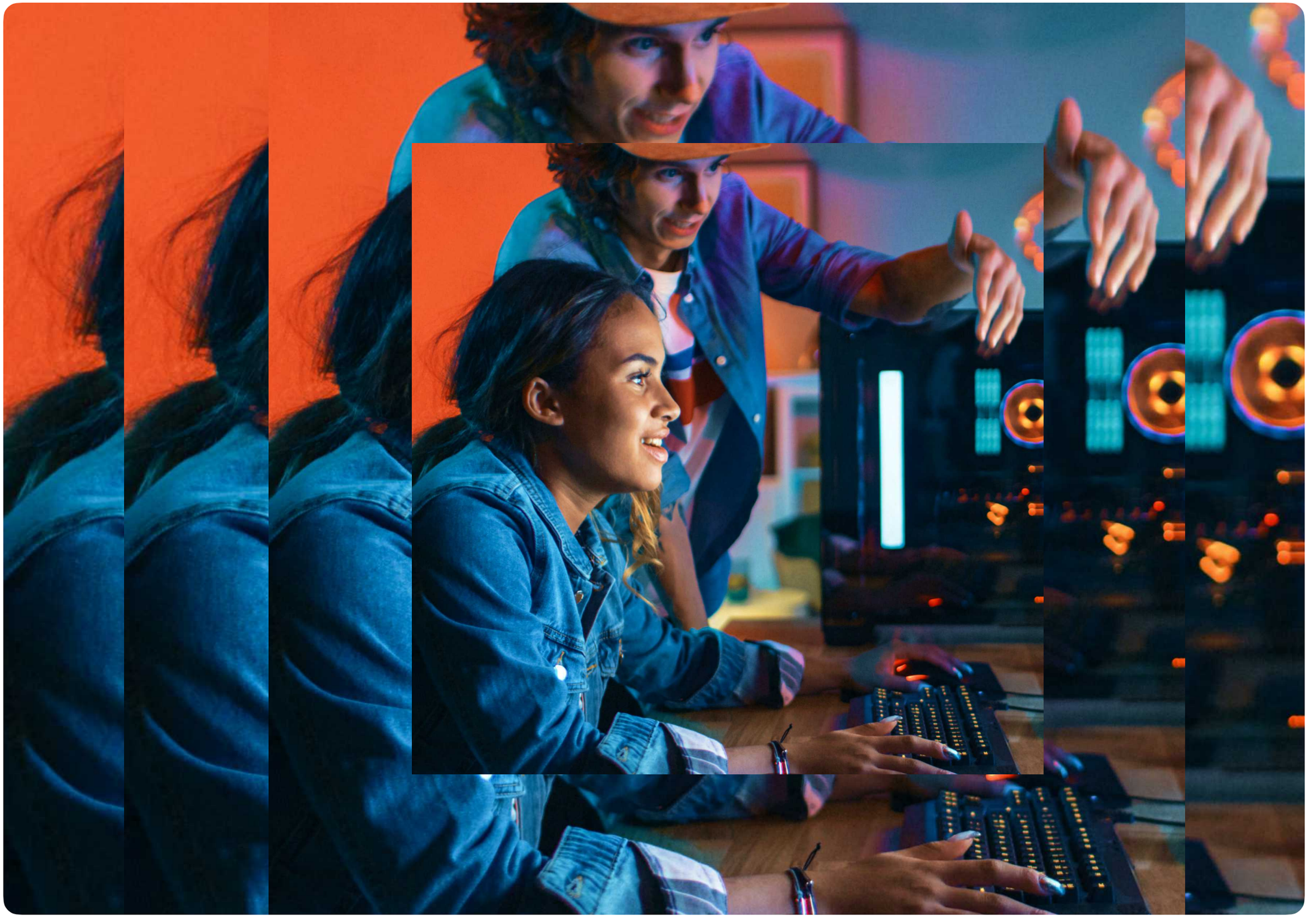
Almost all the databases in the market recently added vector capabilities in their feature sets, from MongoDB to Elastic, PostgreSQL and Databricks to specialty graph-based databases like Neo4J. It is important to note here that SingleStoreDB has had vectors since 2017 when the paper on transformer models was first published by Google. What is different about SingleStoreDB is that you can store vectors alongside all your other data like SQL, JSON and geospatial, so you can do both a keyword and a semantic search in one single query with millisecond response times. This is table stakes for newer use cases where real-time data (Context) is being generated at an incredibly fast rate — and needs to be fed into LLMs to get back responses in a split second.

Let's take a look at a real-life example. Imagine an Amber Alert where time is of the essence, and we have the following data: The make and color of the suspect's car, a picture of the child and one letter on the car's license plate. If you were using a vector library or a vector-only database, you would have to first do a search on the make, model and color in a different database, then vectorize and search the picture. All of this takes valuable resources, especially time and money. With a contextual database like SingleStoreDB, you are able to complete a search of all the data in one single query that gets back the results in a few milliseconds.

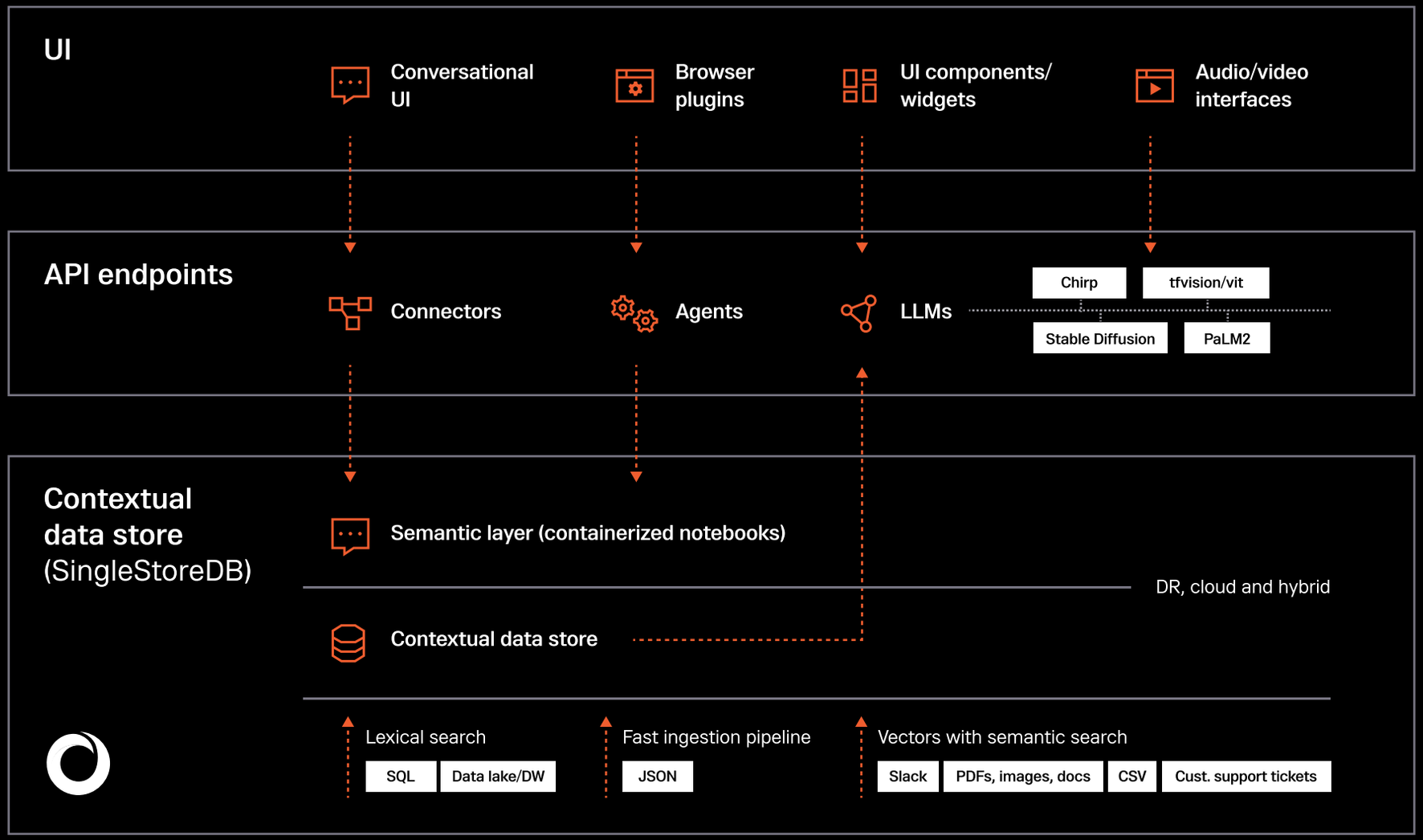
Chapter 4

Building an Enterprise App With the ToI Framework

Using the ToI framework of Information, Context and Choice, we can build a three-layer modular architecture for generative AI applications that not only gives you the flexibility to embed this into your existing application, but also build new ones in a modular and customizable fashion.



The Three-Layer Modular Architecture for Generative AI Applications



Information Layer

This layer consists primarily of assimilating different data points, both transactional and slow-moving analytics data.

The main goal in this layer should be to try and understand patterns in the data, and figure out a strategy of how to take data that is being produced in real time, using that for context in the RAG pattern.

Within SingleStoreDB, you can store multiple different kinds of data and run analytics in a few milliseconds, building SingleStoreDB as a contextual and semantic layer that interfaces with the LLMs (more of this in a bit).

There are a few key things to take into account when building the contextual layer.

Building a data catalog

There are several ways to build a data catalog. But with generative AI, you can now extract and feed metadata to an LLM and optimize the operations, as well as the integrity of the current data. You can then store the catalog in a database like SingleStoreDB, continuously updating it based on triggered changes from downstream data sources with the use of pipelines and external functions.

Data streams and pipeline

Once the catalog has been created, you can add a real-time data stream by adding a pipeline into SingleStoreDB to pull data from a Kafka service.

Contextualizing through vectors

Once the data catalog and the real-time data feeds have been set up, you can then add a layer of vectors for certain kinds of data. To understand what data needs to be vectorized, consider the following factors:

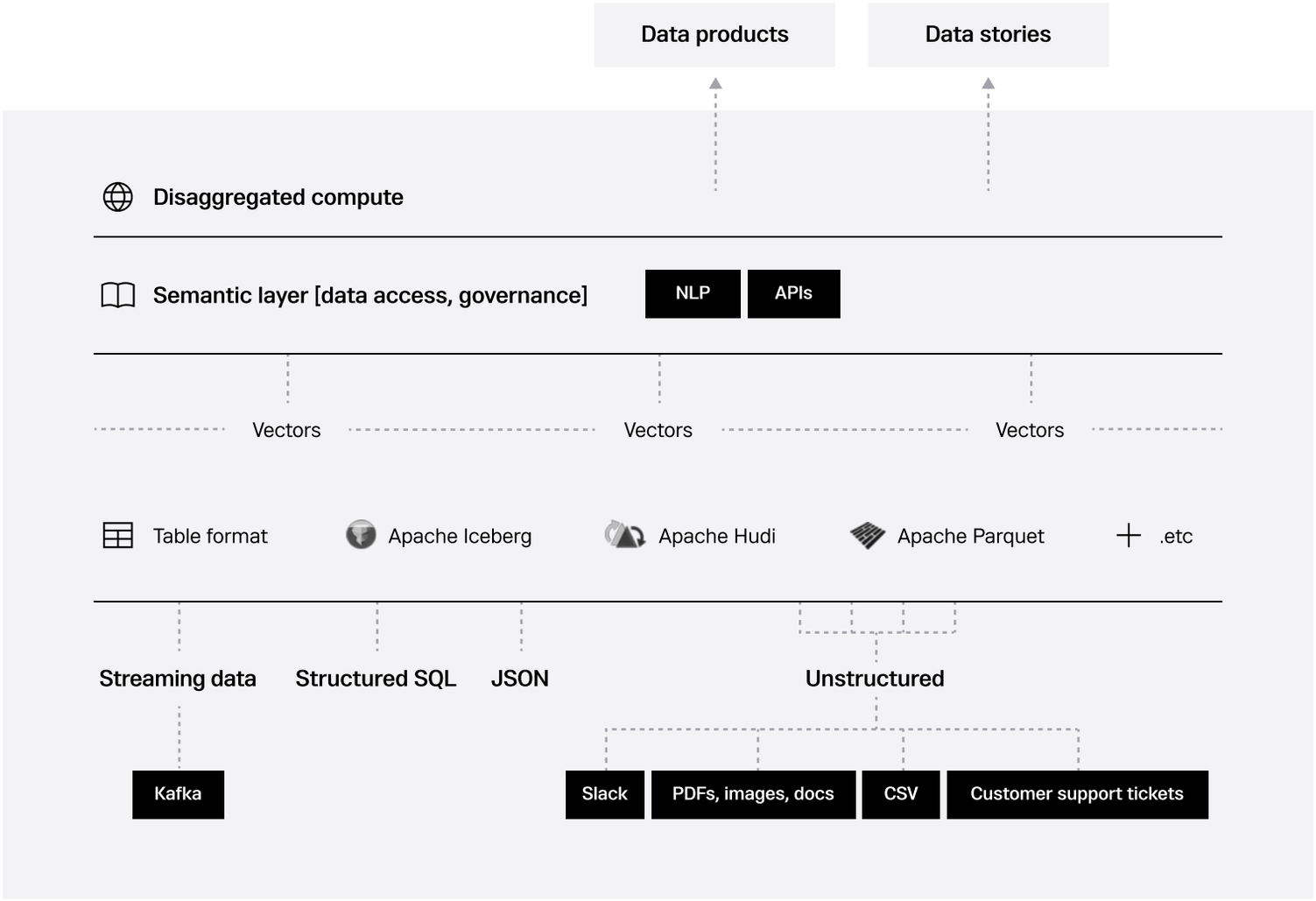
a) If the data is mostly text, image, audio or video based, then you may want to convert it into vectors for future searches. You also need to consider having a mechanism so you can keep these updates based on data changes, and not vectorize the data over and over to keep costs in check. If the data is mostly structured, time-series data or numerical, there is no point in turning it into vectors.

b) If the data is sensitive and should not be accessible to everyone in your organization, then consider putting it in a different database and table altogether, closely managing data access and governance.

c) Finally, only convert data into vectors if you think it is going to be used by the LLMs to perform their tasks.

Once the three steps have been completed, consider taking domains of data and publishing them as API endpoints with CRUD-level access that can be used by other departments, or external customers and users. These are typically called data products. In addition, if you are able to expose a semantic layer to the data products, people can interface with these in natural language — becoming a powerful self-serve semantic layer for your users.

A representation of this layer is visualized in the following diagram.


Data products
Data stories
Disaggregated compute
Semantic layer [data access, governance]
NLP
APIs
Vectors
Vectors
Vectors
Table format
Apache Iceberg
Apache Hudi
Apache Parquet
.etc
Streaming data
Structured SQL
JSON
Unstructured
Kafka
Slack
PDFs, images, docs
CSV
Customer support tickets

Prefrontal Cortex Layer for Decision Making

This layer typically consists of an ensemble of LLMs dedicated to different tasks.

There are several platforms that already exist to help you choose different LLMs, fine-tune them for specific tasks and expose them as APIs. With the use of open-source libraries like LangChain, LangSmith and RAGAs, this layer now becomes the decision-making engine. It can chain multiple prompts from different LLMs, connect to different systems and talk to the contextual layer to get and add context to real-time queries.

Here are some platforms that offer consolidated information and AI decision-making layers that you can consider for your own efforts, depending on your needs.

IBM Watson

IBM Watson is a platform to build either custom models trained on your data, or LLM-based applications with the use of three modular platform components:

watsonx.ai

This offers a few models that you can use as starters, fine-tuning them further for your company's needs. You can then expose these models as APIs to be used by your services and applications. In addition, watsonx.ai also includes Python Notebooks that you can use as disaggregated compute units to run your workloads. Finally, it also offers data connections to databases, including SingleStoreDB, to read and write data into.

watsonx.data

This is another component of the IBM watsonx platform that enables companies to manage data in lakehouse format, a common pattern to store and manage large amounts of data. It is possible to augment this with a database like SingleStoreDB to add fresh or real-time data to feed into LLMs for contextualization.

watson.governance

This is currently not available at the time of writing, but it promises to be a set of tools that will help companies run AI workloads in production by monitoring and managing all AI activities, mitigating risks and costs across the platform.

Google Vertex AI

Similar to IBM Watson, Google Vertex AI is a platform that offers a consolidated set of tools for companies to build end-to-end generative AI applications and services. It has the following main components:

Model Garden

Vertex AI's Model Garden offers 100+ LLMs across languages, speech and video use cases that can be either used and exposed as APIs in a no-code/low-code way, or can be fine-tuned with ease of use within the browser.

Generative AI Studio

Within this section of Vertex AI, companies can fine-tune a prompt (language, speech or vision-based models) and also get started from a very large catalog of existing fine-tuned prompts under classification, extraction, writing, ideation and other use cases.

In addition, Vertex AI also offers components like Feature Store for building custom ML models, datasets for training and fine-tuning models and a set of tools to deploy the applications to Google Cloud.

A few other services to consider while building an entire stack in the ToI framework are Microsoft's Azure, AWS Bedrock (not generally available at the time of writing), Cohere and Anthropic, to name a few. For all of these services, you can use SingleStoreDB as a contextual data store — either by deploying it on one of three hyperscalers (AWS, Azure and Google) or running it in your own private VPC that is fully managed by your team.

Interface layer

This layer is where user queries come in from, and where either a response or recommendation is sent back to perform an action. The usual interfaces for this layer include a conversational chatbot (similar to ChatGPT) or a browser plugin that acts like a co-pilot — or even an audio/video interface to interact with an "agent."

Chapter 5

Conclusion

The world of generative AI is moving fast, and there are new developments almost every day. The one key thing to keep in mind is that your company's data is crucial to taking advantage of any Large Language Models. As we saw in this book, a good framework to demystify AI is the TOI framework, and it can be effectively used to create and manage applications and services that are both modular and disaggregated — so they can be iterated upon as we see more advances in the different areas of AI.

Prepared by:

Raj Verma
Madhukar Kumar
Adee Feiner
Alex Moore

SingleStore.com