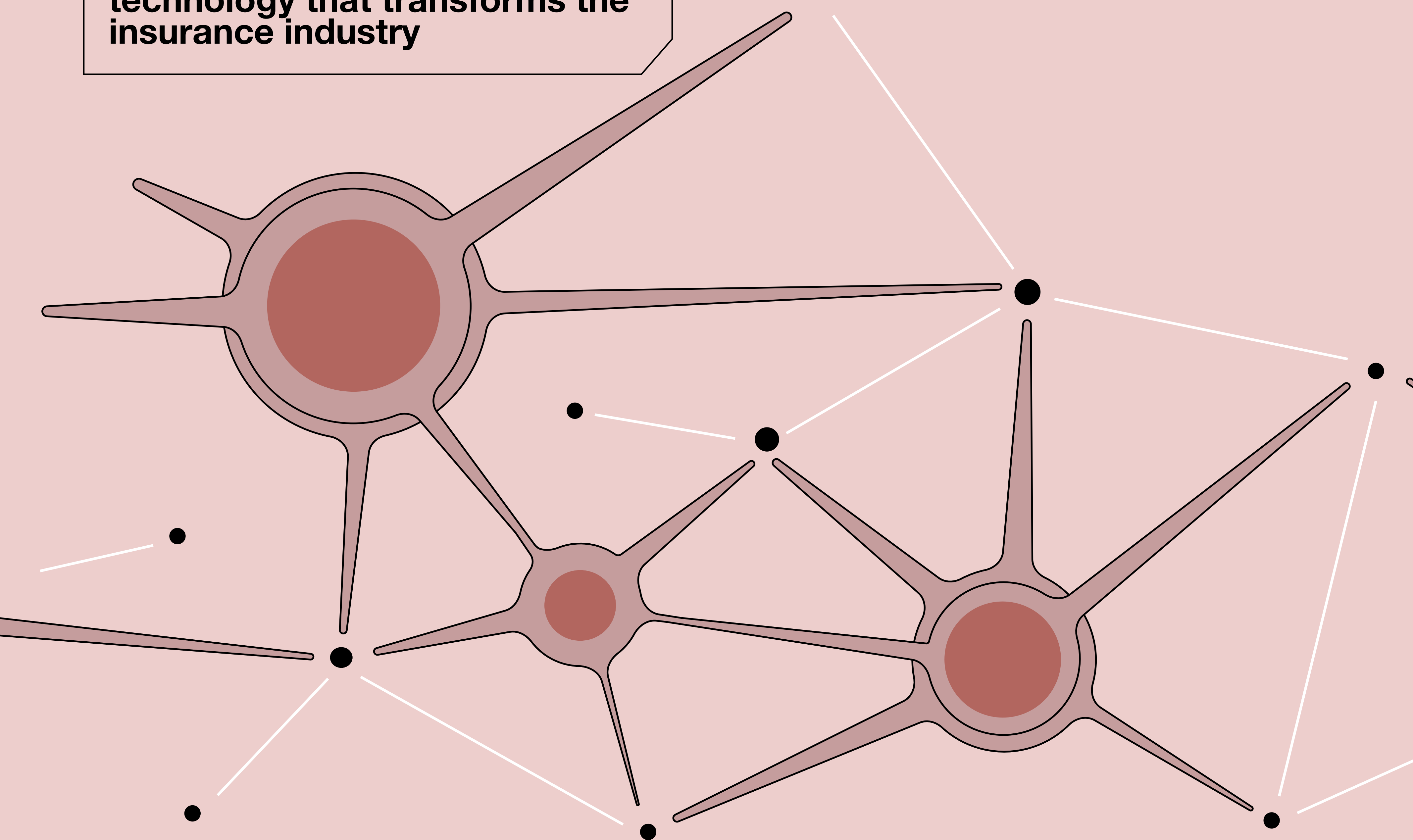


LLMs in insurance: the executive briefing

it's time you finally understand them

all the basics you should know about the **Generative AI** technology that transforms the insurance industry



Key takeaways from this whitepaper:

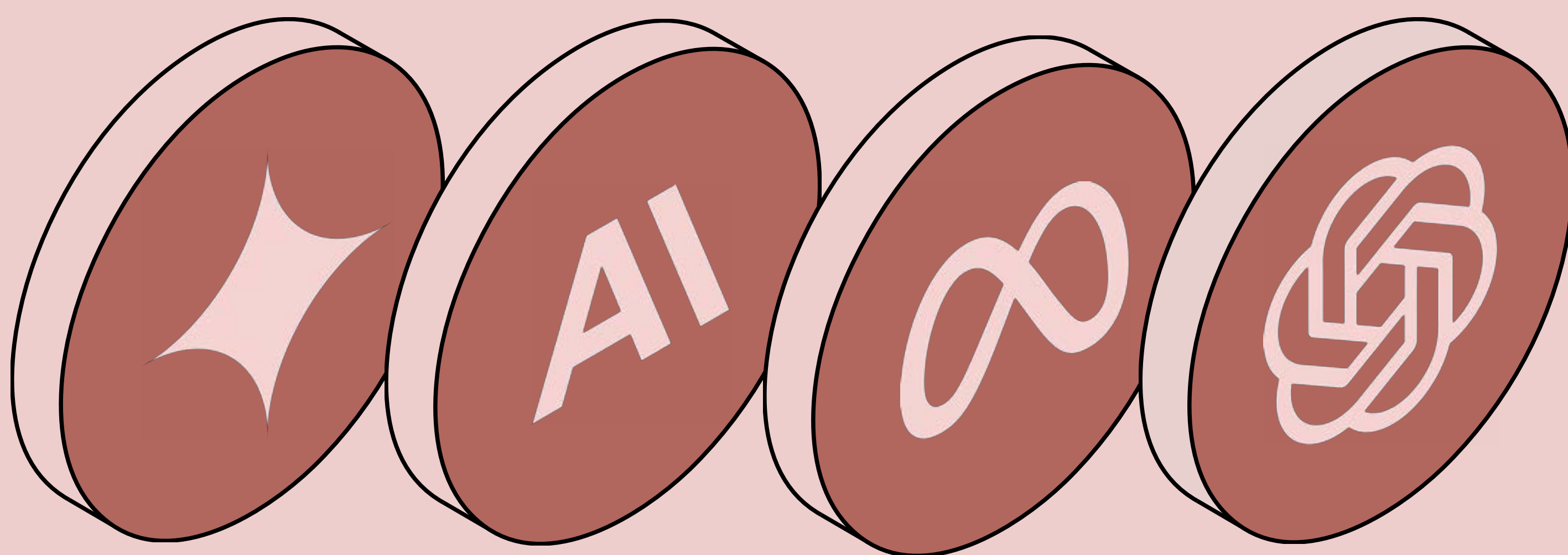
- A tech briefing for insurance business decision makers: understand how LLMs actually work
- Why are LLMs so transformative? A framework to help you think about this new frontier of automation.
- Get a practical view of how modern LLM based systems are built to support your business' unique needs
- Understand reliability concerns and risk mitigation techniques
- A practical framework to set up LLM projects to success

prepare for the ITC by really understanding the technology that everyone is talking about

If you browse the agenda of the 2024 ITC conference, you'll notice that everyone is talking about Generative AI, especially Large Language Models. There are hypes, and then there are technologies that bring sweeping change. Generative AI is not hype—it has the potential to create massive value across numerous business areas. This is a technology that is likely to define the next decade. While it may need some further maturing to be viable for critical business processes, there are already many areas where it's ready for practical applications. Now, you can either leave it to engineers to understand how Large Language Models really work and shape the future of insurance, or you can sit down with a cup of coffee and read through this guide. We guarantee that after reading, you'll better understand the news, the thematic panels at the conference, and be equipped to make more informed decisions about investing in LLM-based systems. Sooner or later, we'll all need to make those decisions—because this AI revolution is coming.



Balazs Fonagy
Chief Strategist
@Supercharge

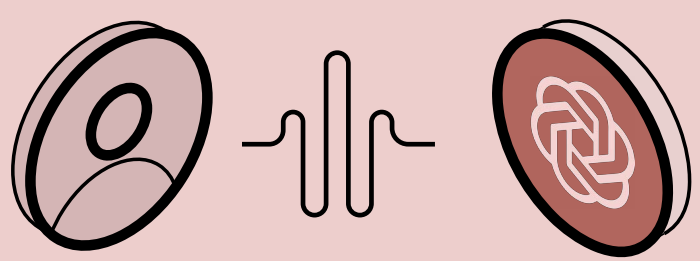


why are Large Language Models (LLMs) revolutionary?

Large Language Models (LLMs) are the branch of Generative AI you're most likely to encounter in your daily life through services like ChatGPT. While image and sound generation offer intriguing possibilities, LLMs' ability to process and generate text makes them particularly impactful for the insurance industry, one where communication and unstructured data is abundant. The best way to think of LLMs is as a new skill that computers can tap into: understanding human language (written or transcribed speech) and generating intelligent responses.

We believe LLMs are truly revolutionary when viewed as a new enabler or skill. They open up countless new use cases in data processing, automation, and human-computer interaction. Instead of focusing on hype cycles and headlines about how much economic value Generative AI has created so far, we urge you to judge a technology's inherent value by asking: "Does it create useful opportunities that were previously unimaginable?" For LLMs, the answer is a resounding yes. While long-term economic impact takes time, LLMs' value is best assessed through their practical applications, which are numerous and impactful.

two revolutions packaged into one technology

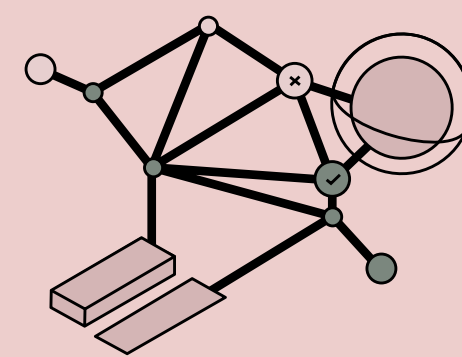


a new era in communicating with machines

Since their invention, communicating with computers has required humans to translate their commands into the machine's abstract language. This process evolved from punch cards to command lines and eventually to today's graphical user interfaces. However, even the most intuitive smartphone interfaces still require users to navigate an abstract world of windows, menus, and icons. In essence, humans must adapt their communication to suit the machine.

Major tech companies attempted to shift this paradigm with natural language-based virtual assistants like Alexa and Siri. Yet these tools have proven inflexible and inconvenient. Traditional chatbots and voice recognition-based call centers so far occasionally helped drive down operational costs but usually led to frustrating customer experiences.

Now, with LLMs, a new era is emerging where computers make a competent effort to understand the often fuzzy and nuanced human language. So far, this mostly takes the form of LLM-based chatbots, but we expect radical breakthroughs in the next few years as LLMs become the interpreter between human and machine. As confirmation of this trend, OpenAI recently announced a new service that enables the creation of real-time speech-to-speech experiences powered by their LLM.



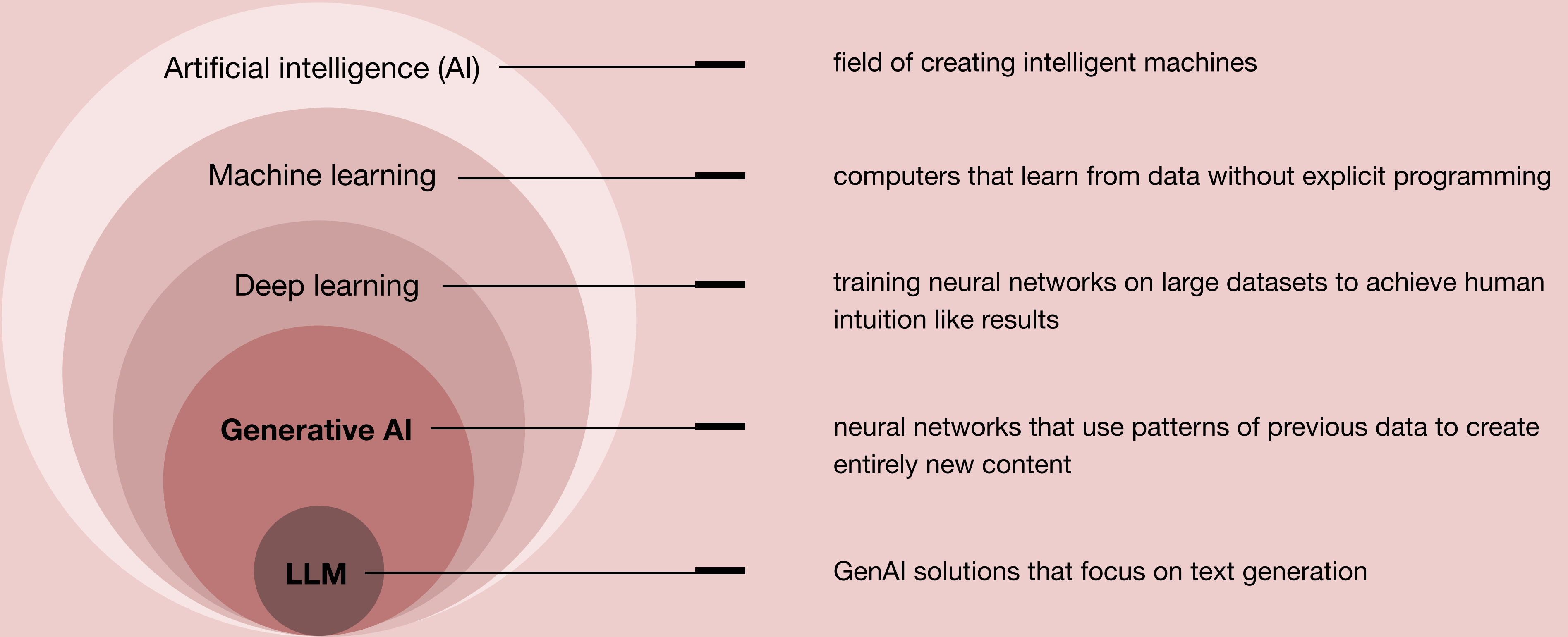
a new era in automating human intelligence

The story of Machine Learning can be framed as an effort to automate processes that do not easily lend themselves to machines or software following strict rules—such as recognizing images or making predictions based on complex patterns. These tasks cannot be broken down into clear, deterministic steps. The ultimate goal is to replicate or even surpass human intelligence by leveraging computers' advanced processing capabilities. LLMs fit seamlessly into this narrative as the latest breakthrough in Machine Learning, enabling the automation of aspects of human intelligence that were once considered uniquely ours: communication and semantic understanding. Practically, we can think of LLMs as systems capable of transforming unstructured data into structured forms—whether it's claim reports, policy documents, or customer complaints.

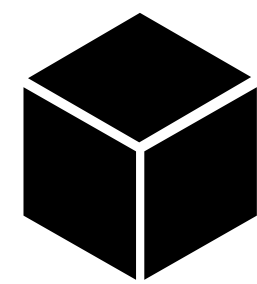
what happens inside the LLM? a super simple explanation

Large Language Models (LLMs) are a type of Generative AI. All GenAI models fall under Deep Learning, the most advanced area of AI. Neural networks with many layers (called ‘deep’ layers, which give Deep Learning its name) are trained on large amounts of text to learn patterns and generate new content. While we won’t go into the details of how neural networks work, it helps to understand that they involve a massive number of complex mathematical operations. During training, these calculations help optimize mathematical functions to learn patterns from huge datasets (like books, articles, and social media content) and produce accurate outputs based on input.

The enormous computational power needed to train these models only became available in the 2020s due to advances in graphics processing units (GPUs). To illustrate the immense computational power required and the rapid pace of innovation in this field, let's consider GPT-4. This model is estimated to have 1.8 trillion parameters— mathematical weights that the model adjusts during training to learn patterns from the data. It was trained over 90 days in 2023 using 25,000 GPUs, with an estimated cost of \$50-100 million. A year later, thanks to more advanced GPUs from Nvidia, the training cost could be reduced by two-thirds.



why do we say it is a blackbox?



The complexity of calculations and their abstract mathematical structure make it impossible for humans to trace back exactly why a model arrived at a certain prediction. Exactly what patterns it picked up during its training and how it arrives at its conclusions remain in the dark recesses of the computer, incomprehensible to the human observer.

imagine the LLM as a text prediction engine

Think of a Large Language Model as a text prediction engine. It takes input text from the user and, based on patterns it learned from training data, predicts what the next word should be.

Without additional post-training refinements (called fine-tuning), the model would merely continue the user’s input—like completing the next sentence of a story. During fine-tuning, models learn to interpret inputs as prompts or user instructions. This enables them to respond in various useful ways instead of just continuing the user’s text: summarizing, translating, explaining, and performing other tasks.

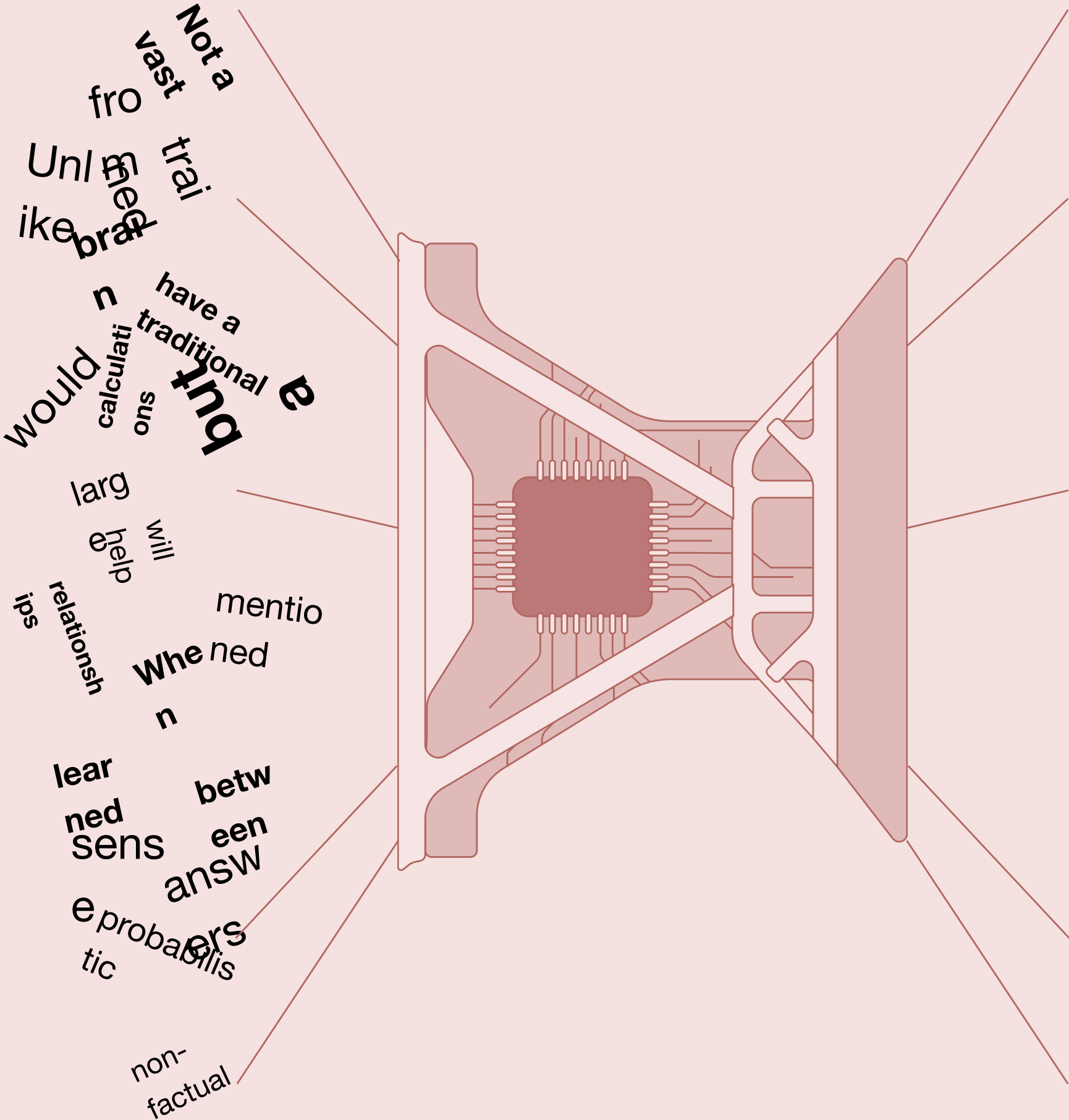
Although their behavior has changed as the result of fine-tuning, LLMs still generate answers through the same statistical process: constructing responses word by word, predicting the most probable next word to create a coherent and meaningful reply. Understanding the inherently probabilistic nature of LLMs helps you grasp some of their limitations.

What is the most famous animal that climbs on trees in the jungle?

The monkey is an **animal**

- animal = 65%
- example = 15%
- insect = 7%
- mammal = 10%
- organism = 3%

The model is choosing the next most likely word for its answer. It considers multiple words with different probabilities.



An LLM is NOT a vast database, but a “brain” that learned billions of patterns

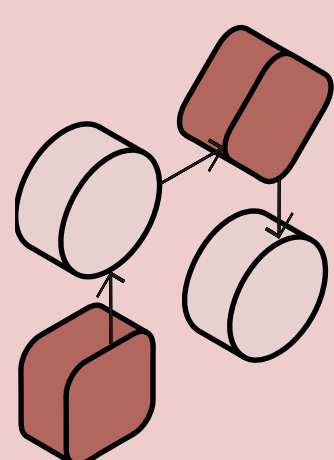
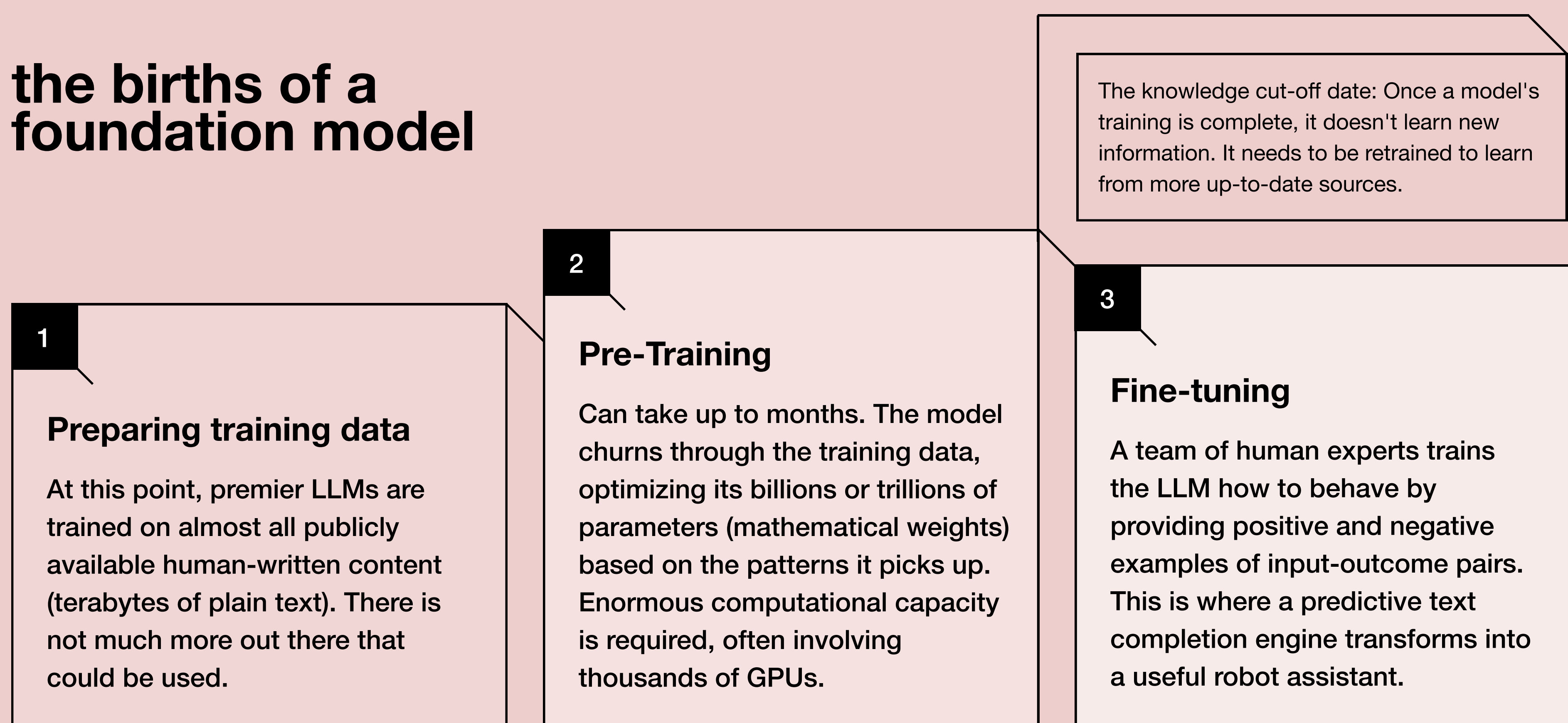
It is important to understand that an LLM does not have a traditional memory or vast database of facts.

When you ask “What is an insurance premium?”, it can’t look up an exact record storing that information. Rather it relies on its internal patterns, parameters, and relationships between words that it learned from text that mention’s insurance, premium and connected concepts. When producing answers, it uses probabilistic calculations to predict what’s the best next word in the sentence based on the context of previous words.

foundation models are what you will build on. what does that mean?

In most real-life LLM projects, the solution will be built on a foundation model. These models are produced by tech companies, spending immense resources on training and fine-tuning them. For most organizations, it wouldn't be realistic to train models by themselves, which is why most decide to pay for a foundation model or use an open-source one. The names that you hear flying around, such as various GPT versions (OpenAI), Gemini (Google), Llama (Meta's open-source model), and Claude (Anthropic), are all such foundation models you can build on. These organizations participate in an arms race, where they keep releasing newer and newer models, trying to outdo the competition in various performance metrics

the births of a foundation model



differences between various LLM models

The LLM market is expanding rapidly, with major developers offering multiple model versions simultaneously. When selecting a model, numerous factors come into play. If you intend to run it on your own, you'll likely opt for an **open-source model** like Meta's Llama. **Pricing** is another crucial consideration, as operational costs for cutting-edge models can escalate significantly as you scale your application. Cloud LLM providers typically charge based on the number of tokens (word chunks) processed, which includes both the input text and the generated output.

While a comprehensive exploration of all aspects of model selection is beyond this guide's scope, we'll introduce you to two key performance indicators you'll definitely encounter when new models are unveiled.

Parameter count: This refers to the number of mathematical weights in the model. A higher parameter count helps the model capture more complex patterns and generate more nuanced, contextually relevant responses. However, more parameters also increase computational costs for both training and use.

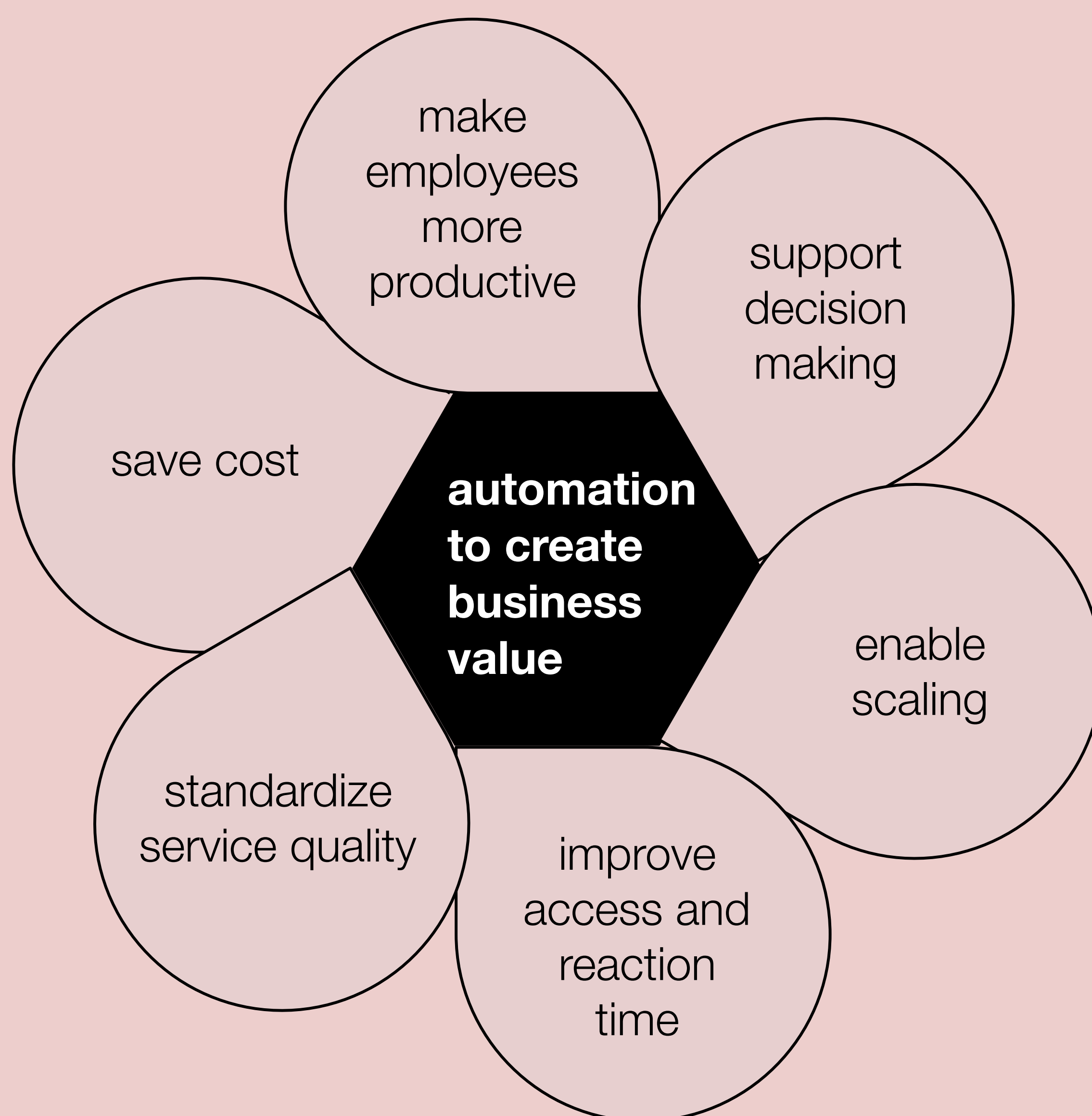
Not every task requires the sophistication of a large model like GPT-4, which has an estimated 1.7 trillion parameters—a smaller, cheaper model can often get the job done.

Context length: Think of this as the model's memory. It represents the maximum number of tokens (word chunks, where 100 words is about 130 tokens) the model can process as input to generate a single response. GPT-4 can handle around 12 pages of text, while its variant, GPT-4 Turbo, extends this to 300 pages. That might seem excessive for a single question, but here's the catch: the input must include all relevant context—chat history, system prompts, and information retrieved from various sources (like RAG and tools). Any information not included in this input will be unknown to the model as it generates its response.

This limitation explains why RAG architectures use external vector databases to store vast knowledge repositories. These databases enable us to input only the relevant information required to answer a question, ensuring it fits within the model's limited context window.

how to use LLMs in your insurance business?

think about LLMs as a massively impactful new building blocks for intelligent automation



At its core, the pursuit of Artificial Intelligence is about automating increasingly sophisticated activities within business processes. Even forecasting and decision-support systems are essentially automating human analysis, supercharged with the massive computational power of computers. The end result is either the **augmentation** of human capabilities or, in some cases, the complete replacement of humans in certain processes through **automated workflows**.

Most AI business cases can be distilled down to three core benefits: machines can perform activities cheaper, faster, or with better outcomes—or any combination of these three. These foundational advantages can be combined to yield more complex benefits such as scalability (cheaper and faster) or process standardization (faster with more consistent outcomes).

As AI technologies advance, the range of processes that can be automated or augmented continually expands. AI becomes capable of replacing more facets of human intelligence or even surpassing it, delivering superior outcomes in areas like pattern recognition within large datasets. You might think of it as an ever-expanding box of building blocks: every new piece (AI capability) allows us to construct more complex and sophisticated systems. And LLMs are extremely important new components, redefining what can be automated.

the new building block that can replace an other layer of human intelligence

We find it a useful framework to think about LLMs as a way to automate those layers of human intelligence that feel the most humane and intuitive. Here, we present the most important new capability building blocks that LLMs give you to build into your workflows.

LLMs can turn unstructured data into structured information and summaries

LLMs can take human-generated content—such as written claim reports or transcripts of doctors' visits—and extract information that can be fed into processes requiring structured input. For example, a policyholder's written incident report can be transformed into structured claims data (date, time, location, type of incident, severity, etc.) and fed into a claim management system. Or a patient's complex medical history documented across multiple doctors' notes, lab reports, and hospital transcripts can be turned into a well-structured, chronological timeline of the patient's medical history.

LLMs enable a new information search experience

LLMs act like highly skilled librarians or research assistants, helping humans find answers from vast bodies of knowledge, including thousands of differently formatted and structured documents. Traditionally, humans had to frame queries in ways machines could understand, but with LLMs, the process is more flexible and conversational, similar to asking a human librarian. For example, an adjuster handling a complex commercial claim could quickly find past similar cases and relevant regulatory guidelines on the machinery used onsite at the claimants facilities.

LLMs can emulate human communication

LLMs can be used to replace human representatives in a multitude of use cases, or to turn static content consumption into a human-like conversational experience. This can mean virtual customer service agents that are miles ahead of the inflexible chatbots of yesterday. Another example can be an SME company's conversational onboarding and risk assessment experience.

LLMs can act as reasoning and decision making agents

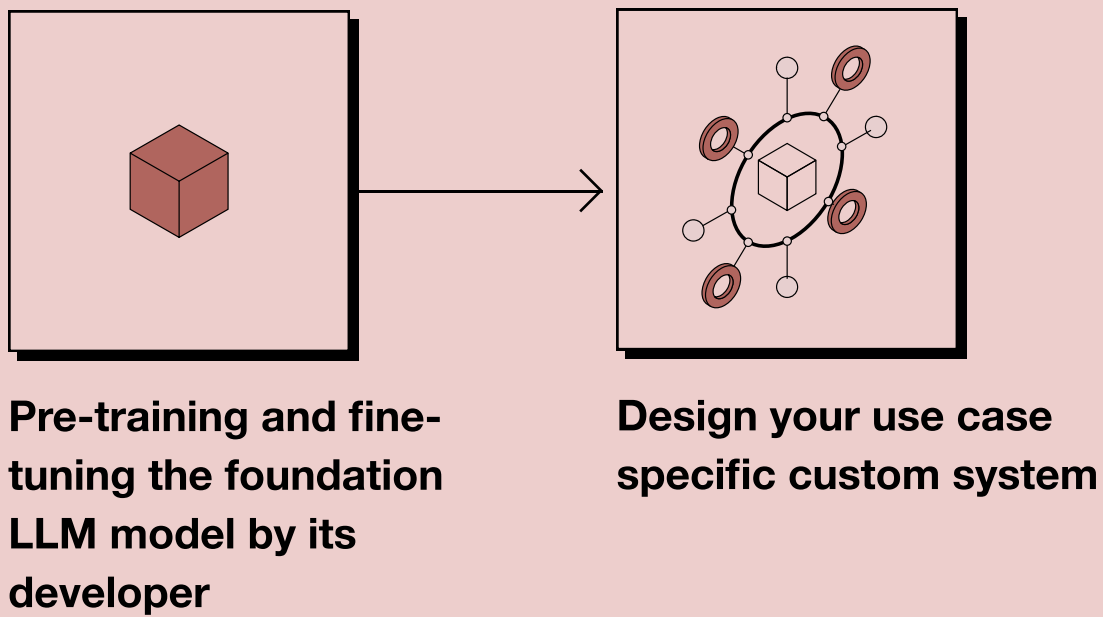
This capability is the most experimental and challenging to reliably harness in LLMs, yet it most closely simulates human intelligence. LLMs can understand context, interpret nuanced information, and perform complex reasoning by deciphering language and meaning. They recognize patterns, draw inferences, generate insights, and even make decisions (such as performing a web search). However, LLM's "black box" nature makes it difficult to use this capability for business-critical applications that require consistent and predictable performance.

Despite these challenges, the potential gains are huge, possibly leading to human-like automation orchestrators. This is why much research is focused on understanding LLMs' internal workings and creating stricter controls and guardrails.

OpenAI is clearly pursuing advanced reasoning, a key feature of their newest model, GPT-o1. It automatically breaks problems into multi-step, chain-of-thought reasoning: understanding the question, structuring the problem, testing strategies, and then summarizing the answer. By also revealing this process, the model's internal logic becomes clearer, reducing its "black box" nature.

It's important to note that the examples we provide would only work if a Large Language Model is used as part of a larger system. This in itself is an important lesson: LLMs are not magical solutions, but rather powerful new building blocks that need to be smartly leveraged within digital tools or data platforms.

how LLMs are customized to your unique use case?



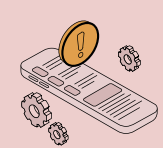
from foundations to a valuable solution for YOUR business

Using ChatGPT or similar models shows the impressive raw power of large language models, but without customization, they are almost never suited for real-world business challenges. It's like hiring a smart graduate who still needs to learn your company's specific processes and information. They need training and context to deliver real value.

Despite Generative AI's rapid evolution, certain solution architectures have begun to gain consistent popularity due to their practicality. In the following pages, we'll explore these options to give you a clear, actionable understanding.

Note: We focus on solutions using cloud bases model-as-a-services and don't delve into the less common case of on-premise LLMs.

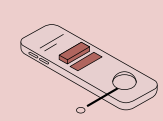
We'll explore the following concepts, ordered by their practical impact in typical use cases, starting with the most significant.



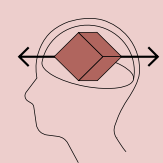
The impactful basics:
Prompt engineering and model settings



Arming the LLM with external knowledge:
Retrieval Augmented Generation (RAG)



Arming the LLM with tools:
LLM Agents



Adapting the model to your context:
Finetuning

Prompt Engineering

How the machine whisperers work.

A **prompt** is a written question or request we give to an LLM. When an LLM produces a response to a prompt, it also takes into account the **context** of the request, that consists of three main elements besides the prompt itself:

- the history of the conversation (think of it as the chat history)
- the system prompt: the written instructions that govern the workings of the LLM
- *information gathered with tools or from embedded documents (we will get there in later sections)*

System prompts are at the core of a custom LLM solution. When someone unfamiliar with the technology first encounters the lengthy instructions behind an LLM performing complex operations, they're often quite surprised. Controlling the LLM you'll find elaborate instructions in plain English. These instructions detail expected outcomes, desired behavior, contextual information and things we explicitly want the model to avoid.

Writing good system prompts requires equal part creativity and engineering mindset. The engineering part comes in with the rigorous evaluation and iteration cycle that leads to an LLM producing accurate results and with the many established prompting techniques the designer can leverage.

Model settings

There are several parameters developers can configure in an LLM, most of which impact its probabilistic behavior and slightly alter its output. One of the most well-known parameters is "temperature." This controls the model's creativity: a value close to 0 makes it consistently choose the most probable next word, resulting in more predictable text. A higher value, closer to 1, allows it to select less likely words, creating more varied and creative results. For business use cases, we typically use very low temperatures to ensure the LLM behaves predictably and reliably.



Instructions (prompt engineering)	<div><div><div>CreateConfigure</div><div><div>Instructions</div><div>Your job is to assist users with insurance-related inquiries across a wide range of insurance topics. You should provide clear, concise, and accurate advice on the following areas:<ul style="list-style-type: none">• Types of insurance (home, auto, health, life, etc.)• Policy comparisons and recommendations based on user input• Claims filing procedures and required documentation• Explanation of premiums, deductibles, and other insurance termsYou should never answer any question outside of these areas. Before answering any question, you must always check the embedded document that contains insurance industry specific information.</div></div><div><div>Conversation starters</div><div><div>What insurance plan is best for me?</div><div>How do I file a claim for an accident?</div><div>What's the difference between comprehensive and liability coverage?</div></div></div><div><div>Knowledge</div><div>If you upload files under Knowledge, conversations with your GPT may include file contents. Files can be downloaded when Code Interpreter is enabled</div><div><div><div>PDF</div><div></div></div><div>Upload files</div></div><div><div>Capabilities</div><div><div><input type="checkbox"/> Web Browsing</div><div><input type="checkbox"/> DALL·E Image Generation</div><div><input type="checkbox"/> Code Interpreter</div></div></div></div></div><div><div>Preview</div><div>Liability coverage is the most basic form of auto insurance, and it's required by law in most places. This type of coverage pays for damages or injuries that you cause to others in an accident, whether it's property damage or medical expenses. However, liability coverage does not protect you or your vehicle; it strictly covers the costs related to the other party involved in the incident. On the other hand, comprehensive coverage goes beyond accidents. It covers damages to your own vehicle from non-collision-related incidents, such as theft, vandalism, natural disasters, or even damage caused by hitting an animal. Essentially, while liability insurance protects you from financial responsibility for damages to others, comprehensive insurance helps protect your own vehicle from a wide range of risks. Comprehensive coverage is especially beneficial if you live in an area prone to natural disasters or high rates of theft or vandalism, as it provides peace of mind that your vehicle is protected against these risks. Additionally, some lenders require comprehensive coverage if you're financing or leasing your vehicle, as it helps safeguard their investment.</div><div>GPT updated</div><div><div>You</div><div>What types of home insurance are available?</div></div><div><div>Insurance Knowledge Bot</div><div>There are several types of home insurance available. The most common include:<ul style="list-style-type: none">• HO-1: Basic coverage, offering limited protection for specific perils like fire or theft.• HO-2: Broad form, covering more perils, including damage from falling objects or weight of ice.• HO-3: Special form, the most popular, which covers your home for all risks except those specifically excluded in the policy.</div></div><div><div>Message Bence Fermentation Chatbot...</div></div></div></div>	Context (chat history + instructions)
Uploaded knowledge (RAG, embedding)		User prompt
Tools & actions (agent)		Response

Prompting techniques

While we won't provide an exhaustive overview of prompting techniques, examining a few popular examples will give you a clearer understanding of how the field works. These techniques highlight the uncanny human-like nature of LLMs - they would be equally effective instructions to help a human intern improve their performance on a task.

Zero Shot Prompting: In zero-shot prompting, the model is given a task without any desirable examples to guide its response.

“Categorize the following claims as either ‘property damage’ or ‘bodily injury’:

1. A driver breaks their arm in a car accident.

2. A tree falls on a house.

3. A pedestrian is hit by a car.

Chain-of-thought prompting:
This technique instructs the model to break down its answer into logical steps, often improving reasoning and accuracy.

“Categorize the following claim: ‘A car runs into a lamppost, causing significant damage to both the car and the lamppost.’

Explain your reasoning step by step before providing the category.”

Model Output:

“Let’s break this down.

1. The claim involves a car hitting a lamppost.

2. ‘Significant damage’ is mentioned, referring to the car and lamppost, which are physical objects.

3. There’s no mention of anyone being injured.

4. Since the damage is only to property, this claim is classified as property damage.

Final category: Property damage.”

OpenAI's newest model, GPT o1, focuses exactly on this kind of step-by-step logical reasoning. This suggests that the industry pioneer believes such an approach could be a promising direction for developing LLMs capable of handling complex human intellectual challenges.

Few Shot Prompting
In few-shot prompting, the model is provided with a few examples of desired or undesired outputs.

“Categorize the following claims as either ‘property damage’ or ‘bodily injury’.

Here are some examples to help you understand the difference.

Example 1: “Claim: A vehicle is damaged after hitting a fence.

Correct Category: Property damage.”

Example 2: “Claim: A passenger sustains whiplash after a rear-end collision.

Correct Category: Bodily injury.”

The idea here is that the model learns from the examples and provides more accurate answers in the future.

10

LLM Agents

using tools opens up the world

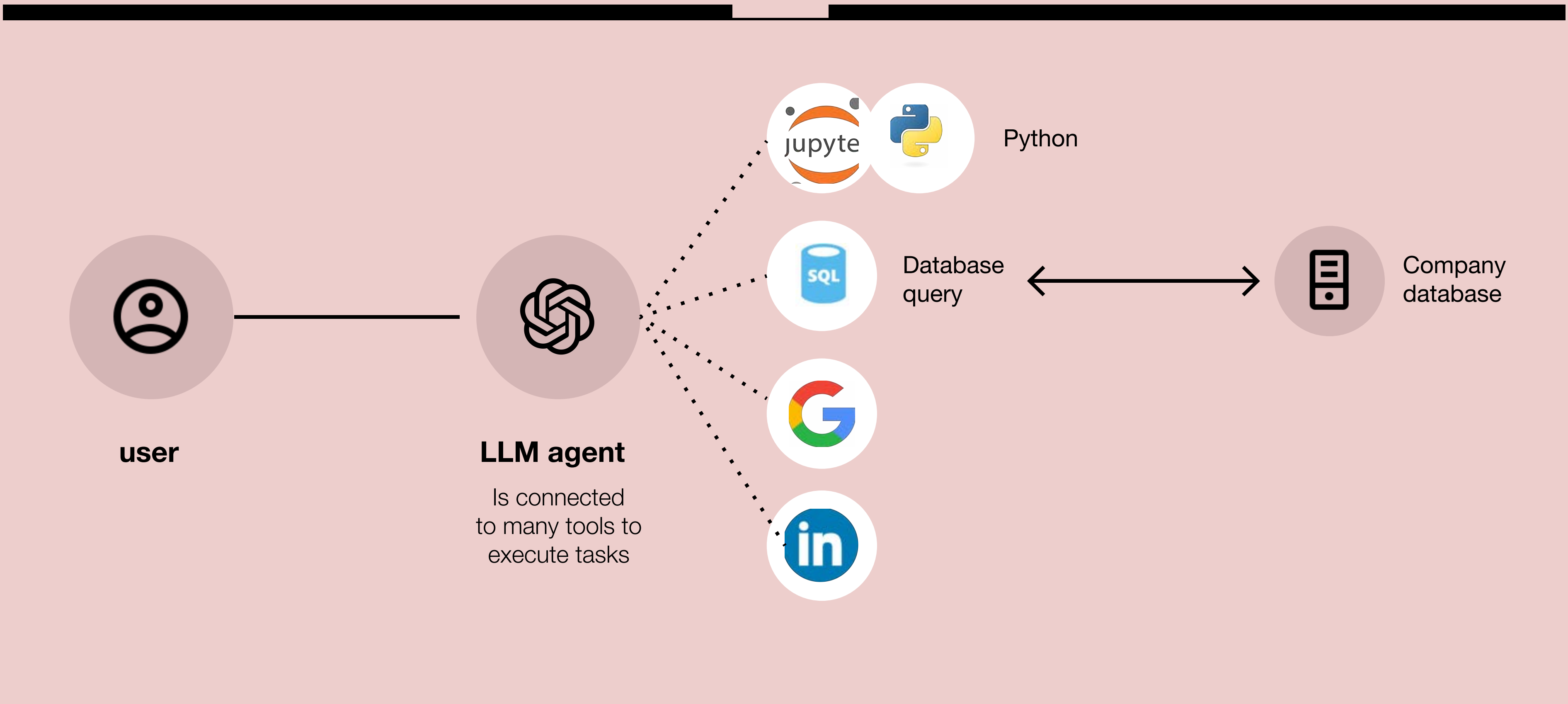
The use of tools was a key turning point in humanity’s evolution from animal to Homo sapiens. Similarly, equipping LLMs (Large Language Models) with tools can massively enhance their usefulness. As the name suggest, the idea is that they become agents, capable of reasoning and acting autonomously to achieve specific goals. In most practical cases their decision-making is strictly limited to selecting the appropriate tool, formulating the right queries, and incorporating the tool’s outputs into their responses.

This area is still highly experimental, some believing that this autonomy can be pushed much further as model’s become better in reasoning and self-reflection. But the use cases we discuss here represent commonly adopted practices in custom LLM-based systems.

How Tools Extend LLM Capabilities

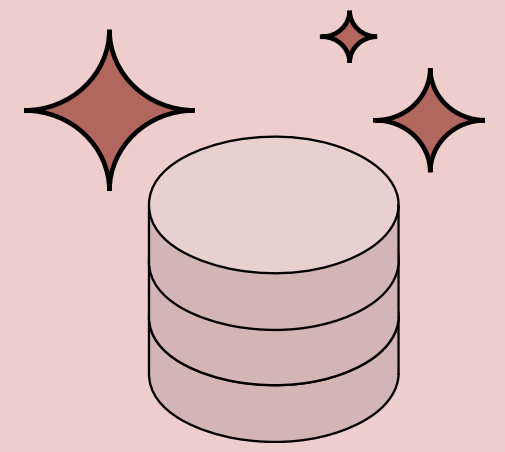
Tools enable LLMs to perform tasks beyond their own capabilities. These are typically applications that the LLM interacts with through APIs. The LLM can often choose from multiple tools based on what appears most effective for achieving its goal. The main practical challenge with LLM Agent systems is their speed: the model may spend considerable time deciding what to do, and then the chosen tool (such as running a Google search) can also take time to execute. This time factor is crucial to consider when evaluating whether an Agent system is a viable solution.

- Here are a few popular examples:
- **Web Search:** LLMs can access up-to-date data by using tools like Google search (even ChatGPT offers this feature). This enables the LLM to search the web and use information found on websites, much like a human would. However, LLM-generated answers based on web searches can sometimes miss the mark because selecting the most credible source and extracting the relevant information is still a challenge for LLMs. A key insight is that LLM accuracy improves with structure: If the LLM is restricted to a pre-vetted range of websites, the results are often more accurate and reliable. The trade-off, of course, is less flexibility for questions that fall outside the scope of those sites.
 - **Access to Public and Internal Databases:** LLMs can be programmed to access and utilize various databases and spreadsheets. For instance a travel insurer’s virtual assistant could retrieve information from a database containing partner hospitals worldwide and their available services.
 - **Access to Specialized Tools:** Certain tasks, such as complex calculations, are best handled by specialized systems rather than LLMs. For example, an LLM might be instructed to use the WolframAlpha API for precise mathematical calculations. An assistant supporting adjusters could be given access to the policy management system—with appropriate access control safeguards in place—to enhance its responses with real-time policyholder data.
 - Additionally, the LLM could be connected to other expert AI systems, consulting them for specialized knowledge in specific domains.



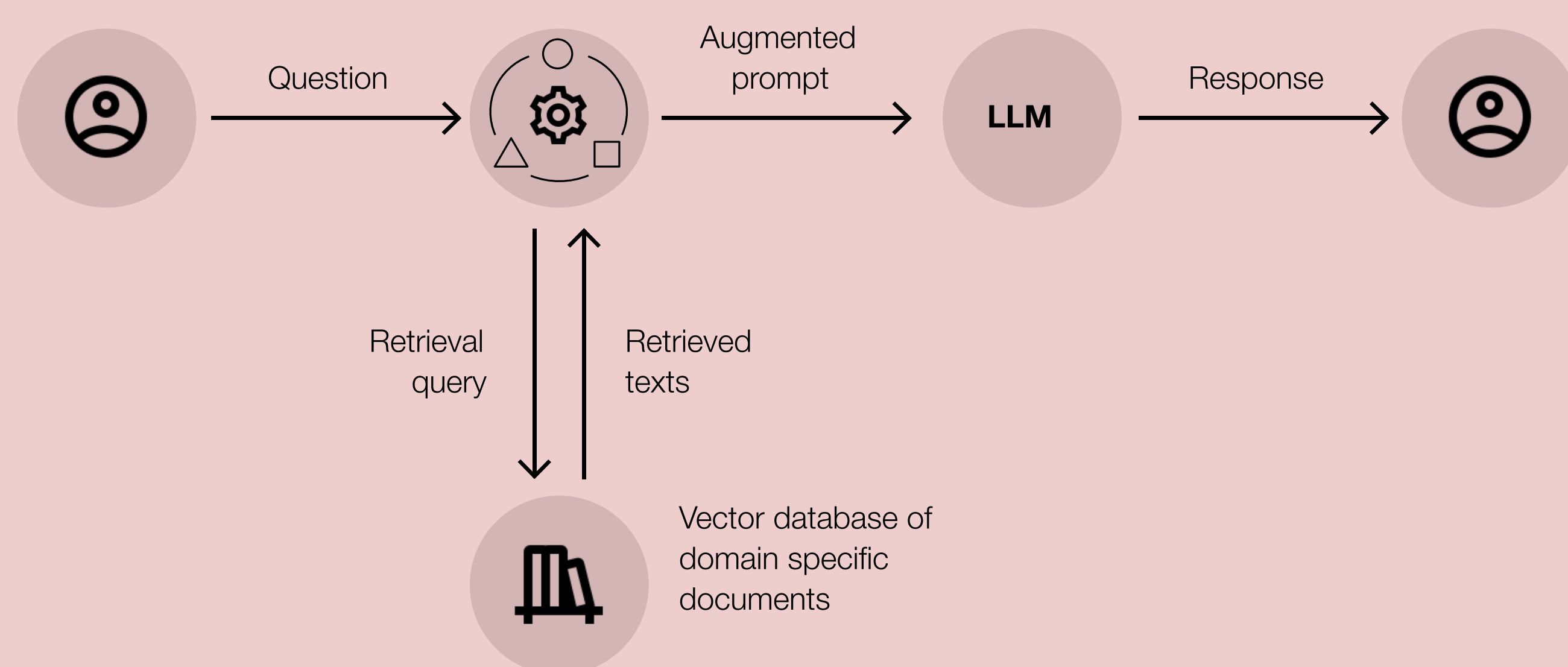
RAG

working from reliable sources of truth



Don't be put off by its technical sounding name: RAG starts to enter the everyday language of talking about LLMs as this is one of the most often used architecture to create reliable systems that can be used to solve real-world business problems. RAG stands for **Retrieval Augmented Generation**: the LLM can search and **retrieve** information from an external vector database and use it to **augment** the input context of the response it **generates**.

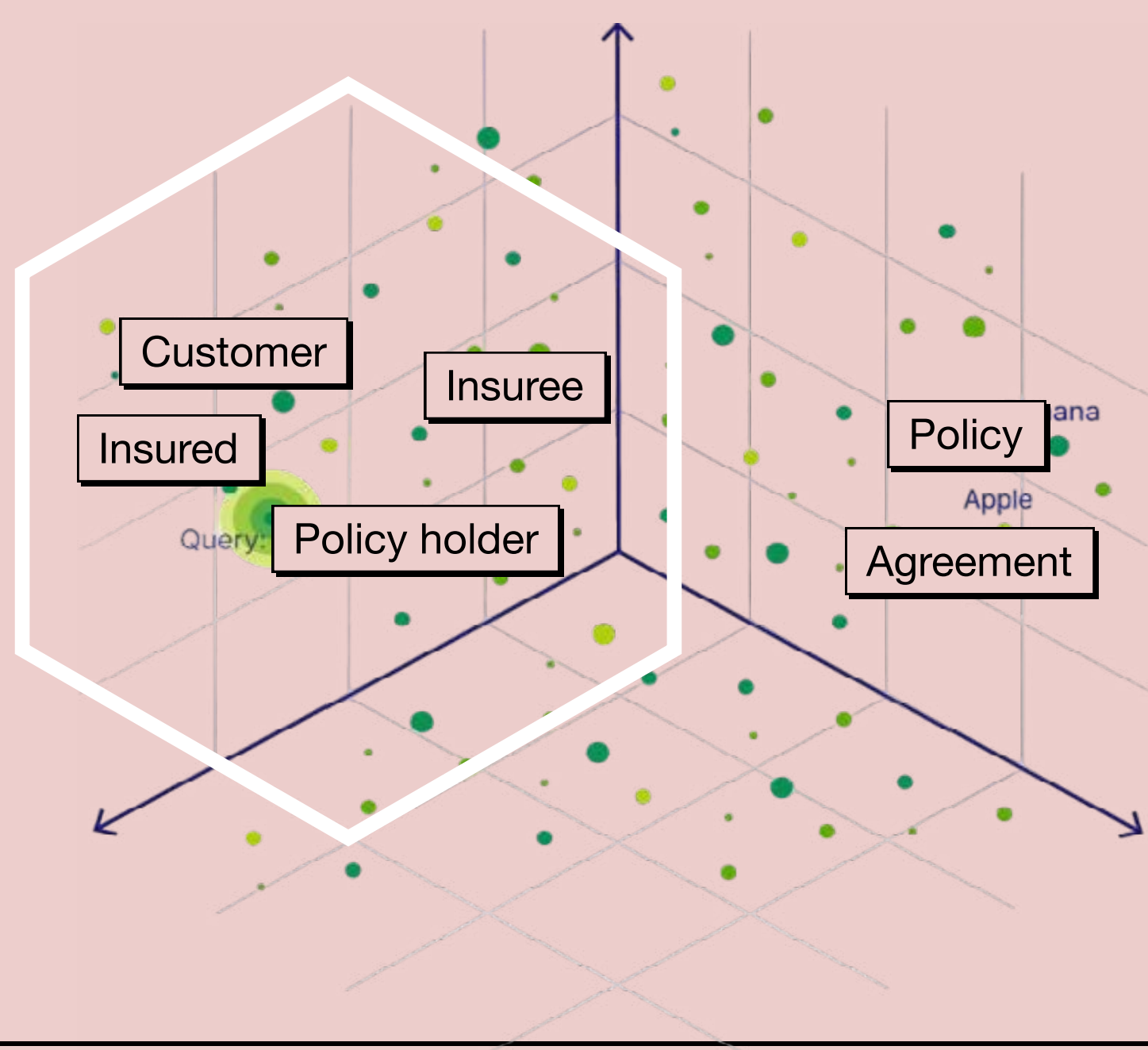
Vector databases have been used for semantic search for quite some time, and with the rise of LLMs, they became a natural fit to augment these models' capabilities. Any kind of textual document can be embedded into a vector database, enabling fast and efficient retrieval **based on meaning** (hence semantic) rather than exact word matches. For example, a search for "insured" would also retrieve relevant information about "policy holders." The retrieval mechanism finds relevant passages, and the LLM incorporates this information into its generated response to provide a more accurate and informed answer.



RAG increases domain specific accuracy by magnitudes

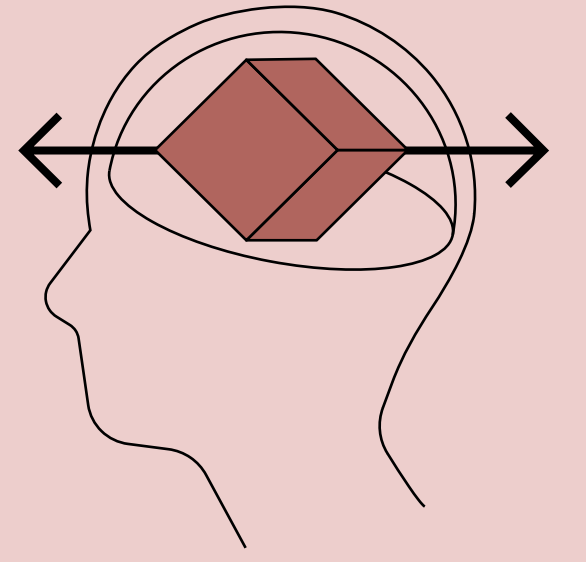
RAG systems can dramatically enhance the reliability of LLMs. Rather than relying on the model's pre-trained knowledge—derived from millions of often unvetted sources—we can ensure it works from carefully curated documents. Imagine an insurance agent answering questions about coverage: would you prefer they rely on hazy memories from a months-old briefing, or consult the insurance carrier's program manual?

In this example, we'd only need to use a single embedded document: the carrier's program manual (likely divided into thematic chunks to improve retrieval accuracy). This is a simple case, but more complex systems can work with tens of thousands of embedded documents—such as all scientific articles on parametric insurance from the last decade. These advanced systems might even include data pipelines to automatically expand the database as new content becomes available. Sophisticated use cases employ algorithms and document labeling to help the retrieval mechanism decide which relevant documents to "pull off the shelf" for searching.



Vector databases represent words or documents as points in a multidimensional space—often with hundreds or thousands of dimensions. Points that are close together in this space have similar meanings.

The illustration shows a simplified version of this concept in just three dimensions.



Fine-tuning

the heavy lifting: retraining the model for your specific context

Fine-tuning models is the most time-consuming and expensive way of building custom LLM systems. Fine-tuning means that by feeding additional training data, a foundation model's parameters (the mathematical formulas representing all the learned patterns) irreversibly change, essentially creating a new unique model version. Even though popular fine-tuning methods only retrain the last couple of layers of the neural network (GPT consists of 120 layers!), this process is still very resource-intensive for three reasons:

1 It necessitates building up high-quality labeled training data consisting of thousands of examples.

For instance, if we want to build an LLM-based decision-making support system for claim adjusters, it could make sense to retrain the model with historical claims data, including inputs and decisions. This would help it recognize patterns and predict outcomes more reliably than a generic model. This exercise would likely start with a month-long process of organizing and cleaning historic claims data to make it suitable for training purposes.

2: The actual training process is computation intensive, as the model updates its internal parameters while processing the examples. This is usually an iterative process that consists of training and validation rounds, until the desired accuracy is reached.

3. After fine-tuning, you now have a custom LLM model that needs to be operated exclusively for you. This means you can't use standard LLM-as-a-service APIs where the model is shared among multiple users. You either need to host the model yourself (which comes with significant infrastructure and MLOps costs), or opt for a dedicated instance from an LLM provider. Some platforms, like Microsoft Azure, offer managed dedicated instances for fine-tuned models, but this incurs much higher costs compared to using shared LLM resources. Additionally, if newer, cheaper, and more capable LLM models become available and you want to leverage one of them, you'll need to fine-tune it again.

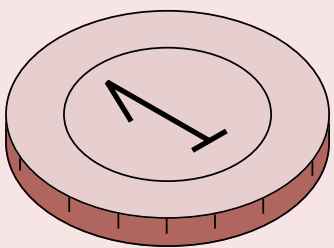
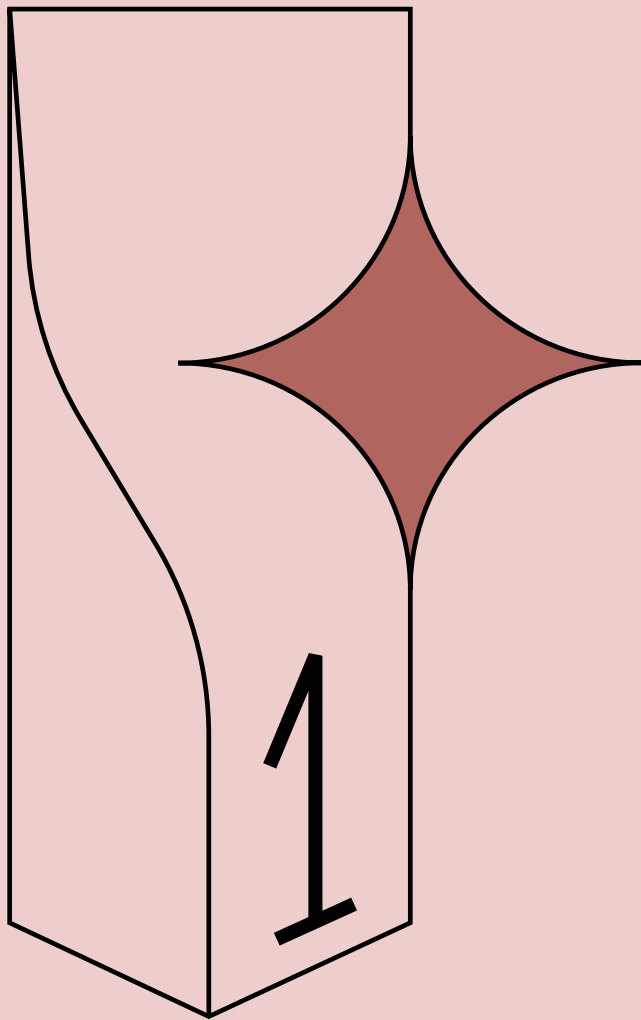
Why would you fine-tune a model?

There are many reasons to fine-tune a model, but due to its cost and complexity, it's usually a "last resort" when other techniques seem insufficient to produce acceptable outcomes. The most common reasons to consider fine-tuning a model are:

- ❑ **Better handling of Domain-Specific Language (e.g., medical, legal, financial, insurance).** Just as with human colleagues, if someone isn't familiar enough with a field's technical terms, accurately interpreting documents becomes impossible. Fine-tuning can help an LLM build a solid grasp of niche domain language. This is especially important for non-English use cases, where the LLM's understanding of niche terms in other languages may be particularly poor.
- ❑ **Improving Performance on Specific Tasks.** An LLM's accuracy can significantly improve when performing certain activities, like claim adjustment, if it receives additional training on proprietary data that allows it to learn decision-making patterns.
- ❑ **Enforce certain approach or principles:** Humans don't always agree on things and viewpoints can evolve in time. LLMs may be pre-trained on diverse viewpoints. Fine-tuning can ensure consistent use of a preferred approach, such as prioritizing customer satisfaction in disputed claims. However, effective prompting may often achieve similar results.
- ❑ **Updating the model's knowledge:** The model's knowledge cut-off date is always somewhat in the past, leaving it unaware of certain new developments (such as the EU's AI Act). Additional training can bring its knowledge up to date.

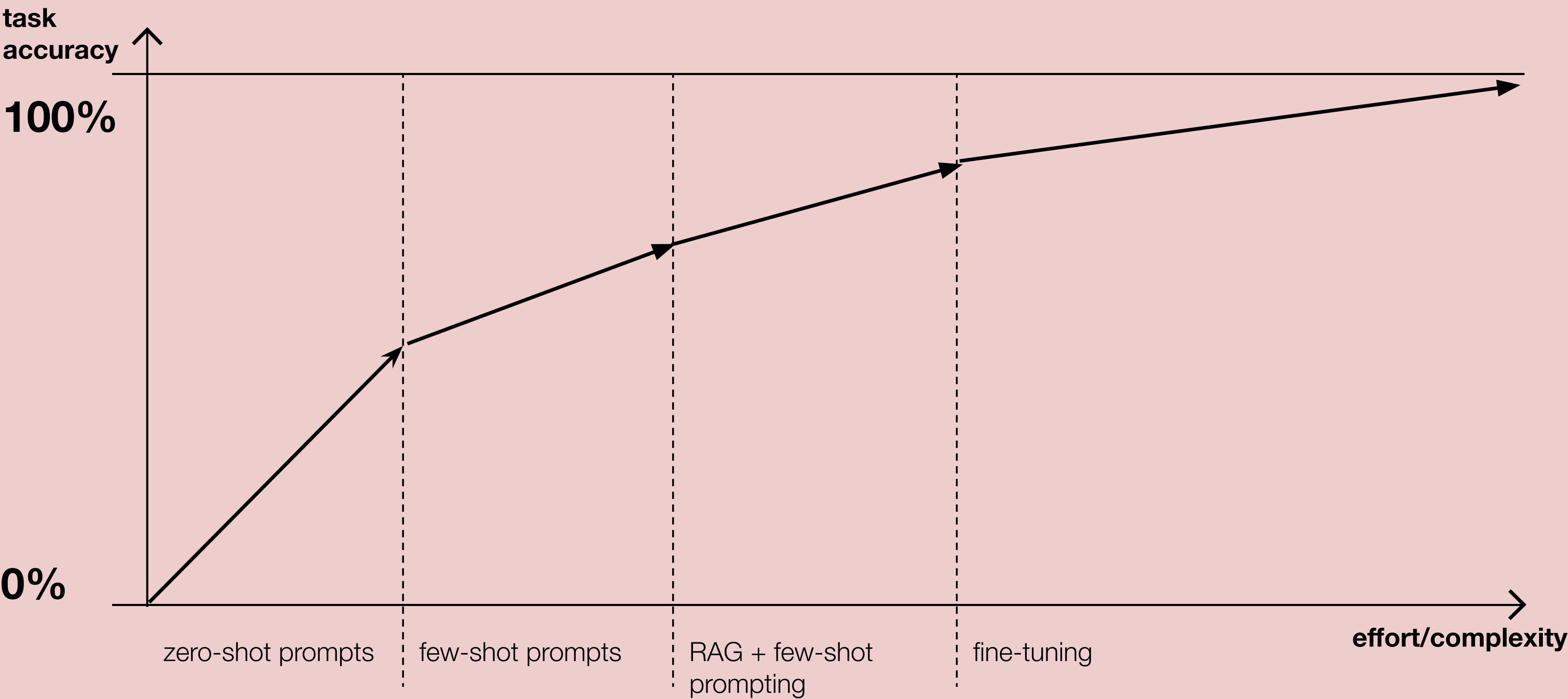
As this is a high-level technology briefing, we won't delve into the possible advantages of fine-tuning smaller models for specific use cases instead of using more generic large models.

choose the tech that gives the most bang for your buck

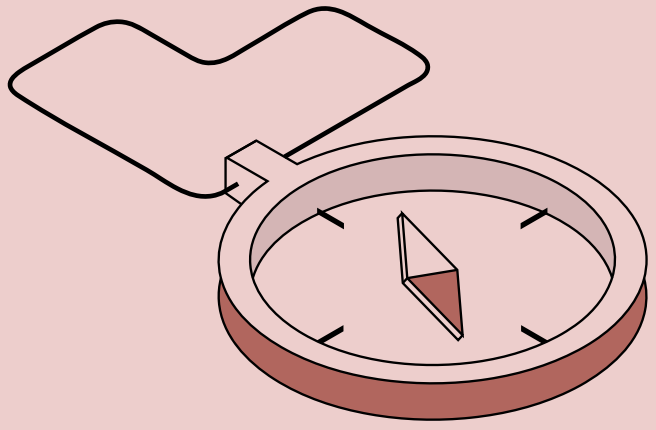


The product development principles of building LLM-based systems are not different from any other digital product: the best ROI can be achieved by finding the simplest possible solution that produces the desired results. The below diagram reflects our real-life experiences: in almost every project, we managed to produce good enough results to automate a process by building a system around carefully crafted system prompts, thoughtful use of RAG, and occasionally augmenting the LLM with tools.

Fine-tuning can significantly increase the cost and complexity of a project. It not only raises initial development expenses but also adds to the ongoing operational costs due to the need for maintaining the customized model. While in some cases it is necessary—particularly when domain-specific expertise or highly customized outputs are required—our first approach should always be to design the system without it. There are many ways to make prompting and the RAG pipeline more sophisticated, and these methods often yield a much better ROI than jumping straight to fine-tuning. However, for specialized tasks, such as in legal, medical, or other high-precision industries, fine-tuning might offer the reliability needed, despite the added complexity.



This illustration serves as a visual representation of a conceptual framework rather than an accurate depiction of actual accuracy improvements.



demystifying the reliability of LLMs

reliable

adjective

Someone or something that is reliable can be trusted or believed because he, she, or it works or behaves well in the way you expect // *Cambridge Dictionary*

Businesses began using software to automate and enhance processes over half a century ago, and today, digital solutions permeate both our professional and personal lives. Through these experiences, we’ve come to expect that computers, aside from occasional bugs, produce predictable results by following rigid rules. Most traditional software is deterministic—“if this, then that.” In contrast, Large Language Models (LLMs) are probabilistic. They generate responses based on statistical patterns learned from vast datasets, introducing variability—given the same input, an LLM might provide different responses. While parameters like temperature can reduce this variability, perfect consistency remains difficult due to the complexity of these models and their probabilistic nature.

LLMs are designed to emulate human communication, which is naturally flexible, so identical inputs rarely occur in real-world use. In short, LLMs are less predictable than traditional software, but far more adaptable. This requires a shift in mindset—approaching LLMs as flexible, yet sometimes unpredictable systems, much like managing human employees.

Humans make mistakes because we aren’t machines. We can be tired, inattentive, distracted, or misjudge situations due to cognitive biases. Many of our decisions are based on mental forecasts, which are inherently probabilistic. When managing employees, we expect this variability and, in return, value their ingenuity and flexible problem-solving. Unsurprisingly, as we strive to create AIs that mimic human intelligence’s flexibility, we introduce a level of unpredictability into our systems.

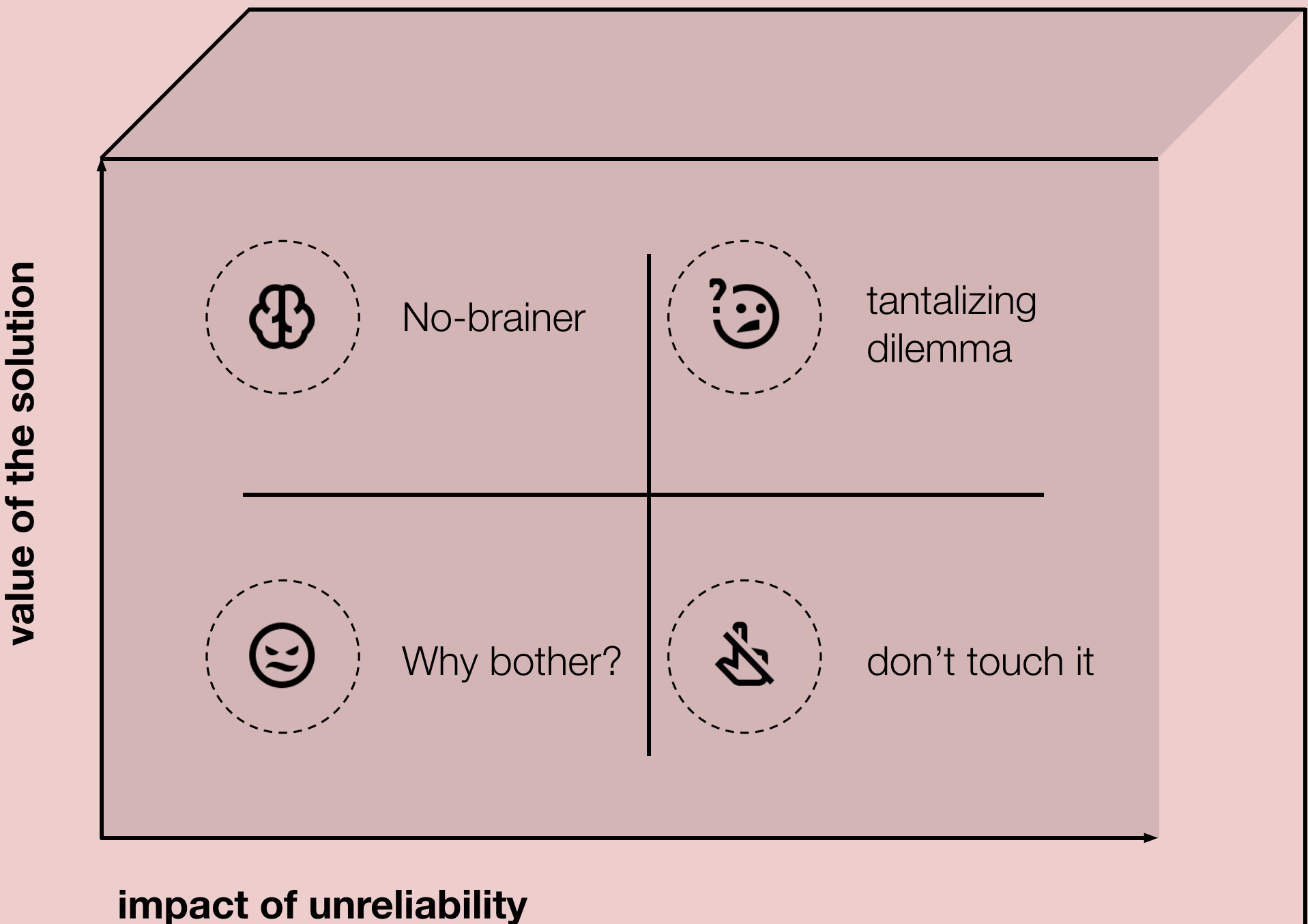
To manage the inherent unreliability of LLM-based systems, there are two key strategies:

- 1. Choose appropriate use cases** where some error or variability can be tolerated without major impact.
- 2. Mitigate unreliability by:**
 - a. Implementing guardrails**, such as content filtering or human-in-the-loop systems, to ensure the LLM operates within defined boundaries.
 - b. Rigorously testing and evaluating** the model’s performance.
 - c. Ensuring comprehensive observability to monitor outputs**, identify biases, and continuously improve reliability.

Choosing the right use case

LLMs aren’t magical tools for every problem. For instance, more predictable ML solutions are often better for categorization tasks. When LLMs seem like the right fit, consider the following:

- ❏ What’s the cost of unreliability? This can include financial, reputational, and legal risks.
- ❏ What’s the current error rate? If humans perform the task, we may already accept some level of error.
- ❏ How do social factors affect our view of mistakes? We tend to forgive human errors, but machine errors can feel less acceptable (like with self-driving cars).
- ❏ If more mature technologies can solve the problem, though with less ideal outcomes, does the LLM solution offer enough extra value to justify the risks?



increasing the reliability of our LLM system

Reliability in LLMs means how consistently the model provides accurate and expected answers. It reflects how well the model avoids mistakes, stays factual, and delivers dependable results across different situations. Ensuring reliability involves using multiple strategies, but the first step is to make reliability measurable—“if you can’t measure it, you can’t improve it.” At a high level, ensuring reliability means building a robust structure for evaluation, monitoring and intervention, along with continuous adjustments to gradually improve performance. Just like optimizing human employees, an LLM system’s outputs can always be refined further, though there is eventually a point of diminishing returns.

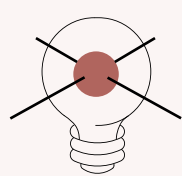
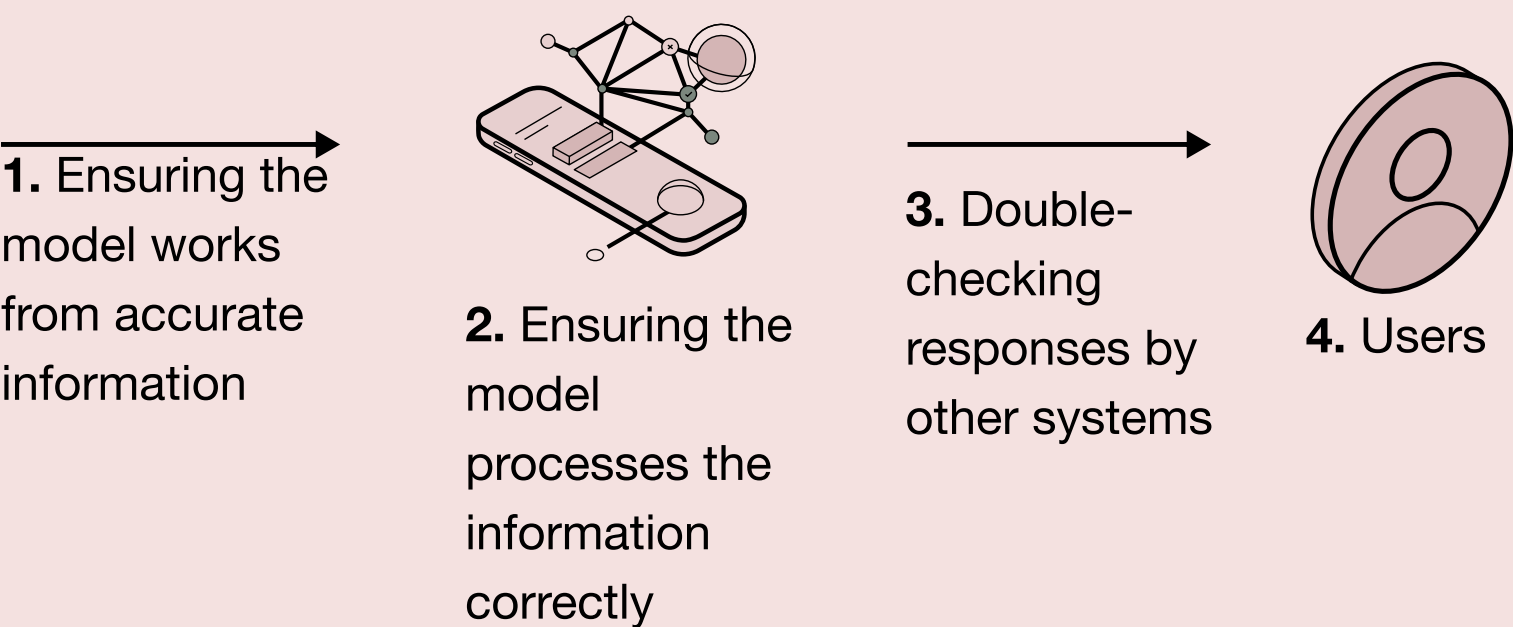
guardrails and other effective mechanisms

The aptly named guardrails stop our system from straying into dangerous or off-limits areas. What are these areas? To name the main offenders:

Factuality issues and hallucination. In most business contexts, the biggest risk is simply that the LLM's answer is not based on facts. You've likely heard about LLMs' tendency to produce hallucinations, so let's demystify that: a hallucination is an answer generated without clear factual foundations in the training data or external sources. Models are fundamentally built to provide answers and find patterns even if little or none exists, and they struggle to simply say "I don't know."

However, factual errors can emerge without any hallucination in play: the LLM may have learned a pattern from a false source, misinterpreted the question, or failed to find or comprehend relevant information when working with external documents.

The greatest risk lies in the variability factor and the smooth-talking nature of LLMs. If most of their responses are factually correct, and the incorrect ones are delivered as plausible-sounding, contextually relevant explanations, it becomes very difficult for humans to spot these errors. This is especially true when incorrect information is "sandwiched" between factually correct statements (intermittent errors). It's easy to see how this underlying doubt can undermine the system's overall credibility and value. (The problem of hidden defects is not unknown in other areas of quality assurance.)



The solutions:

Most of the work that goes into an LLM based system aims to ensure factually correct answers—this isn't something we just mitigate in later stages of a project. Ensuring factual accuracy is an ongoing process that spans the entire development lifecycle.

An essential first step for reducing hallucinations is to instruct the model to admit when it doesn't have sufficient information, such as saying 'I don't know.'

A next crucial concept is providing "ground truth" to the model beyond its knowledge from original training. Depending on the system's scope, certain ground truth information can be part of the system prompt. For example, a policy explanation chatbot could receive a list of industry terms with their exact definitions to use in its replies.

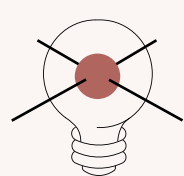
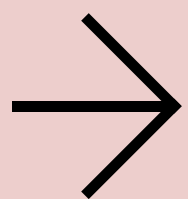
RAG is a popular and more scalable method for equipping LLMs with quickly accessible, curated external information. It provides a strong “ground truth” by having the LLM search provided documents and base its answers on verified data, rather than relying on what it learned during training. This shifts the LLM’s role to reasoning and communication, while the knowledge is stored in an external vector database. To build effective RAG systems, three factors are key: carefully curating and preparing the knowledge base, optimizing the retrieval process so the LLM can find relevant information for each use case, and giving effective instructions on how to use that information. As with most LLM systems, this process requires numerous rounds of evaluation and iteration by the designers.

Post-processing is an advanced method to ensure factual accuracy. While RAG focuses on supplying the LLM with the right source of truth before generating the response, post-processing double-checks the LLMs responses after it has been generated and before they are sent to the user. It effectively works as a censor with veto rights. This is an exciting field that is advancing fast and post-processors can be all kind of additional systems, like other AI models.



Adversarial prompting/prompt injection

Similar to social hacking, this involves tricking the LLM into performing actions beyond its purpose. The main issue is that LLMs interpret user inputs as instructions, not just questions. With the right malicious prompts, users can lead them astray, potentially exposing sensitive information from an accessible database or producing responses that harm a business's public image. (For example, imagine the fallout if a carrier's chatbot gave advice on insurance fraud and it was shared on social media.) This poses a serious risk for customer-facing or externally accessible tools.



Solutions:

Clear System Prompts: Craft the system prompt to include explicit instructions that guide the LLM to refuse inappropriate requests politely and avoid sharing sensitive information.

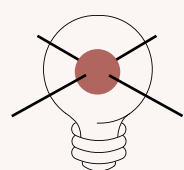
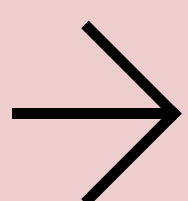
Input validation: A separate, specialized system component can validate and filter incoming user inputs, specifically looking for signs of adversarial prompting. As with fraud detection, flagging false positives can be extremely annoying for users, so these systems must be carefully configured to avoid being overreactive.

Secure design: As with any other software, security best practices should be followed. A crucial principle is to connect the system only to APIs that contain data essential for performing its duties (principle of least privilege).

Post-processing: Similar to ensuring truthfulness, responses can be analyzed and potentially intercepted by another system before they're sent to the user. This check ensures they don't contain sensitive information or content irrelevant to the use case.

Harmful or risky answers

There are topics we want the LLM to stay clear of. This generally includes anything that goes beyond its scope (an insurance bot shouldn't give healthcare advice), but can also include landmine topics within its own specialized field. Users are not aware of what we consider off-limits because of legal, reputational, or operational risks and they may prompt the LLM to discuss it. There are also obvious behaviors that we want the system to avoid, like using toxic language or being passive-aggressive.



Solutions:

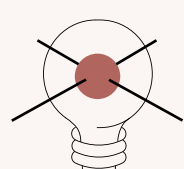
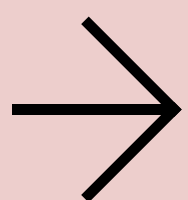
The solutions strategies don't differ much from the previously introduced methodologies.

Prompting: The system prompt gives the LLM its marching orders, often specifying the conversation's permitted scope and topics to avoid. As long as users don't intentionally try to mislead it (see adversarial prompting), this keeps the LLM on track in most cases.

Post-processing: Various tools can verify the LLM's answer before it's sent to the user. This may involve another LLM focused on staying on-topic, specialized tools like Nemo Guardrails, or traditional NLP models checking for certain expressions (like profanity). If necessary, the LLM can be asked to revise its response, or a predefined reply can be sent. Keep in mind, the model's fine-tuning by its developer already aims to minimize harmful content, so LLMs come with some built-in decorum.

Biases

In an LLM, biases happen when the model learns patterns from its training data that favor certain ideas, groups, or perspectives. This can cause the model to produce unfair or unbalanced responses, such as reinforcing stereotypes or overlooking important viewpoints. As a result, its answers might reflect the same biases present in the data it was trained on.



Solutions

AI Ethics is a rapidly expanding field and companies must pay increasing attention to this aspect to avoid reputational risks and potential legal consequences. Ensuring that training data comes from varied and balanced sources is primarily the foundation model developer's responsibility. However, if we fine-tune the model with our own data, we must pay attention to the same aspect.

Even without fine-tuning, biases can emerge from system design choices, such as system prompt formulations. Extensive testing and evaluation for biases are required on every LLM-based system. Once a bias has been identified, the previously discussed strategies can be used to set up effective guardrails for its mitigation.

evaluating the performance and accuracy of LLMs

How do we define a useful LLM system?

Performance in IT often refers to processing speed and response time. These can be important for LLM systems as well, because long response times can radically degrade the user experience. But when we talk about an LLM system's performance, we're not just talking about speed, we are more interested in **accuracy**. We break down what makes it a "valuable business tool" into specific qualities that we can measure and improve.

Below are some key qualities we often consider to measure accuracy, but this list is far from exhaustive. Each of these can be further detailed into specific contributing factors that we can measure and optimize—a process called LLM evaluation.

❑ **Reliability and factuality:** Can we trust the answers the system provides? To figure out why it might give incorrect answers, we need to look at how the system works step by step. How well does it find information from external sources? Are we selecting the most relevant pieces of information? Does the LLM use this information effectively to produce helpful responses? We can measure and improve each of these aspects.

❑ **Relevance:** Do the system's responses truly address the user's questions and context? Are the outputs really meaningful responses?

❑ **Coverage:** Can the system provide useful responses across all scenarios within the target use cases?

❑ **Fluency and Coherence:** Are the LLM's answers easy to understand? The responses should be well-written and make sense to the user without confusion.

❑ **Conversation Relevancy:** The system's ability to remember and utilize context, such as the chat history, which is essential for maintaining genuine conversations.

There are numerous additional aspects worth evaluating especially focused on safety, and ethical operations.

We can break down the performance and accuracy evaluation of an LLM-based system into three types of testing mechanisms:

1. Expert or internal evaluation performed by humans:

The team working on the tool continuously tests the system's outputs and adjusts the design to generate more useful responses. By formulating structured test cases (input, expected output, actual output) and tracking the previously explained factors, this process evolves into a structured scientific activity.

2. Automated evaluation with specialized tools:

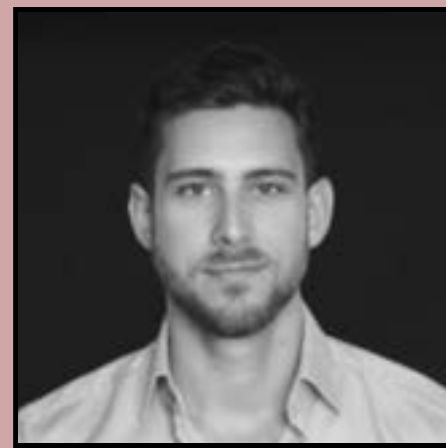
An increasing number of tools are available to run automated tests that continuously evaluate the system down to very granular performance indicators. DeepEval is a promising analytical tool with sound scientific foundations that we have successfully used in past projects.

3. End-user feedback loop: If you've used ChatGPT, you've likely seen the thumbs-down option for unsatisfactory responses. Similar features can be added to any LLM tool. For internal tools or early versions, feedback can be more detailed, with options to select failure categories or add comments. The team should analyze this input to improve response quality. Creating value for end users is the ultimate validation of the solution.

Despite rapid advancements in automated evaluation of LLM-based systems, with new tools and paradigms emerging monthly, human evaluation remains by far the most crucial at this stage.

LLMs aim to automate and augment the most sophisticated areas of human intelligence. Only humans can be the final judge of their most crucial success metric: whether the outputs are meaningful and valuable for the target use case.

authors



Balazs Fonagy
Chief Strategist
@Supercharge



Adam Kovacs
AI Architect
@Supercharge



Bence Lukacs
Experience Design Lead
@Supercharge

seize the opportunity that LLMs offer to rethink your operations and communication channels



Large Language Models are changing reality as you read this. Let's work together on an action plan on how to supercharge your business with generative AI.

Supercharge is an award-winning digital innovation partner. We craft digital strategies, design delightful journeys and build robust, AI-enabled software to create impact in your business with unparalleled speed and quality.

hello@supercharge.io

supercharge.io

