Are you ready to go **beyond**?

Insights Beyond Analytics

We are a consulting firm specialized in **AI, Data & Analytics.**

Business agility is the ability of an organization to adapt quickly to market changes, both internally and externally. This cannot exist without becoming a truly data-driven company.

SDG Group achieves this by co-creating optimal solutions with its customers, leveraging Data & Analytics services through a unique combination of business domain expertise and state-of-the-art technologies delivered by industry-leading talent.

We are pioneers in AI, Data & Analytics consulting and we are committed to unlocking organizations' hidden potential by offering in-depth analytics expertise.

SDG Group at a glance

Global Presence

Customer proximity is our purpose. Our global presence and a leading vision in the practices of Business Analytics and Data-Driven Solutions allow us to serve our customers worldwide at best. **Since November 2020 SDG joined Alten Group** (+57,700 professionals and +4.143 B € Revenues. Listed on Euronext Paris).

+185M Volume of Activity

A growth mind-set to Keep Moving Forward. While maintaining our **own specialized niche player value proposition** we have achieved a considerable volume of activities provided worldwide in consultancy, design support and the creation of business analytic models and solutions.

+2500 Employees

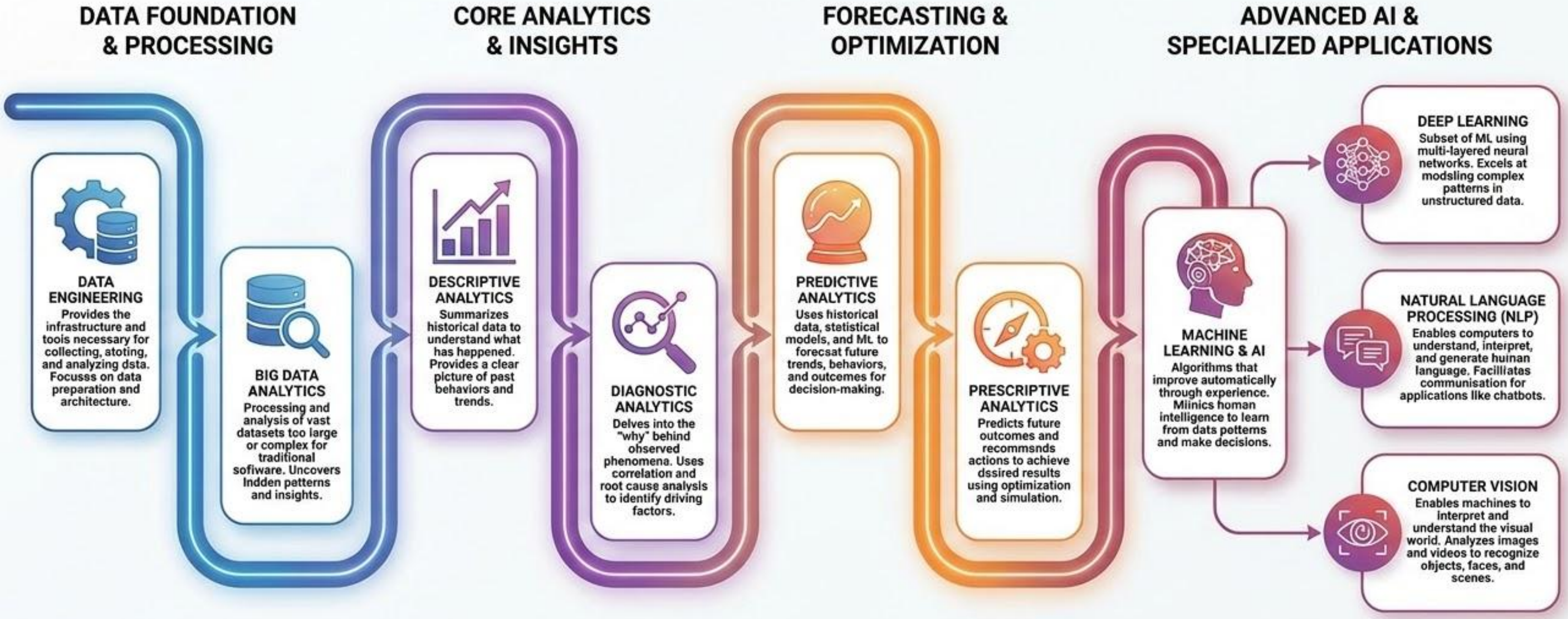
Driven by talent and innovation. We are constantly growing by attracting and retaining the best talents from the market. Our customers can rely on a skilled team passioned for innovation and committed to our shared fundamental **values such as meritocracy, teamwork, integral honesty and committed to excellence**.

+700 Customer Base

Our customer at the center. We serve an ever-increasing number of clients around the world. Working side by side with our clients acting as a **Partner. Collaboration based exclusively on the full achievement of the expected results**, evolving from specialist assignment to long-term partnership.

POPULAR DATA SCIENCE TYPES

Exploring the spectrum of techniques from foundational analysis to advanced artificial intelligence.



PREDICTIVE vs. GENERATIVE AI

Understanding the core differences and how they work together.

While **Predictive AI** tells you what is likely to happen, **Generative AI** creates something new based on what it has learned.

PREDICTIVE AI (Orange Section)	GENERATIVE AI (Purple Section)
Primary Goal: To forecast or categorize future outcomes.	**Primary Goal:** To create new, original content.
Input Data: Mostly structured data (numbers, dates, sales history).	**Input Data:** Massive sets of unstructured data (text, images, code).
Output: A score, a category, or a numerical forecast (e.g., '75% chance of rain').	**Output:** A brand new object (e.g., a poem, an image, or a functional code snippet).
Example: Predicting which customers will cancel a subscription.	**Example:** Writing a personalized email to keep those customers.

HOW THEY WORK TOGETHER (THE "TAG TEAM")

PREDICTIVE AI (The Analyst): It looks at your past data and says, "I predict sales will dip in July because of these three factors." →

GENERATIVE AI (The Creator): It takes that insight and says, "Okay, I've drafted five different summer marketing campaigns and generated three custom images to prevent that dip."

THE "EVOLUTIONARY" VIEW

Descriptive: "What happened?" (Sales were $1M). →

Predictive: "What will happen?" (Sales will be $1.2M next month). →

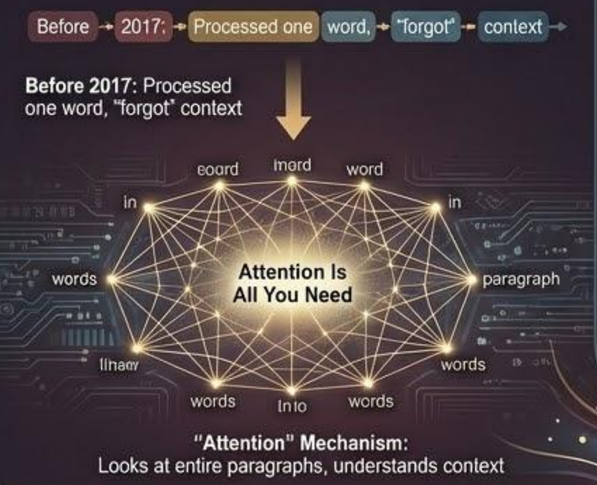
Prescriptive: "What should we do?" (Increase inventory by 10%). →

Generative: "Create the solution." (Generate the purchase orders and email the suppliers automatically).

The Perfect AI Storm: Convergence of Compute, Data, and Architecture

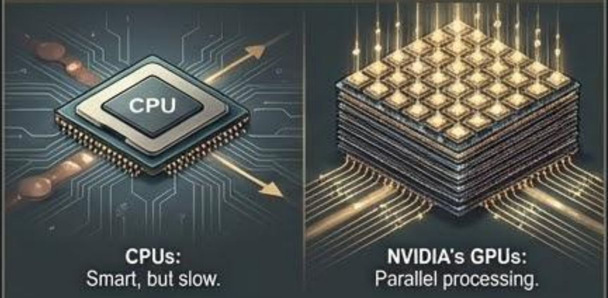
The "explosion" of AI wasn't a single invention, but a "perfect storm" of three specific ingredients finally coming together.

1. The "Transformer" Breakthrough (The Architecture)



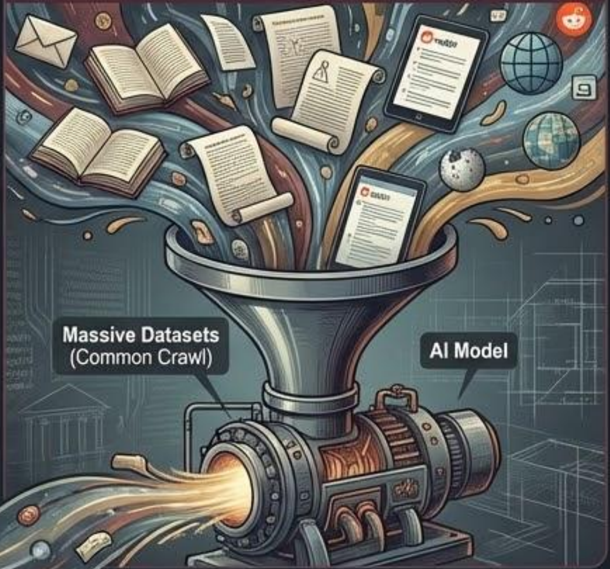
- **Before 2017**: AI processed linearly, "forgetting" context in long sentences.
- '**Attention Is All You Need**' paper introduced the Transformer model.
- '**Attention' Mechanism**: Looks at entire paragraphs at once, understanding long-range dependencies.
- Enabled Generative AI (like ChatGPT) by understanding context.

2. The GPU Revolution (The Compute)



- AI requires billions of math calculations per second.
- **CPUs**: Smart, but slow at parallel tasks.
- **NVIDIA's GPUs**: Originally for gaming, perfect for AI's parallel processing needs.
- **Scaling**: Chips like H100 enabled unprecedented training scale, physically impossible before.

3. The Digitization of Everything (The Data)



- AI is an engine that runs on data ("fuel").
- By 2020, almost every book, scientific paper, and internet content was digitized.
- **Common Crawl**: Massive datasets allowed feeding AI "the entire internet."
- Provided AI with "common sense" and abilities to code, write, and translate.

An introduction to foundation models

Versatility & Scale: Pre-trained on vast, multimodal datasets (text, image, audio, video) with billions of parameters.

Transfer Learning: Designed to apply acquired knowledge to new tasks with minimal additional training (fine-tuning).

From Understanding to Creation: Moves beyond standard NLU (understanding) to **Generative AI** (creating new content).

The "Unified" Solution: Replaces the need for separate models for individual tasks like sentiment analysis or NER.

Large Language Models are a specialized type of **Artificial Neural Network (ANN)**—mathematical models inspired by the human brain's structure and optimized to solve complex problems through pattern recognition.

Core Architecture: The Artificial Neuron

- **The Building Block:** The basic unit is the **node** (neuron).
- **Structure:** Organized into layers (Input, Hidden, Output).
- **Parameters:** Connections between neurons have **weights** that represent the strength of the relationship; these are optimized during training.

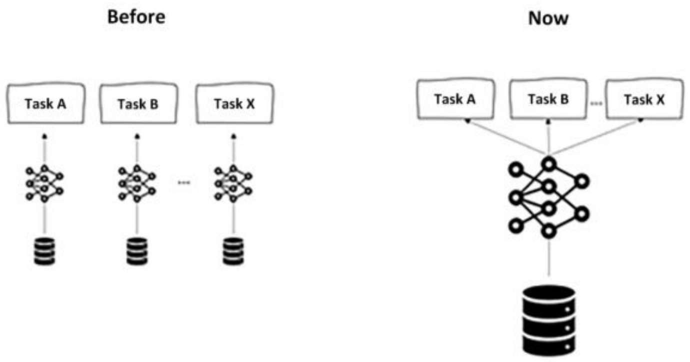


Figure 1.1: From task-specific models to general models

Under the hood of an LLM

Processing Text: From Words to Numbers

Since ANNs only process numerical data, unstructured text must undergo two critical transformations:

- **Tokenization:** Breaking text into smaller units (words, subwords, or characters) to create a structured format.
- **Embedding:** Converting tokens into **dense numerical vectors** in a continuous space. These vectors capture semantic meaning and context.

The Power of Vector Space

Embeddings allow models to perform mathematical operations on language. In a properly trained vector space, semantic relationships are represented by **spatial distance**.

Key Formula: King - Man + Woman ≈ Queen

- **Semantic Similarity:** Words with similar meanings are positioned closer together.
- **Context Awareness:** Captures complex relationships (e.g., gender, royalty, or verb tense) through multidimensional geometry.

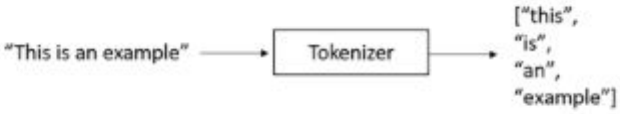


Figure 1.3: Example of tokenization

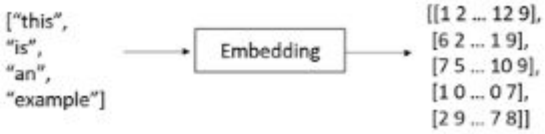


Figure 1.4: Example of embedding

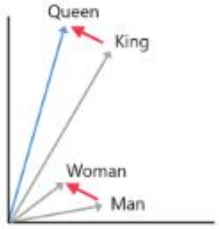


Figure 1.5: Example of words embedding in a 2D space

Under the hood of an LLM

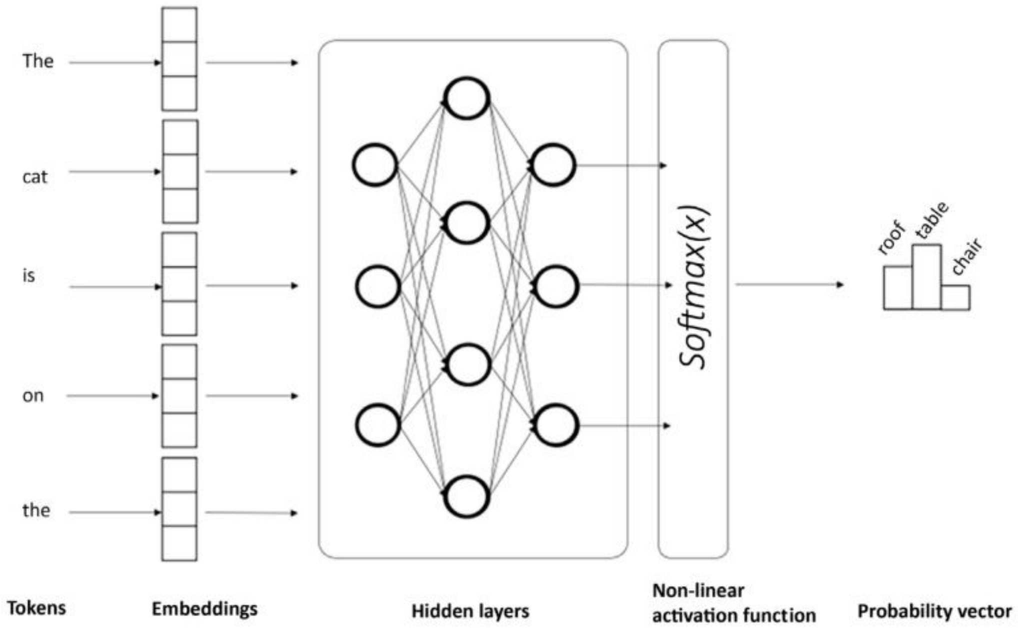


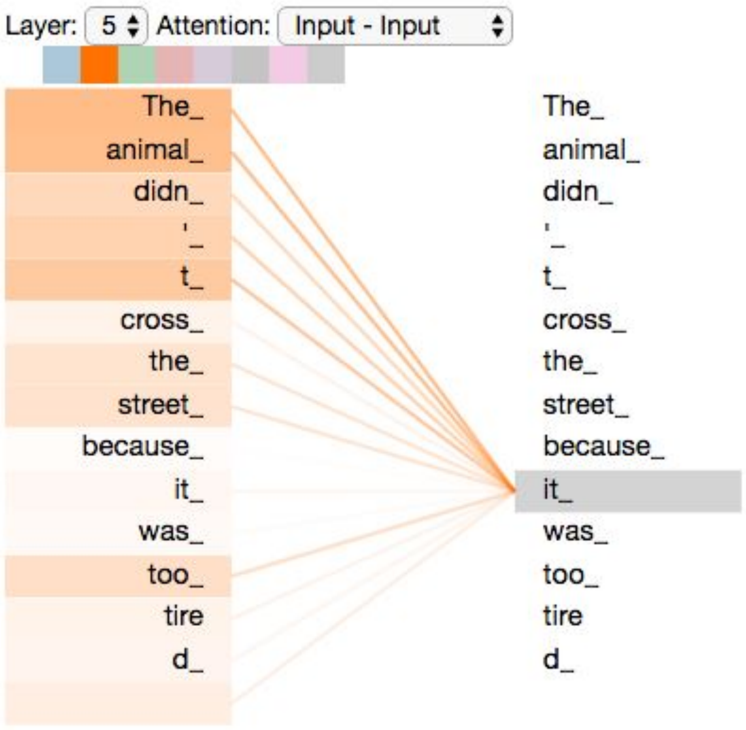
Figure 1.7: Predicting the next most likely word in an LLM

Next-Token Prediction: LLMs function as reasoning engines that predict the most likely next word based on a prompt.
The Role of Bayes' Theorem: The model updates the probability of a "candidate word" by combining:

- **Prior Probability (P(A)):** General frequency of the word in training data.
- **Likelihood (P(B|A)):** How well the word fits the specific current context.
- **Posterior Probability (P(A|B)):** The final score used to select the most coherent completion.

$$P(\text{"table"}|\text{"The cat is on the..."}) = \frac{P(\text{"table"})P(\text{"The cat is on the table"})}{P(\textit{"The cat is on the ..."})}$$

Attention is All You Need (Google, 2017)



The Problem: Contextual Ambiguity

Consider the sentence:

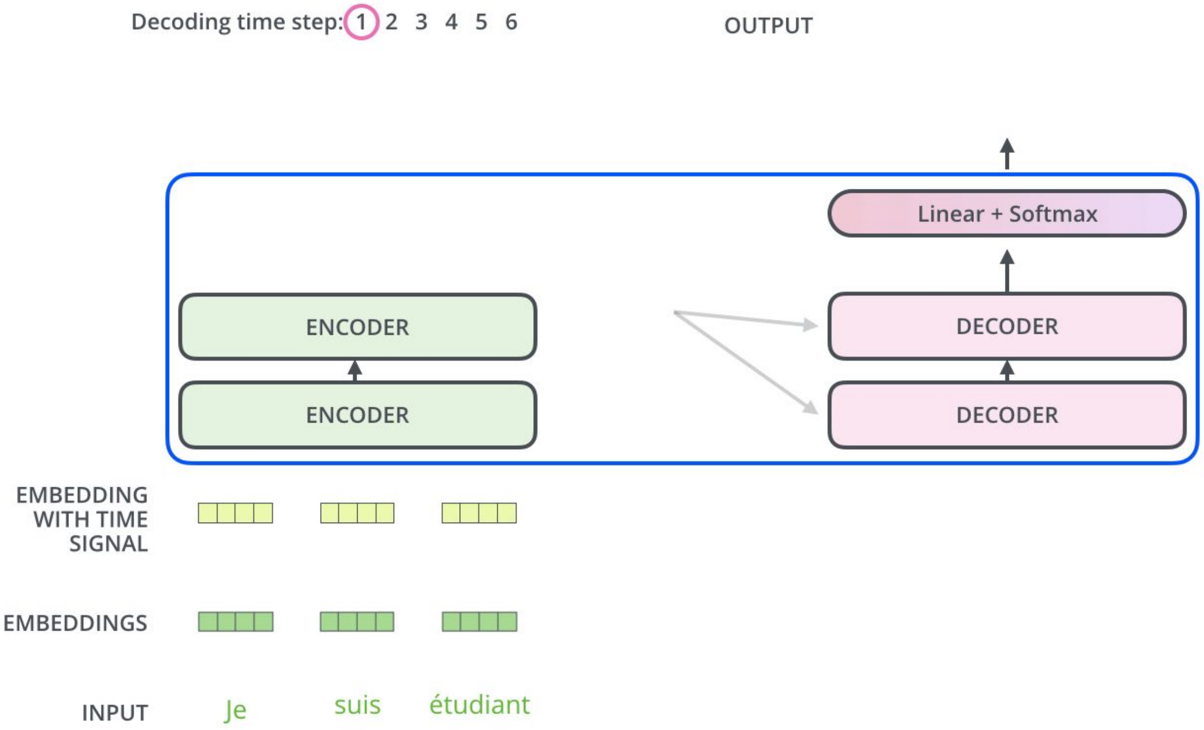
> *"The **animal** didn't cross the street because **it** was too tired."*

- **The Challenge:** For an algorithm, it is unclear if "**it**" refers to the *animal* or the *street*.
- **The Goal:** To create a "richer" encoding for the word "**it**" by incorporating information from the most relevant surrounding words.

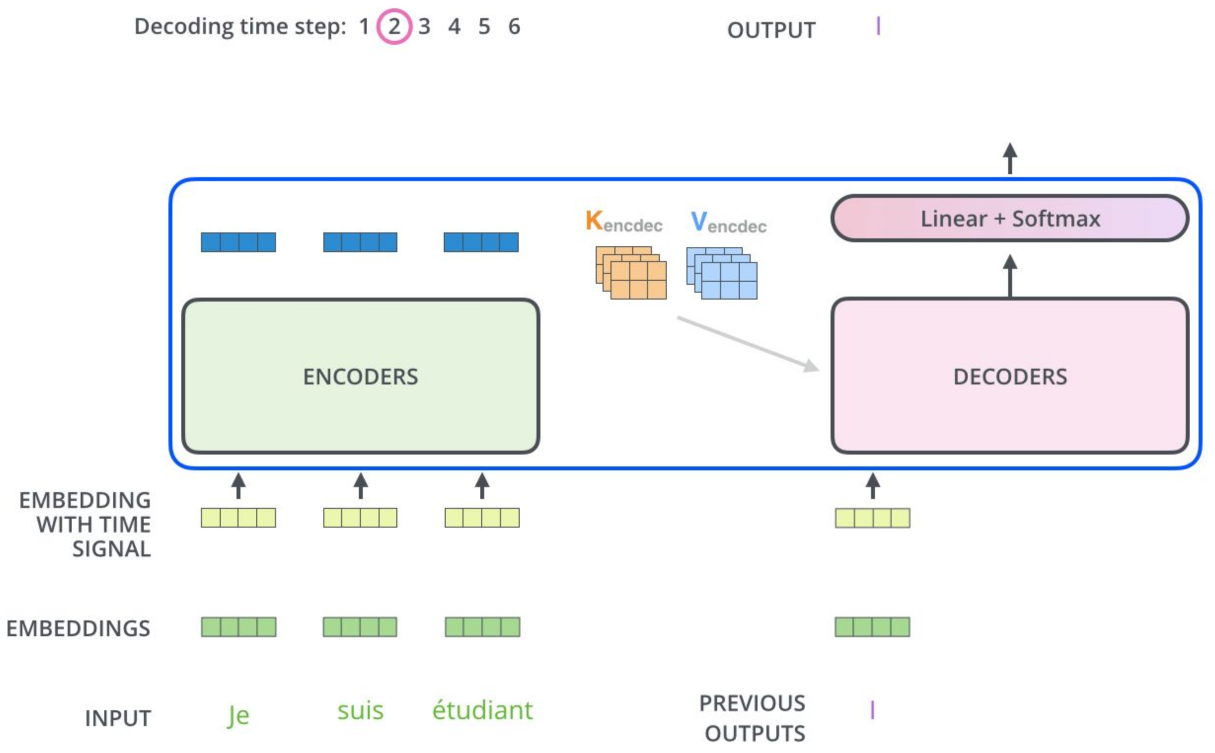
The Solution: How Self-Attention Works: As the model processes each word in a sequence, self-attention assigns different "weights" to other words based on their relevance:

- **Dynamic Relationships:** While encoding the word "**it**," the mechanism places high "attention" on the words "**The animal.**"
- **Information "Baking":** Part of the mathematical representation of "animal" is folded into the representation of "it."
- **Global Context:** Unlike older models (RNNs) that process word-by-word, self-attention allows the model to look at the **entire sequence simultaneously** to find clues.

Encoder - Decoder



Encoder - Decoder



AI orchestrators to embed LLMs into applications

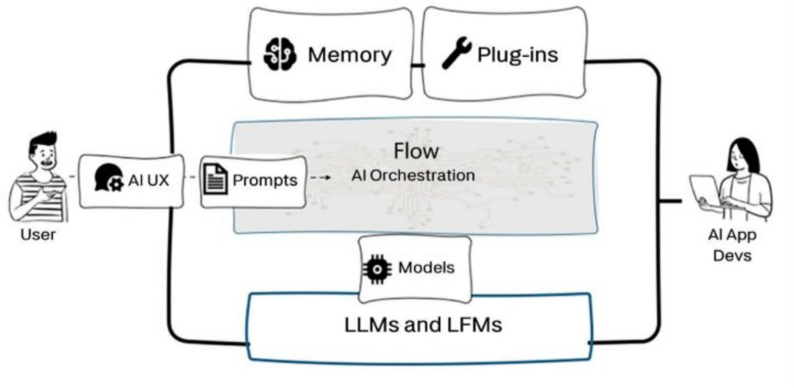


Figure 2.5: High-level architecture of LLM-powered applications

High-Level Architecture of LLM-Powered Apps: Modern LLM applications are built on a modular "Backend" that programs the model using natural language.

- **The Model:** Can be **Proprietary** (e.g., GPT-4, Gemini) or **Open-Source** (e.g., Llama, Falcon).
- **The Orchestrator:** Lightweight libraries (e.g., **LangChain, ADK**) that connect all components.
- **Plug-ins:** Modules that extend the LLM's functionality (e.g., web search, calculators).

Memory & VectorDB: Providing Context: To handle long conversations and specific knowledge, applications use two storage types:

- **Memory:** Stores past interactions so the model can refer back to the conversation history.
- **VectorDB:** A database storing **numerical embeddings**. It enables **semantic search** (finding data by meaning rather than keywords) using tools like FAISS, Pinecone, or Weaviate.

Prompt Engineering “Programming in English”: Prompting occurs at two distinct levels:

- **Frontend (User Prompt):** The natural language query entered by the user.
- **Backend (Meta-Prompts/System Messages):** Hidden instructions that "program" the model’s behavior (e.g., *"Act as a teacher for a 5-year-old"*).

LangGraph

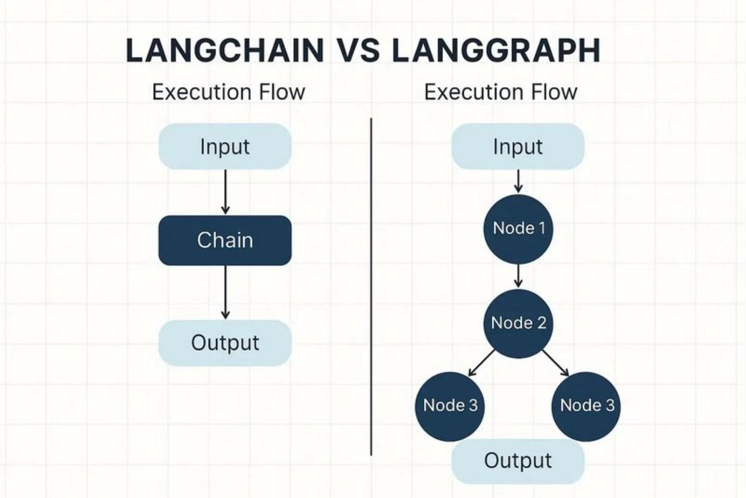
Core Concepts

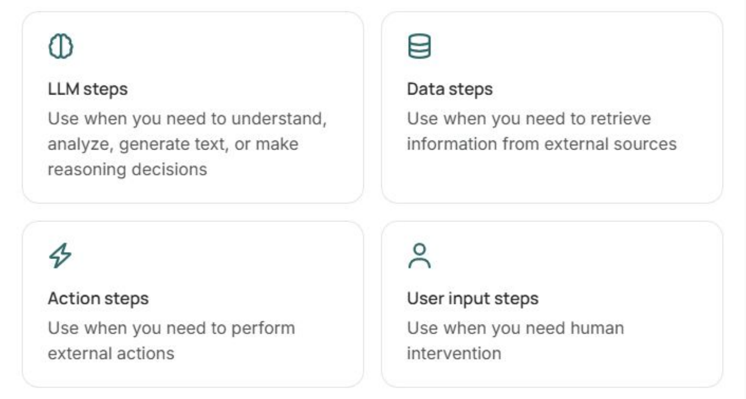
- **State:** A shared "memory" object that tracks information across the entire workflow.
- **Nodes:** Individual steps (LLMs, tools, or functions) that process and update the State.
- **Edges:** The logic paths connecting nodes, including **Conditional Edges** for decision-making.

Why it Matters

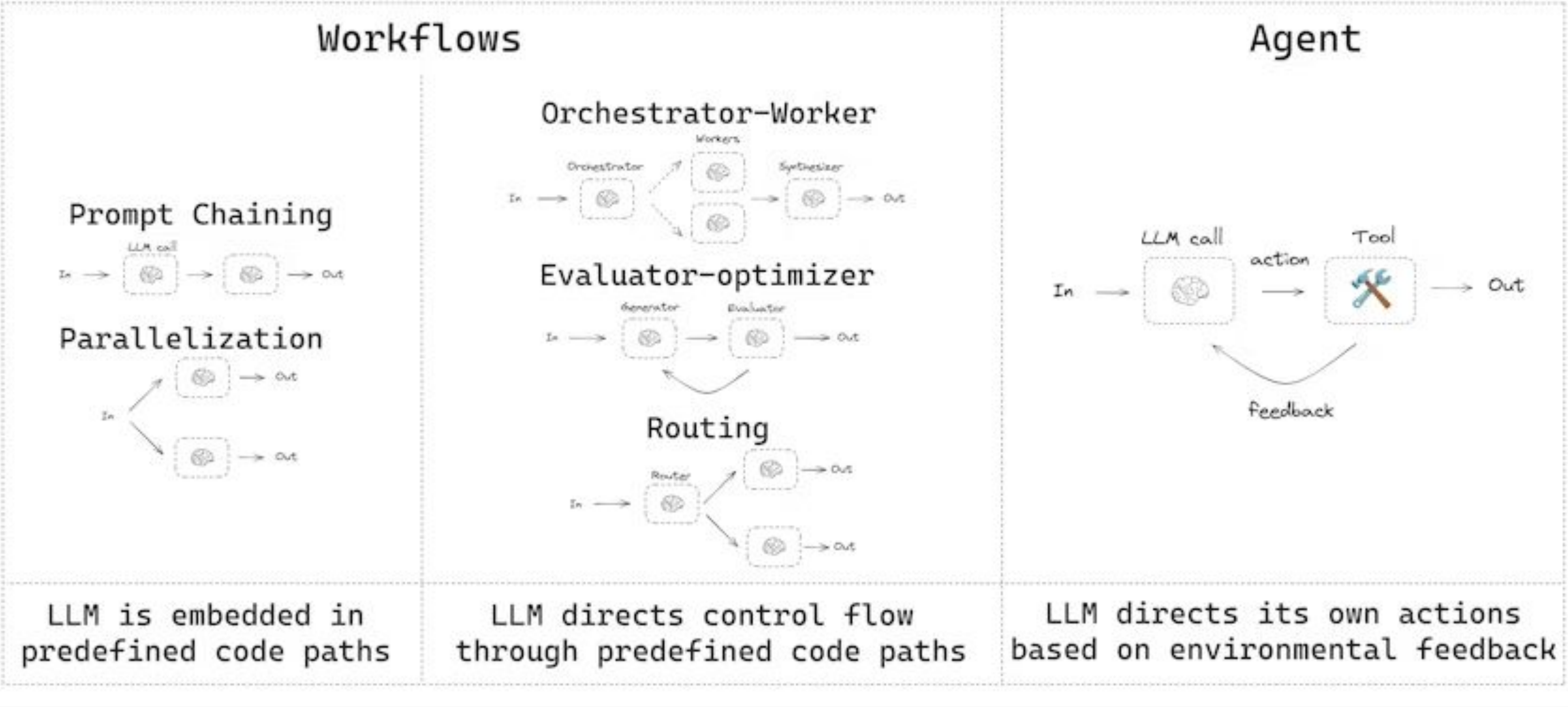
- **Cycles:** Unlike linear chains, it allows for **loops** (e.g., "Review → Edit → Repeat").
- **Human-in-the-Loop:** Built-in "breakpoints" to pause for human approval or manual edits.
- **Persistence:** Saves every step automatically, allowing you to "Time Travel" to any previous state for debugging.

Bottom Line: LangGraph provides the **control** of a flowchart with the **reasoning** of an LLM.





LangGraph


Workflows
Agent
Prompt Chaining
LLM call
In
Out
Parallelization
Orchestrator-Worker
Workers
Orchestrator
Synthesizer
Evaluator-optimizer
Generator
Evaluator
Routing
Router
LLM call
action
Tool
feedback
LLM is embedded in predefined code paths
LLM directs control flow through predefined code paths
LLM directs its own actions based on environmental feedback

Sources

Foundational Concepts

- **The"Attention Is All You Need" |** Vaswani et al. (2017)
  - The original research paper introducing the Transformer architecture.
  - arxiv.org/abs/1706.03762
- **Illustrated Transformer** | Jay Alammar
  - *A visual deep dive into the architecture powering modern Gen AI.*
  - jalammar.github.io/illustrated-transformer

Technical Guides & Documentation

- **Book:** *Building LLM-Powered Applications* by Valentina Alto
  - *A comprehensive guide to designing and deploying AI-driven solutions.*
- **LangGraph Overview** | LangChain Documentation
  - *Frameworks for building stateful, multi-agent applications.*
  - docs.langchain.com/langgraph

Implementation & Hands-on

- **Project Repository:** `agent4bees`
  - *Practical code samples and agentic workflows.*
  - github.com/MaBr84/agent4bees