Advanced Generative AI: Models and Architecture

Large Language Models

Quick Recap

- How do Generative AI features contribute to different domains like healthcare, finance, and others?
- What emerging trends in Generative AI do you foresee shaping the future?

Engage and Think

What if Large Language Models (LLMs) could generate completely original and human-like text in any language or programming language?

How would this revolutionize the way humans communicate and interact with technology?

Learning Objectives

By the end of this lesson, you will be able to:

- Develop an understanding of the core components and architecture of Large Language Models (LLMs)
- Experiment with analyzing the LLM in action and its training process, encompassing tokenization, embedding, neural network training, and fine-tuning
- Identify the functioning of LLMs, focusing on how they generate human-like text and respond to prompts
- Organize a comparison and contrast of various LLMs

Model Evolution

Introduction to NLP

Natural language processing (NLP) refers to the capability of machines to understand and generate human language. It plays a crucial role in bridging the gap between human communication and computer understanding.

- Machine translation
- Speech recognition
- Sentiment analysis

The need for NLP arises from its wide-ranging applications, including machine translation, speech recognition, sentiment analysis, and more.

Types of NLP Models

Rule-based systems

These were among the earliest forms of NLP, relying on predefined linguistic rules.

Statistical models

This category includes Markov chains and n-grams, which utilize statistical methods to predict language patterns.

Deep learning-based models

These advanced models, such as recurrent neural networks (RNNs) and transformers, leverage deep learning techniques to enhance language processing capabilities.

As NLP evolved, early models used probability to predict text. One of the first approaches was Markov chains, which modeled word sequences based on prior words.

Markov Chain: Probabilistic Language Modelling

Markov chains are probabilistic models where the prediction of the next word is based solely on the previous word, following the first-order Markov assumption.

I am → going

I am → banana

What is the most probable continuation of the phrase "I am"?

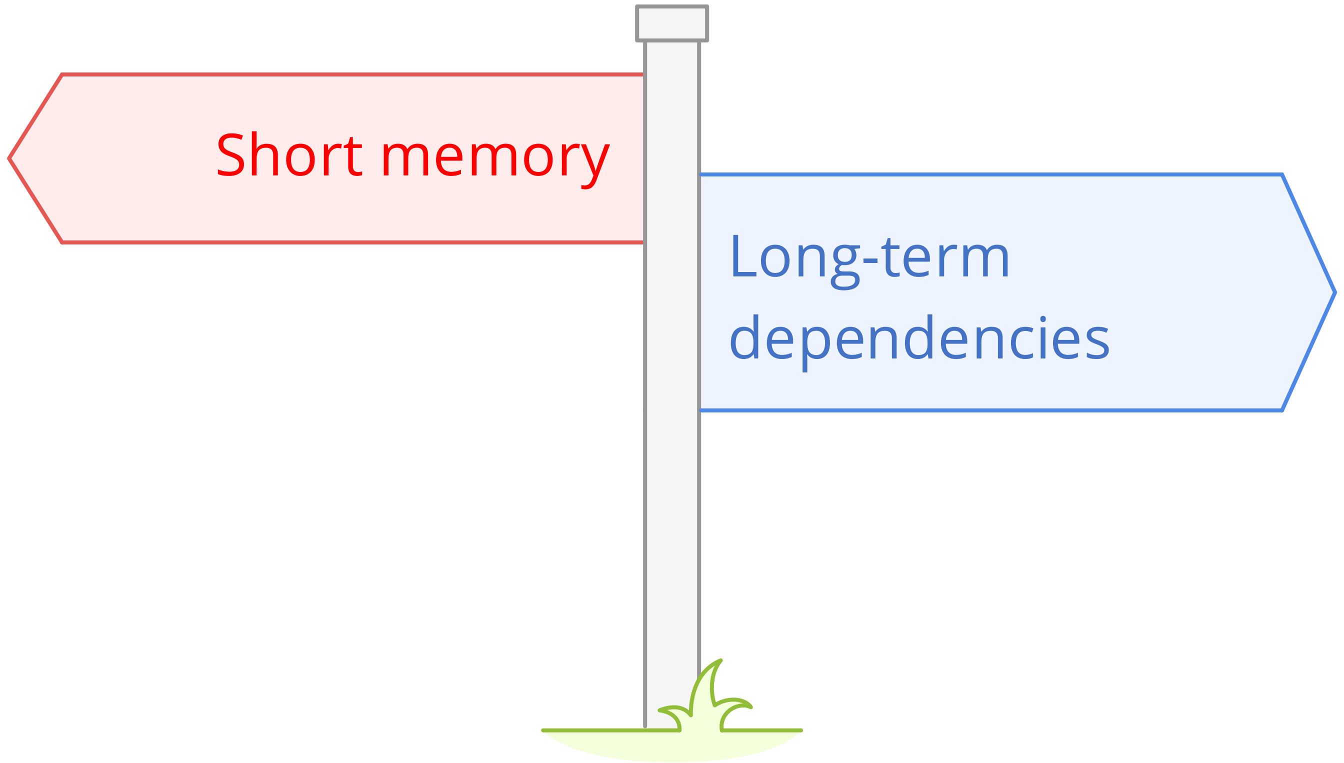
going

High probability of being the next word, indicating a common action

banana

Low probability, as it is an unusual continuation

Limitations of Markov Chain



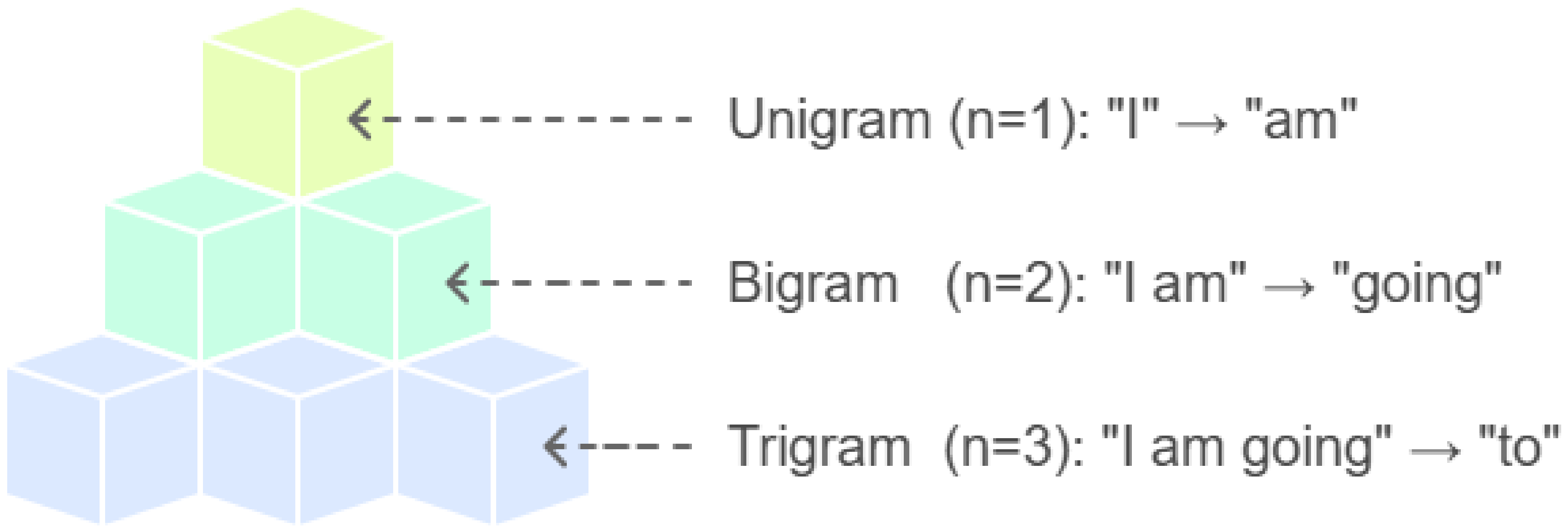
- **Short memory**: They only consider one previous word, limiting context.
- **Long-term dependencies**: They cannot capture dependencies across longer sentences.

While Markov chains improved word prediction, they considered only one previous word, limiting context. To address this, n-grams extended the approach by using multiple words.

Introduction to N-grams

N-grams are sequences of n-words used for predicting the next word in a sentence. They extend Markov chains by considering multiple words rather than just one.

Types of n-grams



Need of N-grams

Markov chains' reliance on a single previous word can lead to poor context understanding. N-grams improve accuracy by analyzing larger sequences of words. They have been utilized in early NLP applications such as:

Speech recognition

Converts spoken language into text

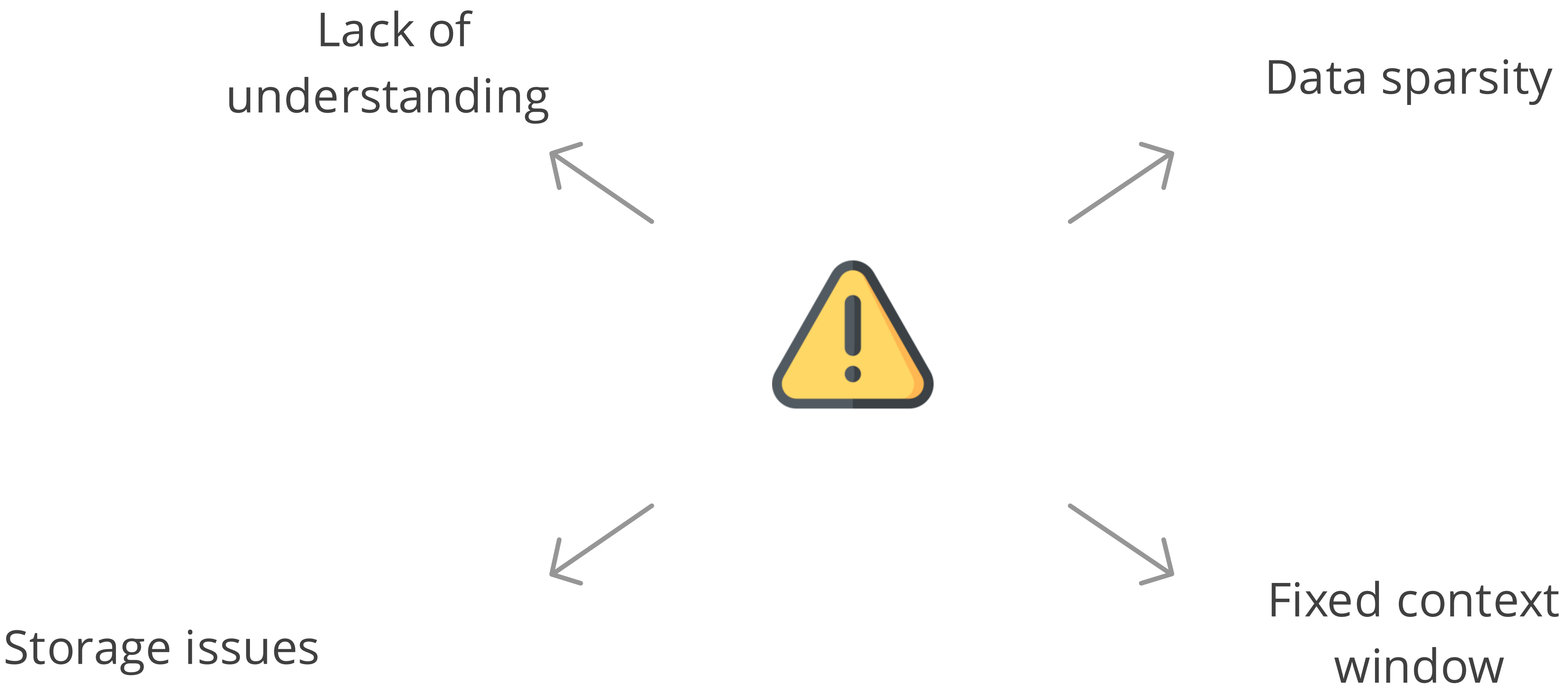
Auto correction

Automatically corrects spelling errors in text

Chatbot responses

Provides basic automated replies to users

Limitations of N-grams



To address these challenges, language models (LMs) were developed, which dynamically predict word sequences using advanced probabilistic techniques.

Language Models

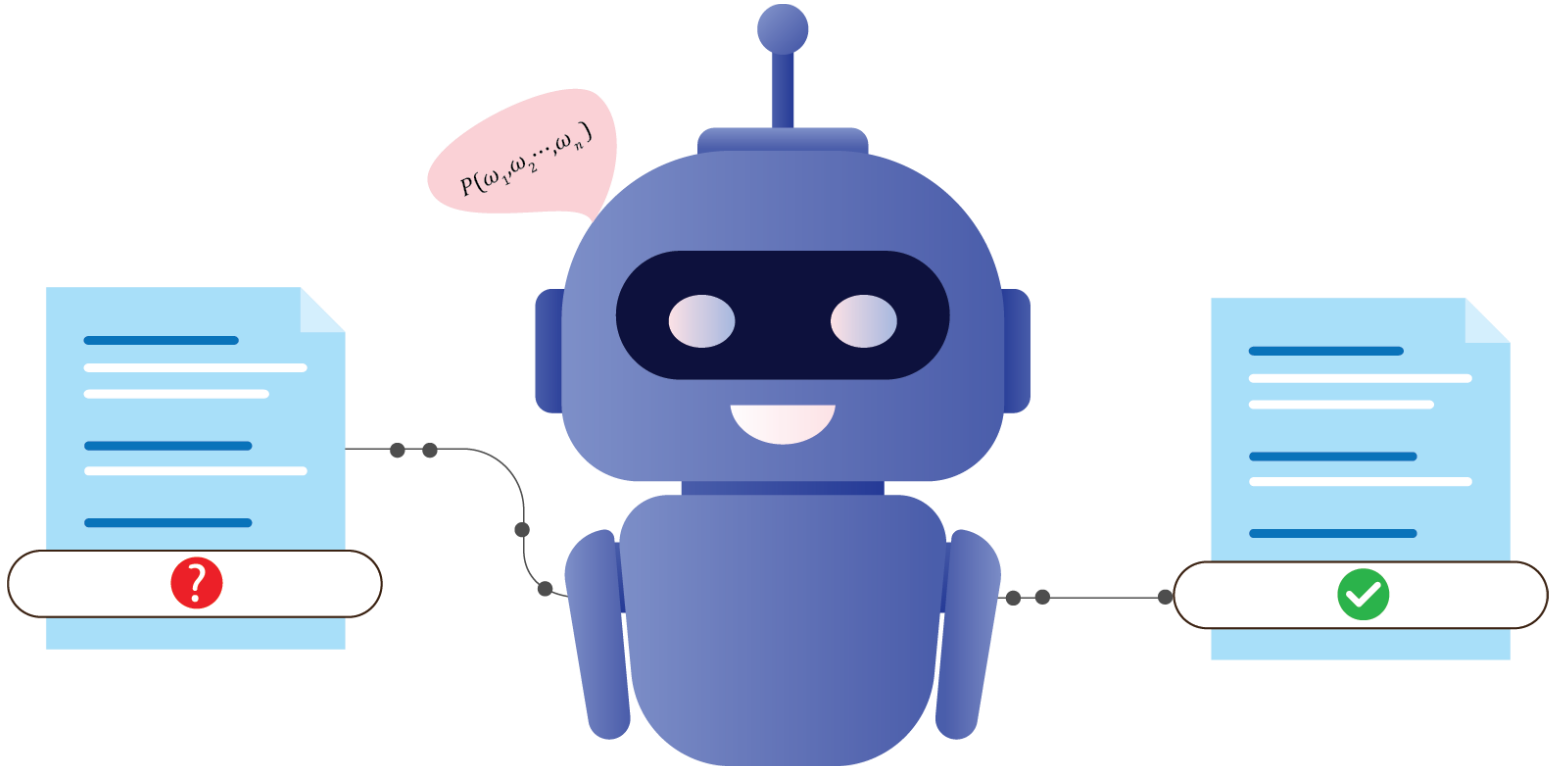
A language model is a probabilistic machine learning entity.

It resembles a complex function, designed to predict the probability of word sequences within a specific language corpus.

It is represented as: P(**Any sentence here**)

Language Models: Equation

Language models operate by assigning probabilities to sequences of words.



Mathematically, it looks like this:

$$P(\omega_1, \omega_2 \cdots, \omega_n) = P(\omega_1) \cdot P(\omega_2|\omega_1) \cdot P(\omega_3|\omega_1, \omega_2) \cdot \ldots \cdot P(\omega_n|\omega_1, \omega_2, \ldots, \omega_{n-1})$$

Language Models: Example

Consider the sentence: This is a new technology.

The language model calculates the probability of the sentence as:

P**(This is a new technology)**

$P(\text{This is a new technology}) = P(\text{This})\ \ P(\text{is} \mid \text{This})\ \ P(\text{a} \mid \text{This is})\ \ P(\text{new} \mid \text{This is a})\ \ P(\text{technology} \mid \text{This is a new})$

Language Models: Calculation

To illustrate, let's calculate the probability of two different sentences:

1. P(This is a fluffy dog.)
2. P(This are a purple flying deer.)

Solution: Sentence 1 gets a high probability, leveraging common context, and in sentence 2, rare and challenging words result in a lower probability.

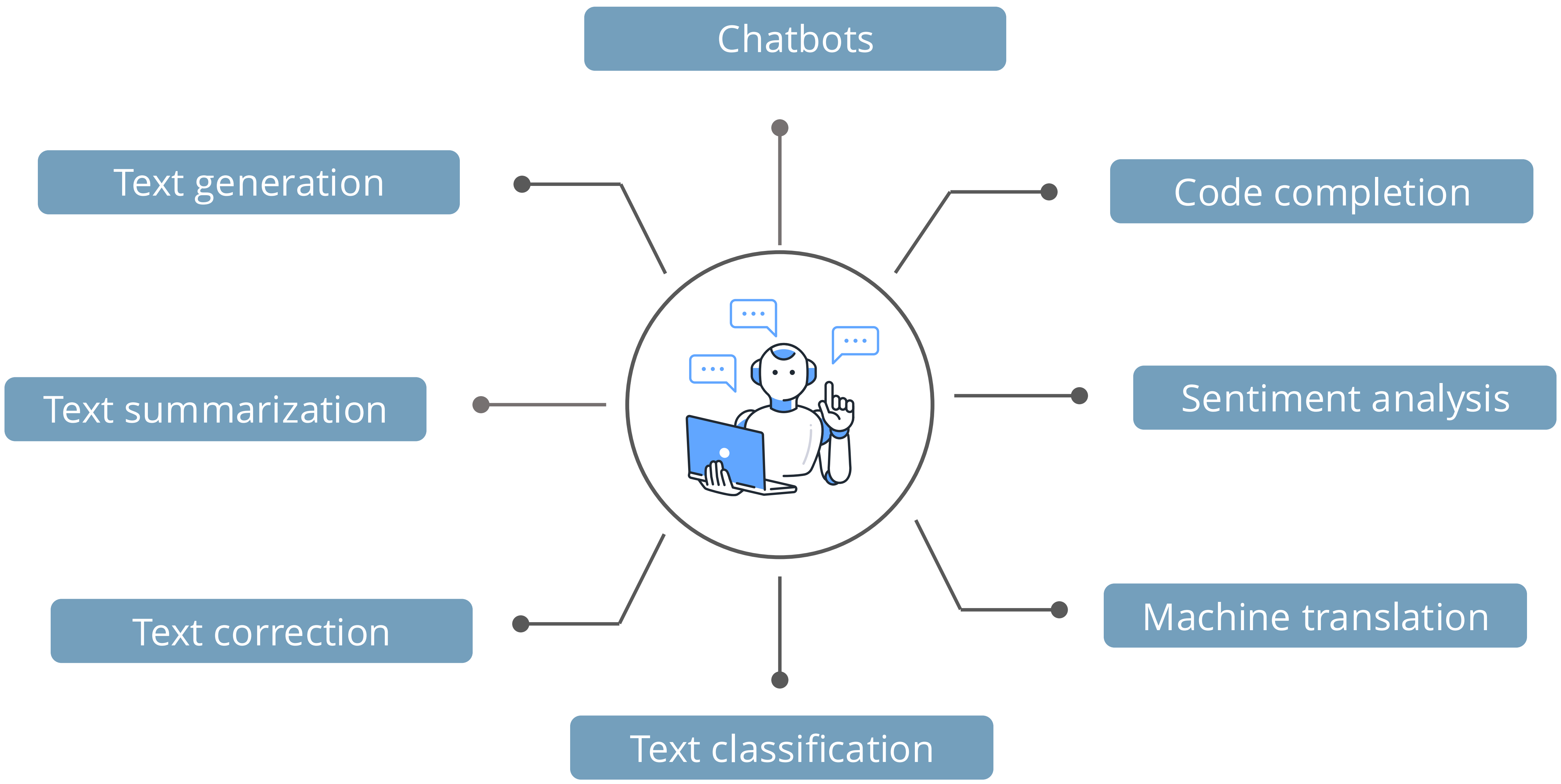
Power of Language Models

The powers of language models extend beyond just sentence prediction.

They are incredibly versatile.

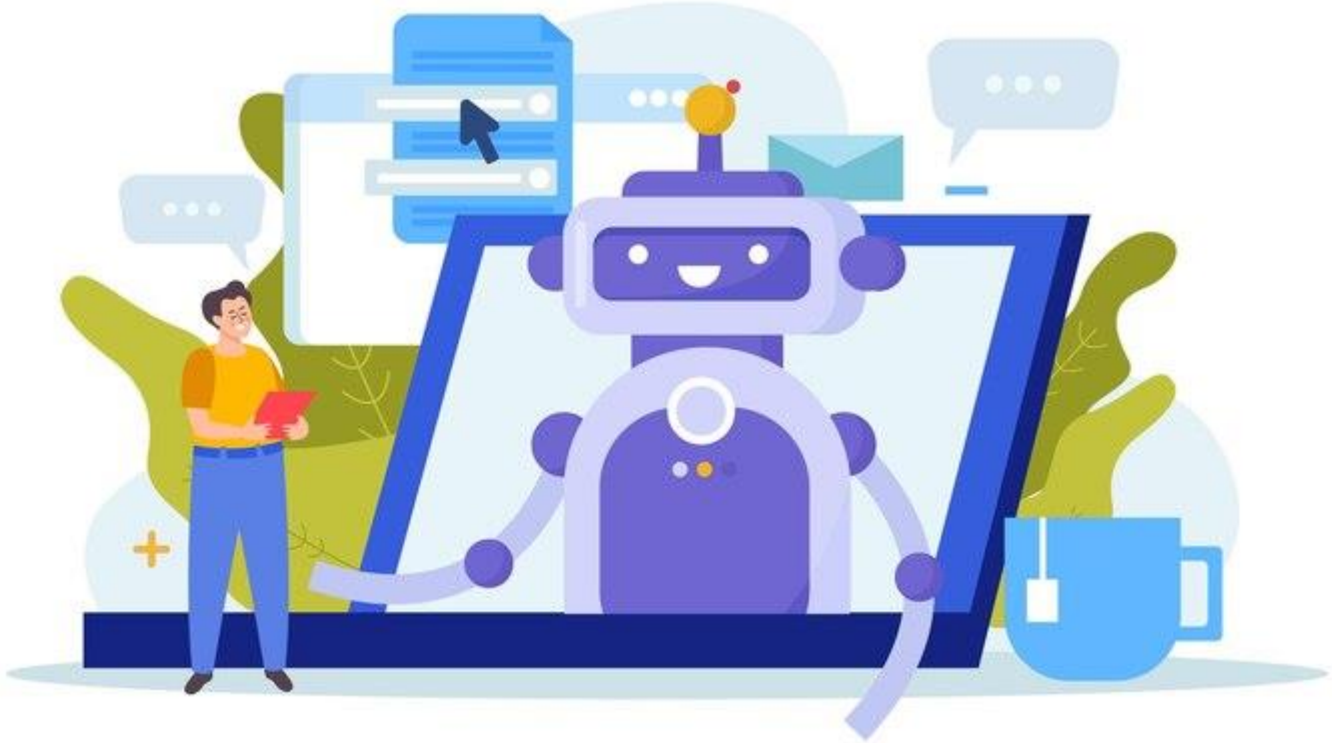
They can answer questions.

Applications of Language Models



Introduction to Large Language Models(LLM)

Large language models (LLMs) are trained on billions of words using deep learning techniques.



They possess the ability to generate, translate, and understand text in a human-like manner, marking a significant advancement in the field of NLP.

Working of Large Learning Models

Tokenization

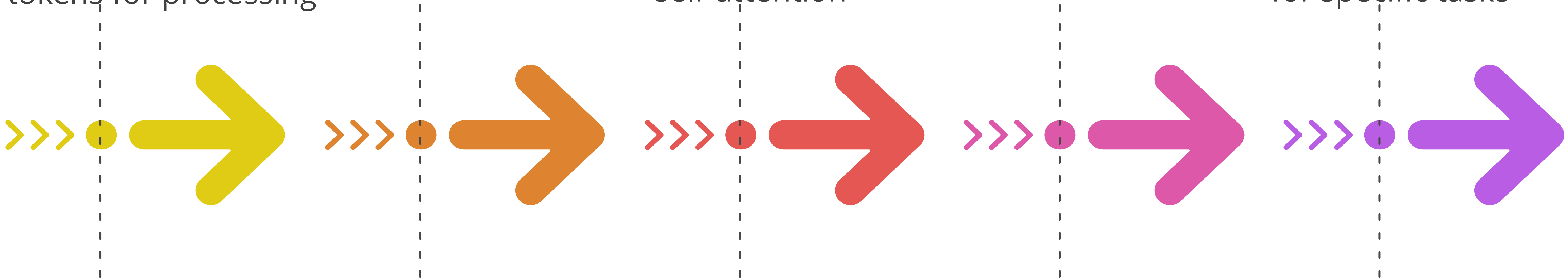
Breaking text into smaller units called tokens for processing

Processing with transformers

Analyzing text using self-attention

Fine-tuning & adaptation

Customizing LLMs for specific tasks



Embedding representation

Converting tokens into numerical vectors to capture meaning and relationship

Generating output

Predicting and constructing text based on learned patterns

Example: Large Learning Models

Tokenization

["Explain", "how", "solar", "panels", "generate", "electricity", "."]

Embedding representation

"solar" → [0.32, 0.89, -0.45, ...]

Processing with transformers

Self-attention identifies important words, linking "solar" with "panels" and "electricity."

Generating output

"Solar panels generate" → "Solar panels generate electricity"

Fine-tuning & adaptation

Customize LLMs for specific tasks (medical AI, legal AI)

Demo: Text Generation

Duration: 20 minutes

Imagine you are on a quest to understand the intricate art of text generation, where a computer learns the patterns of a given writing style and crafts its sentences.

Today's session will explore a Python script designed for educational purposes. This script employs the Natural Language Toolkit (NLTK) and the Brown corpus to demonstrate text generation through a Markov chain model using trigrams.

Note

Please download the solution document from the Reference Material Section and follow the Jupyter Notebook for step-by-step execution.

DEMONSTRATION

Quick Check

Which of the following is not an application of language models?

A. Text generation

B. Machine translation

C. Speech recognition

D. Image processing

Large Language Models

Large Language Models

Large Language Models (LLMs) are state-of-the-art AI models designed to comprehend and generate human language.

Large	Refers to the significant size and complexity of these models, which contains hundreds of millions or even billions of parameters
Language	Denotes their primary function, which is to understand and generate human language
Model	Describes them as mathematical representations that capture the patterns and structure of language data

Components of LLMs

- **Tokenization**
- **Embedding**
- **Attention**
- **Pretraining**

This process involves breaking down text into smaller units called tokens, which can be words, phrases, or even individual characters.

- **Transfer learning**
- **Encoder and decoder**
- **Scaling**

Components of LLMs

- Tokenization
- Embedding
- Attention
- Pretraining

This embedding component maps tokens to a high-dimensional vector space, representing each token with a unique vector.

- Transfer learning
- Encoder and decoder
- Scaling

Components of LLMs

- Tokenization
- Embedding
- Attention
- Pretraining

This attention mechanism lets the model concentrate on specific parts of the input text when generating output.

- Transfer learning
- Encoder and decoder
- Scaling

Components of LLMs

- **Tokenization**
- **Embedding**
- **Attention**
- **Pretraining**

This involves pretraining LLMs on extensive text data to understand the underlying patterns and structures of human language.

- **Transfer learning**
- **Encoder and decoder**
- **Scaling**

Components of LLMs

- Tokenization
- Embedding
- Attention
- Pretraining

This component allows the model to adapt to new tasks by fine-tuning the pre-trained model on a smaller dataset.

- Transfer learning
- Encoder and decoder
- Scaling

Components of LLMs

Tokenization

Embedding

Attention

Pretraining

This employs the Transformer framework in a large language model architecture, comprising two main parts: an encoder and a decoder.

Transfer learning

Encoder and decoder

Scaling

Components of LLMs

Tokenization

Embedding

Attention

Pretraining

This necessitates significant computational resources for training and upkeep, making scaling a challenging but essential part of its architecture.

Transfer learning

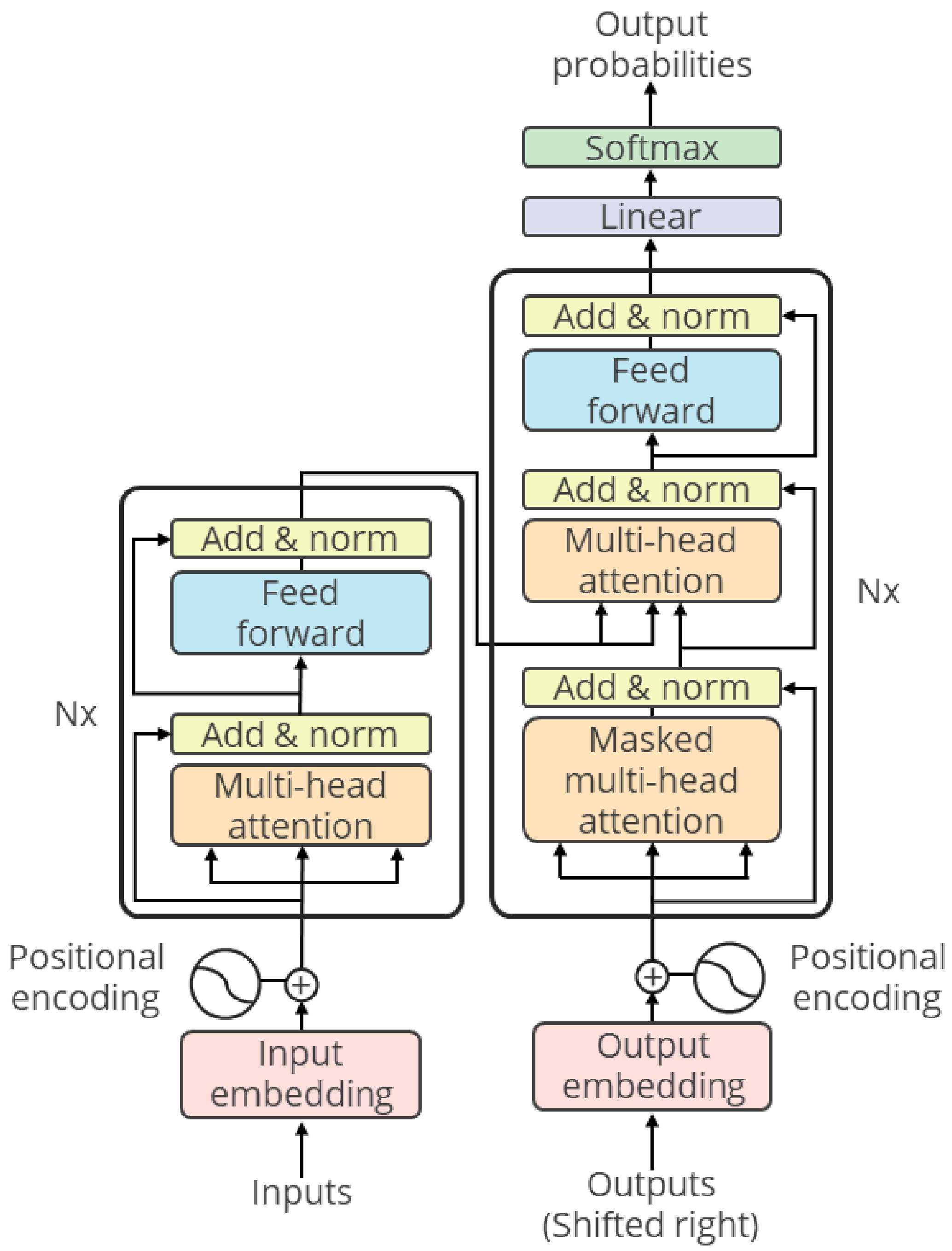
Encoder and decoder

Scaling

LLM Architecture

Components of LLM architecture

- Input embeddings
- Positional encoding
- Encoder
  - Attention mechanism
  - Feed-forward neural network
- Decoder
- Multi-headed attention
- Layer normalization
- Output



LLM Operations

These represent the functions of components within an architecture.

- Input embeddings
- Positional encoding
- Encoder
- Decoder
- Multi-headed attention
- Layer normalization
- Output

- The machine takes in a sentence and breaks it down into smaller pieces.
- Each of these pieces is turned into a special kind of code that the machine can understand.
- This code holds the meaning of the words.

LLM Operations

These represent the functions of components within an architecture.

- Input embeddings
- **Positional encoding**
- Encoder
- Decoder
- Multi-headed attention
- Layer normalization
- Output

- The machine wants to understand not just what words are there but also their order in the sentence.
- So, it adds some extra information to the code to show where each word is in the sentence.

LLM Operations

These represent the functions of components within an architecture.

- Input embeddings
- Positional encoding
- Encoder
- Decoder
- Multi-headed attention
- Layer normalization
- Output

- **Encoder:** Now, the machine gets to work on analyzing the sentence. It creates a bunch of memories to remember what it has read.
- **Attention mechanism:** The machine pays more attention to some words depending on their importance in the sentence.
- **Feed forward:** After paying attention to words, the machine thinks hard about each word on its own.

LLM Operations

These represent the functions of components within an architecture.

- Input embeddings
- Positional encoding
- Encoder
- **Decoder**
- Multi-headed attention
- Layer normalization
- Output

- The machine not only understands but also generates new sentences.
- For this, it has a special part called the decoder.
- The decoder helps the machine predict what word comes next based on what it has understood so far.

LLM Operations

These represent the functions of components within an architecture.

- Input embeddings
- Positional encoding
- Encoder
- Decoder
- Multi-headed attention
- Layer normalization
- Output

- The machine looks at the words in different ways simultaneously.
- This helps the machine grasp different aspects of the sentence all at once.

LLM Operations

These represent the functions of components within an architecture.

- Input embeddings
- Positional encoding
- Encoder
- Decoder
- Multi-headed attention
- Layer normalization
- Output

- This layer is in place to keep everything in check and make sure the machine learns well.
- The machine normalizes its understanding at each step.

LLM Operations

These represent the functions of components within an architecture.

- Input embeddings
- Positional encoding
- Encoder
- Decoder
- Multi-headed attention
- Layer normalization
- Output

- Finally, the machine produces its own understanding or generates new sentences.
- The output depends on what the machine is designed to do.
- For example, if it's predicting the next word in a sentence, it gives a probability for each word.

LLM Training Steps

The steps in the training process of a language model are:

- Corpus preparation
- Tokenization
- Neural network training
- Embedding generation

Quick Check

When considering the architecture of Large Language Models (LLMs), which of the following components is responsible for generating human-like text and responding to prompts?

A. Tokenization

B. Embedding

C. Neural network training

D. Fine-tuning

Types of Large Language Models (LLMs)

Types of LLMs

Below are the various pretrained LLMs available in the market:

GPT 4

DeepSeek

Claude

Cohere

Falcon

LLaMA

Types of LLMs: GPT 4

This model is OpenAI's most advanced version, integrating multimodal capabilities for text, image, and audio processing.

Performance	It performs like a human on tests. It mimics human-like speech and reasoning.
Pros	The model excels in multilingual processing, complex reasoning, and real-time interactions.
Cons	While powerful, GPT-4o still has potential biases and requires significant computational resources.

Types of LLMs: DeepSeek-R1

DeepSeek-R1 is a cutting-edge open-source LLM developed by DeepSeek AI, designed for efficiency and scalability.

Performance	It matches GPT- 4-level performance while being more cost-efficient and requiring fewer computational resources.
Pros	It uses a mixture of experts (MoE) architecture, allowing for efficient and adaptive computing.
Cons	It may require fine-tuning for domain-specific applications, and adoption is still growing.

Types of LLMs: Claude 3.5 Sonnet

This model is the latest addition to Anthropic's Claude series, designed to enhance reasoning, accuracy, and response reliability while prioritizing AI safety and ethical considerations.

Performance	It delivers increased response times, improved contextual understanding, and enhanced reasoning across complex tasks.
Pros	It provides detailed, well-structured responses while maintaining clarity and accuracy.
Cons	It lacks multimodal capabilities, unlike models such as GPT-4o. While more advanced, it remains cautious in responses, sometimes limiting creativity or controversial discussions.

Types of LLMs: Cohere Command R+ 08-2024

This model is Cohere's latest enterprise-focused LLM, optimized for retrieval-augmented generation (RAG) and business applications requiring high accuracy and efficiency.

Performance	It delivers state-of-the-art retrieval-augmented responses, ensuring better factual accuracy and context awareness.
Pros	It is optimized for long-context processing, handling extended conversations and large documents effectively.
Cons	It is primarily designed for enterprise use, making it less accessible to general consumers.

Types of LLMs: Copilot

GitHub Copilot is an AI-powered code assistant that integrates into IDEs to enhance coding efficiency using multiple AI models, including OpenAI's GPT-4o.

Performance	It provides real-time code suggestions, debugging assistance, and context-aware completions across multiple programming languages.
Pros	It seamlessly integrates with Visual Studio Code, JetBrains, and Neovim for a smooth workflow.
Cons	It may cause over-reliance, reducing manual coding skills over time.

Types of LLMs: LLaMA 3

Llama 3.3 is Meta AI's latest open-source large language model, released in December 2024. It offers advanced language processing capabilities while being optimized for efficiency.

Performance	It supports larger context windows (up to 128K tokens) for better comprehension in long conversations.
Pros	It is open-source and customizable, allowing researchers and enterprises to fine-tune models. It improves efficiency by using grouped-query attention (GQA) for faster inference.
Cons	It has high computational requirements for deploying larger versions.

Demo: Testing Different LLM Models

Duration: minutes

Overview:

This demo evaluates the performance of various LLMs by testing their accuracy, response quality, and adaptability using a standardized approach.

DEMONSTRATION

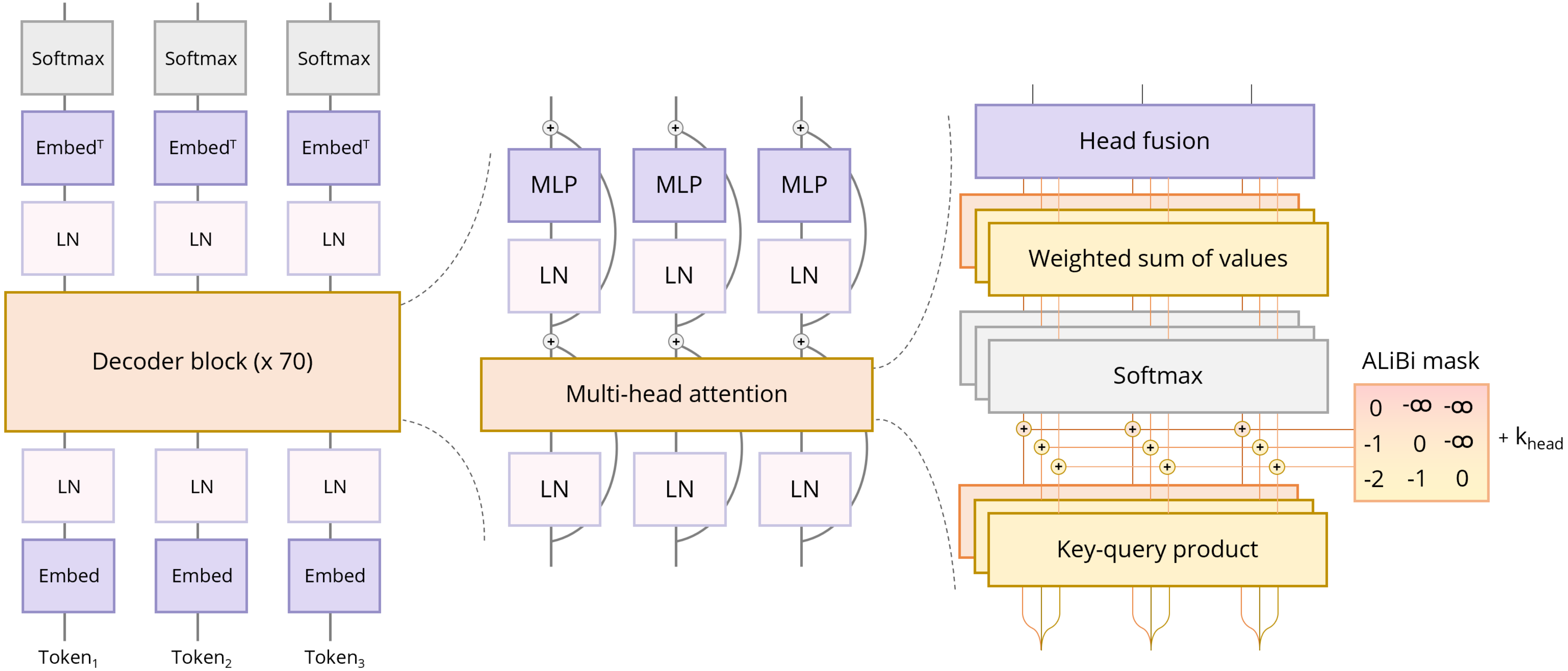
Bloom

Bloom Overview

It is an autoregressive Large Language Model trained on extensive text data using industrial-scale computational resources.

Bloom's Architecture

BLOOM adopts a conventional decoder-only transformer architecture.



Bloom's Architecture

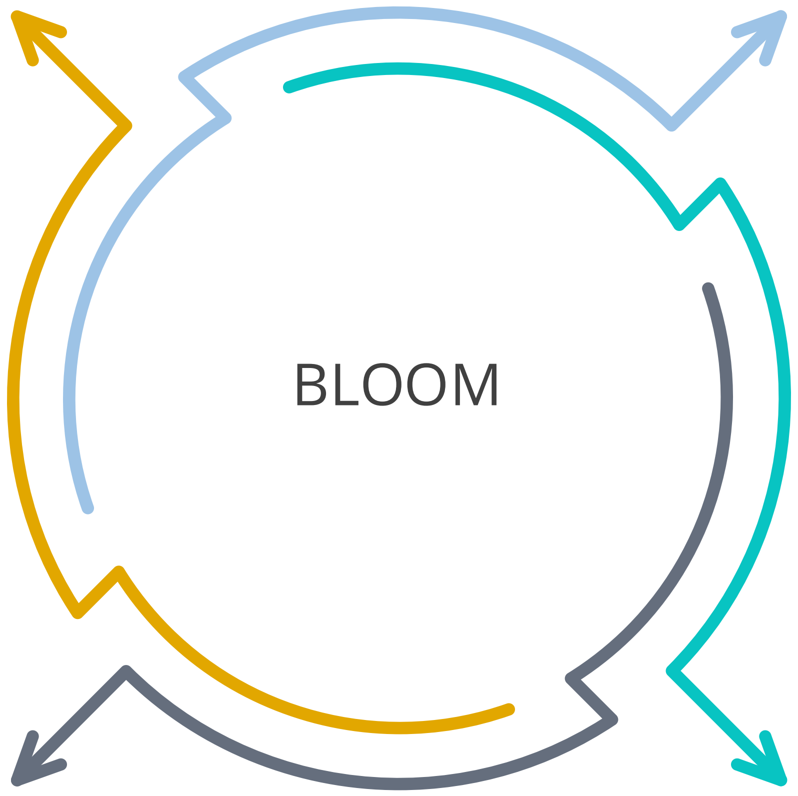
It features several notable modifications, including:

ALiBi	This component enhances the model's capacity to generalize to longer context lengths beyond what it encounters during training.
Embedding layer norm	An additional layer of normalization is introduced after the model's embedding layer, contributing to enhanced training stability.

Unpacking Bloom

It is trained on a massive 1.6TB of text data.

It boasts a staggering 176 billion parameters.



It excels in text generation: 46 natural and 13 programming languages.

It is an architecture, rooted in an auto-regressive model.

LLM Reasoning

Diverse reasoning

The LLM explores varied reasoning, including common sense and math, adapting to diverse contexts.

Eliciting reasoning

Methods like chain-of-thought prompting guide LLMs to stimulate and prompt thoughtful reasoning.

Reasoning contribution enigma

The challenge lies in understanding reasoning's role and impact, differentiating it from factual information.

Quick Check

Which method can be utilized to unleash the reasoning capabilities of LLMs?

A. Cross-Modal Learning

B. Few-Shot Learning

C. Chain-of-Thought Prompting

D. Self-Supervised Learning

LLM Considerations and Future Implications

LLM Considerations

There are two types of considerations for choosing an LLM:

Critical considerations:

Evaluate non-technical aspects, like ethics and biases.

Technical considerations:

Assess performance, architecture, and computational requirements.

Critical Considerations

The critical considerations for choosing an LLM are:

- Licensing and commercial use
- Practical factors for inference speed and precision
- The impact of context length and model size
- Task-specific vs. general-purpose
- Testing and evaluation
- Deployment cost considerations

Technical Considerations

The technical considerations for choosing an LLM are:

- Data security and privacy
- Model inference monitoring
- Scalability and performance
- Version control and updating
- APIs and integration security

Future Implications of LLMs

LLMs have far-reaching implications, which include:

- Job market disruption
- Enhancing productivity and creativity
- Societal impact
- Responsible use
- Evolving opportunities

Quick Check

What is not a potential future implication of using LLMs in real-world applications?

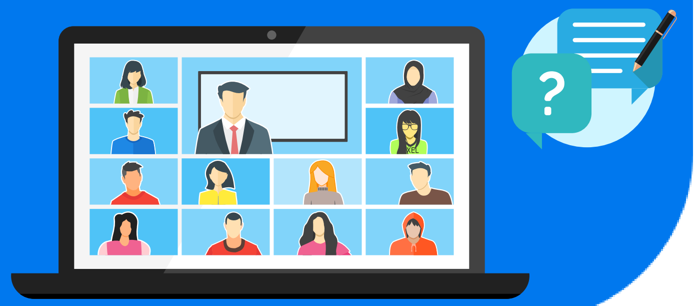
A. Increased job opportunities and economic growth

B. Automation of tasks leading to job market disruption

C. Enhanced productivity and creativity for individuals and businesses

D. Ethical and societal considerations surrounding the use of LLMs

Guided Practice



Overview

Duration: 25 minutes

This activity focuses on testing understanding of diverse language models and their applications. It presents scenarios that require applying learned concepts to solve problems or accomplish tasks.

GUIDED PRACTICE

Key Takeaways

- Language model is a machine learning entity.
- Large Language Models are trained on large datasets, and they can generate human-like text, images, and many more.
- Pretrained LLMs available in the market can be utilized for powerful generative AI solutions
- Bloom is an autoregressive LLM capable of generating text in 46 natural languages and 13 programming languages.

Q&A