

How an LLM Works: A Simple Way to Explain It

[Assem Hijazi – Experience, Knowledge & Ideas](#)

I need with this quick article to simplify the understanding of the LLM, when it is still complicated to explain. I tried to simplify it as much as possible and give examples to make it more understandable.

Okay, so imagine that when you ask an LLM (a language model) a question—let's say, something like "What's the population of Paris and how does it affect the economy?"—you're actually giving it a project to work on.

Now, this project isn't handled by one big brain, but instead by a bunch of teams working together.

Each of these teams is made up of layers of the LLM. These layers are like groups of workers, and they all have different roles in breaking down your question and figuring out the best answer. Some teams handle simple tasks, while others are more skilled for tougher projects.

So, How Does It All Work?

1. **The LLM Gets Your Question** (The Project) First off, you give the model a project by asking a question. It could be something simple like *"Is Paris the capital of France?"*, or it could be more complicated like the example above with Paris's population and the economy.
2. **Teams Start Analyzing** (Layers Jump In) Once the question (the project) is in, the different teams (layers) get to work. If it's a simple question, maybe only a few teams are needed. But for more complex questions, more teams get involved. Each team (layer) looks at a different part of the question, passing their results to the next team.
3. **Building Relationships Between Words** (Parameters) While the teams are working, they use parameters—which are basically the connections "relationship" between words they've learned from past projects (training). So, if the model has seen "Paris is the capital of France" hundreds of times in the past, it knows these words have a strong connection. Parameters are what make these links happen.
4. **Spotting Patterns** "keywords " , As the teams work through the project, they start noticing patterns in the question. For example, if you ask about the population and the economy of Paris, the LLM recognizes that these two concepts are related—it has seen them linked before. So, it follows these patterns to figure out the best way to answer your question.
5. **No Big Boss Making Decisions** Now, you might think there's some kind of central brain inside the model making decisions, but that's not how it works. There's no boss calling the shots. Instead, the layers (teams) work together, and they all adjust what they're doing based on the relationships they've learned. They cooperate and share information, but there's no single decision-maker.
6. **Back and Forth Adjustments (Feedback)** If the teams start heading in the wrong direction, they adjust their approach. This is where **backpropagation** "feedback between teams" comes in during the training process. It's a fancy way of saying the model learns from its mistakes—if something isn't right, it tweaks

how the teams handle things so they get better next time.

7. **More Complicated Questions Need More Teams** (Layers) For something like *“Paris is the capital of France?”*, it’s a simple project, so fewer teams are involved because the model already knows the answer well. But when you ask a more complex question, like *“How does Paris’s population affect its economy?”*, the model brings in more teams (layers) and uses more parameters to connect all the pieces. The harder the question, the more layers work together to figure it out.
8. **Training Data is the Rulebook** All of this—how the layers work together, how the parameters are set up—comes from the training data the LLM has been fed. Think of the training data as a big rulebook. The more accurate the rulebook, the better the model’s understanding. If the model has seen enough reliable data, it knows that “Paris” and “France” are linked and understands more complex things like how population affects the economy.
9. **Delivering the Final Answer** After all the teams have done their work, the LLM puts together an answer. For a simple question, this happens quickly with fewer steps. For complex questions, it takes more steps and more teamwork between the layers to figure out the right connections and deliver a thoughtful response.

This way of thinking about LLMs makes the whole process easier to grasp: each prompt is like a project, and the model’s layers are teams working together, adjusting and learning as they go, without any one boss calling all the shots.

The Role of Training Data: The LLM’s Rulebook

Now, let’s talk about the **training data in more details**, which is the foundation of everything the LLM knows. Think of the training data as the LLM’s **rulebook**—it’s what taught the LLM how words relate to each other and how to handle new projects (prompts).

In short, the **training data** was like the LLM’s teacher. Now that it’s learned everything from that teacher, it can handle new questions based on the **knowledge it has stored** from its training.

Learning from the Rulebook

Before the LLM was deployed, it was trained on massive amounts of text (books, articles, websites). During this training, the model read the data and learned **patterns** and **relationships** between words. This is where it built its **rulebook**.

For example, it learned from this data that **“Paris”** is linked to **“France”** and **“capital”**. The more often it saw these words together, the stronger it understood the connection between them. This process of training is how the LLM filled its **memory** with information that it can use later.

Using the Rulebook Without Going Back to It

Here's an important point: once the LLM is trained, it doesn't need to go back and look at the **training data** every time you ask it a question. It's already **learned** everything it needs from that data—just like how a person doesn't need to re-read a book every time they answer a question about it. Instead, the LLM relies on the **patterns and relationships** it has already memorized.

So, when you ask, *"What's the population of Paris, and how does it affect the economy?"*, the LLM isn't looking up articles or books. Instead, it's using the **knowledge** it gathered from its training data—the **rulebook** it already memorized—to respond to you.

Why the LLM Doesn't Access Training Data in Real-Time

The LLM doesn't need to revisit the training data because:

- **It already learned the material:** The training data taught the LLM how words and concepts fit together. Now, it just uses its memory to apply that knowledge.
- **It's using what's stored in its memory:** Just like how you remember facts after reading a book, the LLM stores the relationships and patterns it learned and applies them in real-time without needing to reference the original data.

Putting It All Together:

- **Prompt = Project:** You give the LLM a project when you ask a question.
- **Teams = Layers:** The layers (teams) work on different parts of the project.
- **Parameters = Relationships:** Parameters are the connections between words, learned from past training.
- **Patterns=keywords:** The teams spot patterns based on what they've seen before.
- **No Central Boss:** No one team is in charge; they all work together without a central decision-maker.
- **Training Data = Rulebook:** The LLM learns from the training data (its rulebook), which helps it provide accurate answers.
- **Complexity:** Simple projects need fewer teams, while complex ones involve more layers and parameters.
- **Training Data = Rulebook:** The LLM learned from the training data, just like how we learn from books, and it doesn't need to keep going back to that data once it's learned the material.

Understanding Parameters and Tokens in an LLM: A Simple Breakdown

We've already established that in an LLM (like GPT), **parameters** are the relationships between words, and **tokens** "they call it tokens" are the words themselves. Now, let's dig a little deeper into how these tokens and parameters work together to help the model understand and respond to prompts.

Tokens and Relationships

First, think of **tokens** as individual **words** or parts of words. Each word has certain **relationships** with other words, and these relationships are built during the model's training. The more training data the model has, the more relationships (or **parameters**) it can establish between different words.

When you give the LLM a **prompt** (we can think of this as a "project"), the model analyzes the relationships between the words in that prompt. It **evaluates** how strongly the words are connected and decides which connections are most relevant. These relationships are measured by **vectors**, which are like numerical scores that show how related two words are. These vectors aren't static—they change dynamically based on the **context** of the prompt.

Visualizing the Relationships

Imagine each word is inside a **circle**, and from that circle, you have multiple **lines** connecting it to other words. Each line represents a potential relationship between the word in the circle and another word. Some of these lines might be **thick and strong** because those words are closely related, while others might be **thin and weak** if the relationship is less relevant.

Note: visit the site for visualize words relationship [Semantically related words for "dubai NOUN" \(nlpl.eu\)](https://nlpl.eu/semantically-related-words-for-dubai-noun/)

For example, if you take the word "**bank**" and place it in a circle, you might see lines connecting it to words like "**money**," "**loan**," "**river**," and "**finance**." The strength of these connections depends on the **context**. If you're talking about banking, the connection between "**bank**" and "**money**" will be strong, but if the context is about rivers, the connection between "**bank**" and "**river**" will become stronger instead.

Changing Contexts and Relationships

The beauty of an LLM is that these relationships **adapt** depending on the context of your prompt. Let's say you move from talking about **healthcare** to **finance** and then to **geography**:

- In **healthcare**, the word "**apple**" might be linked closely to words like "**nutrition**" and "**diet**."
- In **finance**, the word "**bank**" might be linked closely to words like "**loan**" and "**interest**."
- In **geography**, "**bank**" might shift its connection to "**river**" and "**water**."

This means that for each new context or domain, the LLM dynamically adjusts the **vectors** (relationships between words) to match the meaning that's most relevant.

The Role of Training Data in Building These Relationships

So, how does the LLM know to adjust these connections? This is where **training data** comes in. During training, the model is exposed to a huge amount of text across different topics and domains. It learns the **patterns** in language and builds initial vectors (or relationships) between words.

At the start, these vectors are **generic** and might even be random. But as the model processes more data, it refines these relationships. For instance, the more the model reads about **banks** and **finance**, the stronger the connection between "**bank**" and "**loan**" becomes.

Once trained, the model doesn't need to go back to the original data every time it gets a new prompt. Instead, it uses what it already learned—just like how you don't need to look up a fact in a book once you've memorized it. The LLM adjusts the relationships between words on the fly, based on the context of the question or prompt.

In Summary

- **Tokens** are words, and the **relationships** between those words are called **parameters**.
- These relationships are represented by **vectors**, which measure how closely two words are related.
- Depending on the **context** of the prompt, the model will adjust the strength of these relationships.
- The **training data** helps the model learn these relationships, so it can respond to different prompts in various domains (like healthcare, finance, or geography) without needing to revisit the original data.

In short, every word is connected to many other words, and the strength of those connections shifts depending on the project or prompt you're working on. It's like having a **web of words**, and the model knows how to highlight the right connections depending on what you ask.

Going a Bit Deeper: Vector Representation

If you're curious about how these relationships are actually represented in the LLM, here's a bit more detail. In models like GPT, every word (or token) is represented by a **vector**—a long list of numbers. These vectors are usually **1,024 dimensions long**, meaning each word is associated with 1,024 different numbers that capture its meaning.

For example:

- **"apple"** might have a vector like **[0.45, -0.32, 0.78, ..., -0.11]**.
- **"fruit"** could have a vector like **[0.42, -0.30, 0.80, ..., -0.10]**. Since **"apple"** and **"fruit"** are related, their vectors will be similar. In this case, the difference between their vectors is small, which shows that the LLM recognizes them as closely related.

These numbers typically range between **-1 and 1** and reflect the word's relationship with other words in the model. Words with similar meanings will have vectors that are closer together in this **high-dimensional space**, while unrelated words will have vectors that are farther apart. The model uses these vectors to understand and adjust the relationships between words based on the context of your prompt.

So, while we visualize these connections as **lines between words in circles**, under the hood, the model is using complex **mathematical relationships** between these **1,024-dimensional vectors** to decide how words are related. This allows the model to adjust dynamically and provide relevant answers, no matter the domain or topic.