

The LLM Inc

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

Welcome to LLM Inc, a company with one mission: to understand language and respond to any question or statement.

Every time a user inputs a prompt, LLM Inc. springs into action, and each department plays a key role in analyzing, understanding, and responding to that prompt. At LLM Inc., we work as a team of specialized departments, each with specific responsibilities to process language smoothly and efficiently.

The departments details

Department	Role
1. Tokenization Department	Breaks down the input text into manageable parts, or tokens.
2. Embedding Department	Converts each token into a numerical representation, called a vector, capturing its meaning.
3. Attention Department	Focuses on understanding relationships between words, identifying keywords, and recognizing context.
4. Processing Department	Refines connections and clarifies relationships within the prompt using insights from the Attention Department.
5. Decision Department	Generates a final response that directly answers or addresses the user's input based on all prior insights.

Each department brings a unique skill set to the table, and by working together, they ensure every user gets an accurate and well-structured response.

1. Tokenization Department

Department	Role	Algorithms Used
Tokenization Department	Breaks down the input text into manageable parts, or tokens.	Sub-word Tokenization Algorithms (e.g., Byte Pair Encoding (BPE), WordPiece)

Overview of the Tokenization Department's Role and Algorithm:

- **Role:** The Tokenization Department's job is to take the user's input sentence and break it into smaller pieces called tokens. Tokens can be entire words, parts of words, or even punctuation marks, depending on how the sentence needs to be processed.
- **Algorithm Used:**
 - Sub-word Tokenization Algorithms like Byte Pair Encoding (BPE) or WordPiece are commonly used. These algorithms help handle complex language elements, such as rare words and multi-word terms, by breaking them down into more frequently occurring sub-words.
 - For instance, in Byte Pair Encoding (BPE), the algorithm starts by tokenizing common words as whole units but breaks less common or complex words into smaller pieces. For example, "unbelievable" might be tokenized as "un," "believ," and "able," ensuring the model can still understand it, even if "unbelievable" wasn't directly encountered in training.

Example with Algorithm in Action:

Using the example sentence, "What's the weather like today?":

1. The Tokenization Team applies the BPE algorithm to break the sentence into tokens: "What's," "the," "weather," "like," and "today."
2. These tokens are now in a form that can be easily processed by the rest of the departments in LLM Inc.

2. Embedding Department

Department	Role	Algorithms Used
Embedding Department	Converts each token into a numerical representation, called a vector , capturing its meaning.	Embedding Lookup Table and Matrix Multiplication

Overview of the Embedding Department's Role and Algorithm:

- **Role:** After the Tokenization Department has broken the input sentence into tokens, the Embedding Team's job is to convert each of these tokens into a **numerical vector** (a series of numbers). This vector, or **embedding**, captures the basic meaning of the word and makes it readable for the model's processing layers.
- **Algorithms Used:**

- **Embedding Lookup Table:** The embedding process starts with a large “dictionary” or lookup table. This table contains a unique vector for each token in the vocabulary, learned from the model’s training data.
- **Matrix Multiplication:** If the embedding isn’t directly available in the lookup table, the model may use matrix multiplication with learned weights to adjust the vector, refining the meaning based on the token’s use in different contexts.

Example with Algorithm in Action:

Using the example sentence, “**What’s the weather like today?**”:

1. The **Embedding Team** assigns each token a vector:”1024 numbers mainly”
 - “What’s” becomes [1.2, -0.5, 0.8, ...],
 - “weather” becomes [0.4, 1.1, -0.3, ...], and so on.
2. These vectors allow the model to capture the basic meaning of each token, setting up a foundation for the Attention Department to start analyzing relationships.

3. Attention Department

Department	Role	Algorithms Used
Attention Department	Analyzes relationships between tokens to determine which parts of the sentence are most important in context.	Self-Attention Mechanism (also known as Scaled Dot-Product Attention), Multi-Head Attention

Overview of the Attention Department’s Role and Algorithm:

- Role: The Attention Department focuses on identifying the relationships between words in a sentence to understand context. This department uses self-attention to allow each word (or token) to "pay attention" to other words in the sentence. In other words, it calculates which words are most relevant to each other, depending on the overall context.
- Algorithms Used:
 - Self-Attention Mechanism: This algorithm calculates attention scores for each pair of tokens. It does so by creating query (Q), key (K), and value (V) vectors for each token. The model uses these vectors to decide how much focus each word should receive.
 - Multi-Head Attention: This algorithm runs the self-attention mechanism multiple times (in parallel), each time focusing on different aspects of the sentence. Multi-head attention helps capture various relationships and meanings simultaneously.

Example with Algorithm in Action:

Using the example sentence, “What’s the weather like today?”:

1. The Attention Department applies the self-attention mechanism:
 - Team A (Relation Experts): They find that “weather” is a key word in the sentence, with “like” and “today” directly relating to it.
 - Team B (Context Specialists): They understand that “What’s” indicates a question, and that “today” is important for setting the timing (i.e., current weather).
2. Each token is now represented with a stronger sense of context and relationship, based on how much attention each word received.

4. Processing Department

Department	Role	Algorithms Used
Processing Department	Refines and adjusts token meanings using the relationships identified by the Attention Department.	Feedforward Neural Network, Activation Functions (e.g., ReLU, GELU)

Overview of the Processing Department’s Role and Algorithm:

- **Role:** The Processing Department takes the context-enriched tokens from the Attention Department and refines them. It fine-tunes each token’s meaning further, based on its adjusted context, ensuring that the relationships and meanings are as clear as possible.
- **Algorithms Used:**
 - **Feedforward Neural Network:** This is a small, simple neural network that processes each token’s vector independently to refine its meaning.
 - **Activation Functions:** The Processing Department uses activation functions like **ReLU (Rectified Linear Unit)** or **GELU (Gaussian Error Linear Unit)** to add non-linearity, allowing the model to capture complex relationships in the data.

Example of How It Works:

Using the example sentence, “**What’s the weather like today?**”:

1. **Input from the Attention Department:** The word “weather” already has context (it’s related to “like” and “today”), thanks to the Attention Department.
2. **Feedforward Neural Network:** The Processing Department takes this context-rich vector for “weather” and further adjusts it to reinforce the meaning—clarifying, for example, that “like” refers to describing “weather,” not showing preference.
3. **Refinement for Clarity:** After processing, “weather” has an even sharper meaning in context, setting up the token to contribute clearly to the model’s final response.

5. Decision Department's

Department	Role	Algorithms Used
Decision Department	Generates the final response based on all the refined meanings and relationships built up in previous departments.	Softmax Function, Probability Distribution

Overview of the Decision Department's Role and Algorithm:

- **Role:** The Decision Department takes the fully processed tokens from the previous layers and uses them to predict or generate the next word(s) in a sentence. This department works to create a coherent, accurate response that best answers the user's input.
- **Algorithms Used:**
 - **Softmax Function:** The Softmax function is used to calculate probabilities for each possible next token, based on the information refined by the previous departments.
 - **Probability Distribution:** The model uses these probabilities to select the most likely next word, ensuring the response makes sense in context.

The **Decision department** is the one responsible for generating the final reply for the user. This team takes all the refined information processed by the previous departments (Tokenization, Embedding, Attention, and Processing) and uses it to make the best possible choice for each word in the response.

Here's how it works:

1. **Input from Other Departments:** The Decision Department gathers all the context, relationships, and refined meanings developed by the earlier departments.
2. **Selecting the Response:** Using **probability calculations** (via the **softmax function**), it selects the most likely next word or phrase based on the processed input.
3. **Final Reply:** The Decision Department composes the full response by choosing each word step-by-step, ensuring it is contextually accurate and coherent.

The **Decision** Department wraps up all the processing into a clear, final response, delivering it back to the user

Example of How It Works:

Using the example sentence, **“What’s the weather like today?”**:

1. **Gathering Insights:** The Decision Department receives all the refined information from the Attention and Processing Departments, so each word now has a clear meaning and context.
2. **Applying Softmax:** The model applies the Softmax function to create a list of possible next tokens (words) along with probabilities, based on the prompt and refined token vectors.

3. **Selecting the Response:** The Decision Department selects the highest-probability tokens to generate a response. For example, it might choose the response **“The weather today is sunny and 25°C”** by selecting tokens with the highest probability in this context.

Decision Department’s Data Sources and Live Data Connection

The Decision Department relies on two main sources of information to generate accurate responses: **static training data** and **live data through Retrieval-Augmented Generation (RAG)**.

- **Fixed Knowledge vs. Real-Time Data:** The training data is static, meaning the model doesn’t inherently “know” real-time information like the current weather or recent news. To keep responses up-to-date, the Decision Department can access live data sources.
- **Accessing Live Data Sources:** Through **RAG** or predefined APIs, the Decision Department can access real-time sources like weather, news, or stock prices. When the model identifies a prompt that requires fresh data, it sends a query to these trusted sources.
- **Integrating the Response:** Once live data is retrieved (e.g., “The temperature is 25°C”), the Decision Department seamlessly integrates this information with its internally processed context, creating a coherent, contextually relevant response that includes the real-time update.

The Three Pillars of Data Storage in LLM Inc.

In addition to its specialized departments, **LLM Inc.** relies on three distinct types of storage to support its operations.

Each storage type serves a specific purpose, from the initial learning process to the real-time generation of responses. Together, these storage systems enable LLM Inc. to retain knowledge, generate dynamic responses, and continually adapt to new prompts without needing the original training data. Here's a quick overview of these storage types and their roles.

The three pillars

Storage Type	Description	Purpose
1. Source Training Data Storage	Contains the original data (like websites, books, articles) used to train the model.	Used temporarily during training to teach the model patterns, relationships, and meanings in language.
2. Model's Learned Weights Storage	Stores the extracted knowledge as learned weights after training is complete.	Acts as the model's "memory" of what it learned from training; these weights guide all future processing.
3. Layer-Specific Weights Storage	Holds weights specific to each layer (or department) in the model, guiding how tokens are processed.	Allows each department (like Attention or Processing) to dynamically generate Q, K, and V vectors, adapting to context in real-time.

Layer-Specific Weights Storage

Each department within **LLM Inc.** relies on the **Layer-Specific Weights Storage** to carry out its functions effectively. These weights, tailored to each layer, store the knowledge gained during training and allow each department to apply that knowledge dynamically. From Tokenization to Decision-making, every department uses these weights to adjust their processes based on the specific input.

This enables the teams to generate Q, K, and V vectors, refine token meanings, and ultimately produce a contextually accurate response—all without needing the original training data. By tapping into this layer-specific storage, LLM Inc. ensures that each prompt is processed with adaptability and precision

Understanding Q, K, and V Vectors

In LLM Inc., each word (or token) is represented by three essential vectors: **Query (Q)**, **Key (K)**, and **Value (V)**. Each of these vectors plays a distinct role in helping words connect meaningfully in a sentence.

- **Q (Query):** Represents what each word is looking for in other words. It acts as the “question” each token brings to the sentence, helping it identify potential connections with other words.
- **K (Key):** Represents what each word has to offer to others—its unique feature or identity. The K vector allows other words to recognize what this token can contribute to the sentence.
- **V (Value):** Contains the actual information each word provides if it becomes a focus. When a strong connection is made, the V vector supplies relevant details from one token to another.

Together, these vectors enable words to identify and share meaning, setting up the foundation for **dynamic connection testing** with the prompt.

Example 1

Vector Q of word “A” testing vectors K of multi words

Imagine **word A** is trying to understand its relationships with nearby words **B, C, D, E**, and **F**. Each of these words has its own **Q (Query)**, **K (Key)**, and **V (Value)** vectors, allowing them to interact meaningfully.

1. **Q Vector of Word A:** Word A begins by using its **Q vector**, which represents a “question” or focus, to seek out relevant connections. Its Q vector essentially asks, “Which of these words has something valuable to offer?”
2. **K Vectors of Words B, C, D, E, F:** Next, **word A** compares its Q vector with the **K vectors** of each nearby word (B, C, D, E, and F) to see if any have something particularly relevant to offer.
 - This step tests **what each word has to offer** based on its unique identity or characteristics, as captured in their K vectors.
3. **Finding a Match with Word C:** During this comparison, **word A** discovers that **word C’s K vector** provides a strong match with what it’s looking for. This indicates that word C has a meaningful or useful connection for word A.
4. **Vectors V of Words A and C:** Once this connection is identified, **word A** uses its **V vector** to interact with the **V vector of word C**. This allows word A to draw specific information from word C, strengthening their relationship and enriching word A’s understanding based on this context.

In this way, **word A’s Q vector** tests potential connections by evaluating the **K vectors** of nearby words. Once it finds a suitable match with word C, their **V vectors** connect, enabling the exchange of relevant information. This process is repeated for every word in the sentence, allowing dynamic and context-aware relationships to form based on the prompt and stored training knowledge.

Example 2

Connection Testing and prompt dual processes

1. Connection Testing process

When a prompt enters LLM Inc., each word actively begins to test connections with other words to determine meaningful relationships.

This process, known as **connection testing**, is guided by each token's **Query (Q)**, **Key (K)**, and **Value (V)** vectors. As each token's Q vector searches for relevant K vectors in other tokens, it draws on the **stored weight data** within each layer (or department).

These stored weights, learned from vast training data, act as a foundational knowledge base, helping the model recognize typical language patterns and relationships.

For instance, if the prompt involves a question about “weather,” stored weights guide the model to relate “weather” with other timing words like “today” or descriptive words like “like.”

2. Prompt process

However, the model doesn't rely solely on these stored patterns. The specific **context of the prompt** plays a vital role, dynamically shaping how each connection is adjusted.

When a Q-K match is found, the model fine-tunes the **connection weight** based on both the learned knowledge and the unique prompt context, creating a relationship that fits precisely.

Dual process

This dual process—drawing on stored weights and adapting to the prompt—enables LLM Inc. to interpret language flexibly and accurately, delivering responses that are both relevant and contextually aware.