**Artificial intelligence (AI)** is a set of technologies that empowers computers to learn, reason, and perform a variety of advanced tasks in ways that used to require human intelligence, such as [understanding language](https://cloud.google.com/speech-to-text), [analyzing data](https://cloud.google.com/vertex-ai), and even providing helpful suggestions. It’s a transformational technology that can bring meaningful and positive change to people and societies and the world.

**Artificial intelligence (AI): Creating machines or systems that can perform tasks that typically require human intelligence, such as learning, problem-solving, and decision-making.** AI is all about building or creating machines that normally require us humans to use our brains. Now, there are different ways to achieve that goal, and one popular approach is machine learning, or ML for short. **Artificial intelligence (AI):**The broad field of building machines that can perform tasks requiring human intelligence.

Artificial intelligence refers to systems or machines that mimic human intelligence and capabilities to perform different tasks, and that can iteratively improve themselves based on the information they collect. [Artificial intelligence](https://aws.amazon.com/what-is/artificial-intelligence/) is the broader concept of making machines more human-like. It includes everything from smart assistants like Alexa, chatbots, and image generators to robotic vacuum cleaners and self-driving cars.

Artificial intelligence (AI) is a field of computer science focused on creating systems that can perform tasks typically requiring human intelligence, such as learning, reasoning, and problem-solving. AI systems use algorithms to process vast amounts of data, recognize patterns, make decisions, and adapt to new information, enabling them to power applications like voice assistants, self-driving cars, and recommendation engines. [Artificial intelligence](https://aws.amazon.com/what-is/artificial-intelligence/) is the broader concept of making machines more human-like.  
Artificial intelligence (AI) refers to computer systems capable of performing complex tasks that [historically](https://www.coursera.org/articles/history-of-ai) only humans could do, such as reasoning, making decisions, or solving problems. AI is an umbrella term that encompasses a wide variety of technologies, including [machine learning](https://www.coursera.org/articles/what-is-machine-learning), [deep learning](https://www.coursera.org/articles/what-is-deep-learning), and [natural language processing (NLP)](https://www.coursera.org/articles/natural-language-processing).

Artificial intelligence techniques, though diverse, all fundamentally rely on data, algorithms, and computational power. AI systems learn and improve through exposure to vast amounts of data, identifying patterns and relationships that humans might miss. This data serves as the training material, the quality and quantity of which are crucial for the AI's performance.

As mentioned earlier, AI isn't a single technology but a broad field encompassing several key areas:

* [**Machine Learning (ML)**](https://cloud.google.com/learn/what-is-machine-learning)**:** This is a type of AI where systems learn from data to identify patterns and make predictions or decisions without direct programming. Imagine teaching a computer to recognize a bird by showing it thousands of bird pictures; it learns what a bird looks like on its own.
* [**Deep Learning (DL)**](https://cloud.google.com/discover/what-is-deep-learning)**:** A subfield of ML, deep learning uses artificial neural networks with many layers (hence "deep") to learn from data. These networks are inspired by the structure of the human brain and are particularly good at complex tasks like image and speech recognition.
* [**Natural Language Processing (NLP)**](https://cloud.google.com/learn/what-is-natural-language-processing)**:** NLP enables computers to understand, interpret, and generate human language. This is what powers voice assistants like Siri and Alexa, translation services, and chatbots.

Types of artificial intelligence

Artificial intelligence can be organized in several ways, depending on stages of development or actions being performed.

1) AI types of capability (Based on technology)

This classification defines AI models based on their intelligence level and problem-solving abilities.

* **Artificial Narrow Intelligence (ANI)**: ANI, also called weak AI. This is the only form of AI that currently exists. ANI models are designed to perform a single, specific task, such as identifying images, engaging in chat, or filtering emails. Examples include voice assistants, facial recognition technology, and generative AI models like Gemini and other large language models (LLMs). Despite its name, ANI does not possess reasoning or self-awareness; instead, it combines data with an algorithm to make predictions within predefined parameters. While ANI offers many benefits, it also carries risks, as poor training data can lead to biased or inaccurate outputs, which can be critical in applications like loan approvals, hiring decisions, and predictive policing. Cybercriminals can also potentially exploit ANI to create sophisticated AI-driven scams.
* [**Artificial General Intelligence (AGI)**](https://cloud.google.com/discover/what-is-artificial-general-intelligence): Also known as strong AI. This is a proposed future step in AI technology. Theoretically, AGI would be capable of performing a broad range of tasks and would utilize human-like reasoning to learn, adapt, and improve. AGI does not yet exist. Unlike ANI, AGI is expected to be adaptive, autonomous, and capable of learning from its actions. Fictional examples include droids from Star Wars. However, AGI may raise significant safety and ethical concerns, as malicious actors could program AGI with harmful intent, leading to potentially limitless destructive capabilities if unregulated.
* **Artificial Superintelligence (ASI)**: This is the most advanced theoretical form of AI. ASI would be a self-aware entity operating beyond human control, significantly surpassing human intelligence in reasoning, creativity, and even emotional intelligence. Like other forms of AI, there are concerns that ASI could pose an existential threat to humanity, with some AI researchers suggesting a non-negligible chance of extremely bad outcomes, including human extinction.

2) AI types by functionality

This classification categorizes AI based on how it operates and interacts in specific contexts.

1. **Reactive machines:** Limited AI that only reacts to different kinds of stimuli based on preprogrammed rules. It lacks memory and therefore cannot learn from new data. A notable example is IBM’s Deep Blue, which defeated chess champion Garry Kasparov in 1997.
2. **Limited memory:**Most modern AI is limited memory. It can use memory to improve over time by training on new data, typically through an artificial neural network or other training model. This memory is short-term; once a session ends, the memory often resets. Examples include self-driving cars observing other vehicles and chatbots like Gemini remembering previous messages in a conversation.
3. **Theory of mind:**Theory of mind AI doesn't currently exist (yet), but research is ongoing into its possibilities. It describes AI that can emulate the human mind and have decision-making capabilities equal to that of a human, including the ability to recognize and remember emotions and react in social situations as a human would.

One of the first things you need to know while talking about modern AI is big data.

**Big data** refers to extremely large and diverse collections of structured, unstructured, and semi-structured data that continues to grow exponentially over time. These datasets are so huge and complex in volume, velocity, and variety, that traditional data management systems cannot store, process, and analyze them.

**Big data** refers to extremely large and complex data sets that cannot be easily managed or analyzed with traditional data processing tools, particularly spreadsheets. Big data includes structured data, like an inventory database or list of financial transactions; unstructured data, such as social posts or videos; and mixed data sets, like those used to train large language models for AI.

**Big data** refers to massive, complex data sets that traditional data management systems cannot handle. When properly collected, managed and analyzed, big data can help organizations discover new insights and make better business decisions.

Data which are very large in size is called big data. Big data describes large and diverse datasets that are huge in volume and also rapidly grow in size over time.

**What are the Five “Vs” of Big Data?**

Traditionally, we’ve recognized big data by three characteristics: variety, volume, and velocity, also known as the “three Vs.” However, two additional Vs have emerged over the past few years: value and veracity.

Those additions make sense because today, data has become capital. Think of some of the world’s biggest tech companies. Many of the products they offer are based on their data, which they’re constantly analyzing to produce more efficiency and develop new initiatives. Success depends on all five Vs.

* **Volume.** The amount of data matters. With big data, you’ll have to process high volumes of low-density, unstructured data. This can be data of unknown value, such as X (formerly Twitter) data feeds, clickstreams on a web page or a mobile app, or sensor-enabled equipment. For some organizations, this might be tens of terabytes of data. For others, it may be hundreds of petabytes.
* **Velocity.** Velocity is the fast rate at which data is received and (perhaps) acted on. Normally, the highest velocity of data streams directly into memory versus being written to disk. Some internet-enabled smart products operate in real time or near real time and will require real-time evaluation and action.
* **Variety.** Variety refers to the many types of data that are available. Traditional data types were structured and fit neatly in a [relational database](https://www.oracle.com/in/database/what-is-a-relational-database/). With the rise of big data, data comes in new unstructured data types. Unstructured and semistructured data types, such as text, audio, and video, require additional preprocessing to derive meaning and support metadata.
* **Veracity.** How truthful is your data—and how much can you rely on it? The idea of veracity in data is tied to other functional concepts, such as data quality and data integrity. Ultimately, these all overlap and steward the organization to a data repository that delivers high-quality, accurate, and reliable data to power insights and decisions.
* **Value.** Data has intrinsic value in business. But it’s of no use until that value is discovered. Because big data assembles both breadth and depth of insights, somewhere within all of that information lies insights that can benefit your organization. This value can be internal, such as operational processes that might be optimized, or external, such as customer profile suggestions that can maximize engagement.

**The Evolution of Big Data: Past, Present, and Future**

Although the concept of big data is relatively new, the need to manage large data sets dates back to the 1960s and ’70s, with the first data centers and the development of the relational database.

**Past.** Around 2005, people began to realize just how much data users generated through Facebook, YouTube, and other online services. Apache Hadoop, an open source framework created specifically to store and analyze big data sets, was developed that same year. NoSQL also began to gain popularity during this time.

**Present.** The development of open source frameworks, such as Apache Hadoop and more recently, Apache Spark, was essential for the growth of big data because they make big data easier to work with and cheaper to store. In the years since then, the volume of big data has skyrocketed. Users are still generating huge amounts of data—but it’s not just humans who are doing it.

With the advent of the Internet of Things (IoT), more objects and devices are connected to the internet, gathering data on customer usage patterns and product performance. The emergence of [machine learning](https://www.oracle.com/in/artificial-intelligence/machine-learning/what-is-machine-learning/) has produced still more data.

**Future.** While big data has come far, its value is only growing as [generative AI](https://www.oracle.com/in/artificial-intelligence/generative-ai/what-is-generative-ai/) and cloud computing use expand in enterprises. The cloud offers truly elastic scalability, where developers can simply spin up ad hoc clusters to test a subset of data. And [graph databases](https://www.oracle.com/in/autonomous-database/what-is-graph-database/) are becoming increasingly important as well, with their ability to display massive amounts of data in a way that makes analytics fast and comprehensive.

How does big data work?

The central concept of big data is that the more visibility you have into anything, the more effectively you can gain insights to make better decisions, uncover growth opportunities, and improve your business model. Big data works by providing insights that shine a light on new opportunities and business models. Once data has been ingested, getting started involves three key actions:

Making big data work requires three main actions:

* **Integration:**Big data collects terabytes, and sometimes even petabytes, of raw data from many sources that must be received, processed, and transformed into the format that business users and analysts need to start analyzing it.
* **Management:**Big data needs big storage, whether in the cloud, on-premises, or both. Data must also be stored in whatever form required. It also needs to be processed and made available in real time. Increasingly, companies are turning to cloud solutions to take advantage of the unlimited compute and scalability.
* **Analysis:**The final step is analyzing and acting on big data—otherwise, the investment won’t be worth it. Beyond exploring the data itself, it’s also critical to communicate and share insights across the business in a way that everyone can understand. This includes using tools to create data visualizations like charts, graphs, and dashboards.

**Machine learning** is a branch of Artificial Intelligence that focuses on developing models and algorithms that let computers learn from data without being explicitly programmed for every task. In simple words, ML teaches the systems to think and understand like humans by learning from the data.  
**Machine learning** is the subset of artificial intelligence (AI) focused on algorithms that can “learn” the patterns of training data and, subsequently, make accurate *inferences* about new data. This pattern recognition ability enables machine learning models to make decisions or predictions without explicit, hard-coded instructions.

Machine learning is a [subset of artificial intelligence](https://cloud.google.com/learn/artificial-intelligence-vs-machine-learning) that enables a system to autonomously learn and improve using neural networks and deep learning, without being explicitly programmed, by feeding it large amounts of data.

Machine learning allows computer systems to continuously adjust and enhance themselves as they accrue more “experiences.” Thus, the performance of these systems can be improved by providing larger and more varied datasets to be processed.

**Machine learning (ML): A subset of artificial intelligence that uses data to train machines to improve their performance over time without being explicitly programmed.** ML is all about using data to train machines to perform specific tasks. Machine Learning models predict the future based on existing data, much like how humans use experience to make educated guesses. However, while humans might rely on intuition or gut feelings, these models use probability. ML models analyze the data they've been given, identify patterns, and then calculate the likelihood of different outcomes when presented with new information. **ML is** a subset of AI where machines learn from data. ML is a broad field that encompasses many different techniques, one of which is deep learning (DL).

**Machine learning** is a branch of Artificial Intelligence that focuses on developing models and algorithms that let computers learn from data **and make decisions or predictions** without being explicitly programmed for every task. In simple words, ML teaches the systems to think and understand like humans by learning from the data.

**Types of ML**: Machine learning has three primary learning approaches: supervised, unsupervised, and reinforcement learning, each with its own data requirements. Supervised models rely on labeled data (**Labeled data** has tags, such as a name, type, or number), unsupervised models work with unlabeled data (**Unlabeled data** is simply data that is not tagged or labeled in any way. It's raw, unprocessed information without inherent meaning) and reinforcement learning learns through interaction and feedback. The choice of approach depends on the specific task and the nature of the data available.

All machine learning methods can be categorized as one of three distinct learning paradigms: supervised learning, unsupervised learning or reinforcement learning, based on the nature of their training objectives and (often but not always) by the type of training data they entail.

1) [**Supervised Learning**](https://www.geeksforgeeks.org/machine-learning/supervised-machine-learning/)**:** **Supervised machine learning** trains models on labeled data, where each input is paired with its correct output, allowing the model to learn the relationship between them. The model's goal is to identify patterns and relationships within this labeled data, enabling it to accurately predict outputs for new, unseen inputs. **Supervised** **learning** is a machine learning model that uses labeled training data (structured data) to map a specific feature to a label.   
E.g. Predicting housing prices is a common example of supervised learning. A model is trained on a dataset where each house has labeled data, such as its size, number of bedrooms, location, and the corresponding sale price. This labeled data allows the algorithm to learn the relationship between the features of a house and its price. Once trained, the model can then predict the price of a new house based on its features.  
In supervised learning, the output is known (such as recognizing a picture of an apple) and the model is trained on data of the known output. In simple terms, to train the algorithm to recognize pictures of apples, feed it pictures labeled as apples. The most common supervised learning algorithms used today include:

* Linear regression
* Polynomial regression
* K-nearest neighbors
* Naive Bayes
* Decision trees

A diagram of a model training

AI-generated content may be incorrect.

2) [**Unsupervised Learning**](https://www.geeksforgeeks.org/machine-learning/unsupervised-learning/): **Unsupervised ML models** deal with raw, unlabeled data to find natural groupings. Instead of learning from labeled data, it dives headfirst into a sea of unlabeled data. Think of it as exploratory analysis. Unsupervised learning helps you understand the underlying structure of your data and uncover insights that you might not have even known to look for. Finds patterns or groups in unlabeled data (data that only include input), like clustering or dimensionality reduction. Unsupervised learning is a machine learning model that uses unlabeled data (unstructured data) to learn patterns. Unlike supervised learning, the “correctness” of the output isn't known ahead of time. Rather, the algorithm learns from the data without human input (and is thus, unsupervised) and categorizes it into groups based on attributes. For example, if the algorithm is given pictures of apples and bananas, it will work by itself to categorize which picture is an apple and which is a banana. Unsupervised learning is good at descriptive modeling and pattern matching. The most common unsupervised learning algorithms used today include:

* Fuzzy means
* K-means clustering
* Hierarchical clustering
* Partial least squares

A diagram of a brain and gears

AI-generated content may be incorrect.

3) [**Reinforcement Learning**](https://www.geeksforgeeks.org/machine-learning/what-is-reinforcement-learning/): **Reinforcement learning** is all about learning through interaction and feedback. Learns through trial and error to maximize rewards, ideal for decision-making tasks. Imagine a robot learning to navigate a maze. It starts with no knowledge of the maze's layout. As it explores and interacts with the maze, it collects data—bumping into walls (negative feedback) or finding shortcuts (positive feedback). Through this process of trial and error, the algorithm learns which actions lead to the best outcomes. It's like training a pet. You reward good behavior and discourage bad behavior. And over time, the pet learns to perform the desired actions. Similarly, in reinforcement learning, the algorithm learns to maximize rewards and minimize penalties by interacting with its environment. This type of learning is particularly useful in situations where you can't provide explicit instructions or labeled data. For example, you could use reinforcement learning to train a self-driving car to navigate complex traffic situations or to optimize the performance of a robot in a manufacturing plant.

Reinforcement learning is a machine learning model that can be described as “learn by doing” through a series of trial and error experiments. An “agent” learns to perform a defined task through a feedback loop until its performance is within a desirable range. The agent receives positive reinforcement when it performs the task well and negative reinforcement when it performs poorly. An example of reinforcement learning is when Google researchers taught a reinforcement learning algorithm to play the game Go. The model, which had no prior knowledge of the rules of Go, simply moved pieces at random and “learned” the best moves to make. The algorithm was trained by positive and negative reinforcement to the point that the machine learning model could beat a human player at the game.

**Examples of machine learning approaches on Google Cloud**

1) Predictive maintenance with Vertex AI (supervised learning):

By training a model on sensor data from machines like temperature, pressure, and vibration, Vertex AI can predict when a machine is likely to fail, enabling proactive maintenance and reducing downtime.

2) Anomaly Detection with BigQuery ML (unsupervised learning):

BigQuery ML can analyze historical transaction data (amount, location, time, etc.) to identify patterns and flag unusual transactions that deviate significantly from the norm. This helps prevent fraud and minimize financial losses.

3) Product recommendations with Vertex AI (reinforcement learning):

Vertex AI can train a reinforcement learning model to recommend products to users based on their browsing history, purchase behavior, and other factors. The model learns to maximize user engagement and sales by continuously refining its recommendations.

**Data tools and management for ML workloads**

The key to building an ML model is data. But the journey from scattered data to actionable insights isn't always easy. There are a few key steps that it takes to get to having a working solution, and if you are building a solution for an enterprise, along the way you need to consider many factors such as security, robustness, and scalability.

Google Cloud offers a comprehensive suite of data tools and management capabilities tailored for ML workloads. These tools and capabilities are designed to support the entire ML lifecycle, from data preparation and model training to deployment and monitoring. The ML lifecycle encompasses several key stages: data ingestion and preparation, model training, model deployment, and model management. Google Cloud provides a comprehensive suite of tools and capabilities to support each stage of this lifecycle. It includes Vertex AI for model training and deployment, and various data tools for ingestion, preparation, and management. By understanding and effectively managing this lifecycle, organizations can maximize the value of their initiatives and ensure long-term success.

Let's break down the key stages of making data accessible for AI.

**1) Gather your data**: Data gathering, also called data ingestion, involves collecting raw data from various sources. To effectively train and test your model, determine the data you need based on the outcome you want to achieve.

Google Cloud supports data ingestion through several tools.

* Pub/Sub handles real-time streaming data processing, regardless of the structure of the data.
* Cloud Storage is well-suited for storing unstructured data.
* Cloud SQL and Cloud Spanner are used to manage structured data.

**2) Prepare your data**: Data preparation is the process of cleaning and transforming raw data into a usable format for analysis or model training. This involves formatting and labeling data properly.

Google Cloud offers tools like **BigQuery** for data analysis and **Data Catalog** for data governance. These tools help prepare data for ML models.

With BigQuery, you can filter data, correct its inconsistencies, and handle missing values.

With Data Catalog, you can find relevant data for your ML projects. This tool provides a centralized repository to easily discover datasets in your organization.

**3) Train your model:** The process of creating your ML model using data is called model training. Google Cloud's Vertex AI platform provides a managed environment for training ML models.

With Vertex AI, you can set parameters and build your model, using prebuilt containers for popular machine learning frameworks, custom training jobs, and tools for model evaluation. Vertex AI also provides access to powerful computing resources to make the model training process faster.

**4) Deploy and predict:** Model deployment is the process of making a trained model available for use.

Vertex AI simplifies this by providing tools to put the model into action for generating predictions. This includes scaling the deployment, which means adjusting the resources allocated to the model to handle varying amounts of usage.

**5) Manage your model:** Model management is the process of managing and maintaining your models over time. Google Cloud offers tools for managing the entire lifecycle of ML models. This includes the following:

* **Versioning**: Keep track of different versions of the model.
* **Performance tracking**: Review the model metrics to check the model's performance.
* **Drift monitoring**: Watch for changes in the model's accuracy over time.
* **Data management**: Use Vertex AI Feature Store to manage the data features the model uses.
* **Storage**: Use Vertex AI Model Garden to store and organize the models in one place.
* **Automate**: Use Vertex AI Pipelines to automate your machine learning tasks.

All these Google Cloud tools are designed to work together seamlessly. You can also connect them with your existing tools or build your own custom solutions. And throughout the entire process, Google Cloud's infrastructure ensures your solution is available and reliable. At the same time, Identity and Access Management (IAM) keeps your data safe and ensures only the right people have access.

**ML lifecycle**

**Data ingestion and preparation:**  
The process of collecting, cleaning, and transforming raw data into a usable format for analysis or model training.

**Model training:**  
The process of creating your ML model using data.

**Model deployment:**  
The process of making a trained model available for use.

**Model management:**  
The process of managing and maintaining your models over time.

**Deep Learning (DL):** A subset of ML, deep learning uses artificial neural networks with many layers (hence "deep") to learn from data. These networks are inspired by the structure of the human brain and are particularly good at complex tasks like image and speech recognition.

A powerful subset of machine learning, distinguished by its use of artificial neural networks. These networks enable the processing of highly complex patterns and the generation of sophisticated predictions.

Neural networks can leverage both labeled and unlabeled data, a strategy known as semi-supervised learning. They train on a blend of a small amount of labeled data and a large amount of unlabeled data. That way, they learn foundational concepts and generalize effectively to novel examples.

Deep learning is a type of machine learning that uses artificial neural networks with multiple layers to enable digital systems to learn from large amounts of data, recognize complex patterns, and make decisions. This technique is inspired by the human brain and is used for tasks like image and speech recognition, natural language processing, and creating other AI-powered services.

Deep learning is a subset of [machine learning](https://www.ibm.com/think/topics/machine-learning?) driven by multilayered [neural networks](https://www.ibm.com/think/topics/neural-networks?) whose design is inspired by the structure of the human brain. Deep learning models power most state-of-the-art [artificial intelligence (AI)](https://www.ibm.com/think/topics/artificial-intelligence?) today, from [computer vision](https://www.ibm.com/think/topics/computer-vision) and [generative AI](https://www.ibm.com/think/topics/generative-ai) to self-driving cars and robotics.

A diagram of a machine learning

AI-generated content may be incorrect.

Deep Learning is transforming the way machines understand, learn and interact with complex data. Deep learning mimics neural networks of the human brain, it enables computers to autonomously uncover patterns and make informed decisions from vast amounts of unstructured data.

Deep learning is a subset of [machine learning](https://aws.amazon.com/what-is/machine-learning/). Deep learning algorithms emerged to make traditional machine learning techniques more efficient. Traditional machine learning methods require significant human effort to train the software. For example, in animal image recognition, you need to do the following:

* Manually label hundreds of thousands of animal images.
* Make the machine learning algorithms process those images.
* Test those algorithms on a set of unknown images.
* Identify why some results are inaccurate.
* Improve the dataset by labeling new images to improve result accuracy.

This process is called supervised learning. In supervised learning, result accuracy improves only with a broad and sufficiently varied dataset. For instance, the algorithm might accurately identify black cats but not white cats because the training dataset had more images of black cats. In that case, you would need more labeled data of white cat images to train the machine learning models again.

A diagram of machine learning

AI-generated content may be incorrect.

**How Deep Learning Works?**

If you want to know how deep learning works, the first term you need to familiarize yourself with is neural network A neural network is a system of interconnected nodes that aims to mimic how the human brain processes information and recognizes patterns. Deep learning models are neural networks designed after the human brain. A human brain contains millions of interconnected biological neurons that work together to learn and process information. Similarly, artificial neurons are software modules called nodes that use mathematical calculations to process data. Deep learning neural networks, or artificial neural networks, comprise many layers of artificial neurons that work together to solve complex problems.

Neural network consists of layers of interconnected nodes or neurons that collaborate to process input data. In a fully connected deep neural network data flows through multiple layers where each neuron performs nonlinear transformations, allowing the model to learn intricate representations of the data.

In a deep neural network the input layer receives data which passes through hidden layers that transform the data using nonlinear functions. The final output layer generates the model’s prediction.

**Input layer**

An artificial neural network has several nodes that input data into it. These nodes make up the system's input layer.

**Hidden layer**

The input layer processes and passes the data to layers further in the neural network. These hidden layers process information at different levels, adapting their behavior as they receive new information. Deep learning networks have hundreds of hidden layers that they can use to analyze a problem from several different angles.

For example, if you were given an image of an unknown animal that you had to classify, you would compare it with animals you already know. For example, you would look at the shape of its eyes and ears, size, number of legs, and fur pattern. You would try to identify patterns, such as the following:

The animal has hooves, so it could be a cow or deer.

The animal has cat eyes, so it could be a wild cat.

The hidden layers in deep neural networks work in the same way. If a deep learning algorithm tries to classify an animal image, each of its hidden layers processes a different animal feature and tries to categorize it accurately.

**Output layer**

The output layer consists of the nodes that output the data. Deep learning models that output "yes" or "no" answers have only two nodes in the output layer. On the other hand, those that output a wider range of answers have more nodes. Generative AI has a sophisticated output layer to generate new data that matches patterns in its training data set.

**A diagram of a network

AI-generated content may be incorrect.**

**Why is deep learning important?**

Deep learning technology drives many artificial intelligence applications used in everyday products, such as the following:

* Chatbots and code generators
* Digital assistants
* Voice-activated television remotes
* Fraud detection
* Automatic facial recognition

It is also a critical component of technologies like self-driving cars, virtual reality, and more. Businesses use deep learning models to analyze data and make predictions in various applications.

**What are deep learning use cases?**

Deep learning has several use cases in automotive, aerospace, manufacturing, electronics, medical research, and other fields.

* Self-driving cars use deep learning models for object detection.
* Defense systems use deep learning to flag areas of interest in satellite images.
* Medical image analysis uses deep learning to detect cancer cells for medical diagnosis.
* Factories use deep learning applications to detect when people or objects are within an unsafe distance of machines.

These various use cases of deep learning can be grouped into five broad categories: computer vision, speech recognition, natural language processing (NLP), recommendation engines, and generative AI.

**Computer vision**

[Computer vision](https://aws.amazon.com/computer-vision/) automatically extracts information and insights from images and videos. Deep learning techniques to comprehend images in the same way that humans do. Computer vision has several applications, such as the following:

* Content moderation to automatically remove unsafe or inappropriate content from image and video archives
* Facial recognition to identify faces and recognize attributes like open eyes, glasses, and facial hair
* Image classification to identify brand logos, clothing, safety gear, and other image details

**Speech recognition**

Deep learning models can analyze human speech despite varying speech patterns, pitch, tone, language, and accent. Virtual assistants such as Amazon Alexa, [text-to-speech](https://aws.amazon.com/polly/what-is-text-to-speech/), and [speech-to-text software](https://aws.amazon.com/what-is/speech-to-text/) use speech recognition to do the following tasks:

* Assist call center agents and automatically classify calls.
* Convert clinical conversations into documentation in real-time.
* Accurately subtitle videos and meeting recordings for a wider content reach.
* Convert scripts to prompts for intelligent voice assistance.

**Natural language processing**

Computers use deep learning algorithms to gather insights and [meaning from text data](https://aws.amazon.com/comprehend/) and documents. This ability to process natural, human-created text has several use cases, including:

* Automated virtual agents and[chatbots](https://aws.amazon.com/what-is/chatbot/)
* Automatic summarization of documents or news articles
* Business intelligence analysis of long-form documents, such as emails and forms
* Indexing of key phrases that indicate sentiment, such as positive and negative comments on social media

**Recommendation engines**

Applications can use deep learning methods to track user activity and develop [personalized recommendations](https://aws.amazon.com/mxnet/). They can analyze users' behavior and help them discover new products or services. For example,

* Recommend personalized videos and content.
* Recommend customized products and services.
* Filter search results to highlight relevant content based on user location and behavior

**Generative AI**

Generative AI applications can create new content and communicate with end users more sophisticatedly. They can assist in automating complex workflows, brainstorming ideas, and intelligent knowledge searches. For example, with generative AI tools like [Amazon Q Business](https://aws.amazon.com/q/business/) and [Amazon Q Developer](https://aws.amazon.com/q/developer/), users can

* Ask natural language questions and get summarized answers from multiple internal knowledge sources.
* Get code suggestions and automatic code scanning and upgrades.
* Create new documents, emails, and other marketing content faster.

**Benefits of deep learning over machine learning**

A deep learning network has the following benefits over traditional machine learning.

***Efficient processing of unstructured data***

Machine learning methods find unstructured data, such as text documents, challenging to process because the training dataset can have infinite variations. On the other hand, deep learning models can comprehend unstructured data and make general observations without manual feature extraction. For instance, a neural network can recognize that these two different input sentences have the same meaning:

* Can you tell me how to make the payment?
* How do I transfer money?

***Hidden relationships and pattern discovery***

A deep learning application can analyze large amounts of data more deeply and reveal new insights for which it might not have been trained. For example, consider a deep learning model trained to analyze consumer purchases. The model has data only for the items you have already purchased. However, the artificial neural network can suggest new items you haven't bought by comparing your buying patterns to those of similar customers.

***Unsupervised learning***

Deep learning models can learn and improve over time based on user behavior. They do not require large variations of labeled datasets. For example, consider a neural network that automatically corrects or suggests words by analyzing your typing behavior. Let's assume it was trained in English and can spell-check English words. However, if you frequently type non-English words, such as danke, the neural network automatically learns and autocorrects these words too.

***Volatile data processing***

Volatile datasets have large variations. One example is loan repayment amounts in a bank. A deep learning neural network can categorize and sort that data by analyzing financial transactions and flagging some for fraud detection.

**Neural networks** are machine learning models that mimic the complex functions of the human brain. These models consist of interconnected nodes or neurons that process data, learn patterns and enable tasks such as pattern recognition and decision-making.

A neural network is a system of interconnected nodes that aims to mimic how the human brain processes information and recognizes patterns.

A diagram of a network

AI-generated content may be incorrect.

**Natural Language Processing (NLP)** is a branch of AI that allows computers to understand, analyze, and interpret text and spoken word in the same way human beings can.

Natural language processing (NLP) is a form of artificial intelligence (AI) that allows computers to understand human language, whether written, spoken, or even scribbled. Natural language processing (NLP) is a subset of artificial intelligence, computer science, and linguistics focused on making human communication, such as speech and text, understandable to computers.

Natural language processing (NLP) is a subfield of computer science and [artificial intelligence (AI)](https://www.ibm.com/think/topics/artificial-intelligence) that uses [machine learning](https://www.ibm.com/think/topics/machine-learning) to enable computers to understand and communicate with human language.

Natural language processing (NLP) is technology that allows computers to interpret, manipulate, and comprehend human language. Organizations today have large volumes of voice and text data from various communication channels like emails, text messages, social media newsfeeds, video, audio, and more. Natural language processing is key in analyzing this data for actionable business insights. Organizations can classify, sort, filter, and understand the intent or sentiment hidden in language data. Natural language processing is a key feature of AI-powered automation and supports real-time machine-human communication.

Natural language processing (NLP) uses machine learning to reveal the structure and meaning of text. With natural language processing applications, organizations can analyze text and extract information about people, places, and events to better understand social media sentiment and customer conversations. As a branch of artificial intelligence, NLP (natural language processing), uses machine learning to process and interpret text and data. Natural language recognition and natural language generation are types of NLP.

NLP enables computers to understand natural language as humans do. Whether the language is written or spoken, natural language processing uses artificial intelligence to take real‑world input, process it, and make sense of it in a way a computer can understand. So how does it work? NLP has two main phases, data preprocessing and algorithm development. Data preprocessing involves preparing and cleaning text data for machines to be able to analyze it. Preprocessing puts data in workable form and highlights features in the text that an algorithm can work with. First is **tokenization**. This is when text is broken down into smaller units. Tokenization can separate sentences, words, characters, or subwords. This is the first step in any NLP pipeline. Then, we have **stop word removal**. This is when common words such as a, the, how, from, etcetera are removed from text so unique words that offer the most information about the text remain in place. Then we have **lemmatization and stemming**. This is when words are reduced to their root forms to process. And finally, we have **part‑of‑speech tagging**. This is when words are marked based on part of speech they are, such as nouns, verbs, and adjectives. Once the data has been preprocessed, an algorithm is developed to process it. There are many different natural language processing algorithms, but two main types are commonly used. One is a rule‑based system. Rule‑based systems rely on handcrafted grammatical rules that need to be created by experts in linguistics or knowledge engineers. This was the earliest approach to crafting NLP algorithms, and it's still used today. Then, we have machine learning‑based system. Machine learning algorithms use statistical methods. They learn to perform tasks based on training data they are fed and adjust their methods as more data is processed. Using a combination of machine learning, deep learning, and neural networks, natural language processing algorithms hone their own rules to repeated processing and learning. Natural language processing ensures that AI can understand the natural human languages we speak every day. ss

**NLP Use Cases**  
Natural language processing applications are used to derive insights from unstructured text-based data and give you access to extracted information to generate new understanding of that data. Natural language processing examples can be built using Python, TensorFlow, and PyTorch.

**Customer sentiment**

Use entity analysis to find and label fields within documents and channels to better understand customer opinions and find product and UX insights.

**Receipt and invoice understanding**

Extract entities to identify common entries in receipts and invoices, like dates or prices, to understand relationships between request and payment.

**Document analysis**

Use custom entity extraction to identify domain-specific entities within documents without having to spend time or money on manual analysis.

**Content classification**

Classify documents by common entities, domain-specific customized entities, or 700+ general categories, like sports and entertainment.

**Trend spotting**

Aggregate news with text that lets marketers extract relevant content about their brands from online news, articles, and other data sources.

**Healthcare**

Improve clinical documentation, data mining research, and automated registry reporting to help accelerate clinical trials.

**What are NLP tasks?**

Natural language processing (NLP) techniques, or NLP tasks, break down human text or speech into smaller parts that computer programs can easily understand. Common text processing and analyzing capabilities in NLP are given below.

**Part-of-speech tagging**

This is a process where NLP software tags individual words in a sentence according to contextual usages, such as nouns, verbs, adjectives, or adverbs. It helps the computer understand how words form meaningful relationships with each other.

**Word-sense disambiguation**

Some words may hold different meanings when used in different scenarios. For example, the word "bat" means different things in these sentences:

* A bat is a nocturnal creature.
* Baseball players use a bat to hit the ball.

With word sense disambiguation, NLP software identifies a word's intended meaning, either by training its language model or referring to dictionary definitions.

**Speech recognition**

Speech recognition turns voice data into text. The process involves breaking words into smaller parts and understanding accents, slurs, intonation, and nonstandard grammar usage in everyday conversation. A key application of speech recognition is transcription, which can be done using speech-to-text services like [Amazon Transcribe](https://aws.amazon.com/pm/transcribe/).

**Machine translation**

Machine translation software uses natural language processing to convert text or speech from one language to another while retaining contextual accuracy. The AWS service that supports machine translation is [Amazon Translate](https://aws.amazon.com/translate/).

**Named-entity recognition**

This process identifies unique names for people, places, events, companies, and more. NLP software uses named-entity recognition to determine the relationship between different entities in a sentence.

Consider the following example: "Jane went on a vacation to France, and she indulged herself in the local cuisine."

The NLP software identifies "Jane" and "France" as the special entities in the sentence. This can be further expanded by co-reference resolution, determining if different words are used to describe the same entity. In the above example, both "Jane" and "she" pointed to the same person.

**Sentiment analysis**

Sentiment analysis is an artificial intelligence-based approach to interpreting the emotion conveyed by textual data. NLP software analyzes the text for words or phrases that show dissatisfaction, happiness, doubt, regret, and other hidden emotions.

**How does NLP work?**

Typically, NLP implementation begins by gathering and preparing unstructured text or speech data from sources like cloud data warehouses, surveys, emails, or internal business process applications.

**Pre-processing**

The NLP software uses pre-processing techniques such as tokenization, stemming, lemmatization, and stop word removal to prepare the data for various applications.

Here's a description of these techniques:

* Tokenization breaks a sentence into individual units of words or phrases.
* Stemming and lemmatization simplify words into their root form. For example, these processes turn "starting" into "start."
* Stop word removal ensures that words that do not add significant meaning to a sentence, such as "for" and "with," are removed.

**Training**

Researchers use the pre-processed data and machine learning to train NLP models to perform specific applications based on the provided textual information. Training NLP algorithms requires feeding the software with large data samples to increase the algorithms' accuracy.

**Deployment and inference**

AI experts then deploy the model or integrate it into an existing production environment. The NLP model receives input and predicts an output for the specific use case the model is designed for. You can run the NLP application on live data and obtain the required output.

**What are the technologies in NLP?**

Natural language processing (NLP) combines computational linguistics, predictive artificial intelligence, and[deep learning](https://aws.amazon.com/what-is/deep-learning/) models to process human language.

**Computational linguistics**

Computational linguistics is the science of understanding and constructing human language models with computers and software tools. Researchers use computational linguistics methods, such as syntactic and semantic analysis, to create frameworks that help machines understand conversational human language. Tools like language translators, [text-to-speech](https://aws.amazon.com/polly/what-is-text-to-speech/) synthesizers, and speech recognition software are based on computational linguistics.

**Predictive AI**

Predictive AI, also called [machine learning](https://aws.amazon.com/what-is/machine-learning/) or deep learning, is a technology that trains a computer with sample data to perform specific tasks. It involves a [neural network](https://aws.amazon.com/what-is/neural-network/) that consists of data processing nodes structured to resemble the human brain. With deep learning, computers recognize, classify, and correlate complex patterns in the input data.

Human language has several features like sarcasm, metaphors, variations in sentence structure, plus grammar and usage exceptions that take humans years to learn. Programmers use predictive methods to teach NLP applications to recognize and accurately understand these features from the start.

Traditional neural networks deal with data sequences using an encoder/decoder architecture pattern. The encoder reads and processes the entire input data sequence, such as an English sentence, and transforms it into a compact mathematical representation. This representation is a summary that captures the essence of the input. Then, the decoder takes this summary and, step by step, generates the output sequence. This could be the same sentence in another language, or information about sentence intent and sentiment.

**Generative AI** AI technology uses transformers - neural networks that incorporate a self-attention mechanism. Instead of processing data in order, the mechanism enables the model to look at different parts of the sequence all at once and determine which parts are most important.

Because of self-attention, transformers can learn from larger datasets and process very large text where context from far back influences the meaning of what's coming next.

**Applications of Natural Language Processing (NLP)**

* **Analyzing customer feedback:** NLP has become an essential business tool for uncovering hidden data insights from social media channels. Sentiment analysis can analyze language used in social media posts, responses, reviews, and more to extract attitudes and emotions in response to products, promotions, and events, information companies can use in product designs, advertising campaigns, and more.
* **Virtual assistants or chatbots:** We've talked about how virtual assistants that live on your phone or TV help you out in your daily lives. While they work with voice mainly, chatbots perform the same function with typed text queries. The top‑of‑the line stuff also learn to recognize contextual clues about human requests and use them to provide even better responses or options over time.
* **Spam detection:**One of the most irritating things about email is spam. Gmail uses natural language processing (NLP) to discern which emails are legitimate and which are spam. You may not think of spam detection as an NLP solution, but the best spam detection technologies use NLP's text classification capabilities to scan emails for language that often indicates spam or phishing. These indicators can include overuse of financial terms, characteristic bad grammar, threatening language, inappropriate urgency, misspelled company names, and more.These spam filters look at the text in all the emails you receive and try to figure out what it means to see if it's spam or not.
* **Algorithmic Trading:**Algorithmic trading is used for predicting stock market conditions. Using NLP, this technology examines news headlines about companies and stocks and attempts to comprehend their meaning in order to determine if you should buy, sell or hold certain stocks.
* **Questions Answering:**NLP can be seen in action by using Google Search or Siri Services. A major use of NLP is to make search engines understand the meaning of what we are asking and generate natural language in return to give us the answers.
* **Summarizing Information:**On the internet, there is a lot of information and a lot of it comes in the form of long documents or articles. NLP is used to decipher the meaning of the data and then provides shorter summaries of the data so that humans can comprehend it more quickly.

**Benefits of NLP**

NLP makes it easier for humans to communicate and collaborate with machines, by allowing them to do so in the natural human language they use every day. This offers benefits across many industries and applications.

* Automation of repetitive tasks
* Improved data analysis and insights
* Enhanced search
* Content generation

**Why is NLP important?**Businesses these days use massive quantities of text‑heavy, unstructured data and need a way to efficiently process it. This is where NLP is useful.

Generative AI is a subset of ML that generates new content meaningfully and intelligently.

**Generative AI** is a type of artificial intelligence (AI) that creates new content, such as text, images, music, code, or video, by learning patterns from large datasets. Generative AI, sometimes called *gen AI,* is [artificial intelligence](https://www.ibm.com/think/topics/artificial-intelligence) (AI) that can create original content such as text, images, video, audio or software code in response to a user’s prompt or request. Generative AI relies on sophisticated [machine learning](https://www.ibm.com/think/topics/machine-learning) models called [*deep learning*](https://www.ibm.com/think/topics/deep-learning) *models*algorithms that simulate the learning and decision-making processes of the human brain. Generative AI uses the power of deep learning to create new content spanning text, images, audio, and beyond. Deep learning techniques, particularly those centered on neural networks, are the engine behind these generative models.

Generative artificial intelligence (generative AI) is a type of AI that can create new content and ideas, including conversations, stories, images, videos, and music. It can learn human language, programming languages, art, chemistry, biology, or any complex subject matter. It reuses what it knows to solve new problems. For example, it can learn English vocabulary and create a poem from the words it processes. Your organization can use generative AI for various purposes, like chatbots, media creation, product development, and design.

Generative AI took the neural networks of machine learning and deep learning to the next level. While machine learning and deep learning focus on prediction and pattern recognition, generative AI produces unique outputs based on the patterns it detects. Generative AI technology is built on[transformer architecture](https://aws.amazon.com/what-is/transformers-in-artificial-intelligence/) that combines several different neural networks to combine data patterns in unique ways. Deep learning networks first convert text, images, and other data into mathematical abstractions and then reconvert them into meaningful new patterns.

Generative AI, is a subset of machine learning that focuses on creating new content, like images, text, or music. while other types of AI might analyze data to predict an outcome, like whether an email is spam or not, generative AI uses data to create something entirely new. gen AI models learn the underlying patterns and structures of the data they are trained on, and then use what they learned to generate new, similar content. To learn these patterns and structures well enough, these models need so much data. **Generative AI:**An application of AI that creates new content.

**Foundation Models**

Foundation models use deep learning. They are trained on massive datasets that allow them to learn complex patterns and perform a variety of tasks across different domains. They are incredibly powerful machine learning models trained on a massive scale, often using vast amounts of unlabeled data. This training allows them to develop a broad understanding of the world, capturing intricate patterns and relationships within the data they consume. Large language models are probably the most famous type of foundation model.

Foundation models are large AI models trained on enormous datasets. This broad training allows them to develop a deep understanding of the data and be adapted to a wide range of downstream tasks. Gemini is a gen ai model developed by Google.   
Google's suite of foundation models: Gemini, Gemma, Imagen, Chirp, Veo

**Gemini**: Gemini, a multimodal model, can understand and operate across diverse data formats, such as text, images, audio, and video. Gemini's multimodal design supports applications that require complex multimodal understanding, advanced conversational AI, content creation, and nuanced question answering.

**Gemma**: A family of lightweight, open models is built upon the research and technology behind Gemini. They offer developers a user-friendly and customizable solution for local deployments and specialized AI applications.

**Imagen**: A powerful text-to-image diffusion model, it excels at generating high-quality images from textual descriptions. This makes it invaluable for creative design, ecommerce visualization, and content creation.

**Veo**: A model capable of generating video content. It can produce videos based on textual descriptions or still images. Its functionality allows for the creation of moving images for applications such as film production, advertising, and online content.

Collectively, these Google Cloud foundation models empower businesses to enhance customer experiences through intelligent chatbots and personalized content. They increase productivity by automating tasks and improving information retrieval. They foster innovation by generating new ideas and designs. They also derive data-driven insights for improved decision-making.

Think of a foundation model as a highly advanced learner. It's like a student who has read everything in an entire library, absorbing knowledge from countless books, articles, and websites. This deep and extensive learning allows foundation models to be adapted to a wide range of tasks.

**What are the types of foundation models?**

Foundation models encompass various architectures, each designed with unique strengths and applications. Here are a few notable types:

Large language models (LLMs): These models specialize in understanding and generating human language, excelling in tasks like translation, text summarization, and chatbot interactions.

Multimodal models: Trained on diverse data types, including text, images, and audio, these models can analyze and generate content across multiple modalities.

Generative adversarial networks (GANs): GANs are a type of foundation model involving two neural networks contesting with each other in a zero-sum game. One network, the generator, makes new data instances, while the other, the discriminator, assesses their authenticity. This adversarial process leads to the generation of increasingly realistic and complex content.

Computer vision models: These models are trained on image datasets to perform tasks like image classification, object detection, and image generation. They can be fine-tuned for specific applications, such as medical image analysis or object recognition in autonomous vehicles.

**Large Language Model (LLM)**

One particularly exciting type of foundation model is the LLM. These models are specifically designed to understand and generate human language.

They can translate languages, write different kinds of creative content, and answer your questions in an informative way, even if they are open-ended, challenging, or strange. This is likely the most common foundation model you've encountered, such as in popular generative AI chatbots like Gemini. They also help power many search engines you use today.

A **Large Language Model (LLM)** is a type of artificial intelligence model, specifically a deep learning model, trained on massive amounts of text data to understand and generate human-like language. They are used for a wide range of natural language processing tasks, including language generation, translation, and question answering.

In recent years, two of the most exciting advancements in AI have been generative AI and large language models (LLMs). However, the frontier is rapidly expanding with the emergence of AI agents and agentic AI, which represent a significant step towards more autonomous and capable AI systems.

* [**Generative AI**](https://cloud.google.com/use-cases/generative-ai): This is a type of AI that doesn't just analyze data; it creates new content. Think of it as an AI artist, writer, or even coder. Generative AI learns the patterns and structures within vast amounts of data (text, images, code, and more.) and then uses that knowledge to produce entirely new, original content based on prompts. Tools like DALL-E for images and ChatGPT for text are prime examples.
* [**Large Language Models (LLMs)**](https://cloud.google.com/ai/llms): These are the engines powering many of today's most sophisticated AI applications, especially in text-based tasks. LLMs are large AI models trained on massive datasets of text and code. They excel at understanding, generating, and manipulating human language. Because they've processed so much information, they can answer complex questions, summarize documents, translate languages, write creative content, and even generate computer code. These models are becoming increasingly capable, even developing "emergent abilities" like solving math problems and writing code, though it's always wise for developers to review and validate AI-generated code. LLMs are also becoming multimodal, meaning they can understand and process not just text, but also images, audio, and video.
* [**AI agents**](https://cloud.google.com/discover/what-are-ai-agents): These are AI systems designed to perceive their environment, make decisions, and take actions to achieve specific goals. Unlike a simple chatbot that responds to a direct command, an AI agent can:
* **Plan**: Break down a complex goal into a series of smaller, manageable steps
* **Reason**: Use its knowledge and understanding to make decisions at each step
* **Act**: Interact with digital or even physical environments (through APIs or robotic interfaces) to carry out its plan
* **Learn/Adapt**: Potentially learn from its experiences to improve its performance over time
* [**Agentic AI**](https://cloud.google.com/discover/what-is-agentic-ai): This refers to the capability of AI systems to operate autonomously in the manner described above.

Popular Large Language Models (LLMs) in 2026 include proprietary leaders like OpenAI's GPT-4o, Anthropic's Claude 3.5 Sonnet, and Google's Gemini 2.0 Flash/Pro, known for advanced reasoning and multimodal capabilities. Strong open-source/open-weight options include Meta’s Llama 3.3, Mistral, and DeepSeek-V3. These models are heavily used for coding, chat, and reasoning tasks.

The relationship between **LLM (Large Language Model)** and **foundation model** is that **LLMs are a subset of foundation models**. Here's a clearer breakdown of how they relate:

**Foundation Models:**

A **foundation model** refers to any large, pre-trained model designed to serve as a base for a variety of tasks. These models are trained on vast amounts of data and are typically adaptable to multiple applications, meaning they can be fine-tuned or extended for specific tasks across different domains (e.g., language, vision, audio). Foundation models are intended to be general-purpose, often leveraging a multi-modal or multi-task approach.

Examples of foundation models

The foundation model ecosystem is vibrant and competitive. Here are some of the most influential examples from key industry players:

* **Google**: Known for the Gemini family, a series of powerful multimodal models (Gemini 2.5 Pro is a leading example), and Gemma, a family of open-weight, lightweight models for developers; Google has also developed specialized models like Imagen for text-to-image generation and Veo for video generation
* **OpenAI**: The developer of the highly influential GPT (Generative Pre-trained Transformer) series, including the widely used GPT-4
* **Anthropic**: Focuses on AI safety and has developed the Claude family of models; the Claude 3 series (including Opus, Sonnet, and Haiku) is known for its large context windows and strong reasoning capabilities
* **Meta**: A major proponent of open source AI, Meta developed the Llama series; Llama 3 is an open model that has accelerated innovation across the entire community
* **Mistral AI**: A European company that has gained significant traction with high-performing open and commercial models, such as Mistral Large and the open source Mixtral models which use a Mixture-of-Experts (MoE) architecture for greater efficiency

Examples of foundation models include:

* **GPT-3** (for language tasks)
* **CLIP** (for both text and image understanding)
* **DALL·E** (for generating images from text descriptions)
* **BERT** (language understanding for tasks like question-answering, classification, etc.)

**Large Language Models (LLMs):**

An **LLM** is a specific type of foundation model designed primarily for natural language tasks. LLMs are trained on enormous datasets of text and have the ability to generate, understand, and process human language. They are typically **unidimensional**, focused on language processing, unlike foundation models that may span multiple modalities.

Examples of LLMs:

* **GPT (Generative Pre-trained Transformer)** series
* **BERT (Bidirectional Encoder Representations from Transformers)**
* **T5 (Text-to-Text Transfer Transformer)**

**Relationship:**

1. **LLMs are a subset of foundation models**: All LLMs are foundation models, but not all foundation models are LLMs. LLMs specifically focus on natural language tasks, whereas foundation models could be applied across different domains such as vision (image-based tasks), audio, or multi-modal tasks (e.g., processing both text and images).
2. **Common Traits**: Both LLMs and foundation models share key characteristics—massive scale, pre-training on vast datasets, and the ability to be fine-tuned for specific tasks or domains. Both are designed to generalize across a wide range of tasks without needing to be retrained from scratch for every new use case.

**In Summary:**

* **LLMs** are large language models trained specifically for language tasks (a subset of foundation models).
* **Foundation models** are broader, general-purpose models that can handle a variety of tasks, potentially across different modalities (e.g., text, images, audio), with LLMs being a specialized category within this broader framework.

So, an LLM like GPT-3 could be considered a foundation model in the domain of natural language processing, while models like CLIP or DALL·E are multi-modal foundation models that handle both language and images.

**What is the difference between a foundation model and an LLM?**

The terms "foundation model" and "[large language model](https://cloud.google.com/ai/llms)" (LLM) are often used interchangeably, but there's a key distinction. **LLMs are a major type of foundation model, but they aren't the only kind. Think of it as a parent-child relationship: all LLMs are foundation models, but not all foundation models are LLMs.**

The key difference is the type of data they're built on. LLMs, as the name implies, are trained specifically on vast amounts of text and code. The broader category of 'foundation models' also includes models trained on other data types, such as images, audio, and video, or a combination of them (multimodal).

**Diffusion models:**  
  
Diffusion models are another type of foundational model. They excel in **generating high-quality images, audio, and even video** by iteratively refining noise (or unstructured/random data and patterns) into structured data.

Imagen is a powerful text-to-image diffusion model, it excels at generating high-quality images from textual descriptions. This makes it invaluable for creative design, ecommerce visualization, and content creation.

**Foundation model limitations**

Foundation models, while groundbreaking, aren't without limitations. Recognizing these limitations is essential for the responsible and effective utilization of these powerful tools. Foundation models have limitations, including data dependency, knowledge cut-offs, biases, potential for hallucinations, and issues with fairness and edge cases. Techniques like grounding, prompt engineering, fine-tuning, and human-in-the-loop systems can address these challenges and improve the accuracy, reliability, and fairness of foundation models.

**Data dependency**: The performance of foundation models is heavily data-dependent. They require large datasets, and any biases or incompleteness in that data will inevitably seep into their outputs. It's like asking a student to write an essay on a book they haven't read. If the data or questions are inaccurate or biased, the AI's performance will suffer.  
  
**Knowledge cutoff**: Knowledge cutoff is the last date that an AI model was trained on new information. Models with older knowledge cutoffs may not know about recent events or discoveries. This can lead to incorrect or outdated answers, since AI models don't automatically update with the latest happenings around the world. For example, if an AI tool's last training date was in 2022, it wouldn't be able to provide information about events or information that happened after 2022.

**Bias**: An LLM learns from large amounts of data, which may contain biases. You can think of bias as an unbalanced dataset in LLMs. Due to their statistical learning nature, they can sometimes amplify existing biases present in the data. Even subtle biases in the training data can be magnified in the model's outputs.

**Fairness**: Even with perfectly balanced data, defining what constitutes fairness in an LLM's output is a complex task. Fairness can be interpreted in various ways. Fairness assessments for generative AI models, while valuable, have inherent limitations. These evaluations typically focus on specific categories of bias, potentially overlooking other forms of prejudice. Consequently, these benchmarks do not provide a complete picture of all potential risks associated with the models' outputs, highlighting the ongoing challenge of achieving truly equitable AI.  
  
**Hallucinations**: Foundation models can sometimes experience hallucinations, which means they produce outputs that aren't accurate or based on real information. Because foundation models can't verify information against external sources, they may generate factually incorrect or nonsensical responses. These cause significant concern in accuracy-critical applications. The responses might sound convincing, but they are completely wrong. We will cover this in more detail below.  
  
**Edge cases**: Rare and atypical scenarios can expose a model's weaknesses, leading to errors, misinterpretations, and unexpected results.

**Techniques to overcome limitations**

Fortunately, several techniques can significantly improve foundation model performance. Let's explore some of the key approaches.

**1) Grounding**: Generative AI models are amazing at creating content, but sometimes they hallucinate. That's where grounding comes in. It's like a reality check, keeping the AI aligned with how things actually work. Think of it as reminding the AI that cats don't talk and people can't fly unaided. Grounding is the process of connecting the AI's output to verifiable sources of information—like giving AI a reality check. Grounding refers to the ability of the AI model to connect its output to verifiable and specific sources of information. This is a critical aspect of building trust and reliability in AI systems. By providing the model with access to specific data sources, we tether its output to real-world information, reducing the risk of invented content.

Grounding is basically tying an AI model's answers to specific sources. This makes the AI's output more accurate and relevant. In an enterprise context, this means training the AI on a company's specific data. This allows it to generate tailored responses relevant to your organization, not just generic ones. This data could include anything from internal documents and reports to customer data, and even code. It gives the AI the context it needs to understand the company's unique language and provide accurate, relevant responses.

Grounding is essential for building trustworthy and reliable AI applications. By connecting your models to verifiable data, you ensure accuracy and build confidence. It offers several key benefits, including reducing hallucinations, which prevents the AI from generating false or fictional information. Grounding also anchors responses, ensuring the AI's answers are rooted in your provided data sources. Furthermore, it builds trust by enhancing the trustworthiness of the AI's output by providing citations and confidence scores, allowing you to verify the information.

**2)** **Retrieval-augmented generation (RAG)**: There are many different options on how you can ground in data. For example, you can ground in enterprise data or you can ground using Google Search. One common grounding method to do this is with **retrieval-augmented generation**, or **RAG**. One powerful grounding technique is retrieval-augmented generation (RAG). It involves:

1. **Retrieving relevant information:**The AI model first retrieves relevant information from a vast knowledge base (like a database, a set of documents, or even the entire web). This retrieval process is often powered by sophisticated techniques, like [semantic search](https://cloud.google.com/discover/what-is-semantic-search?hl=en) or [vector databases](https://cloud.google.com/discover/what-is-a-vector-database).
2. **Generating output:** The model then uses this retrieved information to generate the final output. This could be anything from answering a question to writing a creative story.

RAG is a grounding method that uses search to find relevant information from a knowledge base and provides that information to the LLM, giving it necessary context.

The first step is retrieval. When you ask an AI a question, RAG uses a search engine to find relevant information. This search engine uses an index that understands the semantic meaning of the text, not just keywords. This means it finds information based on meaning, ensuring higher relevance.

The retrieved information is then added to the prompt given to the AI. This is the augmentation phase.

The AI then uses this augmented prompt, along with its existing knowledge, to generate a response. This is referred to as the generation phase.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

RAG empowers AI models to go beyond simply memorizing information and instead learn to effectively access and utilize external knowledge sources to generate more insightful and reliable outputs.

* **Improved accuracy and relevance:**By accessing and utilizing relevant information directly, RAG models can produce more accurate and informative outputs.
* **Improved explainability and transparency:**RAG can increase transparency and trust in the AI system by showing the specific sources used to generate the output, ensuring that claims can be checked for accuracy.
* **Extending LLM capabilities:**RAG can constrain the output from an LLM so that it only generates a response based on the specific context it was provided.

One way to start using a RAG system without any coding or database development is with a tool called NotebookLM.

**3)** **Prompt engineering:** Prompting offers the most rapid and straightforward approach to supplying supplementary background information to models. This involves crafting precise prompts to guide the model towards desired outputs. It refines results by understanding the factors that influence a model's responses. However, prompting is limited by the model's existing knowledge; it can't conjure information it hasn't learned.

Prompt engineering is the iterative process of designing, refining, and optimizing text-based inputs (prompts) to guide AI models, such as LLMs (e.g., GPT-4, Gemini), to generate accurate, relevant, and high-quality outputs. It bridges the gap between AI capabilities and practical, creative, or analytical applications.

**Key Aspects of Prompt Engineering:**

Goal: To minimize irrelevant, incorrect, or low-quality responses by providing the AI with specific, structured, and contextual instructions.

**Techniques:**

Zero-shot/Few-shot Prompting: Asking for a task with no examples or giving a few examples of desired output.

Chain-of-Thought (CoT): Prompting the model to break down problems into logical, sequential steps, which improves reasoning and accuracy.

Prompt Chaining: Breaking complex, large tasks into smaller, manageable, sequential subtasks.

Persona Adoption: Instructing the AI to act in a specific role (e.g., "Act as a Python expert").

**Components of an Effective Prompt:** A well-engineered prompt often includes a clear task, specific context, a persona, required format, and, if needed, examples.

**Role/Career:** A prompt engineer tests and tunes these inputs to optimize performance across various fields, including creative writing, coding, and data analysis.

Essentially, it is the art of communicating with AI to get the best possible results through, trial-and-error, and, at times, more advanced, structured approaches.

4) **Fine-tuning**: When prompt engineering doesn't deliver the desired outcomes, fine-tuning can enhance your model's performance. Pre-trained models are powerful, but they're designed for general purposes. Tuning helps them excel in specific areas. This process is particularly useful for specific tasks or when you need to enforce specific output formats, especially if you have examples of the desired output.

Tuning involves further training a pre-trained or foundation model on a new dataset specific to your task. This process adjusts the model's parameters, making it more specialized for your needs. Google Cloud Vertex AI provides tooling to facilitate tuning.

Here are some examples of how tuning can be used:

* Fine-tuning a language model to generate creative content in a specific style.
* Fine-tuning a code generation model to generate code in a particular programming language.
* Fine-tuning a translation model to translate between specific languages or domains.

A screenshot of a computer

AI-generated content may be incorrect.

**Humans in the loop (HITL)**

Beyond these techniques, we must remember the invaluable role of humans in the loop (HITL). Machine learning models are powerful, but they sometimes need a human touch. For tasks requiring judgment, context, or handling incomplete data, human expertise is essential. HITL systems integrate human input and feedback directly into the ML process. This collaboration makes models more adaptable, especially in areas like the following.

**Content moderation**: HITL ensures accurate and contextually appropriate moderation of user-generated content, filtering out harmful or inappropriate material that algorithms alone might miss.

**Sensitive applications**: In fields like healthcare or finance, HITL provides oversight for critical decisions, ensuring accuracy and mitigating risks associated with automated systems.

**High-risk decision making**: When ML models inform decisions with significant consequences, such as medical diagnoses or criminal justice assessments, HITL acts as a safeguard, providing a layer of human review and accountability.

**Pre-generation review**: Before deploying ML-generated content or decisions, human experts can review and validate the outputs, catching potential errors or biases before they impact users.

**Post-generation review**: After ML outputs are deployed, continuous human review and feedback help identify areas for improvement, enabling models to adapt to evolving contexts and user needs.

**Secure AI**

Secure AI is about preventing intentional harm being done to your applications. This is about protecting AI systems from malicious attacks and misuse. For all applications, including AI, you need to ensure security throughout the full lifecycle from development through deployment. This includes considering the data, infrastructure, and how and where applications are deployed. AI security is a dual field: securing AI systems (models, data, infrastructure) from threats like tampering and misuse, and using AI itself (like machine learning) to enhance overall cybersecurity by automating threat detection, analysis, and response for better defense. It protects the entire AI lifecycle, ensuring AI applications are trustworthy, data integrity is maintained, and systems operate reliably and ethically against evolving risks.

AI offers significant benefits, but it also introduces security risks like data poisoning, model theft, and prompt injection. To address these challenges, a secure foundation for AI applications is essential. **Google Cloud's** **Secure AI framework (SAIF)**, combined with security tools, can help as you build and maintain secure AI systems.

Google has developed the **Secure AI Framework**, or SAIF, to establish security standards for building and deploying AI systems responsibly. This comprehensive approach to AI/ML model risk management addresses the key concerns of security professionals in the rapidly evolving landscape of AI. Platforms such as Google Cloud can facilitate secure development. In addition to the Secure AI Framework, Google Cloud offers a range of tools to ensure applications remain secure throughout their lifecycle. Google Cloud has built security into its core through a secure-by-design infrastructure, encompassing its global network and hardware, as well as robust encryption in transit and at rest. It provides customers with detailed control over access and usage of cloud resources through Identity and Access Management (IAM). Google Cloud Security Command Center provides a centralized view of your security posture across your entire Google Cloud environment, and Google Cloud also offers tools for monitoring various workloads.

**Responsible AI**

Responsible Artificial Intelligence (Responsible AI) is an approach to developing, assessing, and deploying AI systems safely, ethically, and with trust. Responsible AI is the practice of designing, developing, and using AI technology with the goal of maximizing benefits and minimizing risks.

Ensuring your AI applications don’t cause harm and are used in an ethical manner. Responsible AI needs to be considered throughout the entire AI lifecycle, from data preparation and model training to deployment and ongoing monitori

Responsible AI is a framework for designing, developing, and deploying artificial intelligence systems ethically and safely, ensuring they are fair, transparent, reliable, secure, and accountable, aligning with human values and societal well-being while mitigating potential harm, bias, and negative impacts like privacy violations or discrimination.

Explainable AI makes the decision-making processes of AI models transparent and understandable. This is crucial for building trust, debugging errors, and uncovering hidden biases. Think of it like a judge explaining their verdict; without a clear explanation, it's hard to trust the decision. Tools like Google Cloud’s Vertex Explainable AI can help understand model outputs and identify potential biases.

Developing responsible AI requires a deep understanding of potential issues and unintended consequences. It's important to consider safety, security, and ethical implications—and to actively mitigate potential issues—at every stage of the AI development lifecycle. With the right guardrails, AI can benefit society rather than causing unintended harm.

Responsible AI is the practice of designing, developing, and deploying Artificial Intelligence systems ethically, safely, and transparently, ensuring they align with human values, societal well-being, and legal standards, focusing on fairness, accountability, reliability, privacy, and transparency to prevent harm and build trust. It's about putting ethical principles into tangible AI experiences, making sure AI benefits humanity, not harms it, by addressing potential biases, security risks, and societal impacts.

**Layers of Gen AI (The building blocks of generative AI)  
 :** Generative AI is a powerful technology composed of interconnected layers (from the foundational infrastructure to the user-facing applications), each playing a vital role in its capabilities.  
The Gen AI is composed of five layers. They are infrastructure, models, platform, agents, and gen AI powered applications (user-facing part of generative AI or the frontend). From the foundational infrastructure to the user-friendly applications, each layer of generative AI plays a crucial role in its capabilities.

A diagram of a diagram

AI-generated content may be incorrect.

Think of generative AI as a powerful engine. The infrastructure is its foundation, the models are its core, and the platform connects everything together. Agents act as the drivers, making decisions and taking action, while applications are the vehicles that take us to our destination.

**1) Infrastructure:** The infrastructure layer is the foundation of any AI system upon which everything else rests. It provides the core computing resources needed for generative AI. This includes the physical hardware (like servers, GPUs, and TPUs) and software (operating systems, networking) that provide the necessary computing power, storage, and networking capabilities to train, deploy, and scale AI models. Google has invested heavily in building a robust and cutting-edge AI infrastructure, designed to handle the most demanding AI workloads.

**2) Models:** The "brain" of the agent is the AI model. These models are complex algorithms trained on vast amounts of data. They learn patterns and relationships in the data, allowing them to generate new content, translate languages, answer questions, and much more.  Examples are large language models (LLMs) like Gemini, image recognition models, and recommendation systems.

**3) Platform:** The platform layer sits above the model and infrastructure layers, providing the necessary tools and infrastructure for AI development. It offers APIs, model training platforms like Vertex AI, data management capabilities, and model deployment tools. This layer acts as the backbone of the system, bridging the gap between models and agents while simplifying the complexities of managing the underlying infrastructure. The platform layer provides the foundation for building and scaling your AI initiatives

**4) Agents:** An agent is a piece of software that learns how to best achieve a goal based on inputs and tools available to it. This layer focuses on autonomous action, which describes the ability to independently set goals and carry them out within a defined environment. Agents analyze situations, use multiple tools, and make informed decisions without requiring constant human input. They are also capable of handling multi-step tasks that a model alone cannot, such as researching a topic, troubleshooting code, or accessing a system by chaining together actions. You can have a variety of agents, such as customer agents, code agents, data agents, and many more.

**5) GenAI Applications:** This is the layer that delivers the AI capability to users through interfaces. It's the user-facing part of generative AI, or the frontend. This is the layer that allows users to interact with and leverage the capabilities of AI. Examples of gen AI powered applications include the Gemini app, Google Workspace with Gemini, or NotebookLM.

**Gen AI Agents**

Gen AI agents can process information, reason over complex concepts, and take actions. They can also be used in a variety of applications, including customer service, employee productivity, and creative tasks. Agents enhance applications by adding intelligence and automation and the applications provide the framework and purpose for agents. A gen AI agent is an application that tries to **achieve a goal by observing the world and acting upon it** using the tools it has at its disposal.

With generative AI, you have the flexibility to leverage either pre-built agents and applications, or create custom ones tailored to your specific needs. No matter which path you choose, you can apply generative AI's capabilities in a way that best aligns with your objectives.

**Role of agents in gen AI powered applications**Think of gen AI powered applications as the user-facing layer. They provide the structure and context for agents to operate within. They define the user interface, the overall goals, and the specific tasks that agents will help with. Often, multiple agents with different specializations work together within a single application to create a richer, more dynamic user experience. This is known as a multi-agent system. You can review some examples below.

**A Travel booking app:** An agent could handle the complex task of finding the best flights and hotels based on user preferences, while another agent might specialize in suggesting relevant activities and attractions at the destination. Then, the application provides the user with the interface for browsing options and making reservations.  
**A Customer support app:** An agent could answer common questions, troubleshoot problems, and escalate complex issues to human representatives. The application would provide the chat interface and integrate with other support systems. **A Personalized learning app:** An agent could assess a student's knowledge, recommend relevant learning materials, and even generate personalized exercises. The application would provide the structure for lessons and track progress.

While there are many ways to use gen AI agents, most will fall under two categories: **conversational** agents or **workflow** agents.

**1) Conversational agents**:   
**How it works:**   
Conversational agents are designed to understand what you mean, not just what you say, and respond in a way that makes sense.

* **You provide input:**You can type a message or speak to the agent.
* **The agent understands:**Using powerful AI, it figures out the meaning and intention behind your words.
* **The agent calls a tool:**Based on your request, the agent might need to gather additional information or perform an action. This could involve searching the web, accessing a database, or interacting with another software application.
* **The agent generates a response:** It formulates an answer that's relevant to your request and sounds natural.
* **The agent delivers the response:**You'll see or hear the answer depending on how you interacted with it.

**Examples:**

* **Answering questions:**Imagine you're curious about the weather in Paris. You ask a conversational agent, "What's the weather like in Paris?" And it instantly provides you with the current temperature, forecast, and even suggests what to pack if you're traveling there.
* **Casual conversation:**Feeling bored? A conversational agent can engage in lighthearted banter, tell you jokes, or even discuss your favorite books and movies, just like a friend would.
* **Accessing information:**Need to know the capital of Australia? Or maybe you want a quick summary of the French Revolution? A conversational agent can access a vast amount of world knowledge and provide you with concise and accurate information.

**2) workflow agents:**

**How it works**

**Workflow agents are designed to streamline your work and make sure things get done efficiently and correctly by automating tasks or going through complex processes.**

* You provide input: You define a task or trigger a process like submitting a form, uploading a file, initiating a scheduled event, or even ordering a product online.
* The agent understands: The agent is the software that automates those steps. It interprets the task's requirements and defines the series of steps needed to complete the task.
* The agent calls a tool: Based on the workflow's definition, the agent executes a series of actions. This could involve data transformation, file transfer, sending notifications, integrating with external systems, or initiating other automated processes using APIs.
* The agent generates a result/output: It compiles the outcome of the executed actions, which might be a report, a data file, a confirmation message, or an updated status within a system.
* The agent delivers the result/output: The agent delivers the output to the designated recipient(s) or system(s), such as via email, a dashboard, a database update, or a file storage location.

**Examples**

* Ecommerce order fulfillment: An agent automatically processes orders, updates inventory, sends shipping notifications, and handles returns.
* Customer onboarding: An agent guides new customers through account setup, provides tutorials, and answers frequently asked questions.
* Automated research: An agent can conduct in-depth research on a given topic by autonomously browsing the web, summarizing relevant content, and generating comprehensive reports. (Try this out with [Gemini Deep Research](https://blog.google/products/gemini/google-gemini-deep-research/).)
* Security Log Parsing: An agent that inspects incoming security logs for abnormalities and can flag them, open a ticket, begin triage, and assign to humans for review when necessary.

**Gen AI agents: Beyond just models**

AI agents are designed to observe, act, and achieve goals. While AI agents utilize models, they represent a significant leap beyond their capabilities.

Agents incorporate two key elements that distinguish them from standalone models: a reasoning loop and tools.

A close-up of a person's face

AI-generated content may be incorrect.

The **reasoning loop** is the agent's "thinking process." It's a continuous cycle of observing, interpreting, planning, and acting. This iterative process enables agents to analyze situations, plan actions, and adapt based on outcomes.

A colorful circle with arrows and circles

AI-generated content may be incorrect.

The reasoning loop is the heart of an AI agent's operation. It's an iterative process where the agent:

**Observes**: Gathers information about its environment and the task at hand.

**Interprets**: Processes the information and assesses the current situation.

**Plans**: Plans a course of action to achieve its goal.

**And acts**: Executes the planned action.

This loop continues until the agent reaches its goal or a stopping point. The complexity of the loop varies depending on the agent and its task. Some agents have simple, rule-based processes, while others use more complex logic, potentially involving additional algorithms or probabilistic reasoning.

**Tools** are functionalities that allow the agent to interact with its environment. Tools can be anything from accessing and processing data to interacting with software applications or even physical robots. This empowers agents to connect with real-world information and services, much like apps on our phones.

By combining reasoning and tools, AI agents transcend the limitations of large language models (LLMs). They move beyond text generation to solve complex problems, manage multi-step tasks and produce results beyond just text or media. When solving complex problems, agents analyze information, gather data using tools, and make informed decisions with minimal human intervention. When managing multi-step tasks, agents excel at complex workflows that LLMs alone can't handle, such as conducting in-depth research, troubleshooting code, and automating tasks across multiple systems. AI agents represent a significant leap beyond standard language models by incorporating a **reasoning loop** and **tools** that allow them to interact with their environment. This combination empowers agents to solve complex problems and manage intricate workflows, far surpassing the capabilities of standalone models.

The reasoning loop often utilizes advanced **prompt engineering frameworks** to guide its decision-making process. These frameworks can include:

* Simple rule-based calculations
* Complex thought chains
* Machine learning algorithms
* Probabilistic reasoning techniques

Examples of such frameworks include ReAct or chain-of-thought (CoT) prompting.

**Agents:**

There are generally two kinds of agents: deterministic and generative.

**Deterministic agent**  
A deterministic agent is an agent that is based on predefined paths and actions. It is typically workflow-based and event-driven, and it offers a high degree of control and predictability. A team of people behind the scenes set up very clear directions for the agent on what it has to do every step of the way. These may be very complex directions with lots of decision trees and different logic, but they are deterministic in that the same input will always provide the same output. In addition to being extremely tedious to build, they are not always effective. Unlike generative agents, which have a bit more randomness and creativity and might give different answers to the same input. Deterministic agents only contain one or two components of the agent. The **reasoning loop** and the **tools**. It’s missing the **foundational generative AI model**.

A close-up of a logo

AI-generated content may be incorrect.

It was still using AI to convert your speech to text. Then, some AI for understanding the text and matching it to a list of potential intents of what you meant, usually by matching some common keywords. Then, it might have also used AI to speak back to you. So many of these agents already had AI, but not generative AI. There was no large language model in the background. Which meant the AI was a bit more prescriptive…or deterministic…for that particular use case.

**Generative agents:**Next came generative AI, a whole new world of natural language understanding. Now agents were really brought to the next level. Instead of just recognizing keywords, these agents can actually understand the meaning and intent behind your words. If we go back to the components of the agent, you can now see a **reasoning loop, tooling, and the model** added in there.

A screenshot of a computer

AI-generated content may be incorrect.

Well, the generative AI output from the model just wasn’t necessarily as relevant and catered to the exact use case of the agent nor up to date with the latest information. This was solved with RAG—retrieval augmented generation. This architectural approach enables models to access and integrate information from external data sources. This provides access to more dynamic and up-to-date information, ensuring responses remain grounded in factuality and relevance.

A screenshot of a computer

AI-generated content may be incorrect.

Today, the most complex agents are usually hybrid agents. They have both deterministic and generative capabilities, and the combination is what makes them so powerful. So powerful reasoning loop logic and tooling for the deterministic aspects and for the generative aspects all combined together to form one powerful agent

A diagram of a hybrid agent

AI-generated content may be incorrect.

Today’s generative AI agents can achieve a goal by observing the world and acting upon it using the tools it has at its disposal. This wasn't always the case. Early virtual agents were limited to predefined paths and lacked true understanding. But thanks to advances like generative AI and retrieval augmented generation (RAG), today's agents can understand your intent, access relevant information, and provide much more helpful and natural interactions. All due to the connections of the three components of an agent.

**Components of an agent**

**Foundational model**: This is the underlying language model (LLM) that powers the agent. It could be a small or large language model, a Google model like Gemini, or a model from another provider. The key is to select a model with training data relevant to the agent's intended use case.

**Tools**: Tools enable the agent to interact with the outside world. These can include extensions that connect to APIs, functions that act as mock API calls, and data stores like vector databases. Tools allow the agent to not only observe the world but also act upon it.

**Reasoning loops**: This is the core of the agent, responsible for making decisions and taking actions. It's an iterative process where the agent considers its goal, the available tools, and the information it has gathered. Frameworks like ReAct (Reason and Act) are commonly used to guide the reasoning process.

Keep in mind that **generative AI agents** are powered by **generative AI models**. These models are the brains of the operation. Once your AI models are trained and ready to go, there are still ways to improve your desired results. One such way is by adjusting certain parameters and settings. Generative AI models are the brains behind AI agents, capable of learning patterns and generating creative text formats. By adjusting sampling parameters like temperature, token count, and top-p, you can fine-tune model behavior to achieve desired outputs.

**Sampling parameters and settings**

Sampling parameters act as settings that influence the AI model's behavior, giving you more customized results. Think of these as knobs and dials you can adjust with your prompt input to impact the model's output. By tweaking these settings, you can ensure the model's output aligns with your specific needs, whether it's generating more creative text, providing more concise summaries, or staying within a certain tone. Let's explore some of the most common parameters you can adjust.

* **Token count**: Think of tokens as the basic building blocks of text for the model. Rather than reading whole sentences at once, the model breaks text down into these chunks. A token can be a whole word, part of a word, or even a single space or punctuation mark.

Models have limits on how many tokens they can handle. A higher token count allows for longer conversations but requires more processing power. As a rule of thumb, one token is roughly four characters in English. So, 100 tokens are about 60 to 80 words.

* **Temperature**: This parameter controls the "creativity" of the model, because it adjusts the randomness of word choices during text generation, influencing the diversity and unpredictability of the output. A higher temperature makes the output more random and unpredictable, while a lower temperature makes it more focused, deterministic and repeatable.
* **Top-p (nucleus sampling)**: "Top-p" stands for the cumulative probability of the most likely tokens considered during text generation. This is another way to control the randomness of the model's output. It concentrates the probability on the most likely tokens, making the output more coherent and relevant. A lower top-p value leads to more focused responses (i.e. only the most probable tokens), while a higher value allows for more diversity (i.e. extending to lower probability tokens as well).
* **Safety settings**: These settings allow you to filter out potentially harmful or inappropriate content from the model's output. You can adjust the level of filtering based on your specific needs and preferences.
* **Output length**: This determines the maximum length of the generated text. You can set it to a specific number of words or characters or allow the model to generate text until it reaches a natural stopping point.
* **Top-k sampling**: Top-k sampling is a method used in natural language processing (NLP) and machine learning for generating text by sampling from a restricted subset of the vocabulary. It introduces controlled randomness by selecting the next word in a sequence from the top k most probable candidates, rather than from the entire distribution. This method helps to balance between diversity and coherence, making it suitable for applications where variation in generated text is desirable without deviating excessively from context.

By experimenting with these parameters, you can significantly influence the AI model's behavior. For example, if you need a concise and factual answer, you might use a lower temperature and a smaller output length. If you're looking for a more creative and open-ended response, you could increase the temperature and top-p values.

**How do you access these settings and sampling parameters?**

The way you access and change these settings and parameters varies based on where the models are coming from and how you are accessing the models themselves. Many models can be accessed via APIs, which stands for application programming interfaces. Essentially, an API is a way for different software systems to communicate and exchange information. It acts like a messenger, allowing one application to request services or data from another. For example, when you use an app to check the weather, it's likely using an API to get that information from a weather service. In the case of generative AI, your application would use an API to send a prompt to the model and receive a response. These APIs often allow you to customize the model's behavior by taking settings and parameters as input along with your prompt.

**Agent Tooling**

Agent tooling equips agents with the resources they need to be effective. Think of it as providing the agent with the right skills, connections, and knowledge to achieve its goals. These tools allow agents to access information, perform actions, and interact with various systems.

We can categorize agent tools into four key types.

**1) Extensions (APIs)**: Extensions bridge the gap between an agent and external APIs. APIs (application programming interfaces) are sets of rules that govern how software interacts. Extensions provide a standardized way for agents to use APIs, regardless of the API's specific design. This simplifies API interaction, making it easier for agents to access external services and data.

**Example:** An agent designed to book travel might use an extension to interact with a travel company’s API. The extension handles the complexities of communicating with travel company’s systems, allowing the agent to focus on the task of finding and booking flights.

**2) Functions**: Functions are like specialized tools within the agent's toolbox. They represent specific actions the agent can perform. An agent's reasoning system selects the appropriate function based on the task at hand. Functions can encapsulate complex logic or interactions, making them reusable and manageable.

**Example:**A "calculate\_price" function might take flight details and passenger information as input and return the total cost. The agent can call this function whenever it needs to calculate a price.

**3)** **Data stores**: Data stores provide agents with access to information. This can include real-time data, historical data, or knowledge bases. Data stores ensure that the agent's responses are accurate, relevant, and up-to-date.

**Example:**An agent might use a data store to access current weather conditions, stock prices, or a database of customer information.

**4) Plugins**: Plugins extend the agent's capabilities by adding new skills or integrations. They can connect the agent to specific services, provide access to specialized tools, or enable interaction with particular platforms.

**Example:**A plugin could enable an agent to interact with a calendar application, allowing it to schedule appointments. Another plugin might integrate with a payment gateway, enabling the agent to process transactions.

**How the reasoning loop works with tools**

The ReAct (reasoning and acting) cycle prompting technique describes how agents use tools:

* **Reasoning (tool selection)**: The agent analyzes the task and determines which tools are needed. It considers the available extensions, functions, and data stores to choose the most appropriate resources.
* **Acting (tool execution)**: The agent executes the selected tool. This might involve calling an API via an extension, invoking a function, or querying a data store. The agent provides the necessary inputs to the tool.
* **Observation**: The agent receives the output from the tool. This output becomes the "observation" in the ReAct cycle.
* **Iteration (dynamic iteration)**: Based on the observation, the agent reasons about the next steps. It might need to select different tools, refine its approach, or gather more information. This cycle repeats until the task is complete.

Example: Scheduling a garden consultation

* **Reasoning**: The agent needs to find an available time slot for a garden consultation. It selects the scheduling plugin, which integrates with the gardener's calendar.
* **Acting**: The agent uses the scheduling plugin to check the gardener's availability for the next week, specifying the desired consultation duration (e.g., 30 minutes) and the client's preferred days (e.g., weekdays). It also accesses the customer database (via a data store) to retrieve the client's name and contact information to include in the appointment details.
* **Observation**: The scheduling plugin returns a list of available time slots.
* **Iteration**: The agent presents the available slots to the client. If the client requests a specific time that's unavailable, the agent might use a function to suggest alternative times or offer to put the client on a waiting list. Once the client confirms a time, the agent uses the scheduling plugin again to book the appointment and sends a confirmation email to both the client and the gardener. This cycle continues until the consultation is successfully scheduled.

Agent tooling is all about enabling agents to take action, whether that's accessing databases, making purchases, interacting with social media, controlling smart home devices, scheduling appointments, booking travel, or performing other necessary tasks. By understanding the range of tools available, you can effectively equip your AI agents with the resources they need to perform complex tasks and interact intelligently with the world.

**Google's agent tooling**

Whether building your agents inside or outside of Google Cloud, the options for tooling are extensive. You can custom-build tools, leverage third-party solutions, or, as we'll focus on here, utilize the powerful suite of tools available within Google Cloud.

Here are some of the most relevant Google Cloud services for agent tooling:

* **Cloud Storage**: A highly scalable and durable object storage service. Use Cloud Storage to store and retrieve data that your agent needs.
* **Databases** (Cloud SQL, Spanner, Firestore): Google Cloud offers a variety of database solutions to suit your needs. Your agent can use these databases to store and retrieve information, manage user data, or track its own progress.
* **Cloud Run Function**: Create serverless functions that act as specialized tools for your agent. Cloud Run functions can be used to connect to databases, call external APIs, perform complex calculations, or handle other specific tasks. They are easily triggered by your agent and scale automatically.
* **Cloud Run**: For more complex agent tools that require containerized environments. Cloud Run provides a serverless platform for deploying and running stateless containers. This is ideal for custom tools that have specific dependencies or require more control.
* **Vertex AI**: Agents can use other agents as tooling. You can use Vertex AI to create models or agents that are called as tooling by other agents.

Along with these services, Google Cloud also offers a wide range of pre-built APIs that provide powerful functionalities for your agent. These APIs can be used for tasks like understanding natural language, analyzing images, translating text, and more.

* The **Speech-to-Text API** accurately converts speech into text. This is useful for transcribing meetings, customer service calls, or even video content.
* The **Text-to-Speech API** converts text into natural-sounding speech, which is helpful for creating voice user interfaces in devices and applications, and for personalized communication.
* The **Translation API** can translate text, documents, websites, and even audio and video files into more than 135 languages.
* The **Document Translation API** can translate formatted documents while keeping the original layout.
* The **Document AI API** allows businesses to extract data from various document formats. It can be used to automate data capture and document processing. This is useful for many kinds of documents, from invoices to contracts to medical records. Document AI can also summarize large documents using generative AI.
* The **Cloud Vision API** allows developers to understand the content of an image by applying powerful machine learning models. It can detect objects, faces, and landmarks, and even read text within images (OCR). This API is useful for a wide range of applications, including image tagging, content moderation, and visual search.
* The **Cloud Video Intelligence API** allows developers to analyze video content and extract meaningful information. It can identify objects, actions, and events within a video, and even transcribe spoken words. This API can be used for applications like content recommendation, video search, and media analysis.
* The **Natural Language API** helps derive insights from unstructured text using Google's machine learning. This API can be used to understand the sentiment of text, classify content, and extract important entities such as people, places, and events. This API can also analyze text in multiple languages.

Google Cloud also offers a vast API Library with APIs linked to many other Google products such as Google Maps, Google Workspace, YouTube, Google Photos, and more.

Google Cloud provides an extensive suite of tools and APIs for building and powering your agents. By leveraging services like Cloud Functions, Databases, and pre-built APIs such as Document AI and Google Maps, you can create agents that perform complex, multi-step tasks. This allows you to orchestrate multiple tools to solve real-world problems and build powerful, intelligent agents.

**The model layer**

At the heart of every AI and machine learning system lies the model. Think of it as the brain of the operation. These aren't just any algorithms. They're sophisticated mathematical structures trained on massive amounts of data. This training process allows them to learn patterns and relationships, ultimately enabling them to perform a wide range of tasks, such as generating content, analyzing data, and classifying information. The model lies at the core of any AI system, acting as its "brain" and enabling a wide range of functions from content generation to data analysis. Vertex AI provides a comprehensive platform for leveraging the power of models, offering access to a vast repository of pre-built models in Model Garden and the ability to create custom models.

Within the Vertex AI platform, you have multiple options for how to handle AI models for your project. You can choose to build a model from scratch or use an existing model.

For many use cases, your organization won’t need to develop a new model. There are many AI models that already exist today that you can fine-tune and leverage for so many different use cases. Model Garden on Vertex AI is a service that lets you discover, customize, and deploy existing models from Google and Google partners. Model Garden gives you options to pick from over 160 models and offers options of Google models, third-party models, and open-source models. With Vertex AI, you can customize these models with your own data and deploy them to your custom applications. Additionally, some pre-trained models can be used out-of-the-box without tuning.

With Vertex AI, you have two options for training and using your own models. You can go fully custom and create and train models at scale using any ML framework (PyTorch, TensorFlow, scikit-learn, or XGBoost), or there are also options for using AutoML to create and train models with minimal technical knowledge and effort.

Standard workflow when creating your models in Vertex AI.

**1)** **Gather your data**: Determine the data you need for training and testing your model based on the outcome you want to achieve.

**2) Prepare your data:** Make sure your data is properly formatted and labeled.

**3) Train:** Set parameters and build your model.

**4) Manage your model:** Review model metrics to evaluate the model’s performance and monitor changes over time.

**5) Deploy and predict:** Make your model available to use.

**Edge computing**

While hosting infrastructure on the cloud is powerful, it's not always the ideal solution. Imagine a self-driving car needing to make split-second decisions. It can't wait for data to travel to the cloud and back. That's where edge computing comes in. Edge computing allows AI to run locally on devices or servers closer to the data source, ensuring real-time responsiveness and increasing data control.

**Why go local or edge?**

Imagine a drone navigating a complex environment. It needs to react instantly to obstacles, making cloud processing too slow. Running AI locally on the drone ensures real-time responsiveness. Other benefits include increased data privacy and reduced reliance on internet connectivity. To run powerful AI models on edge devices and mobile phones, Google provides tools like **Lite Runtime (LiteRT)**. Think of LiteRT as a platform that helps machine learning models work efficiently on your device. One example of an AI model designed for edge is Gemini Nano. Google provides tools like Lite Runtime (LiteRT) and Gemini Nano that are specifically designed for edge computing.

**Gemini Nano** is Google's most efficient and compact AI model, specifically designed to run on the edge on devices like smartphones and embedded systems. It's part of the larger Gemini family of models. Think of Gemini Nano as a miniature version of the powerful AI that usually lives in Google's data centers. This "on-device" or edge approach offers several benefits.

* **Privacy**: Your data stays on your device, enhancing your privacy.
* **Speed**: You get fast responses since there's no need to send data to the cloud.
* **Offline** **access**: Gemini Nano can work even without an internet connection.

**Gemini Nano** brings the power of AI directly to your devices, making them smarter, more helpful, and more privacy-focused. It is currently being integrated into various Google products and services, including:

* **Pixel phones**: Gemini Nano powers features like **Call Notes**, which summarizes phone conversations and **Pixel Recorder**, which summarizes voice recordings.
* **Android**: Gemini Nano is available to Android developers through the AI Edge SDK, enabling them to build innovative AI experiences into their apps.

Even when your goal is to deploy on the edge, Vertex AI provides a powerful platform for building, training, and refining your models. You can leverage Vertex AI's capabilities for tasks like data preparation, model training, tuning, model evaluation, collaboration, and MLOps.

Once your model is ready for the edge, Vertex AI offers tools to streamline the deployment process.

You can:

* **Convert models:** Convert your models to Lite Runtime (LiteRT) for optimal performance on edge devices.
* **Package and deploy:**Package your models and dependencies into containers for deployment on various edge hardware.
* **Manage and monitor:** Manage your edge deployments, track their performance, and gather insights to improve your models over time.

Google provides the tools to deploy your AI models in these different locations, giving you more control and flexibility.

**Gen AI project resources: People, cost, and time**

**Roles and responsibilities**

It is important to understand the resources you have available to build out your solution. Let’s dig deeper into some of the roles and responsibilities at each layer of the stack.

The AI stack is designed to support different roles with specific needs. Before jumping into building out a solution, ensure you have the proper expertise and talent for what you want to build. Business users can improve their tasks, developers can build custom AI solutions, and AI practitioners can develop and deploy advanced AI models responsibly and securely. This collaborative ecosystem allows your organization to leverage the full potential of AI across various business functions.

**Business Leaders**: Business leaders typically interact with pre-built gen AI solutions to enhance daily operations and improve customer experiences. For example, Google Workspace with Gemini can be used for content creation, data analysis, and document summarization.

**Developers**: Developers in your organization are responsible for building and deploying custom AI agents and integrating AI capabilities into existing applications. They can use AI Applications for custom agent creation, AI code generation, and AI-driven data processing. Developers can also leverage pre-trained APIs to rapidly integrate AI into applications, or they can use the Vertex AI platform which is designed to help developers build and deploy AI agents with tools for orchestration, grounding, and action.

**AI Practitioners**: AI practitioners play a valuable role in customizing, deploying, and optimizing generative AI models. They leverage tools within Vertex AI to accelerate development and ensure responsible AI practices. Their expertise extends to scaling AI workloads, integrating models with BigQuery, and implementing responsible AI measures such as bias detection and adversarial testing.

**Cost**

The pricing model of gen AI can vary based on which part of the gen AI landscape you are looking at or even which specific product or company you are using. It is important to have a realistic budget and understanding of the cost for your project.

When building gen AI solutions, you pay for three primary activities:

* Training the model.
* Deploying the model to an endpoint.
* Using the model to make predictions.

Training and deploying models often involve paying for the compute time used as well as the storage for the training data and model outputs.

**Time**

It probably won’t surprise you to learn that the more custom your solution is, the more time and resources it takes to build. Using a prebuilt gen AI application takes seconds. Building your own custom solution with a custom AI model, can take months. Choosing to build a custom agent with AI Applications takes much less time.

Think about your project timelines and evaluate them against your needs and requirements. Determine if the timeline is realistic based on your requirements.

**Google Workspace with Gemini**

**Google Workspace** is a collection of cloud-based productivity and collaboration tools that help people create, communicate, and collaborate, such as Gmail, Google Docs, Google Sheets, Google Meet, and Google Slides. Google Workspace is a productivity solution designed to help you safely connect, create and collaborate with tools like Gmail, Docs, Meet and more.

Gemini is a gen AI model that is then used in many other applications. Some of those applications were built in Google Workspace, so you can access Gemini directly within your Workspace apps through Google Workspace with Gemini. Google Workspace with Gemini is a powerful collaborative partner that can act as a coach, thought partner, source of inspiration, and productivity booster—all while ensuring every user and organization has control over their data.

**Google Workspace with Gemini features**

1. **Gemini side panel**: One way to access Google Workspace with Gemini is through the Gemini side panel, which is present on the side of many Workspace apps. With the side panel, Gemini can assist you with summarizing, analyzing, and generating content by utilizing insights gathered from your emails, documents, and more—all without switching applications or tabs.

For example, new employees can use the Gemini in Drive side panel to get up to speed on projects and company initiatives. They can ask Gemini questions like "summarize the goals and current status of Project Phoenix" and easily find the information they need, enabling them to quickly become familiar with the project.

1. **Gemini in Gmail**: Gemini in Gmail uses generative AI to help users write and refine emails more efficiently.

An HR representative can use Gemini to write a company-wide email announcing a new employee benefit. They can provide a prompt that includes information about the new benefit, who is eligible, and the effective date, and Gemini will generate a draft email.

1. **Gemini in Google Docs**: Gemini in Google Docs helps you streamline the process of content creation, refining, and proofreading right within your document as you write.

A marketing team member can utilize Gemini to generate captivating taglines and draft social media posts for a new athletic shoe campaign. The user can provide a prompt like "Generate three social media posts that highlight the benefits of our athletic shoes, focusing on comfort and performance." Gemini then generates text options for the posts.

1. **Gemini in Google Slides**: Gemini in Google Slides can help you and your team work more effectively by using AI to generate photorealistic images that enhance your presentations.

By providing Gemini with a descriptive prompt that outlines the subject, setting, or style you desire, you can quickly create images that otherwise would require time and effort to source or design. This streamlines the presentation creation process and frees up your team to focus on content and delivery.

1. **Gemini in Google Sheets**: Gemini in Google Sheets helps you generate formulas using AI.

You can provide Gemini with prompts using plain language or generic cell names to create complex formulas. For example, you can ask Gemini to "create a formula to find cell C1 in range D:G and output the value in column G." Gemini then creates the formula.

1. **Gemini in Google Meet**: Gemini in Google Meet can help your global team communicate by providing real-time transcription and translation, facilitating seamless communication in multilingual meetings. It could also automate meeting summaries and action item extraction, saving time and increasing productivity for participants.

These Google Workspace with Gemini features allow you to use **AI-enhanced workflows** in your daily work to become**more productive and effective**. Gemini's capabilities also extend to other powerful applications within the Google Workspace ecosystem, like Google Vids (**Google Vids** is an online video creation and editing app available to Google Workspace. With Help me Create in Vids, you can use Gemini to generate a first draft of your video) and AppSheet (**AppSheet** is a no-code app development tool included with Google Workspace enterprise editions. With Gemini in AppSheet, you can quickly create apps using AI by describing your needs in a prompt using natural language). Google Workspace with Gemini is more than a content generation tool; it's a solution for boosting organizational efficiency and customization.

**Prompt engineering techniques**

As you continue experimenting with different gen AI applications, you'll quickly discover the crucial role of prompting. Developing your prompting skills is essential for achieving optimal results and tailoring generative AI to your specific needs. The key to using gen AI tooling is to ensure you get experience prompting. There are many different prompting techniques that one can use, but we will highlight two of the most common for this course: ReAct prompting and chain-of-thought (CoT) prompting.

**ReAct prompting**:

ReAct is a prompting framework that allows the language model to reason and take action on a user query, with or without in-context examples. ReAct, which stands for "reasoning and acting," is like giving an LLM a brain and a pair of hands. It allows the LLM to not only think about a problem but also take actions to solve it. This is a big step up from traditional prompting, where the LLM could only passively generate text.

Imagine you're asking an LLM to find you a good Italian restaurant nearby. With ReAct, the LLM can:

* **Think**: Okay, I need to find Italian restaurants that are close by and have good reviews.
* **Act**: I'll search online for Italian restaurants in this area.
* **Observe**: Ah, here are some options with high ratings.
* **Think**: Which one is the best fit based on the user's preferences?
* **Act**: I'll check the menus and see which one has the most appealing dishes.
* **Observe**: This one has delicious-looking pasta!
* **Respond**: *I recommend [restaurant name]. It has great reviews, is located nearby, and has a fantastic pasta selection.*

This "thought-action-observation" loop allows the LLM to interact with the real world and gather information, making it much more powerful than a traditional LLM.

**Why is ReAct important?**

* **Dynamic problem solving:** ReAct allows LLMs to tackle complex tasks that require interacting with external resources and adapting to new information.
* **Reduced hallucination:**By grounding the LLM's reasoning in real-world data, ReAct can help reduce the risk of generating incorrect or nonsensical information.
* **Increased trustworthiness:** The ability to see the LLM's reasoning process and how it interacts with external sources makes its responses more transparent and trustworthy.

**Key components of ReAct**

* **Think:**The LLM generates a thought about the problem, similar to CoT.
* **Act:**The LLM decides what action to take, such as searching the web, accessing a database, or using a specific tool; the LLM specifies the input for the action, like a search query or database command.
* **Observe:**The LLM receives feedback from the action, such as search results or database entries.
* **Respond:**The LLM generates a response, which could involve providing an answer to the user, taking further actions, or formulating a new thought for the next iteration.

**ReAct in action**

ReAct has been successfully used in various applications, including:

* **Question answering:** LLMs can use ReAct to access external knowledge sources and answer questions more accurately.
* **Fact verification:** LLMs can verify claims by searching for evidence online.
* **Decision making:** LLMs can use ReAct to gather information and make informed decisions in interactive environments.

**Chain-of-thought (CoT)**

Chain-of-thought prompting is a technique where you guide a language model through a problem-solving process by providing examples with intermediate reasoning steps, helping it learn to approach new problems in a more structured and logical way.

Remember prompt chaining? Where you keep prompting in the same thread so the LLM keeps your chat history and learns more as you go? Well, you can do something very similar behind the scenes for your user using chain-of-thought (CoT) prompting. CoT enables reasoning capabilities through intermediate steps.

Think of CoT as a way to make LLMs even smarter by teaching them to think step-by-step, just like a human would. Instead of just giving the LLM a prompt and expecting an answer, you guide it through the reasoning process. You provide examples of how to solve similar problems, showing the steps involved. This helps the LLM learn to approach problems in a more logical and structured way. It is similar to teaching a student to think out loud. You're teaching the LLM to think out loud. By showing it the intermediate steps, you're helping it develop a "chain of thought" that leads to the correct answer.

**Why is CoT important?**

CoT is a powerful technique that helps LLMs think a bit more like humans. By guiding them through the reasoning process, we can unlock their full potential and achieve even more impressive results.

* **Improved reasoning:** CoT helps LLMs solve complex problems that require logical thinking.
* **Better accuracy:** By breaking down problems into smaller steps, CoT can lead to more accurate results.
* **Enhanced explainability:**CoT makes it easier to understand how the LLM arrived at an answer, which is important for building trust and transparency.

**Key components of CoT**

Just like there are different ways to solve a problem, there are different ways to implement CoT. Some popular techniques include:

* **Self-consistency:** Encouraging the LLM to generate multiple solutions and choose the most consistent one.
* **Active-prompting:**Allowing the LLM to ask clarifying questions or request additional information.
* **Multimodal CoT:**Combining text with other forms of data, like images or videos, to enhance reasoning.

**CoT in action**

Chain-of-thought prompting has been successfully used in various applications, including:

* **Complex reasoning tasks:**LLMs can use CoT to break down complex problems into smaller, more manageable steps, leading to more accurate solutions in tasks like math word problems or logical reasoning puzzles.
* **Explanation generation:** LLMs can use CoT to generate step-by-step explanations for their answers, making their reasoning process transparent and understandable, which is crucial for building trust and identifying potential errors.
* **Multi-step planning:**LLMs can use CoT to plan and execute complex tasks that require multiple steps, such as writing a story, planning a trip, or debugging code.

**Choosing a prompting framework**

There are many other frameworks and techniques that can be used for prompt engineering and within your reasoning loop. All these different techniques can be combined and leveraged in different ways depending on the use case.

The best framework for you will depend on your use case. For example, while both ReAct and CoT enhance LLM reasoning, they have different strengths:

* **CoT** focuses on internal reasoning, guiding the LLM through a chain of thought.
* **ReAct** focuses on external interaction, allowing the LLM to gather information and take actions in the real world.

In fact, ReAct and CoT can be combined for even more powerful results. By integrating both techniques, we can create LLMs that are capable of both deep reasoning and dynamic interaction with the world around them.

Prompt engineering techniques like ReAct and chain-of-thought (CoT) prompting can enhance the reasoning capabilities of agents, leading to more accurate, reliable, and human-like interactions. By understanding prompt engineering techniques, you can effectively guide the development of AI agents that can transform your organization.

When it comes to **foundational (basic)** **prompting techniques**, there are a few different approaches.

* **Zero-shot prompting** is like asking a foundation model to complete a task with no prior examples, relying solely on its existing knowledge.
* **One-shot prompting** involves showing the foundation model just one example, allowing it to learn and apply that knowledge to similar situations.
* **Few-shot prompting**, on the other hand, provides the foundation model with multiple examples to learn from, which helps it better understand the task and improve its performance.

**Role prompting** is a technique used to guide the behavior of LLMs by assigning them a specific role or persona. This can be anything from a business analyst or Shakespearean actor to a helpful customer service agent. By instructing the model to adopt a particular role, you influence the style, tone, and focus of its responses.

A black rectangular object with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

When working in a tool like Gemini, it's important to understand how previous prompts may impact future prompts.

**Prompt chaining** is a powerful technique for getting more complex and nuanced results from large language models like Gemini. It's like having a conversation with the AI where each response builds upon the previous one, leading to a more sophisticated and refined outcome. This is where the concept of a chatbot comes in. Prompt chaining lets you continue your conversation in the same chatbot so the chatbot knows your history before continuing on with the new prompt.

Prompt chaining is a technique for refining AI outputs by building on previous prompts. Instead of trying to achieve the final result in a single request, start with a broad, initial prompt. Then, use subsequent prompts to ask clarifying questions, provide more specific information, or request a different format to refine the AI's output incrementally until it meets the desired goal.

**Gemini Advanced** is an upgraded version of Gemini with extra features, making it far more capable at highly complex tasks, like coding, logical reasoning, following nuanced instructions, and creative collaboration. You can adjust your data retention settings to determine how long information is stored and delete your conversation history at any time.

**Gems**  
In Gemini, you can even name your chats and pin them for easy access. But even saved information won’t always work. Because that will apply to all your Gemini chats. Sometimes you need specific context for particular tasks or conversations. This is where Gems come in. Like agents, these AI assistants can process information and reason over complex ideas within the context of your chosen task. You can have Gems dedicated to:

* **Creative writing:** This Gem could be pre-loaded with your preferred writing style, tone, and commonly used resources. It could even help you brainstorm ideas, generate different creative text formats (poems, scripts, articles), and provide feedback on your drafts.
* **Coding:**A coding Gem could be equipped with your favorite libraries, coding conventions, and even access to relevant documentation. It could assist with debugging, suggest code optimizations, and even generate code snippets based on your instructions.
* **Marketing:**This Gem could be pre-loaded with your target audience demographics, competitor information, and campaign performance data. It could help you brainstorm campaign ideas, generate different marketing copy (social media posts, email newsletters, website content), analyze campaign performance, and suggest optimizations.

**How do Gems work?**

Gems are your personalized AI assistants within Gemini, allowing you to create custom tools for any task. By using role-prompting and a focused knowledge base, you can give them clear and specific instructions, turning your well-crafted prompts into powerful, reusable assistants that streamline your workflows and save time.

**Personalized responses**: Gems can be tailored with specific instructions and information. This allows them to provide responses that are more relevant to your needs and preferences within that particular use case.

**Streamlined workflows**: Gems can help you streamline repetitive tasks by providing templates, prompts, or guided interactions. This can save you time and effort when working on familiar projects or activities.

**Reset context**: You can set the context for a Gem, like giving it background information or specific instructions. Then, you can start multiple chats with that Gem, each with its own focus and flow. These chats remain separate, so information from one won't spill over into another. It's like having different conversations with the same expert, each tailored to a specific purpose.

You can choose to use Google’s existing Gems that were built for many different common tasks, or you can create your own gem tailored to your use case.

**NotebookLM (Google's AI-powered research assistant)**

NotebookLM is an AI-first notebook, grounded in your own documents, designed to help you gain insights faster. NotebookLM is an AI-powered notebook designed to help you quickly gain insights from your documents. It acts as a virtual research assistant, able to summarize information, answer your questions, and even create different types of summaries based on your uploaded documents, presentations, or audio and video files.

Unlike general AI tools that draw from a broad knowledge base, NotebookLM focuses solely on the sources you provide. This "grounding" ensures that the information it provides is accurate, relevant, and traceable back to your original materials.

It’s a unique tool that is built using Gemini models and is designed to help you learn and understand information better by acting as a virtual research assistant. You can "ground" NotebookLM in specific sources like documents, presentations, or even audio and video files. This means your AI assistant will only use information from those sources to answer your questions and generate summaries. Shared NotebookLM notebooks foster a culture of collaborative learning and knowledge sharing, empowering your team to work smarter and more effectively. It's like having a collective "brain" that everyone can contribute to and benefit from.

​**How is NotebookLM different from Gemini and Gems?​**

While Gems excel at general tasks and workflows, NotebookLM is designed to help you deeply understand and learn from specific sets of information. It's like having a dedicated expert who has thoroughly analyzed your chosen materials and is ready to answer any question you have.

**NotebookLM versus Gems**

* **Hyper-focused knowledge:**Instead of drawing from a broad knowledge base, NotebookLM focuses solely on the sources you provide. This could be anything from research papers and articles to meeting notes and presentations.
* **Interactive learning:**NotebookLM goes beyond simply summarizing information. It encourages active learning by allowing you to ask questions, generate different types of summaries, and even create quizzes to test your understanding.
* **Source-based answers:**Every answer and insight provided by NotebookLM is directly grounded in your uploaded sources. This ensures accuracy and allows you to easily trace back to the original information. If you ask NotebookLM a question that isn't covered in the materials you've provided, it will honestly tell you that it can't answer. It won't invent information or speculate. This ensures that the information you get is always grounded in your sources and reliable.

**Use cases**

* **Create training materials and documentation:** Onboard new team members and facilitate knowledge transfer by creating a central repository for essential resources. Create a dedicated notebook to share with new hires with relevant onboarding materials (employee handbook, product documentation, training presentations). The new employees can then use NotebookLM to ask questions, clarify information, and gain a deeper understanding of their role and the company.
* **Researching a new topic:** Gather articles and videos on a subject you're interested in and have NotebookLM create a comprehensive summary or answer specific questions you have.
* **Learn by listening to audio:** With audio overviews in NotebookLM you can learn about the material covered in your sources through a podcast-like experience. NotebookLM will summarize your material, connecting topics, and produce a downloadable audio with back-and-forth conversation between two AI hosts.
* **Preparing for presentation:** Feed NotebookLM your presentation slides and ask it to generate speaker notes or practice questions to help you rehearse.
* **Summarize documentation:** Investment firms and legal teams need to thoroughly analyze documents and data. Upload financial statements, legal documents, and market research reports into NotebookLM. The team can then use the AI to quickly identify key information, summarize findings, and flag potential risks or inconsistencies.
* **Project proposal and plans:** Keep everyone aligned and informed by creating a shared NotebookLM with all project-related information.

NotebookLM helps businesses unlock knowledge hidden within scattered data. By centralizing a wide range of documents in a shared workspace, NotebookLM enables you to ask targeted questions, generate insightful summaries, and collaborate more effectively, leading to faster data-driven decisions.

**Gemini for Google Cloud**

Just like with Google Workspace, Gemini is revolutionizing how we interact with Google Cloud. From simplifying data analysis to boosting developer productivity, Gemini for Google Cloud offers a suite of AI-powered tools designed to enhance your cloud experience.

* **Gemini Cloud Assist** is like having an AI expert on your team who helps you design, manage, and optimize applications on Google Cloud. It provides personalized guidance and is integrated with your Google Cloud environment to provide application lifecycle management assistance. It analyzes your cloud environment, your resources deployed, as well as metrics and logs to deliver actionable insights tailored to your needs.
* **Gemini in BigQuery** makes data analysis easier and more accessible. It can help you write code, understand your data, and even generate insights automatically, regardless of your SQL experience. This means faster and more efficient data exploration.
* **Gemini Code Assist** acts as an AI pair programmer, helping developers write better code, just like when two programmers work together to solve problems. Gemini Code Assist can provide code suggestions, generate code blocks, and even offer explanations. It supports over 20 popular programming languages, code editors, and developer platforms, all of which help developers increase productivity.
* A Colab Enterprise notebook is an interactive environment that lets you write and execute code. **Gemini in Colab Enterprise** can use AI to help you write Python code in your notebook by suggesting code segments as you type and generating code based on your descriptions. This streamlines your data analysis and machine learning workflows.
* **Gemini in Databases** helps developers and database administrators manage their databases more effectively. It uses AI to simplify many aspects of using a database, from building applications with natural language to managing an entire fleet of databases from a single interface.
* With **Gemini in Looker**, you can analyze data and gain insights faster. As an intelligent assistant, it helps you understand your data, create visualizations, and even generate reports, making data exploration more intuitive.
* **Gemini in Security** helps security teams detect, contain, and stop threats from spreading. It provides near-instant analysis of security findings and potential attack paths. Gemini in Security also summarizes prevalent tactics, techniques, and procedures used by threat actors, giving customers around the world detailed and timely threat intelligence.

**Google’s generative AI APIs**

Google’s generative AI APIs offer pre-trained large language models that can be fine-tuned for specific tasks. These APIs include capabilities for text completion, multi-turn chat, code generation, and image generation. For example, the Imagen API can be used to generate and customize images. Google provides two tools where you can easily start to access and experiment with generative AI model APIs.

**Google AI Studio**: Google AI Studio is a web-based tool that allows developers, students, and researchers to try Gemini models and begin building with the Gemini Developer API. It is designed for ease of use and accessibility, targeting a broad audience, including non-technical users who want to leverage AI capabilities without deep expertise in machine learning.

**Vertex AI Studio**: Vertex AI Studio is a Google Cloud console tool for rapidly prototyping and testing generative AI models. It provides developers with a space to test models using prompt samples, design and save prompts, and tune foundation models.

**What is the difference between Google AI Studio and Vertex AI Studio?**

Both Google AI Studio and Vertex AI Studio allow you to experiment with and utilize Google's Gemini API, but they cater to different needs and levels of expertise. Here's a breakdown of their key differences:

**A screenshot of a computer

AI-generated content may be incorrect.**

**Google AI Studio:** small scale projects and prototypes

**Vertex AI Studio:** large-scale enterprise projects

Google provides tools like Vertex AI Studio and Google AI Studio for experimenting with generative AI APIs and customizing models for specific tasks. These tools empower both developers and non-technical users to harness the power of generative AI and integrate it into their applications. Choosing the right tool depends on your specific needs and expertise. If you're just starting out, Google AI Studio is a great place to learn and experiment. If you need a more powerful and scalable solution for professional use cases, Vertex AI Studio is the better choice.

You'll need to generate an API key for Google AI Studio or set up authentication and authorization for Vertex AI Studio. This process involves obtaining credentials that allow your application to securely access and communicate with the Gemini API. These are essential steps for integrating Gemini into your own projects. So, let’s say I have the proper authentication or API key in place. What happens then? With those in place, you’re now able to start making API calls from your application code. These calls will send your prompts and parameters to the Gemini model and receive the generated text in response. This allows you to dynamically generate creative text formats, answer questions, translate languages, and much more. All within your own applications.

You can integrate the Gemini API into virtually any application that can make HTTP requests. This can be a wide range of environments, including web applications, mobile apps, desktop software, and even embedded systems.

**Adding prompt engineering to your reasoning loops** (enhance your reasoning loop through prompt engineering)

We can take basic agent to the next level just by adding some reasoning and logic into it. This is what a lot of people will refer to as prompt engineering. You can create this agent layer before the end user application where you add in prompting logic. The prompt engineering pros frequently don’t actually even write those prompts themselves. They use generative AI to generate their prompts. It’s called meta prompting.

[Meta prompting](https://www.google.com/search?q=Meta+prompting&rlz=1C1GCEB_en&oq=meta+prompting&gs_lcrp=EgZjaHJvbWUyCQgAEEUYORiABDIHCAEQABiABDIHCAIQABiABDIHCAMQABiABDIHCAQQABiABDIHCAUQABiABDIHCAYQABiABDIHCAcQABiABDIHCAgQABiABDIHCAkQABiABNIBCDQyODZqMWo3qAIAsAIA&sourceid=chrome&ie=UTF-8&ved=2ahUKEwikpOigoYOSAxXh1zgGHXTIMo8QgK4QegYIAAgAEAM) is an advanced AI technique where you use one prompt to generate, refine, or analyze other prompts, essentially prompting the AI to improve its own instructions for better, more structured results, focusing on the *how* (structure, logic, format) rather than just the *what* (content) of the task. It's like teaching an AI to write better prompts for itself, allowing for dynamic self-optimization, moving beyond simple commands to creating abstract templates for complex, repeatable tasks like summarization or Q&A.

Meta prompting is an advanced [prompt engineering](https://www.geeksforgeeks.org/blogs/prompt-engineering-best-practices/) technique where prompts are used to generate, refine, or analyze other prompts, rather than directly answering a user’s question. This higher-level approach helps guide large language models (LLMs) to create, improve or interpret prompts for specific tasks, making AI interactions more dynamic, flexible and effective.

Metaprompting is about creating prompts that guide the AI to generate, modify, or interpret other prompts. Metaprompting is a powerful tool for interacting with AI, enabling more dynamic, flexible, and adaptable prompt creation and interpretation. It's a key technique for unlocking the full potential of large language models.

**The reasoning loop**

The reasoning loop is a key component of a generative AI agent that governs how the agent takes in information, performs internal reasoning, and uses that reasoning to inform its next action or decision. It is an iterative, introspective process that continues until the agent achieves its goal or reaches a stopping point. The complexity of the reasoning loop can vary greatly depending on the agent and the task it is performing. The reasoning loop is a central, iterative process that enables a generative AI agent to take in information, perform internal reasoning, and make decisions. Here's a breakdown of the key aspects of the reasoning loop:

* **Interactive process**: The reasoning loop is not a one-time operation, but rather a cyclical process where the agent continuously evaluates its progress and determines the next best action to take. This loop involves steps for action, tool selection, and observation.
* **Internal reasoning**: The agent uses its underlying language model to think through the steps it needs to take to complete a task. The language model provides the agent with reasoning and logic capabilities.
* **Decision making**: Based on its internal reasoning, the agent decides on the next course of action. This involves choosing the appropriate tools to use and determining the necessary inputs for those tools.
* **Reasoning frameworks**: The reasoning loop utilizes various prompt engineering frameworks and techniques to guide its reasoning and planning.

**The power of search agents**

Search is the primary way we navigate the digital world. It's how we find information, products, services… everything. Google’s mission is to organize the world's information and make it universally accessible and useful. Tools like Vertex AI Search use AI, including generative AI, to create powerful and smart search experiences within organizations.

Vertex AI Search combines information retrieval, natural language processing, and large language models to understand what people are really asking, even if they phrase it poorly. Vertex AI Search offers both **search** and **recommendation** solutions.

Vertex AI search is a powerful tool leverages advanced AI and machine learning to understand customer intent and deliver highly relevant search results.

Vertex AI Search goes beyond simple keyword matching. It understands the meaning behind search queries, including synonyms, related terms, and even misspellings, to deliver more accurate and comprehensive results. This means customers are more likely to find what they're looking for, even if they don't know the exact name of the product.

One of the most exciting features is the AI-powered image search. Customers can now simply upload a photo of an item they like—perhaps an old favorite they're trying to replace, or a style they spotted on social media—and Vertex AI Search will find visually similar products in the company's current catalog. Vertex AI Search retail recommendations learns from customer behavior, including past purchases, browsing history, and search queries, to provide personalized product recommendations. This helps customers discover new items they might love, increasing engagement and driving sales.

Vertex AI Search for commerce enhances online shopping by understanding customer intent. It goes beyond keywords with features like image search and personalized recommendations, which boosts customer satisfaction and drives sales.

**Search solutions**

Search allows you to create a powerful search experience for your public website. It can index and search across a variety of data types, including structured data in BigQuery and unstructured documents stored in Google Cloud Storage. This ensures your users can easily find the information they need on your website, regardless of how it's stored.

Search also extends to include some specific forms of search such as:

* **Document search**: If your primary need is to search across a large repository of unstructured documents stored in Google Cloud Storage, this is the ideal solution. It's optimized for understanding and retrieving information from text-heavy data, making it perfect for internal knowledge bases, document archives, and more.
* **Media search**: For organizations dealing with rich media libraries, media search provides specialized capabilities. It's designed to understand and search within images, videos, and audio files. This allows users to find specific media assets based on their content, metadata, or even spoken words within audio or video.
* **Healthcare search**: Addressing the unique needs of the healthcare industry, this option enables searching across healthcare data while supporting regulatory compliance. It allows professionals to access patient records, research articles, and medical guidelines efficiently and securely.
* **Search for commerce**: Specifically designed for e-commerce, this option focuses on building a search app for your retail catalog. It optimizes product discovery, understanding product attributes, and handling complex product queries.

**Recommendation solutions**

The general-purpose recommendation engine can be used to recommend similar content within websites, documents, and other structured content. It analyzes user behavior and content attributes to provide personalized recommendations, increasing user engagement and content discovery.

This also extends to include some specific forms of recommendations such as:

* **Media recommendations**: Tailored for consumer-focused media applications like audio/video streaming and digital publishing, this option provides specialized recommendations. It understands user preferences and consumption patterns within media content to suggest relevant audio, video, articles, and more.
* **Retail recommendations**: Optimized for e-commerce, this solution drives sales by providing personalized product recommendations. It leverages user browsing history, purchase data, and product attributes to suggest items that are likely to be of interest to each individual shopper.

**How Vertex AI Search works**

Regardless of the specific search or recommendation option you choose, Vertex AI Search operates on a foundation of intelligent data connection, grounding, and generative AI. It seamlessly connects to your existing data stores, whether they are structured databases, unstructured document repositories, or a combination of both. This connection is crucial as it allows Vertex AI Search to act as an agent, observing the user's query or context (the environment) and acting by retrieving relevant information or suggesting relevant items (using the data stores as tools) to achieve the goal of providing the right information or recommendation at the right time.

**A key strength of Vertex AI Search** lies in its ability to ground gen AI LLM responses with your first-party data, curated third-party data, and even Google's knowledge graph (Grounding with Google Search), minimizing "hallucinations" and ensuring trustworthy information.

This grounding is where the connection to RAG comes in. By using your own data sources as the foundation for LLM responses, Vertex AI Search implements a RAG approach, ensuring that the information provided is relevant, accurate, and grounded in the context of your specific data.

Vertex AI Search also gives you the options of adding **extra generative AI features** to your search functionality.

* **Search summaries:**Vertex AI Search can generate concise and informative summaries of search results, saving users time and effort. These summaries can be tailored to different needs, providing a quick overview of a document, a comparison of multiple products, or a synthesis of key findings from a set of search results.
* **Answers and follow ups:** Vertex AI Search can add AI generated answers to your search results. Users can ask questions in natural language to your website’s search and get AI generated answers based on the search results. Users can then follow up with further questions.

Vertex AI Search is **built for enterprise**and offers granular access controls to help ensure data security. It also provides advanced analytics to understand search trends and user behavior and scalable infrastructure to handle large volumes of data and search requests. It's designed to meet the demanding requirements of organizations of all sizes. Vertex AI Search is built to easily integrate with existing enterprise systems through APIs and SDKs. Whether you're building a customer-facing search experience or an internal knowledge base for employees, Vertex AI Search can be seamlessly integrated into your workflow.

By building Vertex AI Search capabilities into your application, you can empower your customers to find information faster, understand it better, and make more informed decisions. This capability significantly enhances the overall search experience and drives efficiency.

**Gen AI for customer engagement**

Customer Engagement Suite is a cohesive way to build extremely powerful agents for your company that leverage generative AI technology along with other powerful technology to create a seamless experience for your customers.

**Google’s Customer Engagement Suite**

Websites, even with the best search, are not necessarily enough for your company. Customers frequently don’t want to search. They want to connect with your company directly to get the answers and support they are looking for.

These points of communication can be a pain point between customers and companies, but they are also an extremely important point of connection. Positive customer engagement could be the difference between the success and failure of your company.

Google’s Customer Engagement Suite has tools designed to support your company in engaging with customers effectively. The tools are built using conversational AI, some of which is also generative AI.

With the Customer Engagement Suite, you can build:

* **Conversational Agents** to act as effective chatbots communicating with your customers.
* **Agent Assist**to support your live human contact center agents.
* **Conversational Insights**to gain insights into all your communications with customers (through chatbot agents or human agents).

All of this can be built on top of Google’s **Contact Center as a Service (CCaaS)**, an enterprise-grade contact center solution that is native to the cloud.

**Conversational Agents**

You may be a bit skeptical of customer service chatbots. But the types of conversational agents you can build today have come a long way.

Remember, there are two primary methodologies you can use when building conversational agents: deterministic and generative.

While deterministic AI alone can be rigid, and generative AI alone may lack structure, their combination creates a powerful solution. A hybrid approach allows you to build conversational agents that can handle a wide range of customer needs effectively.

With Conversational Agents in Google Cloud, you have flexible options for building and integrating your agents with your existing tools. Create simple agents with generative AI by using natural language to instruct your agent or complex hybrid agents with custom rules and logic.

* **Deterministic**: Deterministic is more associated with some historical agents. It is a rule-based, very defined system for your chatbot agents to follow. It will use very defined logic such as if the user presses this number, go to this route. Everything you want a deterministic agent to do needs to be explicitly defined. Deterministic agents usually require low to medium code to build.
* **Generative**: Generative is based on new generative AI technology. It uses large language models to give a real conversational feel to your chatbot. These agents will determine what to do on their own based on your prompt unless you specifically tell it otherwise. Generative agents usually require prompting and either no code or low code to build.
* **Hybrid**: Google Cloud Conversational Agents empowers you to build hybrid agents that combine the strengths of deterministic and generative AI. This approach offers strict control while leveraging generative AI's flexibility to better address customer needs.

Conversational Agents can leverage generative AI playbooks and a company's own data stores to provide accurate, instant responses to customer inquiries. This can significantly enhance customer satisfaction and drive tangible business results.

A **Gen AI Playbook** is a strategic guide or roadmap that helps businesses understand, implement, and scale Generative AI technologies to achieve specific goals like innovation, efficiency, and growth, providing practical steps, best practices, use cases, and governance for safe and effective adoption across various departments like marketing, finance, or legal. It bridges the gap between AI's potential and real-world business value by offering structured frameworks, from initial experiments to large-scale integration, focusing on measurable returns and responsible use.

When building a generative AI agent with Conversational Agents, you create what is called a playbook for how you want your agent to act. In the playbook, you define your agent's goal, such as providing customer support, answering user questions, or even generating creative content. You then provide detailed instructions on how the agent should act and any rules you would like it to follow. In the playbook, you also have the option to link it to external tools such as data stores. Once your playbook is defined, you are ready to go and can start testing your agent and interacting with it.

**Agent Assist**

There will always be use cases where conversational agents are not enough or tricky situations where the human touch is needed. This is where live agents come in. But using human agents comes with its own challenges. Not all human agents are the same and they have different levels of experience. There can be a lot of training needed, especially when a new agent starts.

This is where Agent Assist comes in. It is a tool that supports live human agents with in-the-moment assistance, generated responses, and real-time coaching to help them resolve customer issues faster and with greater accuracy. Using AI and generative AI, Agent Assist can recommend agent responses to customers, suggest the appropriate knowledge base content to solve a customer’s issue, transcribe or translate calls in real time, summarize conversations and more.

**Conversational Insights**

Whether using live or virtual agents, these connections with your customers provide lots of value and insights for your company. Conversational Insights analyzes conversational data from across the customer journey to provide contact center leaders and managers with the data-driven insights to boost efficiency, improve agent performance, and create better customer experiences. This tool uses machine learning analytics to provide you with information such as agent and caller sentiment, entity identification, and call topics. It can automatically identify interesting customer interactions in need of further review.

With Generative FAQ in Insights, you can also see the common questions that customers are asking your contact center and how these questions are being answered. This information can help you identify gaps in your FAQs, track trending questions, and improve customer service responses.

**Contact center as a service (CCaaS)**

Today's contact centers are complex, needing to handle 24/7 communication across multiple channels (phone, text, email, etc.) while maintaining security and customer privacy.

CCaaS, part of Google's Customer Engagement suite, simplifies this by providing a complete contact center solution. It manages the infrastructure, integrates with CRMs, and offers omnichannel support (consistent experience across all channels, like websites, apps, phone, and text). CCaaS handles simultaneous multichannel communication, channel switching, multimodal interactions (text, voice, images), and agent routing, freeing businesses to focus on customer experience. CCaaS seamlessly integrates with other Customer Engagement Suite tools, including Conversational Agents for automated support, Agent Assist for real-time agent guidance, and Conversational Insights for valuable data analysis.

An [AI agent](https://www.ibm.com/think/topics/ai-agents) is an autonomous AI program, it can perform tasks and accomplish goals on behalf of a user or another system without human intervention, by designing its own workflow and using available tools (other applications or services).

[**Agentic AI**](https://www.ibm.com/think/topics/agentic-ai) is a system of multiple AI agents, the efforts of which are coordinated, or orchestrated, to accomplish a more complex task or a greater goal than any single agent in the system could accomplish.

**Agentic AI** is a type of artificial intelligence (AI) that can act independently and autonomously without constant human supervision. It's also known as AI agents. Agentic AI is considered the next wave of AI, and it's characterized by its ability to:

**Learn and adapt**

Agentic AI can learn from experience and feedback, and adapt to changing circumstances.

**Make decisions**

Agentic AI can understand the user's goal and the context of the problem, and then make decisions based on that information.

**Solve complex problems**

Agentic AI can handle complex tasks and sequences of activities, and can independently search databases or trigger workflows.

Agentic AI can be used in a variety of industries and applications, including:

**Healthcare**

AI agents can help doctors analyze medical data, automate administrative tasks, and provide 24/7 support to patients.

**Grid management**

Agentic AI can learn to recognize consumption patterns, and can help utility managers and grid operators manage resources autonomously.

**Supply chain management**

Agentic AI can predict demand, optimize inventory, and coordinate shipments.

**Manufacturing**

Agentic AI can help manufacturing robots respond to unexpected scenarios, such as anomalies on a conveyor belt.

**Finance**

Agentic AI can execute trades at high speeds and identify arbitrage opportunities.

Agentic AI uses a combination of machine learning, natural language processing, and automation technologies.

Build Agentic AI Application with the help of **frameworks such as Langflow, Phidata, LangGraph, Autogen, LangChain, Huggingface, Crew AI, Agno**

**Langchain** (One of the most popular framework used to create GenAI application)LangChain is an open-source orchestration framework that simplifies building applications with [large language models](https://cloud.google.com/ai/llms) (LLMs). It provides tools and components to connect LLMs with various data sources, enabling the creation of complex, multi-step workflows.

Available as libraries in Python and JavaScript, LangChain helps developers enhance LLM capabilities beyond text generation by linking them to external data and computation. This helps facilitate the development of advanced AI applications like intelligent chatbots, sophisticated question-answering systems, and automated data analysis tools.  
  
**LLM models**: OpenAI, Grok, ollama, anthropic, Google gemini, mistral AI, chatgpt

**Vertex AI:**Vertex AI is Google Cloud's unified machine learning (ML) platform. Vertex AI streamlines the entire ML workflow by providing the necessary infrastructure, pre-trained models, and tools to build, deploy, and manage your ML and generative AI solutions. By using Vertex AI, companies can unlock the power of generative AI to streamline workflows, improve decision-making, and drive innovation.

Vertex AI is a managed machine learning platform that provides a comprehensive set of tools and services for building, training, deploying, and managing machine learning models. It is a good choice for organizations that need to create custom end-to-end AI models. Vertex AI is a managed machine learning (ML) platform that developers, data scientists, and researchers use to build, deploy, and manage high-quality, scalable ML models. These models are then capable of making predictions from data.

It empowers you to build, train, and deploy ML models and AI applications. For multiple modalities (text, code, images, speech), Vertex AI gives you access to Google's large generative AI models through [Model Garden](https://cloud.google.com/vertex-ai/generative-ai/docs/model-garden/explore-models). You can tune Google's large language models (LLMs) to meet your needs and then deploy them for use in your AI-powered applications. Simply put, Vertex AI is the key to creating and deploying custom generative AI solutions for your business.

**Vertex AI** is a unified, fully managed platform on Google Cloud that allows data scientists and developers to build, train, and deploy machine learning (ML) and generative AI models. It provides a single environment for the entire ML lifecycle, from data preparation and model training to deployment and management, with access to Google's advanced AI models like [Gemini](https://www.google.com/search?sca_esv=3fb0e2d94fbc189e&rlz=1C1GCEB_en&q=Gemini&sa=X&sqi=2&ved=2ahUKEwieq8WMiqGRAxXlyzgGHc_JDuUQxccNegQIVRAB&mstk=AUtExfDG9uu5cbpz7WrqxGhtT6413xMCNEMluwQyas2v4re2Dmj4RUwNvWmyxFSpvPHVs4cthzi9JmhSgS1F5dGDqlPh4R901YbdvGP8T4f-ynpdSvCmeCScCnunrGe028KsFf4BMqaPquxjnq92_6gvWCn3t5ikPs7FrEexjHOwpKY-ANXjgsbRESIMq8gEfXkdq9Nm&csui=3). Key features include the [Model Garden](https://www.google.com/search?sca_esv=3fb0e2d94fbc189e&rlz=1C1GCEB_en&q=Model+Garden&sa=X&sqi=2&ved=2ahUKEwieq8WMiqGRAxXlyzgGHc_JDuUQxccNegQIVhAB&mstk=AUtExfDG9uu5cbpz7WrqxGhtT6413xMCNEMluwQyas2v4re2Dmj4RUwNvWmyxFSpvPHVs4cthzi9JmhSgS1F5dGDqlPh4R901YbdvGP8T4f-ynpdSvCmeCScCnunrGe028KsFf4BMqaPquxjnq92_6gvWCn3t5ikPs7FrEexjHOwpKY-ANXjgsbRESIMq8gEfXkdq9Nm&csui=3) for a wide selection of pre-trained models, tools for MLOps (ML operations), and a unified interface for data analysis and AI development.

Vertex AI streamlines the integration of advanced artificial intelligence capabilities into business applications, allowing for seamless discovery, deployment, and customization. These models empower businesses to leverage cutting-edge AI, providing the flexibility to work with many different models without the need for extensive in-house model development.

**Key features and benefits of Vertex AI**

Open and flexible: Vertex AI supports open frameworks, allowing you to use your preferred tools and models without vendor lock-in.

Comprehensive tooling: Develop, deploy, and manage your models with ease using Vertex AI's integrated tools.

Powerful infrastructure: Leverage Google Cloud's scalable and reliable infrastructure to handle your ML workloads.

Customization: Fine-tune and adapt models to your specific needs using the built-in IDE.

Pre-trained models: Choose from a wide variety of pre-trained models or build your own.

Easy integration: Integrate your models into applications seamlessly using available APIs.

**Vertex AI's MLOps tools**

Vertex AI also includes ML operations (MLOps) tools built in to help you orchestrate end-to-end ML workflows, perform feature engineering, run experiments, manage and iterate your models, track ML metadata, and monitor and evaluate model quality. MLOps helps automate, standardize, and manage the ML project lifecycle, enabling better collaboration and continuous improvement. By using Vertex AI, you get a fully managed compute infrastructure, high-performance ML optimized training jobs, distributed training, hyperparameter optimization, and enterprise security.  
Vertex AI simplifies the complexities of machine learning, making it easier to build and deploy AI solutions. Its unified platform, comprehensive tooling, and MLOps capabilities empower developers and data scientists to innovate and accelerate their AI initiatives.

With Vertex AI you can access models developed by Google including **Gemini**, **Gemma**, **Imagen**, and **Veo**. You can also access proprietary third-party models, and openly available models.

**What Vertex AI offers:**

* **Unified platform:**

Provides a single environment for all ML and AI workflows, from data preparation to model deployment and monitoring, which helps teams collaborate more effectively.

* **Model Garden:**

A repository of over 200 pre-trained models, including Google's first-party models (like Gemini), third-party models, and open-source models, that you can use to build applications.

* **Generative AI capabilities:**

Offers access to Google's most capable generative AI models, including Gemini, with tools for building applications through simple interfaces like Vertex AI Studio.

* **MLOps tools:**

Includes a suite of MLOps features to streamline the end-to-end ML workflow, such as Vertex Explainable AI to understand model decisions, Vertex Pipelines to manage workflows, and Vertex Model Monitoring.

* **Custom model development:**

Supports building and training custom models with options ranging from no-code or low-code solutions like AutoML to fully customized, code-based training.

* **Vertex AI Workbench:**

Provides a managed notebook environment for data analysis and AI development, natively integrated with other Vertex AI services to accelerate model development

**Model Garden** is an AI/ML model library that helps you discover, test, customize, and deploy models and assets from Google and Google partners.

Model Garden provides access to pre-trained foundation models. While these can be fine-tuned, AutoML is specifically designed to build custom models from scratch using the company's own data with high levels of automation.

Model Garden on Vertex AI is a service that lets you discover, customize, and deploy existing models from Google and Google partners.

Model Garden gives you options to pick from over 160 models and offers options of Google models, third-party models, and open-source models. With Vertex AI, you can customize these models with your own data and deploy them to your custom applications. Additionally, some pre-trained models can be used out-of-the-box without tuning.

Advantages of Model Garden

When you're working with AI models, Model Garden provides the following advantages:

* Available models are all grouped in a single location
* Model Garden provides a consistent deployment pattern for different types of models
* Model Garden provides built-in integration with other parts of Vertex AI such as model tuning, evaluation, and serving
* Serving generative AI models can be difficult—Vertex AI handles model deployment and serving for you

To view the list of available Vertex AI and open source foundation, tunable, and task-specific models, go to the Model Garden page in the Google Cloud console.

The model categories available in Model Garden are:

A screenshot of a computer

AI-generated content may be incorrect.

To filter models in the filter pane, specify the following:

* **Tasks**: Click the task that you want the model to perform.
* **Model collections**: Click to choose models that are managed by Google, partners, or you.
* **Providers**: Click the provider of the model.
* **Features**: Click the features that you want in the model.

**Vertex AI Studio:** Vertex AI Studio is a Google Cloud console tool for rapidly prototyping and testing generative AI models. Test sample prompts, design your own prompts, and customize foundation models to handle tasks to meet your application's needs. Vertex AI Studio is a tool to interact with generative AI models, prototype business ideas, and launch them into production.

**Vertex AI Workbench:** Vertex AI Workbench is a Jupyter notebook-based development environment for the entire data science workflow. You can interact with Vertex AI and other Google Cloud services from within a Vertex AI Workbench instance's Jupyter notebook. Vertex AI Workbench provides a JupyterLab experience and advanced customization capabilities

Vertex AI Pipelines (MLOps tool) lets you automate, monitor, and govern your machine learning (ML) systems in a serverless manner by using ML pipelines to orchestrate your ML workflows.

**AutoML** in Vertex AI -- > To allow users to build and train AI models with minimal technical expertise. (AutoML simplifies the model-building process, making AI accessible to a wider range of users, regardless of their coding skills.)

**Vertex AI AutoML** enables developers with limited machine learning expertise to train high-quality models specific to their business needs. It automates tasks like feature engineering, model selection, and hyperparameter tuning for various data types, including tabular data (like customer purchase history) for prediction tasks.

**MLOps** in Vertex AI --> To automate, standardize, and manage the ML project lifecycle. (MLOps tools in Vertex AI help streamline the ML workflow, enabling better collaboration and continuous improvement in AI projects.)

**Vertex AI Search** helps developers build secure, Google-quality search experiences for websites, intranet and RAG systems for generative AI agents and apps. Vertex AI Search is a part of Vertex AI Agent Builder. Vertex AI search is a Google cloud service that allows developers to easily integrate advanced search capabilities into their applications. It’s a fully managed, scalable service that can handle large volumes of multimodal data and complex search queries.

**Vertex AI Search** is for building search and recommendation applications, often leveraging pre-built capabilities, rather than training custom prediction models from tabular data in an automated fashion.

**Vertex AI Search is a fully managed Google Cloud service that lets you build and deploy customized search engines across your own data, enhanced by the Gemini Generative AI model for natural language responses. You can index and query websites, structured data, documents, databases, Google Drive, Cloud Storage, and third-party sources with minimal setup. This enables developers, even those with limited AI expertise, to leverage cutting-edge innovations in Generative AI, Semantic Search, Embeddings, and Retrieval Augmented Generation (RAG) to create robust search experiences.**

**Vertex AI Search offers the following key capabilities and configurations:**

* **search**

**Search**

**Out-of-the-box natural language understanding and semantic search.**

* **recommend**

**Recommendations**

**ML-based content and metadata understanding so that your users quickly find content that is similar to the content that they're looking for.**

* **text\_analysis**

**Generative AI**

**Generative AI-powered answers and summarization with conversational search.**

* **terminal**

**Console or API**

**Use the AI Application page in Google Cloud console or the Discovery Engine API to set up a search widget for your users.**

* **screen\_search\_desktop**

**Widget**

**Integrate the search into your website using the widget.**

* **web\_traffic**

**Self-learning models**

**Get self-learning ranking models and advanced analytics. This requires user events that based on user's clickstream.**

* **movie\_info**

**Optimization for media**

**Create recommendation and search apps optimized for media content.**

* **health\_and\_safety**

**Natural language querying of healthcare data**

**Search FHIR resources without prior knowledge of any query language.**

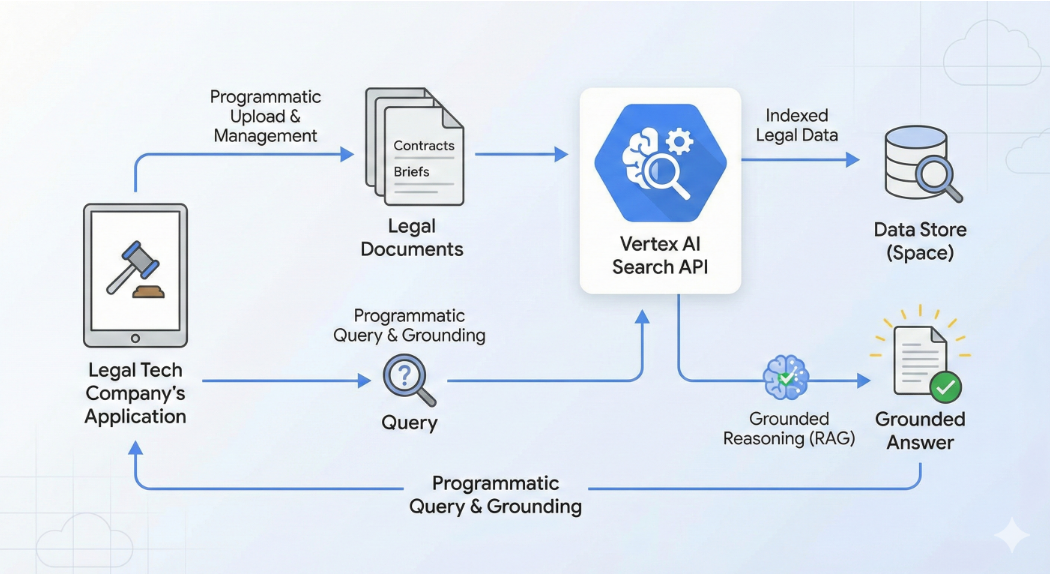
* **medical\_information**

**Context-aware healthcare searches**

**Find search results with semantic relevance that a structured FHIR search might miss.**

**TensorFlow** (**building advanced machine learning models**) - TensorFlow is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [software library](https://en.wikipedia.org/wiki/Library_(computing)) for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). It provides a comprehensive, flexible ecosystem of tools, libraries, and community resources that let researchers push the state-of-the-art in ML, and developers easily build and deploy ML-powered applications. TensorFlow supports deep learning along with a suite of tools for model building, training, and deployment, making it ideal for a wide variety of machine learning applications across different platforms and languages. **TensorFlow** is an open-source machine learning framework that can be used to build and deploy custom machine learning applications. It is also compatible with TPUs, which are Google's custom-designed machine learning accelerators.

**Vertex AI Search API** provides programmatic control to create and manage data stores (serving as document "spaces"), upload and index legal documents, and execute semantic search queries with grounded reasoning over private document collections. The API supports RAG (retrieval-augmented generation) by connecting Gemini models to enterprise data, enabling applications to generate answers grounded in specific document corpora—precisely what legal e-discovery applications require for querying case law, contracts, and legal briefs programmatically.



**Agent Development Kit (ADK)** is a flexible and modular framework for **developing and deploying AI agents**. While optimized for Gemini and the Google ecosystem, ADK is **model-agnostic**, **deployment-agnostic**, and is built for **compatibility with other frameworks**. ADK was designed to make agent development feel more like software development, to make it easier for developers to create, deploy, and orchestrate agentic architectures that range from simple tasks to complex workflows.

We recommend deploying your ADK agent to [Vertex AI Agent Engine](https://docs.cloud.google.com/agent-builder/agent-engine/overview) Runtime, a fully managed Google Cloud service specifically designed for deploying, managing, and scaling AI agents built with frameworks such as ADK.

Vertex AI Agent Engine, a part of the Vertex AI Platform, is a set of services that enables developers to deploy, manage, and scale AI agents in production. Agent Engine handles the infrastructure to scale agents in production so you can focus on creating applications. Vertex AI Agent Engine offers the following services that you can use individually or in combination:

* **Runtime**:
  + [Deploy](https://docs.cloud.google.com/agent-builder/agent-engine/deploy) and scale agents with a managed runtime and end-to-end management capabilities.
  + Customize the agent's container image with build-time installation scripts for system dependencies.
  + Use security features including VPC-SC compliance and configuration of authentication and IAM.
  + Access models and tools such as [function calling](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/multimodal/function-calling).
  + Deploy agents built using [different Python frameworks](https://docs.cloud.google.com/agent-builder/agent-engine/overview#supported-frameworks) and the [Agent2Agent open protocol](https://docs.cloud.google.com/agent-builder/agent-engine/develop/a2a).
  + Understand agent behavior with [Google Cloud Trace](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/manage/tracing) (supporting [OpenTelemetry](https://opentelemetry.io/)), [Cloud Monitoring](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/manage/monitoring), and [Cloud Logging](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/manage/logging).
* **Quality and evaluation** (Preview): Evaluate agent quality with the integrated [Gen AI Evaluation service](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/evaluate) and optimize agents with Gemini model training runs.
* [**Example Store**](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/example-store/overview) (Preview): Store and dynamically retrieve few-shot examples to improve agent performance.
* [**Sessions**](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/sessions/overview) (Preview): Agent Engine Sessions lets you store individual interactions between users and agents, providing definitive sources for conversation context.
* [**Memory Bank**](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/memory-bank/overview) (Preview): Agent Engine Memory Bank lets you store and retrieve information from sessions to personalize agent interactions.
* [**Code Execution**](https://docs.cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/code-execution/overview) (Preview): Agent Engine Code Execution lets your agent run code in a secure, isolated, and managed sandbox environment.

Retrieval-augmented generation (RAG) enhances the capabilities of large language models (LLMs) by grounding their responses in external knowledge sources. This process allows the model to access and process information beyond its training data, leading to more accurate, relevant, and up-to-date responses.

**Retrieval-Augmented Generation (RAG)** is the process of optimizing the output of a large language model, so it references an authoritative knowledge base outside of its training data sources before generating a response. Large Language Models (LLMs) are trained on vast volumes of data and use billions of parameters to generate original output for tasks like answering questions, translating languages, and completing sentences. RAG extends the already powerful capabilities of LLMs to specific domains or an organization's internal knowledge base, all without the need to retrain the model. It is a cost-effective approach to improving LLM output so it remains relevant, accurate, and useful in various contexts.

RAG (Retrieval-Augmented Generation) is an AI framework that combines the strengths of traditional information retrieval systems (such as search and databases) with the capabilities of generative large language models (LLMs). By combining your data and world knowledge with LLM language skills, grounded generation is more accurate, up-to-date, and relevant to your specific needs.

Retrieval-Augmented Generation (RAG) is an advanced AI framework that combines information retrieval with text generation models like GPT to produce more accurate and up-to-date responses. Instead of relying only on pre-trained data like traditional language models, RAG fetches relevant documents from an external knowledge source before generating an answer.

Before retrieval-augmented generation (RAG), models could not directly learn from tool data. They could use tools to fetch information, but lacked the ability to process and integrate that data into their knowledge. Their understanding was limited to their training data, and tool-retrieved data was used only for the immediate query, not for learning. RAG solves this by enabling models to retrieve and learn from data provided by tools.

A close-up of a document

AI-generated content may be incorrect.

Retrieval augmented generation, or RAG, is an architecture for optimizing the performance of an [artificial intelligence (AI)](https://www.ibm.com/topics/artificial-intelligence) model by connecting it with external knowledge bases. RAG helps [large language models (LLMs)](https://www.ibm.com/topics/large-language-models) deliver more relevant responses at a higher quality.

A screenshot of a computer

AI-generated content may be incorrect.

RAG, or Retrieval-Augmented Generation, is a technique that combines large language models (LLMs) with external knowledge bases to improve the accuracy and relevance of AI-generated responses. Instead of relying solely on its training data, a RAG system first retrieves relevant information from a specified source and then uses that information to generate a more contextually appropriate and up-to-date answer. This approach helps overcome LLM limitations like factual errors and outdated information without requiring expensive model retraining.

**Benefit of RAG**

1. **Access to Updated Knowledge:**LLMs are trained on fixed datasets but RAG allows them to fetch fresh and real time information from external sources.
2. **Improved Accuracy:** It reduces hallucinations in LLMs and makes answers more factually correct.
3. **Domain Specific Expertise: It l**ets us use specialized datasets like medical records and legal documents to get expert-level responses without retraining the model.
4. **Cost Efficiency:** Instead of retraining massive LLMs with new data, we simply update the external knowledge base hence saving time and resources.
5. **Personalization:** RAG can retrieve user specific information like past interactions or personal data to provide more tailored and relevant responses.

**Components of RAG**

The main components of RAG are:

1. **External Knowledge Source:**Stores domain specific or general information like documents, APIs or databases.
2. **Text Chunking and Preprocessing:**Breaks large text into smaller, manageable chunks and cleans it for consistency.
3. **Embedding Model:** Converts text into numerical vectors that capture semantic meaning.
4. **Vector Database:**Stores embeddings and enables similarity search for fast information retrieval.
5. **Query Encoder:** Transforms the user’s query into a vector for comparison with stored embeddings.
6. **Retriever:**Finds and returns the most relevant chunks from the database based on query similarity.
7. **Prompt Augmentation Layer:**Combines retrieved chunks with the user’s query to provide context to the LLM.
8. **LLM (Generator):**Generates a grounded response using both the query and retrieved knowledge.
9. **Updater (Optional):**Regularly refreshes and re-embeds data to keep the knowledge base up to date.

**How RAG works**

After the user submits a query or request to the LLM, here's how RAG works with tools.

**Retrieval:** When a user asks a question, the RAG system performs a search on an external knowledge base to find information that is relevant to the query. This knowledge base can include anything from the latest internet data to a company's internal documents. The LLM, equipped with retrieval tools, identifies relevant information from external sources. These tools can include:

* Data stores: These can be internal databases or other sources of structured and unstructured data.
* Vector databases: These databases store embeddings (numerical representations) of text, allowing the LLM to find semantically similar information to the user's query. The LLM uses the user query to create an embedding and searches the vector database for matching documents or passages.
* Search engines: The LLM can use search engines (via extensions or APIs) to find relevant web pages, articles, or other online content.
* Knowledge graphs: These structured databases store information about entities and their relationships. The LLM can query knowledge graphs to retrieve facts and relationships relevant to the user's query.

LLM uses these tools to retrieve a set of potentially relevant documents or passages.

**Augmentation:** The retrieved information is then incorporated (or "augmented") into the prompt that is fed to the LLM. This augmented prompt now contains both the user's original query and the relevant context retrieved from external sources.

**Generation:** The LLM processes the augmented prompt and generates a response that is now grounded in the information it was just given, making the answer more accurate, specific, and relevant. Because the prompt includes relevant external information, the LLM can generate a more informed, accurate, and contextually appropriate response. It can also cite the sources of its information, increasing transparency and trustworthiness.

**Iteration** **(optional)**: In some RAG systems, the LLM might iterate on the retrieval process. For example, if the initial retrieval doesn't yield satisfactory results, the LLM might refine its query or use different retrieval tools to find more relevant information. It could also ask clarifying questions of the user. This iterative process allows the LLM to continuously improve the quality and relevance of its responses.

A screenshot of a computer

AI-generated content may be incorrect.

A diagram of a chat with a document application process

AI-generated content may be incorrect.

**Data Stores**

RAG allows LLMs to access and process information beyond their training data, making them more accurate and relevant. A key component of this process are data stores in AI Applications. They act as structured and unstructured knowledge bases that your agent can draw upon.

You can access a variety of data store options to connect your agent to your data through AI Applications, including:

* **Websites:** Access and process information directly from web pages, enabling your agent to stay up-to-date with current events or access publicly available information.
* **Structured data:**Connect to information organized in tables or JSON format, such as product catalogs, customer databases, and internal knowledge bases. Vertex AI can automatically understand the structure of your data or allow you to define it.
* **Unstructured data:**Utilize files in various formats such as HTML, PDF, and DOCX, making it possible for your agent to access information from a wide range of sources.

Data stores in AI Applications are essential for building powerful and reliable AI agents. They provide a way to manage and organize your agent's knowledge, enabling it to provide accurate, relevant, and up-to-date responses. By leveraging the power of RAG and data stores, you can create agents that are truly helpful and informative.

**Example: Answering a question about a recent event**

**1) User query**: "What were the main developments at the recent climate conference in Dubai?"

**2) Retrieval (**Fetching relevant documents from a knowledge base based on user query):

The LLM uses a combination of tools:

* It queries a vector database of news articles related to climate change.
* It uses a search engine to find recent news reports and official conference documents.

**3) Augmentation** (Adding context to the query by including additional information from the knowledge base): Creating a response to the user's query in natural language. The LLM combines the user's query with the retrieved information (summaries of key discussions, agreements, and announcements from the conference).

**4) Generation** (Creating a response to the user's query in natural language): The LLM generates a response summarizing the main developments at the climate conference, citing the news articles and conference documents it used as sources. Because it has access to up-to-date information, the LLM's response is more accurate and comprehensive than if it had relied solely on its training data.

**5) Iteration (optional)**: If the initial search results are insufficient, the LLM might try different search terms or query other data sources to get more complete information.

By combining the power of LLMs with the ability to access and process external knowledge, RAG systems create more powerful and reliable AI agents.

An **MCP (Model Context Protocol) server** is a specialized service that connects Large Language Models (LLMs) to real-world data and tools, acting as a standardized bridge for AI to access files, databases, APIs, and other systems, enabling them to perform actions and provide grounded, real-time information beyond their training data. It defines a standard way for AI applications ([hosts](https://www.google.com/search?q=hosts&rlz=1C1GCEB_en&oq=what+is+mcp+server&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQIRiPAjIHCAIQIRiPAtIBCDcxMzRqMGoxqAIAsAIA&sourceid=chrome&ie=UTF-8&ved=2ahUKEwj3veP74IuSAxUE2TgGHR_vBywQgK4QegYIAQgAEAY)) to request and receive context from servers that provide specific resources (like document data) or tools (like code execution).

**Key Functions**

* **Data Access:** Provides LLMs with up-to-date information from internal or external sources (e.g., company documents, databases, web search).
* **Action Execution:** Allows AI to interact with external systems, like running code, updating CRM records, or fetching data via APIs.
* **Standardization:** Simplifies integration by acting as a universal adapter, removing the need for complex, custom code for each tool.
* **Context Management:** Manages the flow of data and instructions, ensuring the AI has the right context for its responses.

**Examples of MCP Servers**

* **File System Servers:** Access local or cloud files.
* **Database Servers:** Query databases for specific data.
* **GitHub Servers:** Interact with code repositories.
* **Slack Servers:** Connect to team communication.
* **Microsoft Learn Server:** Provides access to Microsoft's technical documentation for coding assistants.

**How it Works**

1. An [**AI Host**](https://www.google.com/search?q=AI+Host&rlz=1C1GCEB_en&oq=what+is+mcp+server&gs_lcrp=EgZjaHJvbWUyBggAEEUYOTIHCAEQIRiPAjIHCAIQIRiPAtIBCDcxMzRqMGoxqAIAsAIA&sourceid=chrome&ie=UTF-8&ved=2ahUKEwj3veP74IuSAxUE2TgGHR_vBywQgK4QegYIAQgGEAE) (the main application) uses an **MCP Client** to connect to an **MCP Server**.
2. The client requests specific data (a **Resource**) or an action (a **Tool**) from the server.
3. The server fetches the data or performs the action and sends the results back to the client in a structured format.
4. The LLM uses this real-world context to generate more accurate and capable responses, reducing hallucinations.

embedding convert text ----> vector

An **embedding** is a numerical representation of complex data, like words or images, that captures its meaning and relationships in a format that machine learning models can process. These representations are essentially vectors—arrays of numbers—where similar items are placed closer together in a multi-dimensional space, allowing algorithms to find patterns and understand relationships between them. This is foundational for AI applications like semantic search, which finds similar documents based on meaning rather than just keywords.

**Vectorization** is the process of converting data into numerical vectors (arrays of numbers) or changing an algorithm to operate on sets of values at once instead of one at a time. This conversion is essential for machine learning and other data processing tasks because algorithms need numerical data to work with. It can also refer to transforming a graphic from a pixel-based image (raster) into a scalable vector image or a hardware optimization that allows a single instruction to operate on multiple data points simultaneously.

DAG, RAG, Vector Database, Fine tunning, Prompt engineering, MCP, Embedding, Auto ML (Model training, build model), ML Ops tool (like vertex ai pipeline), **Context window**

Vertex AI Pipelines lets you automate, monitor, and govern your machine learning (ML) systems in a serverless manner by using ML pipelines to orchestrate your ML workflows.

traditional vs Agentic RAG

chunking, parsing, embedding, vector database, vector embedding, vectors

Nano Banana Pro (Imagen latest model)

Imagen4 model to generate image  
  
**Guardrails** in technology are safety mechanisms, protocols, and filtering systems designed to ensure that AI models, particularly Large Language Models (LLMs), operate within set boundaries. They validate, monitor, and filter inputs and outputs to prevent issues like hallucinations, toxic content, data leaks, and prompt injections, thereby ensuring AI reliability.  
  
Regular Expression (RegEx),   
tenseflow and pytorch

langchain and langraph