Unit 1 - What are LLMs?

UNIT 1 - WHAT ARE LLMS?

Unit 1
What are LLMs?

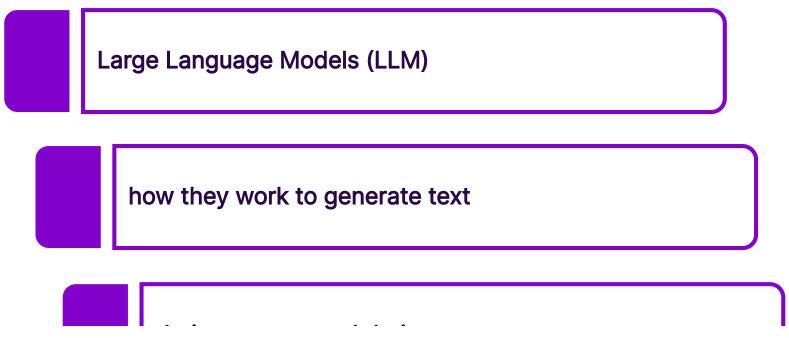
1.1 Unit Introduction

Welcome to the first unit of the course Mastering LLMs.

In the previous course, you learned about AI: what it is and how it learns. You have also started to learn about generative AI (GenAI).

Over the next few units, you will learn how **GenAI for text generation** works.

You will focus on:



their structure and their components

Understanding their underlying mechanism will allow you to know which parameters you can tweak to modify their behavior.

Time to start!

Continue to 1.2: What are LLMs?

1.2 What are LLMs?

Large Language Models (LLMs) are what's behind the scenes of GenAI, helping it understand and produce text content that sounds natural.

If you think of the text generated by GenAI as a delicious Michelin-star meal, the LLMs are like Matteo the Master Chef, using his skills and experience to create each dish.

LLMs are AI models designed to **understand and create human language**. They are trained using **text data**, allowing them to learn from a variety of sources and generate text content based on the input they receive. They can respond in conversations, answer questions, and even write code. Examples include well-known models like **GPT**.

Matteo the Master Chef has learned about cuisines from all over the world. With years of practice, he mixes techniques and flavors from many cultures to create his star dishes.

Continue to 1.2.1: How do LLMs work?

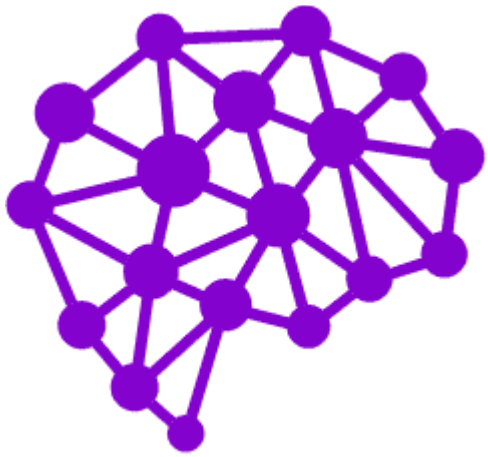
1.2.1 How do LLMs work?

LLMs use a combination of neural networks and machine learning to understand and create new text.

Click each one to learn more.

NEURAL NETWORKS | **MACHINE LEARNING**

Neural networks act as the LLM's **brain**. They **learn from data provided during the training phase and build connections** to understand patterns, language rules, and context in the text.



NEURAL NETWORKS

MACHINE LEARNING

Machine learning then helps the LLM get better in **predicting the next word based on the context of the words that come before**.

Imagine Matteo the Master Chef learning from many different recipes, understanding their flavors and techniques. As he cooks, he chooses the ingredients based on what's already in the pot.

Pro Tip

LLMs are referred to as **foundation models** because they are **trained on vast amounts of data**, allowing them to be used in different contexts and applications.

Let's explore this in more detail in the next lessons.

Continue to 1.3: What are neural networks?

1.3 What are neural networks?

To understand how LLMs work, let's begin with neural networks, which form the foundation of their architecture.

Neural networks are **artificial intelligence models inspired by the structure and function of the human brain**. They consist of a series of **algorithms** designed to recognize patterns and relationships in data, similar to how the brain works. They are made of interconnected **nodes** (like neurons) that process information. At each node, **information is received, processed using a mathematical algorithm, and passed to the next node in the network** to help generate the final result.

Click each + to learn more.

Architecture _

The **architecture** of a neural network is the **way it is built** and **how its parts are connected**.

Algorithms _

An **algorithm** is a **set of instructions or rules**.

Node _

A **node** is a fundamental processing unit in a neural network that **receives input**, **applies a mathematical operation** to it, and **produces an output**.

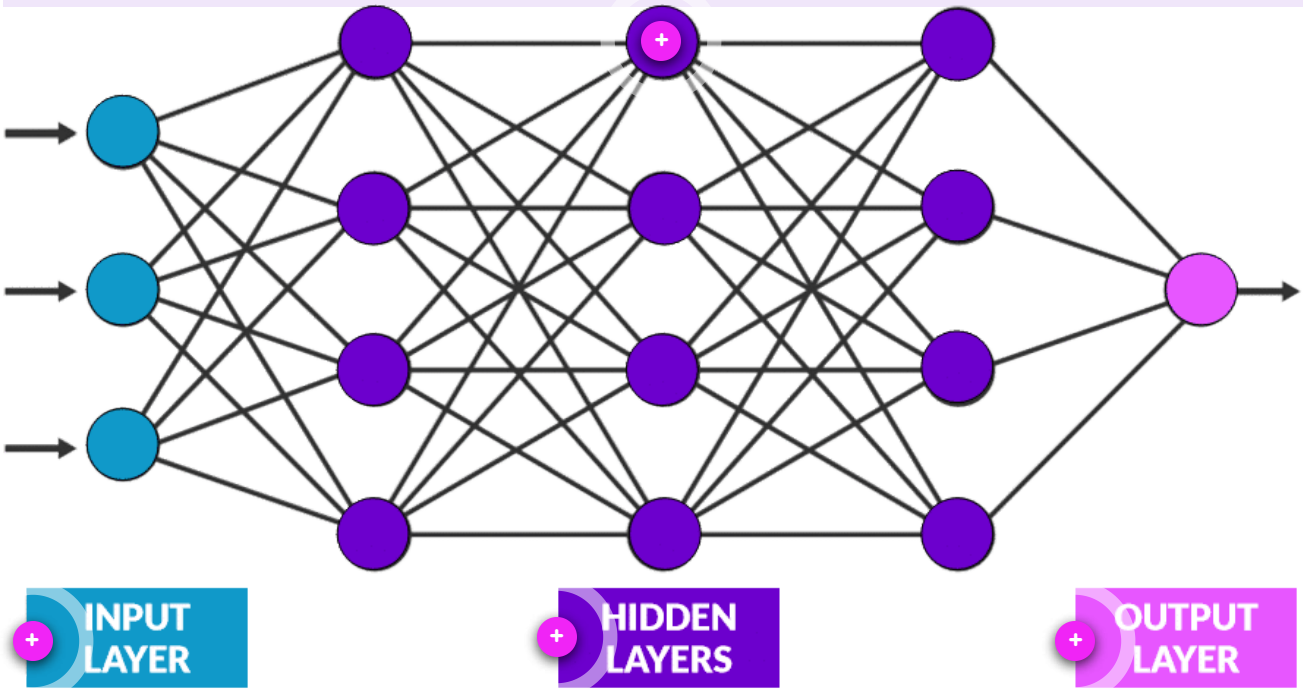
Matteo's kitchen is the neural network where each node is a different cooking station. Each station processes ingredients and contributes to the final dish.

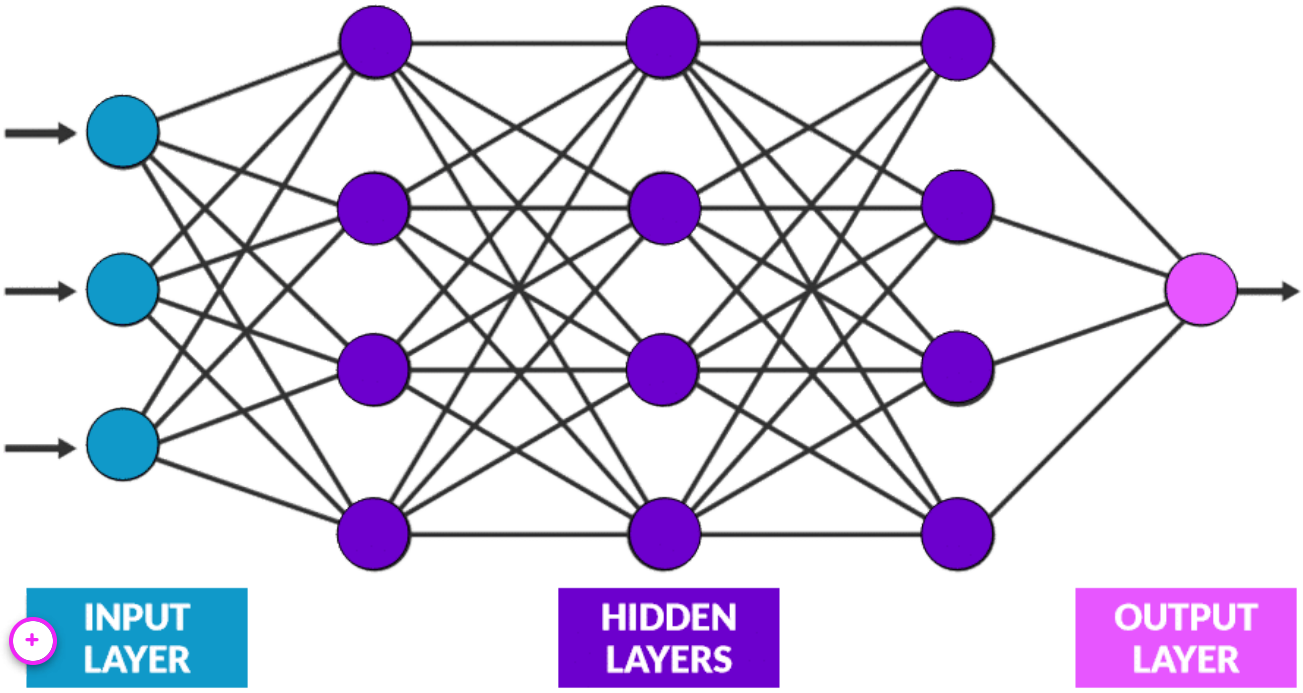
Continue to 1.3.1: What is their structure?

1.3.1 What is their structure?

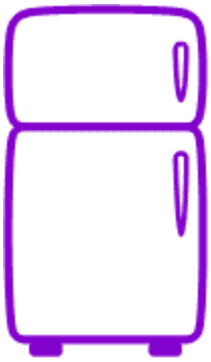
Let's look at the structure of neural networks in more detail. Neural networks have three main layers.

Click each + to learn more.



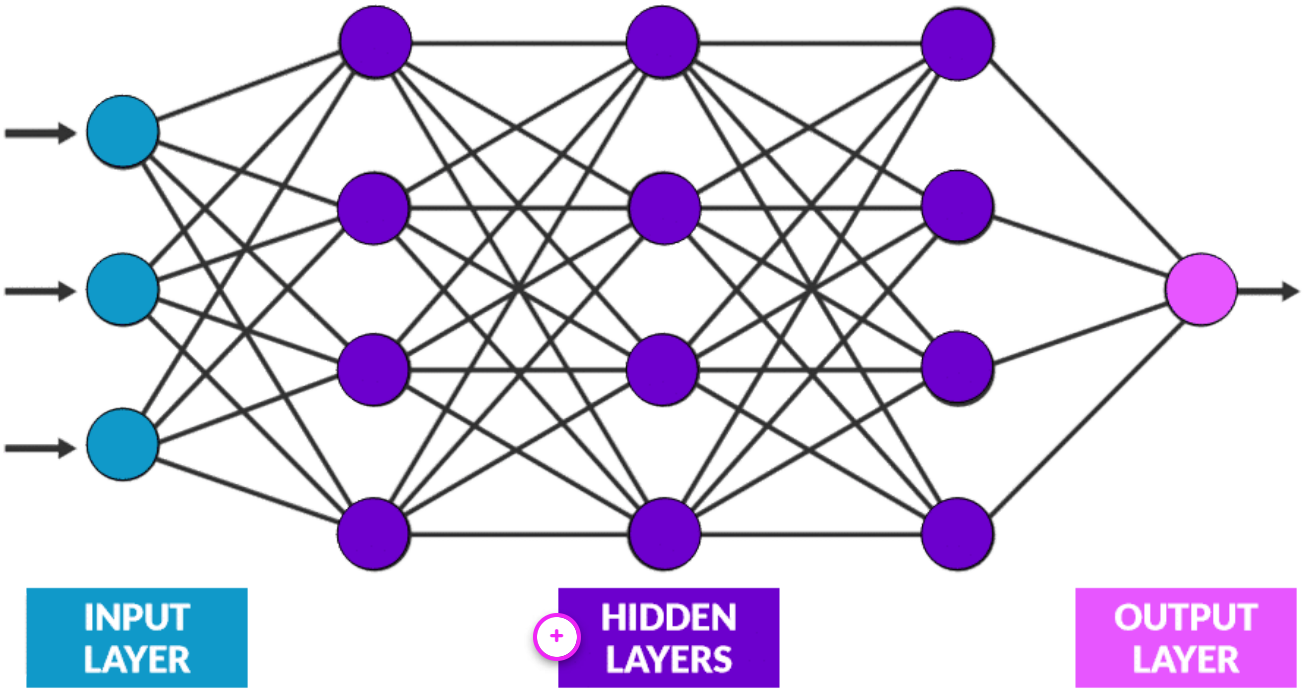


Input layer



The input layer is where the data first enters. The input nodes **process the data to prepare it for the next layers**.

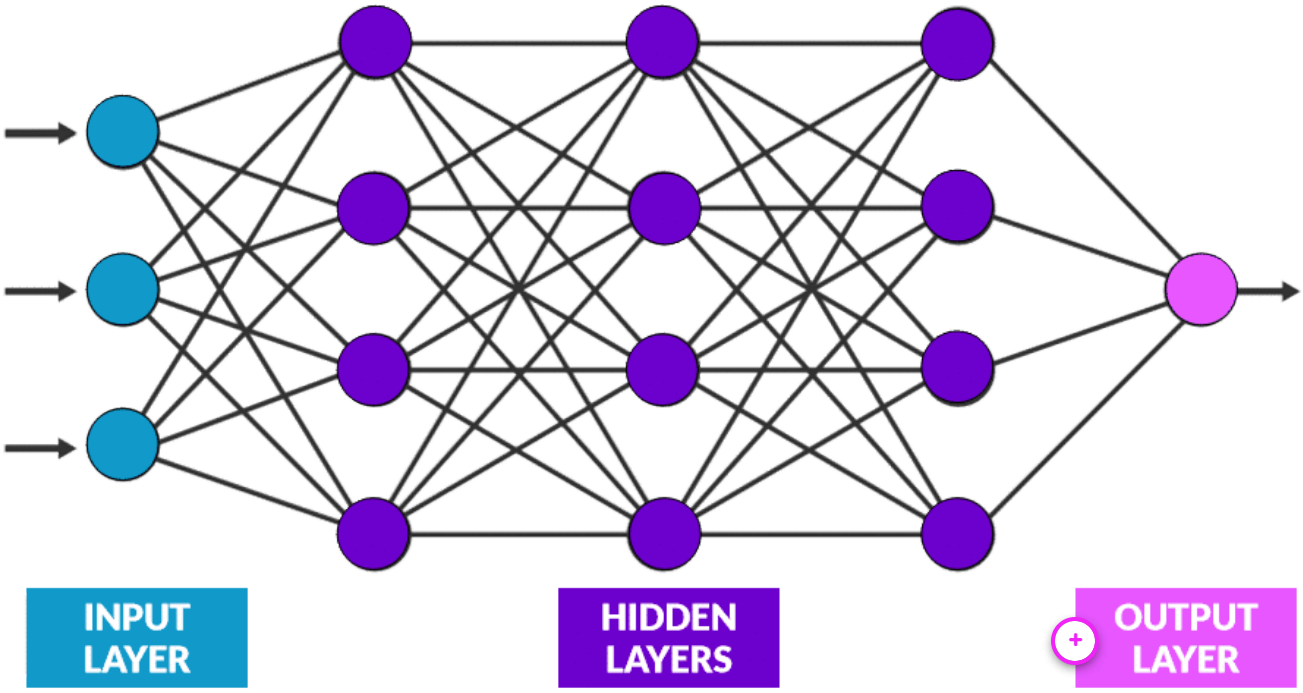
It is like the fridge, where all the fresh ingredients are kept and ready for use. The ingredients are selected and sent to different cooking stations for processing.



Hidden layers

The hidden layers receive the data from the previous layer, process it, and pass it along to the next layer. When processing the data, the hidden layers perform **mathematical calculations** that adjust and combine the data, helping the network recognize patterns, filter out noise, and refine details.

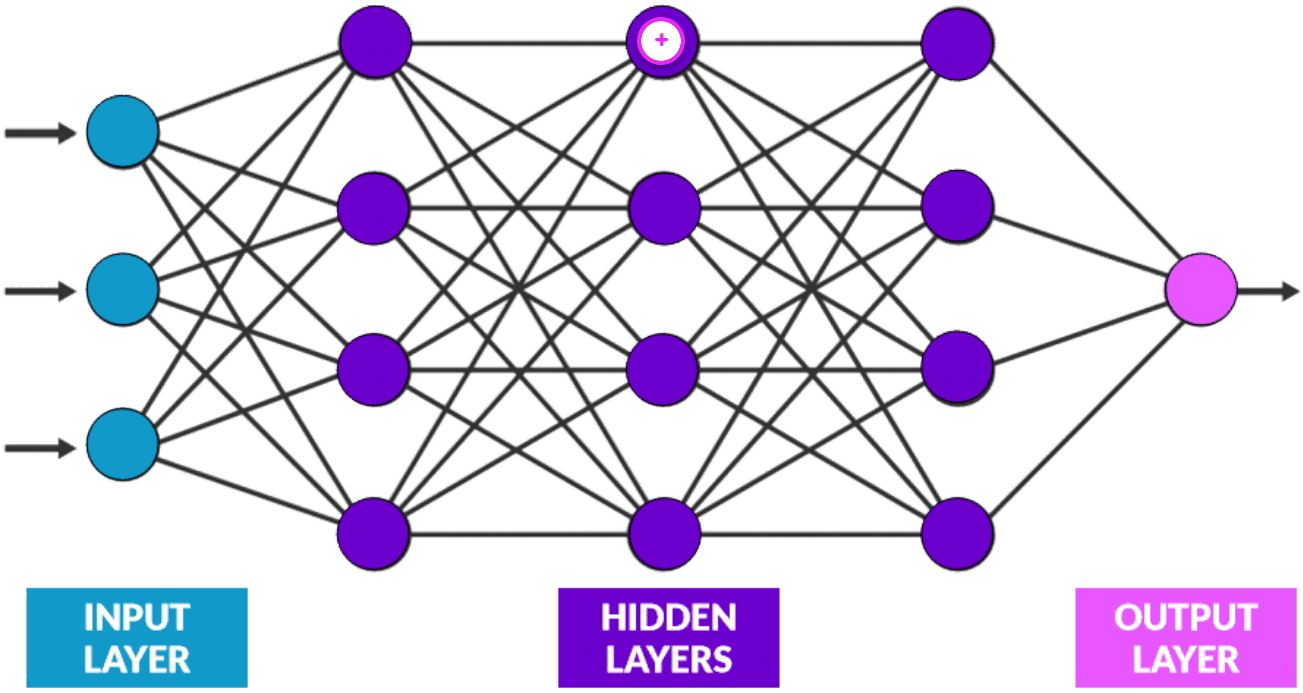
These are the different cooking stations where the ingredients are chopped, mixed, and prepared. Each station adds its own touch or transformation to the ingredients before they move on to the next stage.



Output layer

The output layer produces the **final result** by taking all the processed information from the previous layers and using it to generate an answer.

This is the plating station where all the prepared ingredients come together to create the final dish, ready to serve.



Node

This is a node! It takes the data from the nodes in the previous layer, makes some calculations and passes the results to the nodes in the next layer.

Continue to 1.3.2: How do they work?

1.3.2 How do they work?

Neural networks handle various data types (text, images, sounds, etc) turning them into numbers and using mathematical operations to find patterns and make predictions.

Input data moves through **layers** to create an output. Each layer has nodes that perform **calculations** on the received data. They assign a **weight** (a number) to show its importance, apply a **mathematical function**, and then pass the result to the next layer.

Pro Tip

During **training**, the model defines the weights. The neural network adjusts them to improve the model's accuracy. The model learns from errors and updates the weights to make better predictions.

When making a **prediction**, the model uses the weights defined during the training phase to make the calculations and obtain the final result.

In Matteo's kitchen, each station receives ingredients, adjusts them based on the taste of the customer, refines them, and then passes them along to the next station for further preparation.

Continue to 1.3.3: What is a transformer?

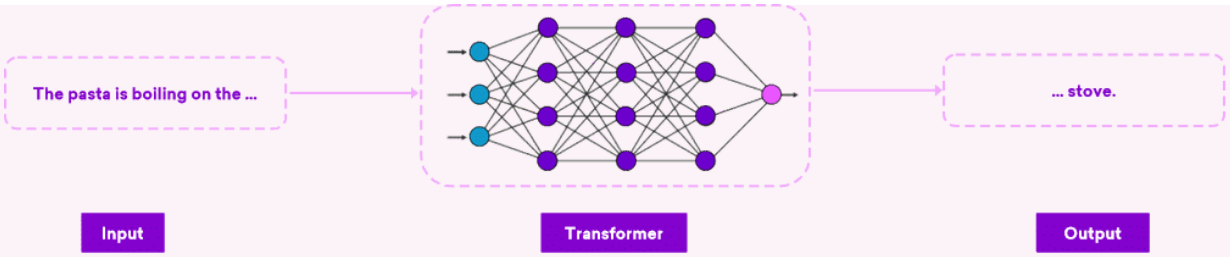
1.3.3 What is a transformer?

A transformer is a type of neural network designed to produce text.

It **understands context and meaning by analyzing relationships between words and phrases**. Think of a transformer as a box where text is input, processed, and transformed into a coherent output.

It processes text by breaking it into smaller parts called **tokens** and passing them through multiple layers. Each layer refines the data, identifying patterns and connections between words to understand their meaning within the sentence or text.

For example, if the input is **“The pasta is boiling on the...”**, the transformer uses a technique called **self-attention** to examine the relationship between words like **“boiling”** and **“on”**, recognizing that something should follow, like a cooking surface. With this context, it predicts **“stove”**, completing the phrase. This process of **analyzing words in context** helps the transformer create coherent text.



Input

Transformer

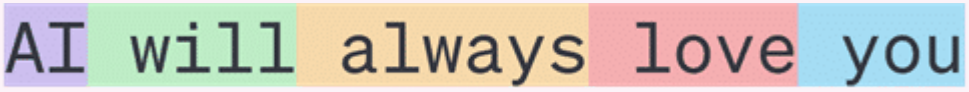
Output

Continue to 1.4: How do tokens work?

1.4 How do tokens work?

Tokens are the basic units of text that language models process.

They can be words, parts of words, or even characters, depending on how the text is broken down. They are a manageable unit that a model can work with. Each token below is represented by a color:



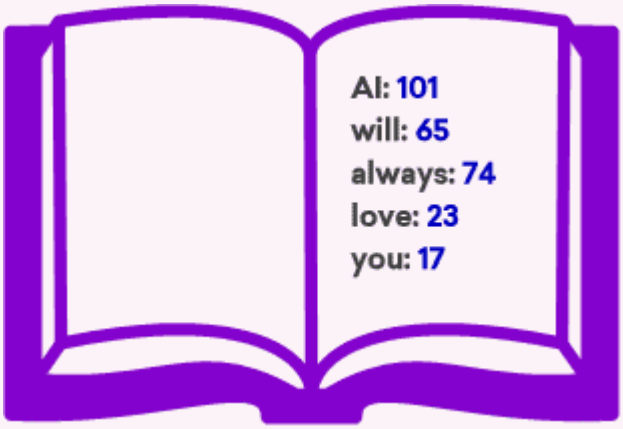
In Matteo's kitchen, tokens are like the individual ingredients that make up the dishes.

Tokenization is the process of **breaking raw text into tokens**. It can be done at different levels depending on the application:

- **Word-level:** splits text into individual words.
- **Subword-level:** breaks down words into smaller units.
- **Character-level:** divides the text into individual characters.

Continue to 1.4.1: Token ID

1.4.1 Token ID



Okay, you have broken down your text into tokens, what next? Each token is assigned a **token ID**, that is a **unique numerical identifier for each token**.

The token ID is assigned based on the model's **vocabulary**. The vocabulary is a **list of all possible tokens that have been generated during the training step**. Each token has a unique number (the token ID) so that the model can **process text as numbers**.

When the model processes a text, it splits the text into these tokens from its vocabulary and assigns the relative token ID.

Click the + to learn more.

What about tokens that are not in the vocabulary?

When a token is not present in a language model's vocabulary, it is considered an **out-of-vocabulary (OOV)** word.

To handle OOV words, LLMs use an approach called **open vocabulary**:

1. **Subword tokenization:** break down unknown words into smaller subword units that are in the vocabulary.
2. **Special tokens:** use special tokens like **<UNK>** (unknown) to represent the unknown parts.
3. **Character-level fallback:** break down completely unknown words into individual characters.
4. **Contextual understanding:** Even if a specific word is unknown, the model can often figure out its meaning from the surrounding context.

This approach allows language models to handle a much larger effective vocabulary than their actual token vocabulary size, and to make reasonable guesses about the meaning and usage of previously unseen words.

For example, if the word **unfathomable** is not in the vocabulary, a subword tokenizer might break it down into **un**, **fathom**, and **able**, all of which could be in the vocabulary. If **fathom** is also unknown, it might be further broken down or replaced with **<UNK>**.

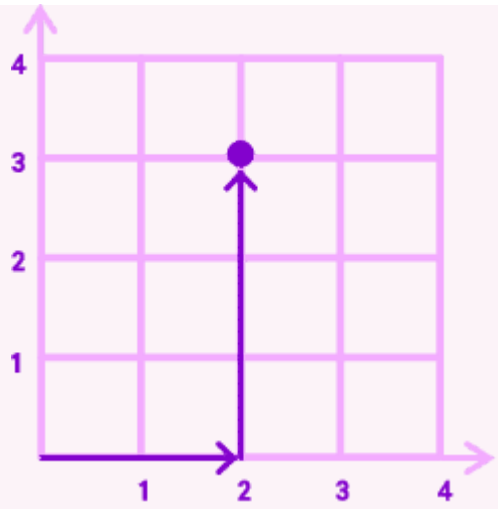
Token IDs are like pantry labels for ingredients. Matteo has a list of all the ingredients he might need, each with a unique number. For example, salt is ID 1, pepper is ID 2, and so on. When following a

recipe, he uses these IDs to quickly find what he needs, making cooking faster and easier.

Continue to 1.4.2: Vectors and embeddings

1.4.2 Vectors and embeddings

The token ID is converted into a vector, which is a list of numbers, for example:
[`0.8, -0.5, 0.3, 0.1`].



In math, a vector is **a set of numbers that defines the coordinates of a position in a space or a graph**. For example, in 2D, a vector like `[ 2, 3 ]` tells you to move 2 steps to the right and 3 steps up. In this case the vector has two dimensions (two numbers) and it represents the position of a point in a space.

The token ID is converted into a **vector**, which is a list of numbers, like for example **[0.8, -0.5, 0.3, 0.1]**. The process of mapping a token ID (a unique number for each token) to its vector is called embedding, and the resulting vectors are known as embeddings.

> For embeddings, the vectors have at least **512 dimensions**, which means they represent **positions in a much larger, high-dimensional space**.
>
> **Click the + to learn more.**

Why 512 dimensions? _

512 dimensions have been chosen as it's a good compromise between having enough info and

being able to perform calculations without needing too much computational power.

Each dimension captures a specific feature of the word, such as its meaning, context, or relationships to other words. You will see an example in the next section.

Think of embeddings as a pantry with hundreds of shelves, each representing a different characteristic of an ingredient. For example, one shelf might represent sweetness and another spiciness.

Continue to 1.4.3: How do embeddings work?

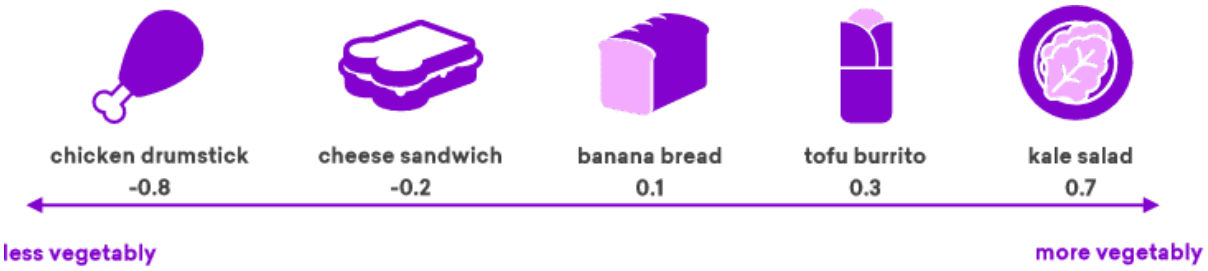
1.4.3 How do embeddings work?

Let's have a deeper look at how embeddings are used to represent the meaning of the tokens.

Work through the different points to learn more.

Step 1

1D Space

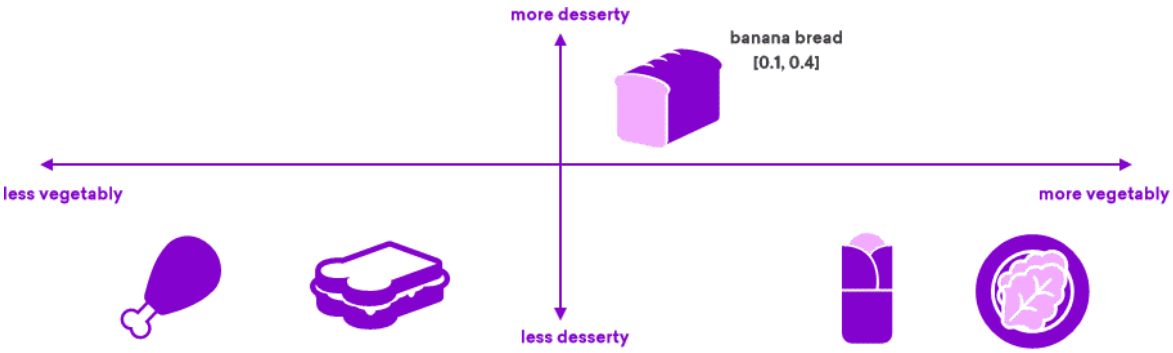


The position of embeddings in the space reflects the **meaning of the tokens**. Each dimension represents a **specific feature** of the token. How does this work?

To make things easier, let's start with a **1D** example. You want to represent the following food according to its **vegetabliness**. The value in the vector represent the position of the food from the least vegetable based to the most vegetable based.

Step 2

2D Space

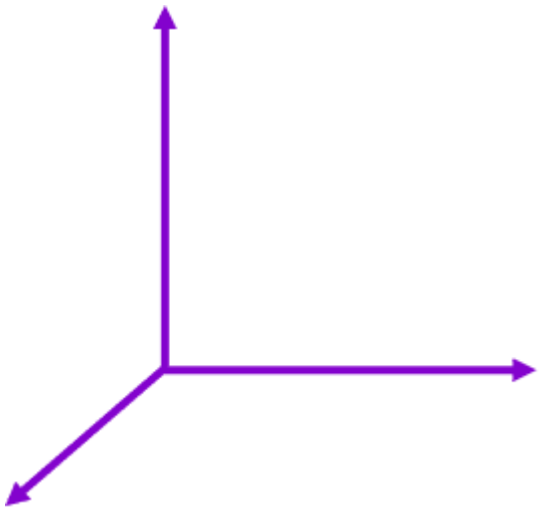


Apart from the **vegetabliness**, you can also add another dimension, like **dessertiness** for example, and place the food in this **2D space** according to their different characteristics.

> In this case **0.1** represents the **vegetabliness** and **0.4** the **dessertiness** and they are reflected by the position of the food in the 2D space.

Step 3

3D Space and more



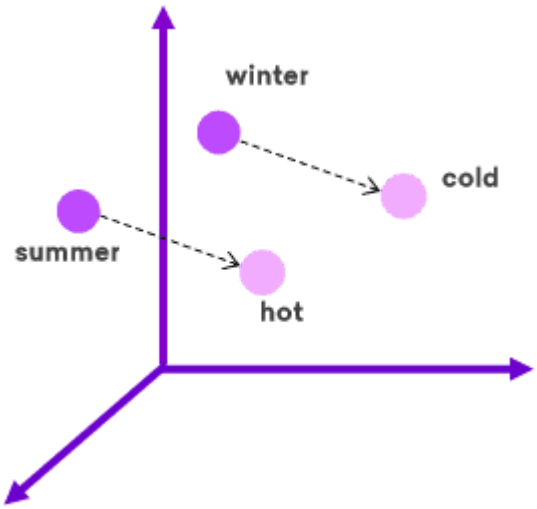
If you add another dimension like **spiciness**, you would have a **3D space** in which your food is placed according to its characteristics.

You can continue adding even more dimensions, like **crunchiness**, **saltiness**, or **texturiness**, and the space would keep expanding.

Note: A space with more than three dimensions can't be directly visualized because humans are limited to perceiving only three dimensions. However, you can think of it as a space where each point has more than three coordinates, representing different features or characteristics.

Step 4

Embeddings



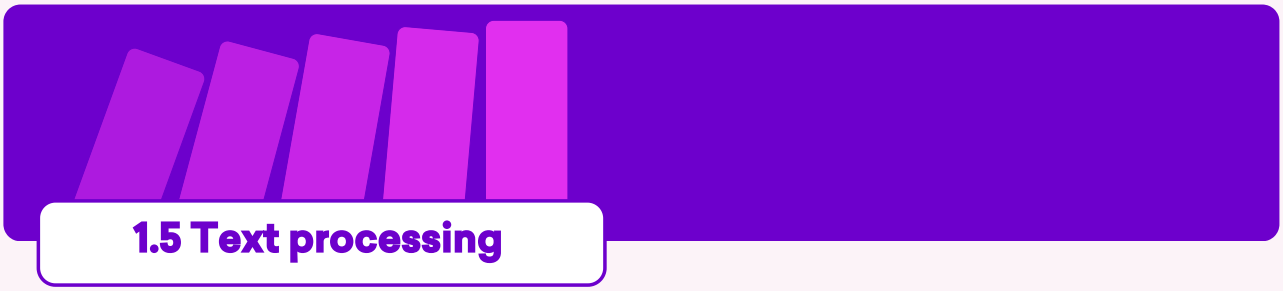
The same happens with the embeddings, **each dimension represents a characteristic of the token** like:

- Meaning
- Grammatical properties
- Associations
- Synonymy
- and many more

Knowing the position of the tokens in the space helps the model **understand their relationships with other words**. Tokens that have similar meanings are placed closer together, while those with different meanings are farther apart.

The distance between tokens helps the model understand **how related or different they are**.

Continue to 1.5: How do LLMs process data?



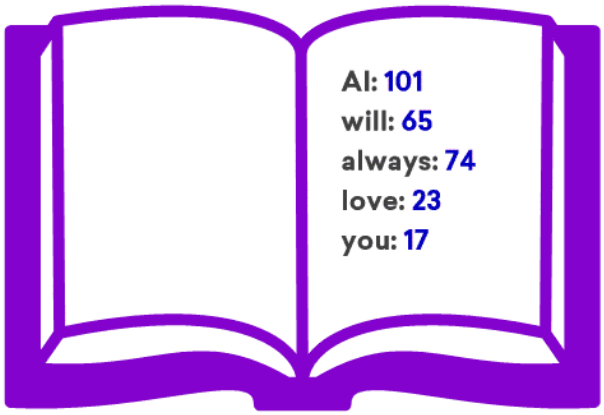
1.5 Text processing

Let's recap the process up to now before we have a look at the next step in which LLMs understand and generate text (Processing).

Click each card to review.

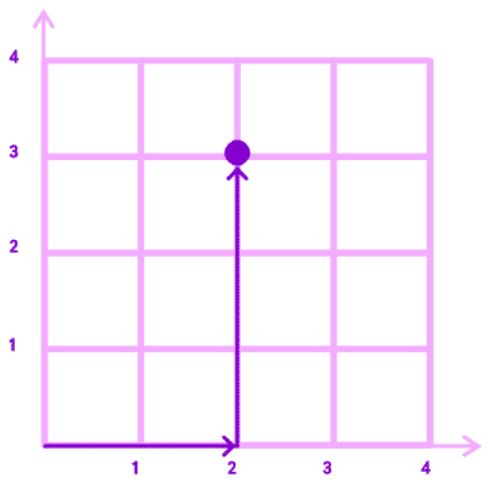
Tokens

Text is broken down into smaller pieces called **tokens**, which can be words, parts of words, or single characters that help the model understand the meaning and structure of the text.



Token ID

Each token is given a unique number, called a **token ID**, to help the model recognize and process it.



Embeddings

Then a list of numbers is generated based on the tokens and their positions in the text. These numbers are called vectors or embeddings, where each number in the list represents a different feature or aspect of the token's meaning. The position of the vector in the space shows how it relates to other tokens.

LLMs process embeddings. At each layer, the values in the vectors are used for calculations like identifying patterns, adjusting weights, and capturing relationships between words to refine the model's understanding.

The **self-attention mechanism** helps the model figure out **which words in a sentence are more important from the context point of view by giving them a weight** (a number). This number is then added to the calculations done by the layers, so the model can focus more on the important words.

The pasta is boiling on the ...

LLM

... stove.

Matteo decides which ingredients to add based on their flavor impact. For each dish, he evaluates and ranks the ingredients, focusing on the ones that will enhance the recipe the most.

Continue to 1.5.1: Training LLMs

1.5.1 Training LLMs

As you've learned in the AI Fundamentals course, the first step with any machine learning model is training, and LLMs also need to be trained before they can function properly.

There are two different training stages, each with distinct goals and methodologies.

Click each one to learn more.

PRE-TRAINING | **FINE TUNING**

In pre-training, the model processes large amounts of **unstructured text using unsupervised learning**, predicting missing or next words to understand grammar, context, and word relationships.

PRE-TRAINING | **FINE TUNING**

Fine tuning trains a pre-trained LLM to do a specific job, like translating text or identifying sentiment. It uses a **labeled dataset with examples of inputs and their correct outputs**, teaching the model through supervised learning to handle similar tasks accurately.

In Matteo's kitchen, pre-training is like learning about all kinds of ingredients and how they work together, while fine-tuning is like focusing on specific recipes, such as mastering Italian cuisine.

Continue to 1.5.2: Text generation

1.5.2 Text generation

After the training phase, an LLM generates text by using the knowledge it learned from vast amounts of data.

When given an **input**, the model breaks it down into **tokens**, processes them through its neural network **layers**, and uses the relationships it learned during training to **predict** the most likely next tokens.

This prediction is based on the patterns, contexts, and associations between words it encountered during training.

The model then generates an **output**, word by word, refining its predictions by considering the **probability distribution of possible tokens**, ultimately producing coherent and contextually relevant text.

Click the + to learn more.

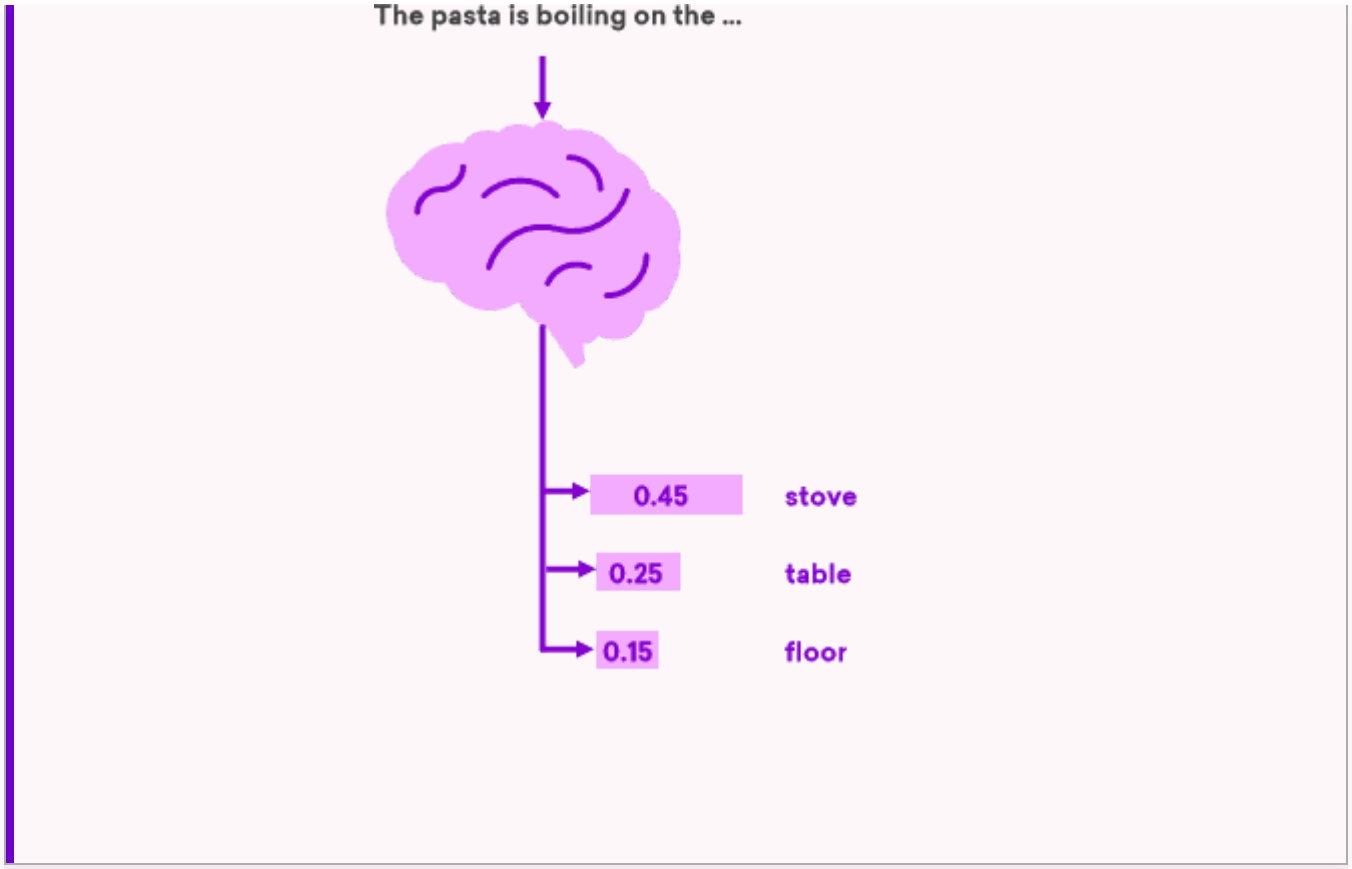
Token probability distribution

A parameter that plays a crucial role in this process in the **token probability distribution**.
It is a set of probabilities assigned to all possible tokens in the model's vocabulary, representing **how likely each token is to appear next in a sequence**. It is calculated by the LLM.

During training, this distribution helps the model adjust its weights by comparing predicted probabilities with actual data.

In predictions, it is used to **select the most likely next token**, enabling the model to generate coherent and contextually relevant text.

The token probability is like trying to predict the chances of a certain ingredient being picked for a recipe based on previous choices. The more often an ingredient has been used in similar dishes, the higher the chance it'll be picked again.



After training, Matteo is a skilled chef using learned recipes to choose the right ingredients to prepare his Michelin star meal.

Continue to the wrap up for this unit

1.6 Wrap up

1

Large Language Models (LLMs) are **neural networks trained on vast amounts of text data to understand and generate human-like language**.

2

LLMs, like GPT, use **transformers** to process language. Transformers **convert words into tokens and embeddings**, allowing the model to capture and understand the **meaning of each word based on its context in the sentence**.

3

When they receive an input text, LLMs process it by converting the words into tokens and embeddings, then **use probability to predict the most likely next word or token**, generating text that responds to the input provided.

Unit complete!

You've completed the first unit and discovered the secret of LLMs. Well done!

By now you should have an understanding of:

- What are LLMs and how they are structured
- How LLMs understand and generate text
- The key components of the text generation process

Great work! Let's keep this momentum going. In next unit you will discover how to interact with LLMs.

make | academy

Mark this task complete to continue to the next unit.