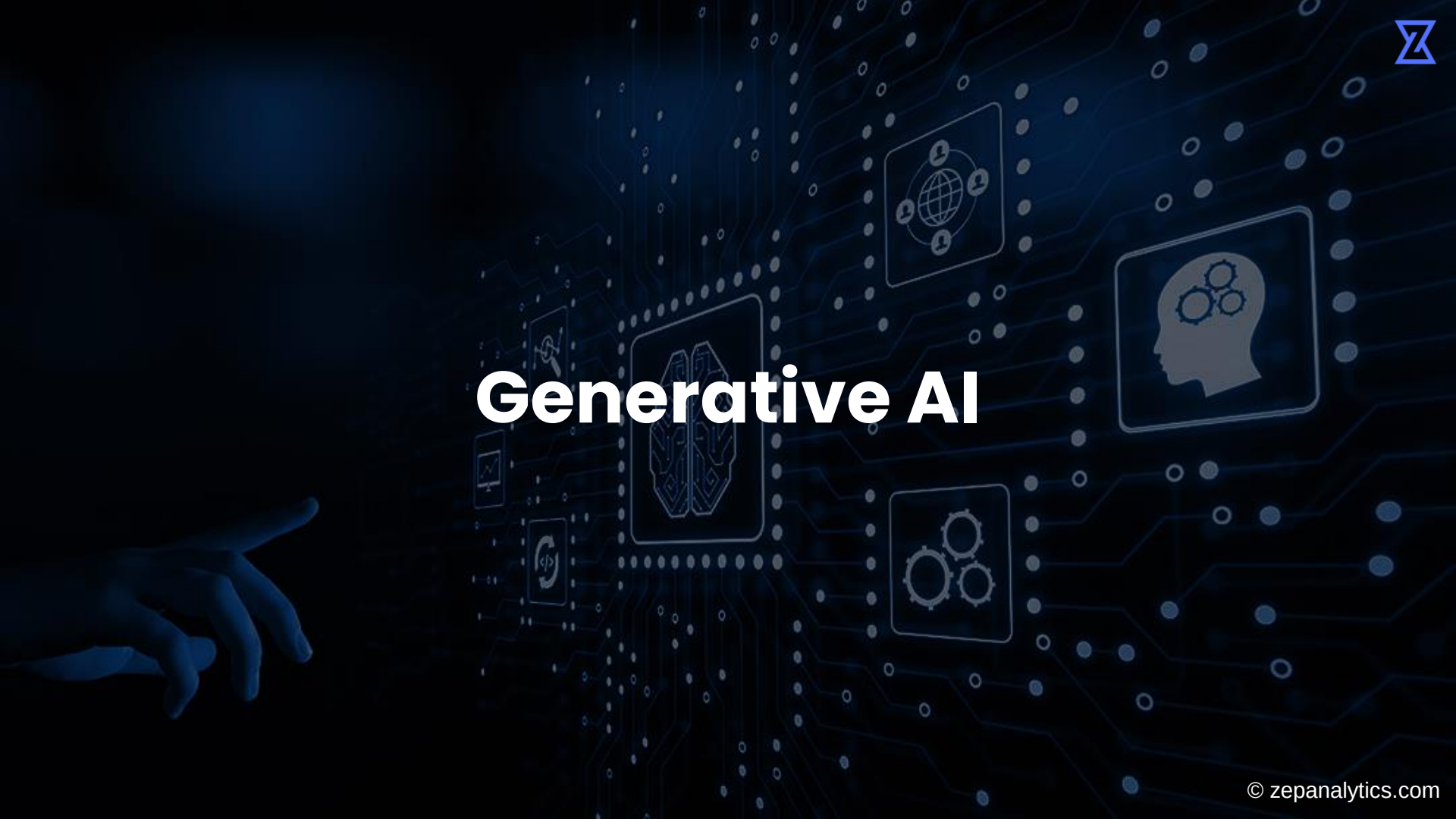


Agenda

- Introduction to Generative AI
- Fundamentals of Neural Networks
- Transformers and Self-Attention
- BERT (Bidirectional Encoder Representations from Transformers)
- Large Language Models (LLMs)
- Sequence-to-Sequence Models
- Language Modelling
- Prompting and Prompt Engineering
- Applications of Generative AI
- Ethical Considerations and Limitations
- Future Trends and Research Directions

Introduction to Generative AI

Understanding the World of Human Language and Computers

Generative AI

Generative AI is a type of artificial intelligence technology that can produce various types of content, including text, imagery, audio and synthetic data.

Generative AI refers to deep-learning models that can generate high-quality text, images, and other content based on the data they were trained on.

Overview of GEN AI Applications

01

Create Text - NLP enables computers to translate text or speech from one language to another, facilitating cross-language communication and accessibility.

02

Image-to-Image Conversion - It involves changing the external components of an image while maintaining its internal components, such as colour, media, or shape.

Such a transformation may involve changing the daytime image into the night-time image.

03

Increase Image Resolution - While creating new documents from existing content, generative AI employs various techniques. One such technique is called a Generic Adversarial Network (GAN).

The generator and the discriminator form a GAN, which generates new data and ensures that it is factual.

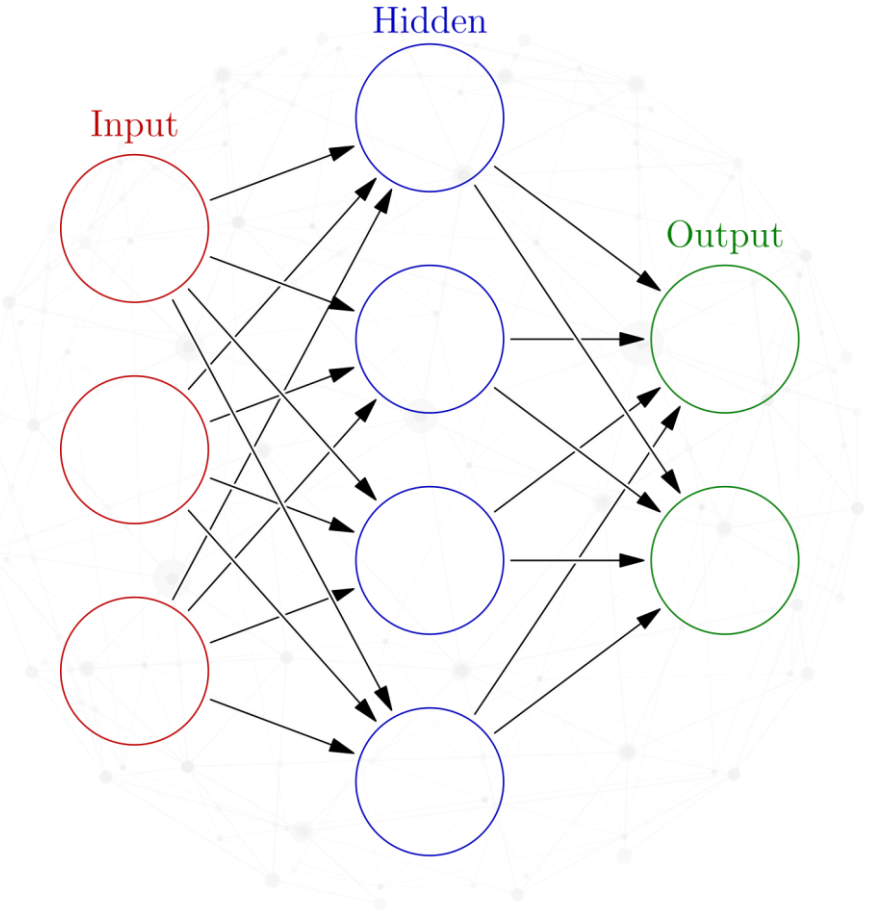
Fundamentals of Neural Networks

What is a Neural Network ?

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain.

It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain.

It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy.



Components of Neural Network

A neural network has four main components

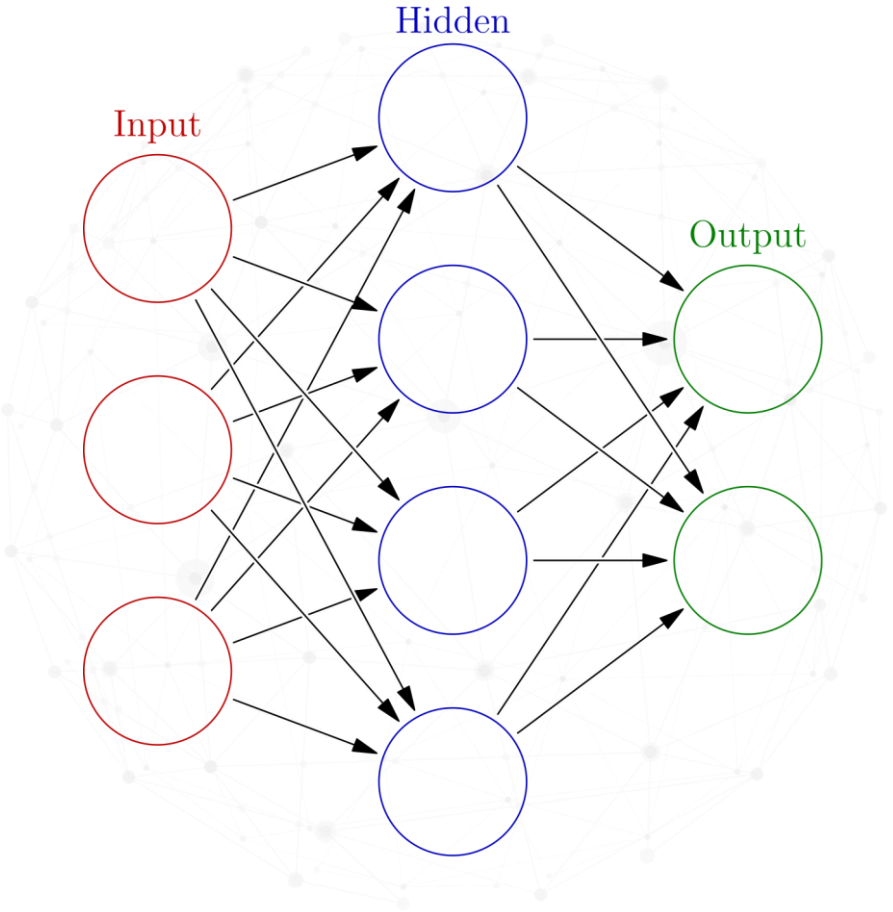
1. **Input layer**
2. **Hidden layer**
3. **Output layer**
4. **Activation Function (Node)**

1. **Input layer**

   Information from the outside world enters the artificial neural network from the input layer. Input nodes process the data, analyze or categorize it, and pass it on to the next layer.

1. **Hidden layer**

   Hidden layers take their input from the input layer or other hidden layers. Artificial neural networks can have a large number of hidden layers. Each hidden layer analyzes the output from the previous layer, processes it further, and passes it on to the next layer.



Components of Neural Network

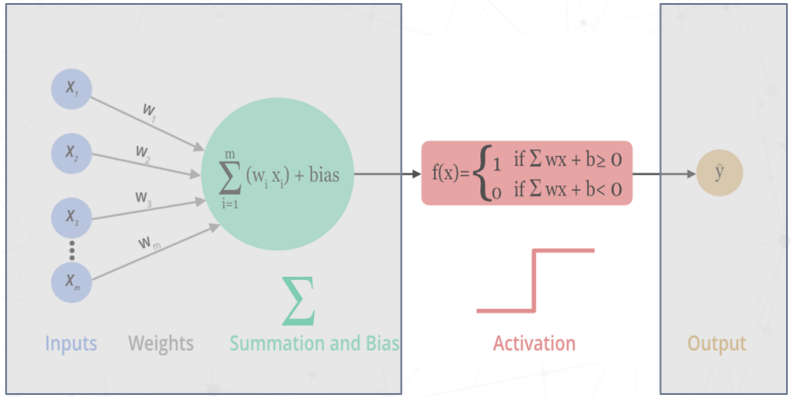
3. Output layer

The output layer gives the final result of all the data processing by the artificial neural network. It can have single or multiple nodes. For instance, if we have a binary (yes/no) classification problem, the output layer will have one output node, which will give the result as 1 or 0.

4. Activation function

an activation function is a mathematical function that determines the output of a neuron (node) given its input. Activation functions introduce non-linearity into the output of a neuron, which is crucial for learning complex patterns in data.

The primary role of activation functions is to decide whether a neuron should be activated (fired) or not, based on the weighted sum of its inputs and a bias term.



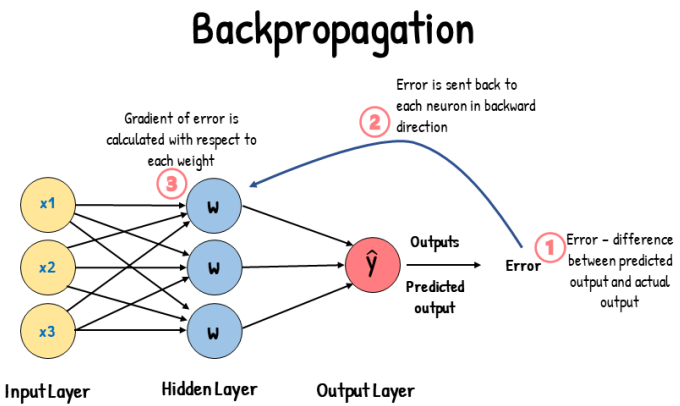
Backpropagation and gradient descent

Backpropagation

Backpropagation is an algorithm used to calculate the gradients (partial derivatives) of the error function with respect to the weights and biases in a neural network.

It is a algorithm that allows the network to adjust its weights and biases based on the error between the predicted output and the expected output.

Backpropagation is an efficient way to train multi-layer neural networks and is a key algorithm that enabled the success of deep learning.

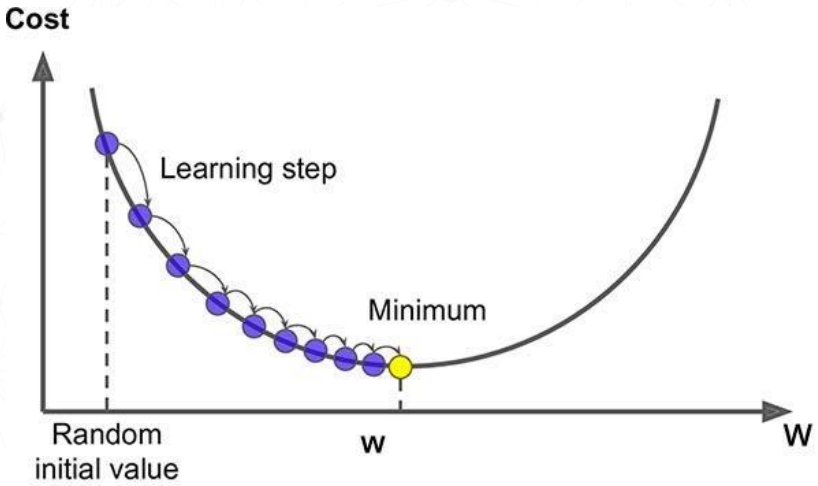


Backpropagation and gradient descent

Gradient Descent

gradient descent is an iterative optimization algorithm that finds the minimum of the loss function.

- Once we have calculated the gradients (derivatives) of the loss function with respect to each weight in the network using backpropagation.
- The gradients tell us how much the loss function changes for small changes in each weight.
- To minimize the loss, we update each weight in the opposite direction of its gradient. If the gradient is positive (loss increases as weight increases), we decrease the weight. If the gradient is negative (loss decreases as weight increases), we increase the weight.
- By updating the weights in the opposite direction of the gradients, we move the weights towards values that decrease the overall loss function.



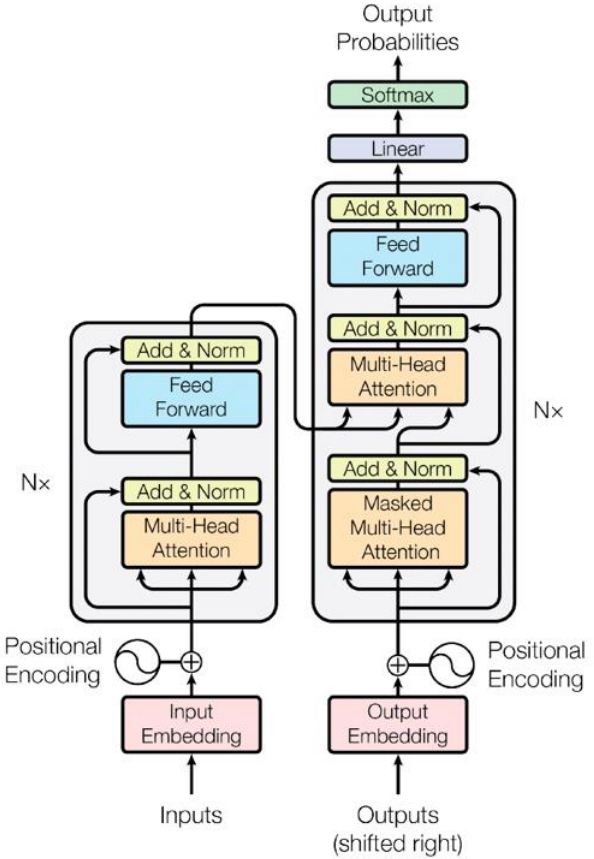
Transformers

What is a Transformer ?

Transformers are a type of neural network architecture that can process and understand sequences of data, such as text, audio, or time-series data. They work by transforming or converting an input sequence into an output sequence.

The main idea behind transformers is that they use attention mechanisms to learn the relationships and context within the input sequence.

Attention allows the model to focus on the relevant parts of the input sequence when producing the output.

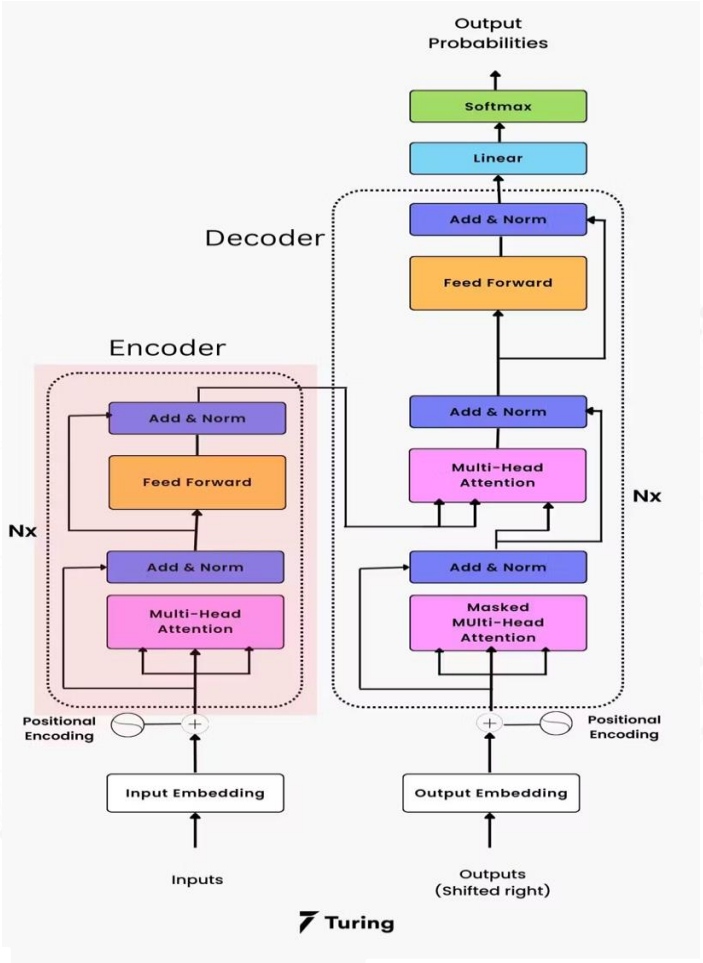


Architecture of Transformer

Transformers consist of two main networks an **encoder** and a **decoder**.

The encoder takes the input sequence and creates a representation that captures the relationships and context within the input.

The decoder then uses this representation, along with the attention mechanism, to generate the output sequence one element at a time.

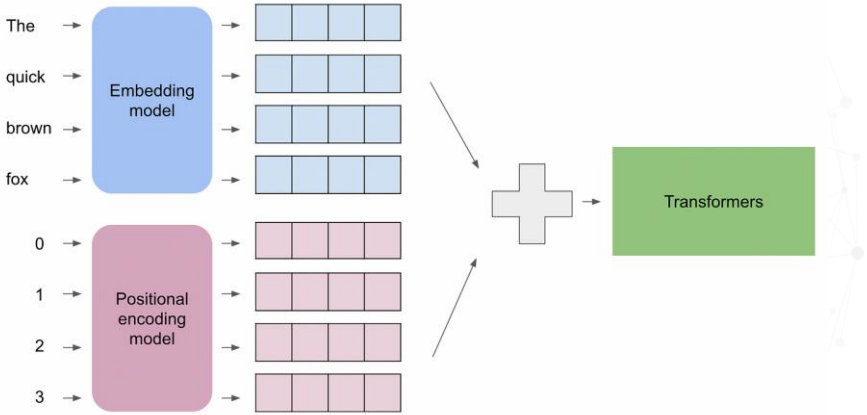


Working of transformer

The working of a transformer can be explained in the following steps:

Input Embedding: The input sequence (e.g., a sentence or a series of tokens) is first converted into numerical vectors called embeddings. Each token in the sequence is represented by a unique vector.

Positional Encoding: Since transformers do not have an inherent notion of order like recurrent neural networks, positional encoding is added to the embeddings. This encodes the position of each token in the sequence, allowing the model to understand the order of the tokens.

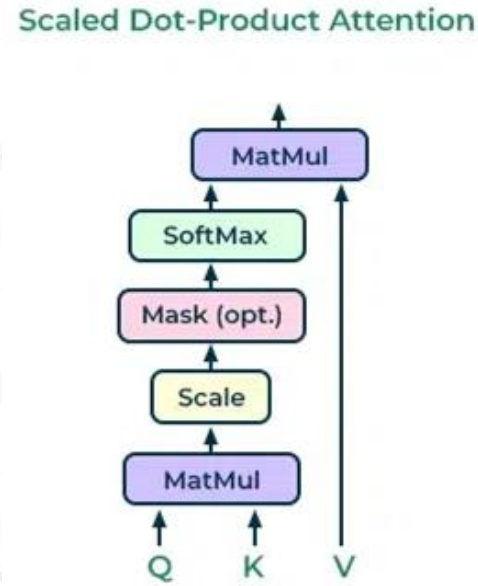


Self-attention mechanism

Self-attention operates by transforming the input sequence into three vectors: query, key, and value. These vectors are obtained through linear transformations of the input.

The attention mechanism calculates a weighted sum of the values based on the similarity between the query and key vectors.

The resulting weighted sum, along with the original input, is then passed through a feed-forward neural network to produce the final output. This process allows the model to focus on relevant information and capture long-range dependencies.



How Key, Query and Value vectors are obtained ?

The calculations for Q, K, and V are as follows:

input sequence of embeddings $\boldsymbol{X = [x_1, x_2, ..., x_n]}$, where each x_i is a d-dimensional embedding vector representing a token in the sequence.

Query (Q): $\boldsymbol{Q = X \times W_\varphi}$
The Query matrix Q is obtained by multiplying the input embeddings X with a weight matrix W_φ of shape (d, d_φ), where d_φ is the dimension of the Query vectors.

Key (K): $\boldsymbol{K = X \times W_k}$
The Key matrix K is obtained by multiplying the input embeddings X with a weight matrix W_k of shape (d, d_k), where d_k is the dimension of the Key vectors.

Value (V): $\boldsymbol{V = X \times W_v}$
The Value matrix V is obtained by multiplying the input embeddings X with a weight matrix W_v of shape (d, d_v), where d_v is the dimension of the Value vectors.

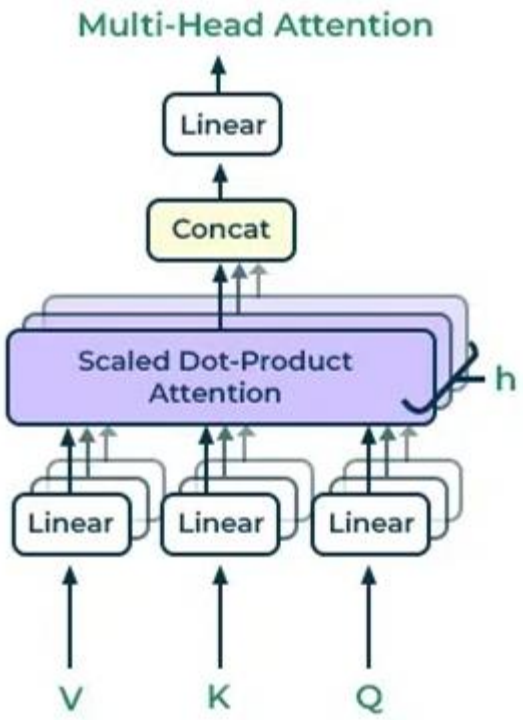
The weight matrices W_φ, W_k, and W_v are learnable parameters of the self-attention mechanism, and they are updated during the training process.

Multi head Attention

The multi-head attention mechanism is a concatenation of multiple single-head attention computations running in parallel. Each attention head independently projects the input embeddings into its own set of Query, Key, and Value vectors.

The outputs from all attention heads are concatenated, allowing the model to attend to different representations of the input simultaneously. This concatenated output goes through a final linear projection to produce the multi-head attention output.

This parallelization allows the model to capture diverse relationships and dependencies within the input sequence, enhancing its ability to model complex data effectively.



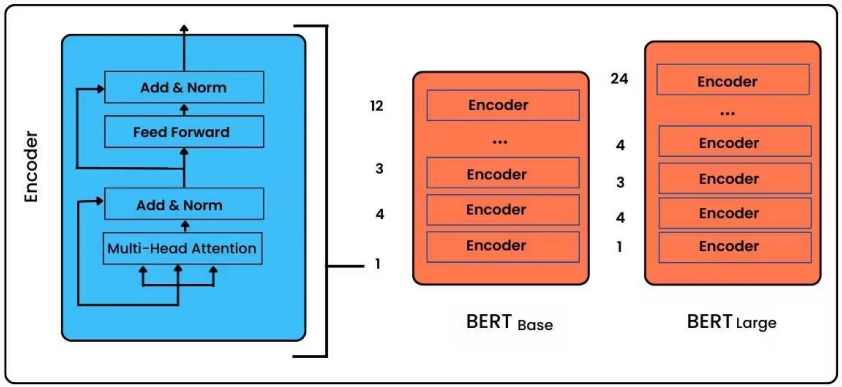
BERT (Bidirectional Encoder Representations from Transformers)

What is BERT ?

BERT is pre-trained on a large corpus of unlabeled text data using two unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction. This pre-training allows BERT to develop deep bidirectional representations of language.

Unlike traditional language models that process text sequentially, BERT uses the transformer architecture to attend to all words in a sentence simultaneously, allowing it to capture bidirectional context.

BERT's architecture is based on the transformer encoder, which uses multi-head self-attention mechanisms to encode the input sequence.



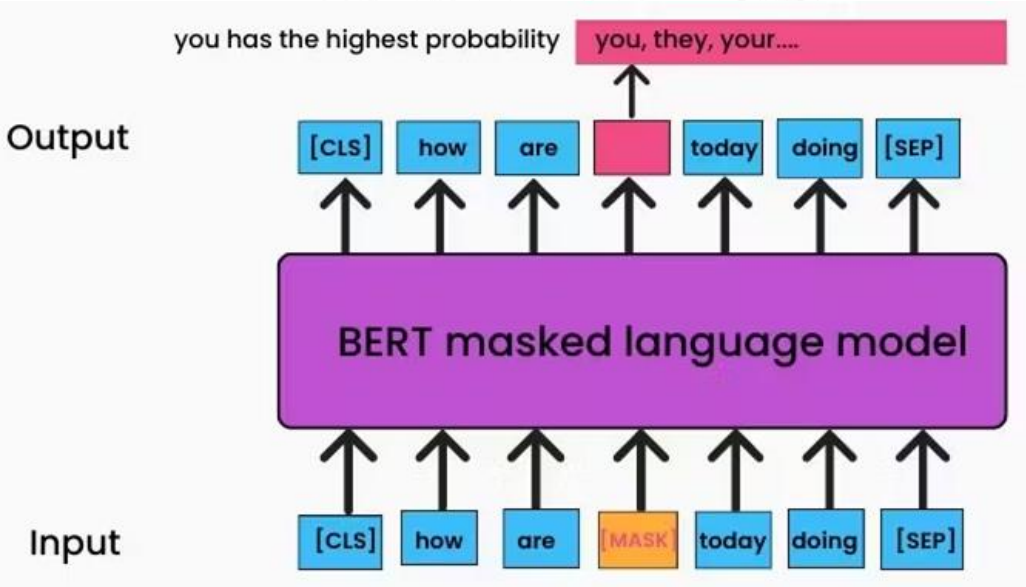
Role of Masked language modeling

MLM enables bidirectional learning by masking (hiding) some words in the input text sequence.

BERT is trained to predict the masked words by utilizing the context from the remaining words both before and after the masked word.

This bidirectional context consideration allows BERT to develop a deeper understanding of language by leveraging information from both directions.

BERT's objective is to accurately predict these masked words based on the bidirectional context.

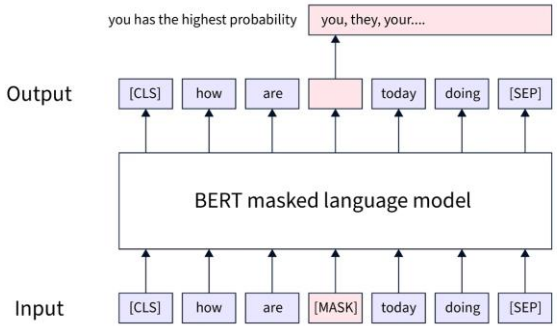


Pre-training and fine-tuning BERT

Pre-training BERT involves two main tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP).

1. Masked Language Modeling (MLM)

- Input text is tokenized, and 15% of tokens are randomly masked with [MASK].
- BERT is trained to predict the original masked tokens from the remaining context tokens.
- This forces BERT to learn bidirectional representations by leveraging context from both sides of the masked tokens.



Next Sentence Prediction (NSP).

- Input sequences are formed by concatenating two sentences, separated by [SEP].
- BERT is trained to predict if the second sentence follows the first in the original text.
- NSP allows BERT to learn sentence relationships, beneficial for tasks like summarization and QA.

Fine tuning BERT

We extend the pre-trained BERT with an additional output layer and then fine-tune the resulting model's entire set of parameters to train task-specific models.

Extending BERT with Task-Specific Layers

- **Additional Output Layer:** A new output layer is added to the pre-trained BERT model to adapt it to the specific task (e.g., classification layer for text classification tasks or span prediction layers for QA tasks).

- **Fine-Tuning Entire Model:** Unlike freezing the pre-trained layers, the entire set of parameters of the BERT model, along with the new task-specific layers, is fine-tuned. This means all weights are updated during training on the task-specific dataset.

Variants of BERT

1. RoBERTa (Robustly optimized BERT approach)

RoBERTa is an optimized version of BERT that emphasizes training with more data and computational resources, as well as some modifications in the training procedure.

Key Improvements:

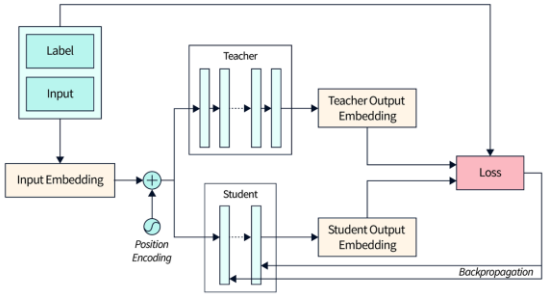
- Training Data: Uses much larger training datasets.
- Training Duration: Trains for longer periods.
- Dynamic Masking: Changes masking patterns dynamically during training instead of using static masking.

2. DistilBERT

DistilBERT is a smaller, faster, and cheaper version of BERT that uses knowledge distillation to retain a significant portion of BERT's capabilities while being lighter.

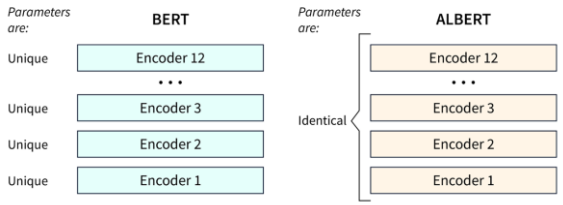
Key Improvements:

- Model Size: 40% fewer parameters than BERT.
- Speed: Runs 60% faster.
- Performance: Retains about 97% of BERT's performance on language understanding benchmarks.



3. ALBERT (A Lite BERT)

ALBERT is designed to be more parameter-efficient by sharin parameters across layers and factorizing the embedding parameter leading to a smaller model with faster training times.



Key Improvements:

- Parameter Sharing: Shares parameters across layers to reduc model size.
- Factorized Embedding Parameterization: Splits the embedding matrix into smaller matrices to improve efficiency.

4. TinyBERT

TinyBERT is another distilled version of BERT, focusing on compressing the model size while maintaining a good balance of performance.

Key Improvements:

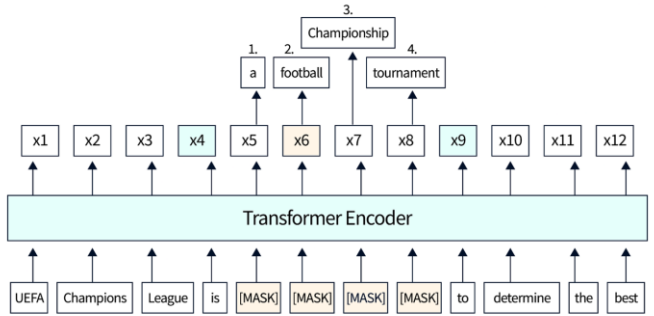
- Model Compression: Applies both knowledge distillation and data augmentation techniques.
- Size and Speed: Significantly smaller and faster than BERT.

5. SpanBERT

SpanBERT is an enhancement of BERT that focuses on better capturing span-level representations, making it more effective for tasks involving spans of text like question answering and coreference resolution.

Key Improvements:

- Span Prediction: Trains the model to predict masked spans of text instead of individual tokens.
- Objective Modification: Uses a span-boundary objective to improve the model's ability to understand the context of spans.



LLMs (Large Language Models)

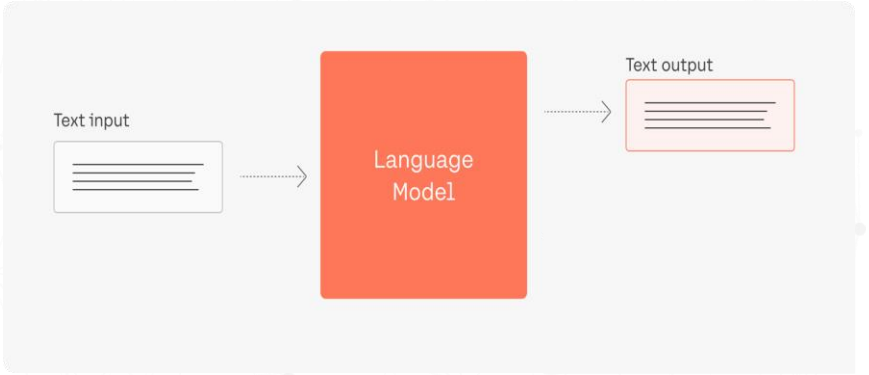
LLMs (Large Language Models)

Large language models (LLMs) are deep learning algorithms.

Large language models use transformer models and are trained using massive datasets.

They also have a remarkable amount of general knowledge about the world and can “memorize” an enormous number of facts during training.

Large language models are also referred to as neural networks (NNs), which are computing systems inspired by the human brain.



GPT 3.5 (Generative Pretrained model)

GPT 3.5, Sometimes called InstructGPT that trained on the same datasets of GPT-3 but with additional fine tuning process that adds a concept called ‘reinforcement learning with human feedback’ or RLHF to the GPT-3 model.

Instead of relying solely on labeled data, humans provide rewards, corrections, or adjustments to guide the model's behavior toward desired outcomes.

RLHF aims to incorporate human knowledge and preferences into ML systems, enabling them to learn complex tasks more effectively.

Capabilities of GPT 3.5

- Understand and generate human-like text using natural language comprehension and generation to complete various natural language-related tasks.
- Translate text from one language to another with some fluency and accuracy.
- Answer questions by providing relevant information, making it suitable for chatbots and virtual assistants using GPT-3.5 Turbo, which is tailored to working with the Chat Completions API.
- Generate concise summaries of longer text, such as documentation and reports.
- Generate content for a wide range of use cases and writing projects, such as emails and code.

GPT 4

GPT-4's extended capabilities include the following:

Multimodality - GPT-4 can accept images as inputs to generate captions, classifications and analyses.

Larger context windows - GPT-4 offers two context window sizes -- 8,192 and 32,768 tokens -- with which the model can handle over 25,000 words of text.

Broader general knowledge - GPT-4's expanded knowledge base enables the model to generate, edit and iterate various tasks such as composing songs, writing screenplays or learning a user's writing style.

Enhanced safety and alignment features - GPT-4 is 82% less likely to respond to requests for disallowed content and 40% more likely to produce factual responses than GPT-3.5.

LLaMA: LLM by Meta

LLaMA (Language Model from Meta AI) is a large language model developed by Meta (formerly Facebook) AI Research.

Architecture: LLaMA is based on the transformer architecture, specifically the decoder part, similar to models like GPT-3. It uses self-attention mechanisms to capture long-range dependencies in text.

Open-Source: In a move towards more open and accessible AI research, Meta has released several versions of LLaMA under a permissive license, allowing researchers and developers to study, fine-tune, and build applications with the model.

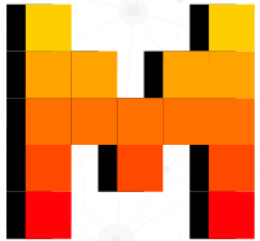
Mistral : Open-source LLMs

Mistral AI is a French company focusing on Artificial Intelligence (AI) and specifically known for its open-source large language models.

Mistral's models are based on a transformer architecture.

These models are completely free to use and come under the permissive Apache 2.0 license.

While many LLMs are only proficient in a single language, most of Mistral's models are natively fluent in English, French, Spanish, German and Italian — meaning they have a more "nuanced understanding" of both grammar and cultural context,



MISTRAL AI_

Application of LLMs

- **Code development**
  Large language models can assist programmers in writing, reviewing, and debugging code.

  Real world app :
  Starcoder, This open-source LLM is trained on a diverse and extensive dataset sourced from GitHub, which includes a wide array of programming languages.

- **Sentiment analysis**
  Large language models applications can be utilized for sentiment analysis.

  Real world app :
  Grammarly uses powerful large language models to understand the context and offer suggestions to enhance writing style, tone, and clarity.

- **Virtual assistant**
  At the core of AI-powered virtual assistants are LLMs that understand and process natural language.

  Real world app :
  Alexa

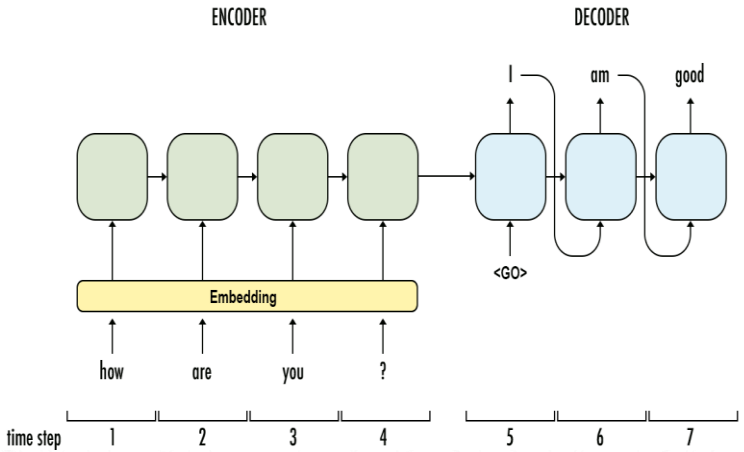
Sequence To Sequence Models

Sequence-to-sequence models

Sequence-to-sequence (seq2seq) models are a type of neural network architecture used for transforming one sequence into another sequence.

A seq2seq model typically consists of two main components:

- **Encoder**: Processes the input sequence and converts it into a fixed-length context vector (also known as a thought vector or embedding).
- **Decoder:** Takes the context vector produced by the encoder and generates the output sequence.



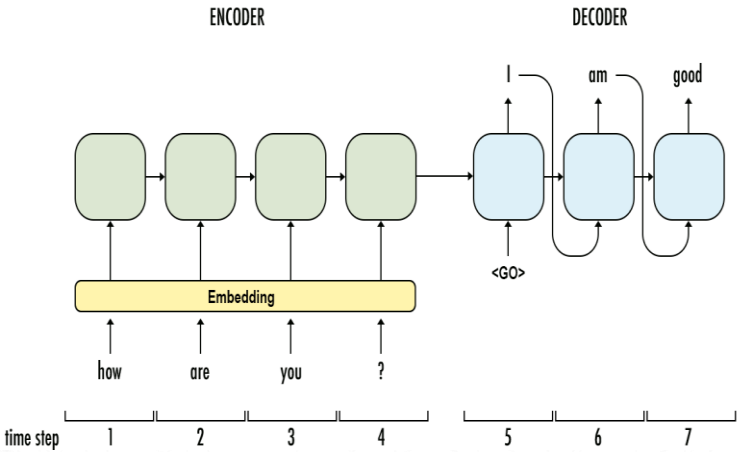
Encoder - Decoder architecture

Encoder:

- Usually a Recurrent Neural Network (RNN) variant, such as LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit).
- Processes the input sequence token by token and updates its hidden state at each step.
- The final hidden state encapsulates the information of the entire input sequence.

Decoder:

- Also typically an RNN variant, LSTM or GRU, but conditioned on the context vector from the encoder.
- Generates the output sequence one token at a time, using the previous token as input at each step.



Attention mechanisms

Need for attention mechanisms *(problems with traditional models)*

Fixed-Size Context Vector: Traditional models like RNNs and LSTMs encode the entire input sequence into a single fixed-size context vector.

As the length of the input sequence increases, it becomes increasingly difficult for the context vector to capture all relevant information

Long-Term Dependencies: Long sequences make it hard for traditional models to maintain information about distant words or tokens.

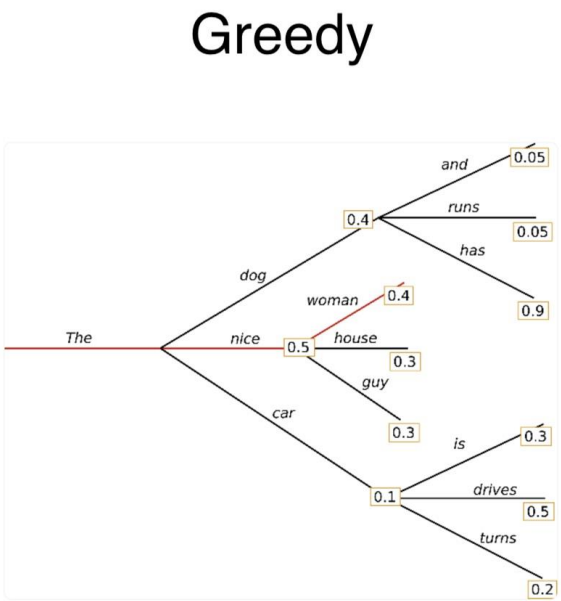
Information Bottleneck: Compressing all the input sequence information into a single vector creates an information bottleneck.

Beam search and greedy decoding

Greedy Decoding

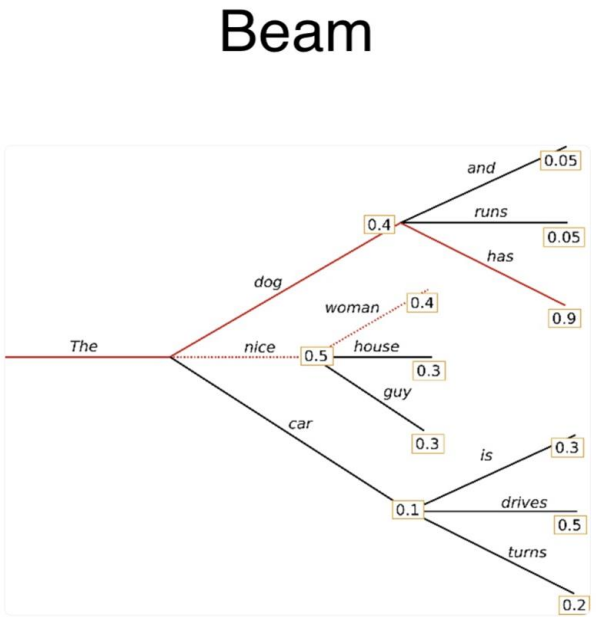
Greedy Decoding is a simple and straightforward approach where, at each step of the sequence generation, the model selects the token with the highest probability as the next token. This process is repeated until the end-of-sequence token is generated or the desired sequence length is reached

The problem with greedy decoding is that it creates suboptimal results because it doesn't take full possibilities into consideration.



How beam search solves the issue ?

Beam addresses this issue by not only looking at the probabilities of the next word, but the next possible chains of words. In the example it detects that 'The dog has' has a high probability than 'The nice woman'. The number of beams defines how far the model should look into the potential futures/next word combinations.

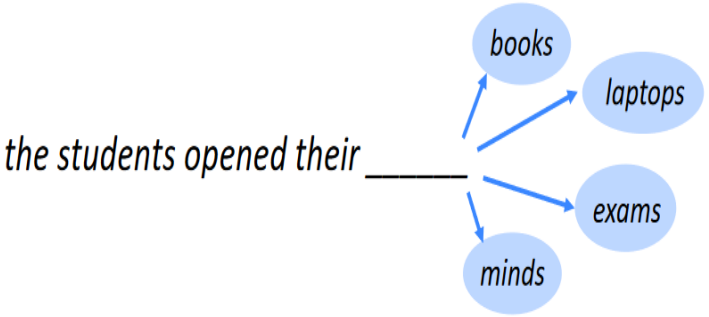


Language Modelling

What is language modeling ?

Language modeling, or LM, is the use of various statistical and probabilistic techniques to determine the probability of a given sequence of words occurring in a sentence.

Language models determine word probability by analyzing text data. They interpret this data by feeding it through an algorithm that establishes rules for context in natural language. Then, the model applies these rules in language tasks to accurately predict or produce new sentences.



Techniques used for language modeling

Some of the techniques for language modeling

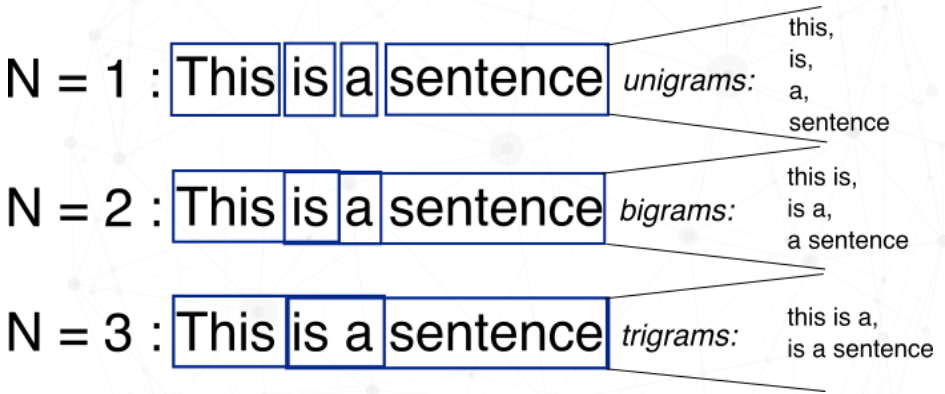
- **N-gram models**
- **Recurrent Neural Networks (RNNs)**
- **Long Short-Term Memory (LSTMs)**
- **Generative Adversarial Networks (GANs)**
- **Variational Autoencoders (VAEs)**

N-Grams

An N-gram is a contiguous sequence of N items from a given sample of text or speech. In language modeling, these items are typically words or characters.

Suppose we have the sentence "I am Sam. Sam I am." The bigrams are:

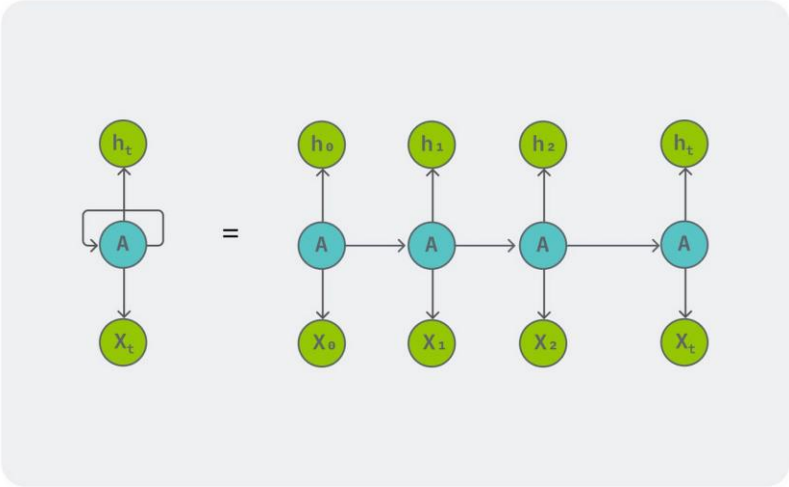
- "I am"
- "am Sam"
- "Sam I"
- "I am"



RNN (Recurrent Neural Networks)

Recurrent Neural Networks (RNNs) are a class of artificial neural networks specially designed to handle sequential data.

Unlike feedforward neural networks, which process all inputs independently, RNNs maintain a memory of past inputs by looping the outputs of neurons back into the network as additional inputs.



Key components of RNN

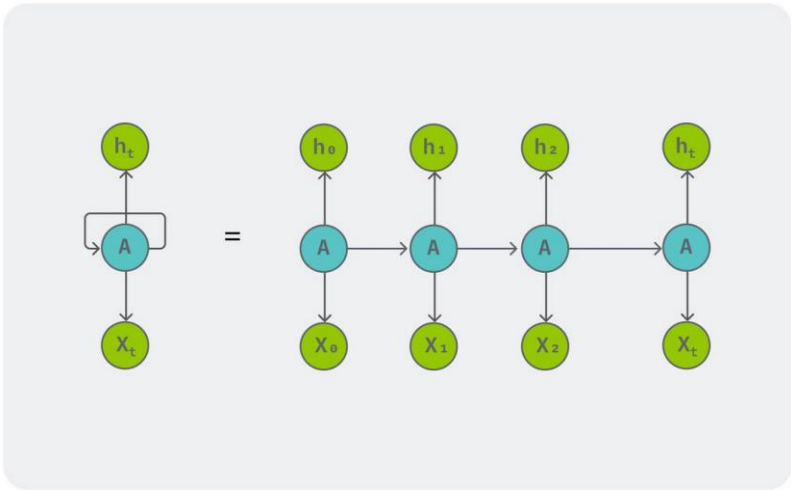
Key Components of RNNs

Recurrent Connections:

- RNNs have recurrent connections that allow information to persist over time.
- Each neuron in the network receives input not only from the current time step but also from the previous time step's output, creating a form of memory.

Hidden State:

- At each time step t, the RNN maintains a hidden state h_t, that represents the network's internal memory.
- The hidden state is updated based on the current input x_t, and the previous hidden state h_{t-1}.



Limitations of RNN

Vanishing Gradient Problem:

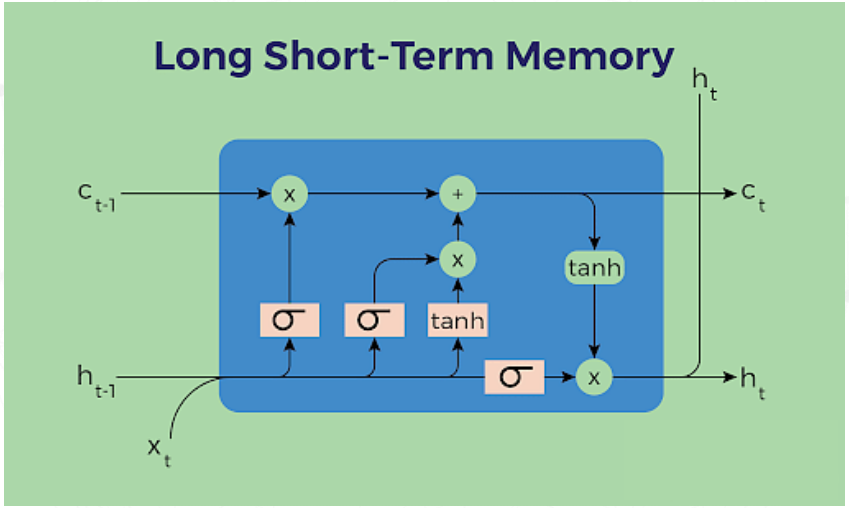
Traditional RNNs are prone to the vanishing gradient problem, where gradients become extremely small as they propagate back through time during training.
Exploding Gradient Problem:

Exploding Gradient Problem:

In addition to the vanishing gradient problem, RNNs can also suffer from the exploding gradient problem, where gradients grow exponentially as they propagate back through time.

LSTM (Long Short Term Memory)

Long Short-Term Memory (LSTM) networks were introduced to address some of the limitations of traditional Recurrent Neural Networks (RNNs), particularly with regards to learning long-term dependencies in sequential data.



Working of LSTM

The LSTM has three gates Forget gate, input gate, output gate.

Forget Gate:

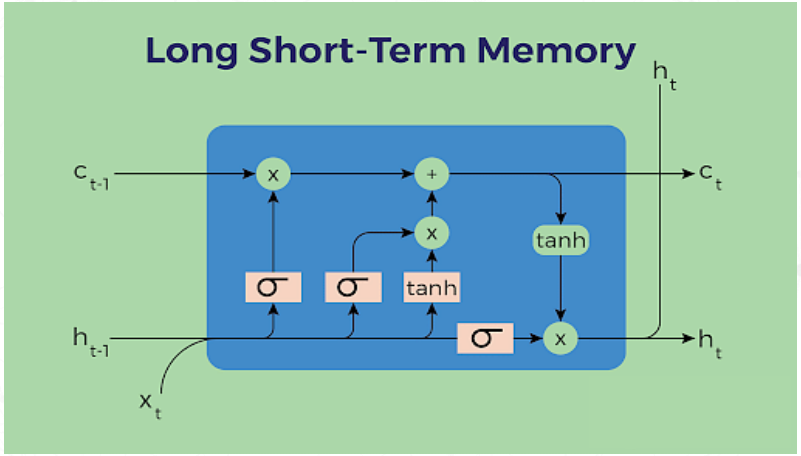
The forget gate determines which information from the previous cell state c_{t-1}, should be retained or discarded.

Input Gate:

The input gate determines what new information should be stored in the cell state.

Output Gate:

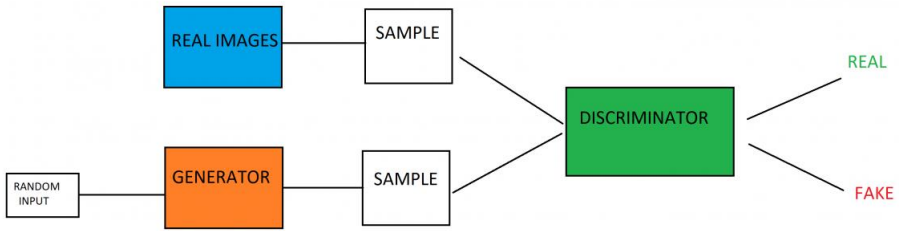
The output gate determines which information from the current cell state c_t should be output to the next time step.



GAN (General Adversarial Network)

GANs are a model architecture for training a generative model.

The GAN model architecture involves two sub-models: a generator model for generating new examples and a discriminator model for classifying whether generated examples are real or fake.



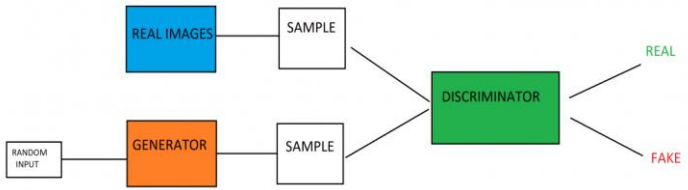
Key components of GAN

Generator:

- The generator's role is to create synthetic data (e.g., images) that resemble the real data.
- It takes a random noise vector (sampled from a simple distribution like Gaussian noise) as input and transforms it into a data sample.

Discriminator:

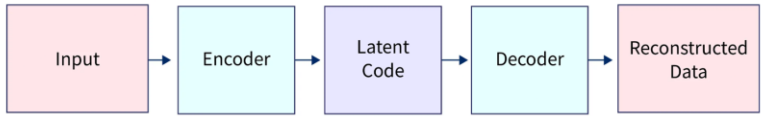
- The discriminator's role is to distinguish between real data (from the actual dataset) and fake data (produced by the generator).
- It outputs a probability value indicating whether a given data sample is real or fake.



Variational Autoencoders

Variational Autoencoders are generative models that learn a dataset's underlying probability distribution and generate new samples.

VAE uses a encoder decoder architecture.



Key components of VAE

Encoder:

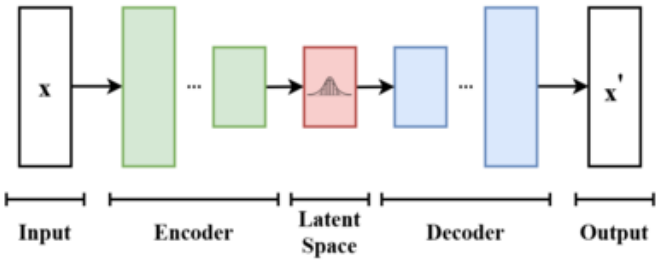
- The encoder network maps input data x to a latent space z using a probabilistic distribution.

Latent Space:

- The latent space is a low-dimensional representation where each point corresponds to a potential data sample.

Decoder:

- The decoder network generates new data samples from points in the latent space.
- It takes samples from the latent space and maps them back to the original data space.

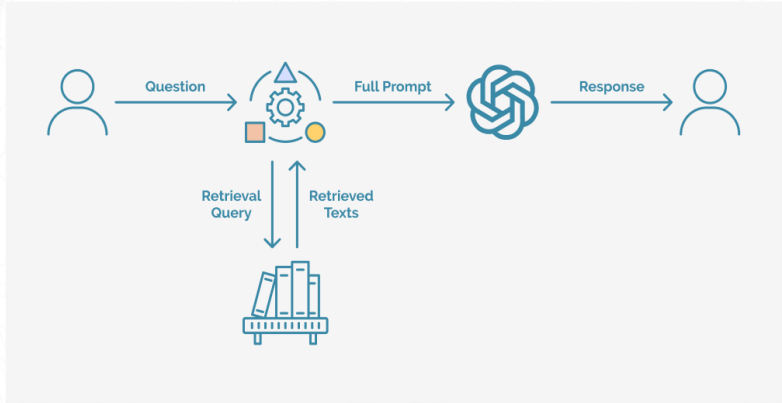


RAG (Retrieval Augmented Generation)

Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is the process of optimizing the output of a large language model, so it references an authoritative knowledge base outside of its training data sources before generating a response.

RAG extends the already powerful capabilities of LLMs to specific domains or an organization's internal knowledge base, all without the need to retrain the model.



Need of RAG

The challenges LLMs face are

- Presenting false information when it does not have the answer.
- Presenting out-of-date or generic information when the user expects a specific, current response.
- Creating a response from non-authoritative sources.
- Creating inaccurate responses due to terminology confusion.

RAG is one approach to solving some of these challenges. It redirects the LLM to retrieve relevant information from authoritative, pre-determined knowledge sources.

Working of RAG

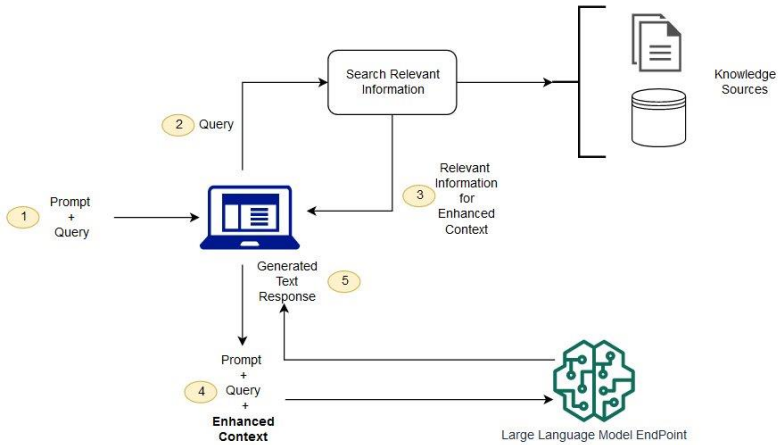
With RAG, an information retrieval component is introduced that utilizes the user input to first pull information from a new data source. The user query and the relevant information are both given to the LLM. The LLM uses the new knowledge and its training data to create better responses.

Step by Step process of how RAG works

1. **Create external data:** External data beyond the language model's original training set is stored in a vector database using embedding techniques.

1. **Retrieve relevant information:** When a user provides a query, it is converted into a vector representation and matched against the vector database to retrieve the most relevant documents/passages.

1. **Augment the LLM prompt:** The user's original query is augmented by appending the relevant retrieved information. This augmented prompt provides additional context to the large language model.



Working of RAG

4. **Generate output:** Finally, the language model generates an output response conditioned on the augmented prompt containing both the original query and the retrieved relevant information.

Neural Retrieval models

When you pass a query to a neural retrieval model (NRM), it will attempt to retrieve and return the most relevant documents or passages from the corpus that the model was trained on.

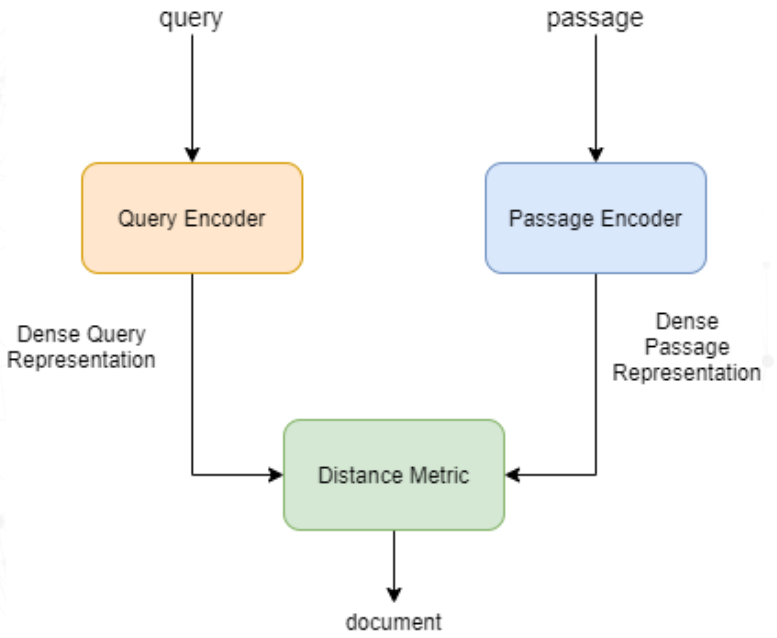
Here's how it typically works:

- The user provides a query or information need, which is text input representing what they are searching for.
- The neural retrieval model encodes the query text into a dense vector representation or embedding using its neural network architecture (e.g. BERT).
- The model then computes the similarity between the query embedding and the pre-computed embeddings of all the documents/passages in its corpus.
- Based on this similarity scoring (e.g. cosine, dot-product), the model ranks all the documents by their predicted relevance to the query.
- The top-k most relevant documents according to the model's scoring are returned as the retrieval results.

Some Neural Retrieval models

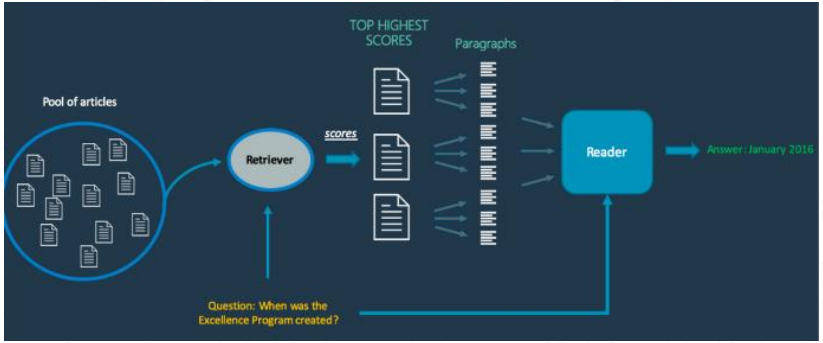
Dense Passage Retriever (DPR):

- DPR is a popular bi-encoder model that was pre-trained on a large question-answering dataset to produce high-quality dense vector representations of questions and passages.
- It uses two separate BERT encoders - one for encoding questions and another for encoding passages.
- During retrieval, the dot product between the question and passage vectors is used to score relevance.
- DPR is efficient for retrieving from large corpora and has been widely used in many retrieval-augmented systems.



ColBERT:

- ColBERT (Contextualized Late Interaction over BERT) is a late interaction model that combines the benefits of bi-encoders and cross-encoders.
- It uses a lightweight token-level interaction model on top of a pre-trained bi-encoder.
- This allows it to effectively model fine-grained interactions between query and document while still being efficient for retrieval over large corpora.
- ColBERT has shown strong performance on retrieval tasks like MS MARCO.



RAG for question answering and information synthesis

Retrieval-Augmented Generation (RAG) models have shown particular effectiveness for question answering and information synthesis tasks that require retrieving and reasoning over relevant evidence from large corpora.

RAG models can be applied for these use cases:

Open-Domain Question Answering:

- Given a natural language question, the retrieval component of RAG first retrieves relevant passages or documents from a large corpus like Wikipedia.
- The retrieved evidence is then provided as additional context along with the question to a pre-trained language model like BART or T5.
- The language model generates the final answer by conditioning on both the question and the retrieved relevant information.

RAG for question answering and information synthesis

Multi-Document Summarization:

- For summarizing information spread across multiple documents, the retrieval component can retrieve clusters of relevant documents for the given topic.
- These retrieved documents act as the evidence set which is passed to the generation component.
- The language model then synthesizes and generates a coherent summary by abstracting and fusing the key information from the multiple input documents.

Langchain

What is Langchain?

- LangChain is a framework for developing applications powered by language models.
- LangChain is a framework designed to simplify the creation of applications using large language models (LLMs)
- It provides a set of tools and libraries that make it easy to connect LLMs to other systems and data sources.

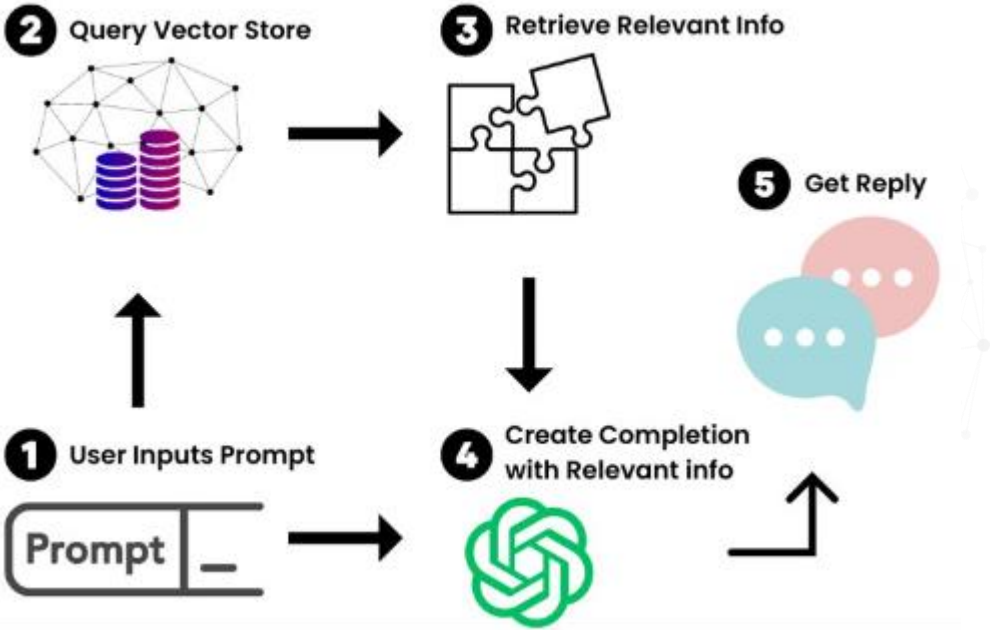
How Langchain works?

LangChain isn't complicated at all!!

In short, LangChain just composes large amounts of data that can easily be referenced by a LLM with as little computation power as possible. It works by taking a big source of data, take for example a 50-page PDF, and breaking it down into "chunks" which are then embedded into a Vector Store.

Open AI + Langchain

When we insert a prompt into our new chatbot, LangChain will query the Vector Store for relevant information. Think of it as a mini-Google for your document. Once the relevant information is retrieved, we use that in conjunction with the prompt to feed to the LLM to generate our answer.

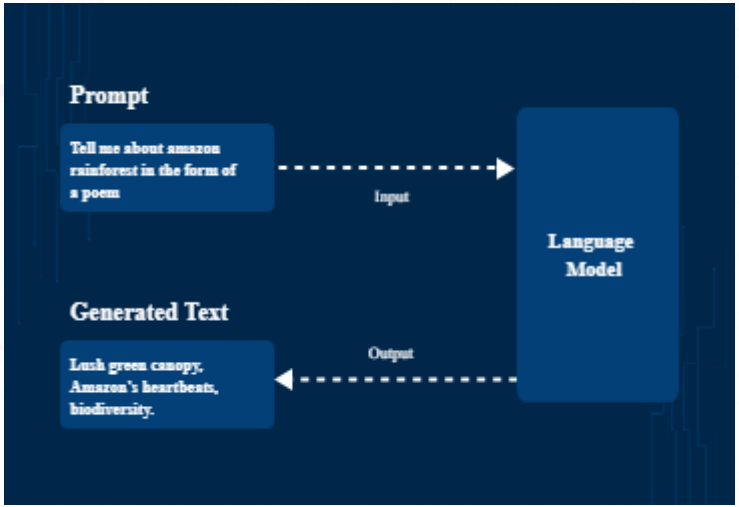


Prompt engineering

What is prompt engineering ?

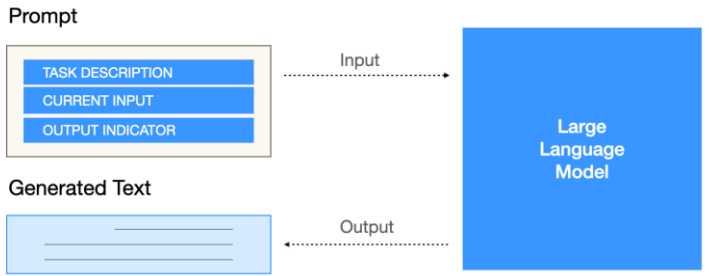
Prompt engineering is the practice of designing and refining prompts—questions or instructions—to elicit specific responses from AI models.

It is the process where you guide generative artificial intelligence (generative AI) solutions to generate desired outputs.



What is a prompt?

- A prompt is a natural language text that requests the generative AI to perform a specific task.
- Generative AI systems require context and detailed information to produce accurate and relevant responses.
- When you systematically design prompts, you get more meaningful and usable creations.



prompt engineering techniques

Chain-of-thought prompting

Chain-of-thought prompting is a technique that breaks down a complex question into smaller, logical parts that mimic a train of thought. This helps the model solve problems in a series of intermediate steps rather than directly answering the question.

You can perform several chain-of-thought rollouts for complex tasks and choose the most commonly reached conclusion.

For example, if the question is "What is the capital of France?" the model might perform several rollouts leading to answers like "Paris," "The capital of France is Paris," and "Paris is the capital of France." Since all rollouts lead to the same conclusion, "Paris" would be selected as the final answer.

prompt engineering techniques?

Tree of Thoughts

ToT maintains a tree of thoughts, where thoughts represent coherent language sequences that serve as intermediate steps toward solving a problem.

This approach enables an LM to self-evaluate the progress through intermediate thoughts made towards solving a problem through a deliberate reasoning process.

The LM's ability to generate and evaluate thoughts is then combined with search algorithms (e.g., breadth-first search and depth-first search) to enable systematic exploration of thoughts with lookahead and backtracking.

prompt engineering techniques

Complexity-based prompting

This prompt-engineering technique involves performing several chain-of-thought rollouts. It chooses the rollouts with the longest chains of thought then chooses the most commonly reached conclusion.

For example, if the question is a complex math problem, the model might perform several rollouts, each involving multiple steps of calculations. It would consider the rollouts with the longest chain of thought, which for this example would be the most steps of calculations. The rollouts that reach a common conclusion with other rollouts would be selected as the final answer.

prompt engineering techniques

Generated knowledge prompting

This technique involves prompting the model to first generate relevant facts needed to complete the prompt. Then it proceeds to complete the prompt.

This often results in higher completion quality as the model is conditioned on relevant facts.

For example, imagine a user prompts the model to write an essay on the effects of deforestation. The model might first generate facts like "deforestation contributes to climate change" and "deforestation leads to loss of biodiversity." Then it would elaborate on the points in the essay.

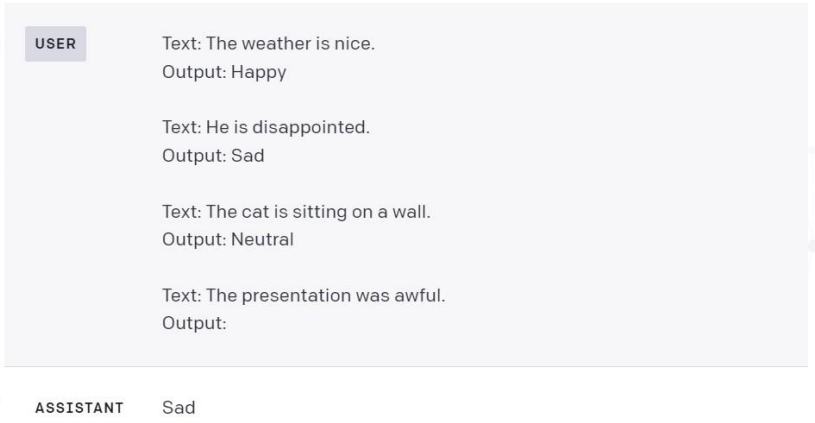
Few-shot Prompting

Few-shot Prompting

Few-shot prompting can provide demonstrations to steer the model to better performance. It is a technique for providing LLMs with a few examples of the desired output, in addition to the prompt.

The examples help the model to better understand the task and to generate more accurate and informative responses. If We provide vast and different examples to the model, instead of multiple similar examples, It ensures the model learns as much as possible about the task.

Standard few-shot prompting is a good technique for many tasks, but not reliable for complex reasoning tasks. Therefore, more advanced prompting techniques, such as chain-of-thought, active prompting, and fine-tuning, are needed.



Applications of Generative AI

Generating text

AI systems, particularly large language models, generate text through a process called language modeling.

The main types of models used for generating text with AI are large language models, especially transformer-based models like:

- GPT (Generative Pre-trained Transformer) models: GPT, GPT-2, GPT-3 by OpenAI
- BERT (Bidirectional Encoder Representations from Transformers) and variants like RoBERTa, ALBERT
- T5 (Text-to-Text Transfer Transformer) by Google

They are first pre-trained on massive text corpora (billions of words/parameters) using self-supervised objectives like masked language modeling and next token prediction. This allows them to learn robust representations of language.

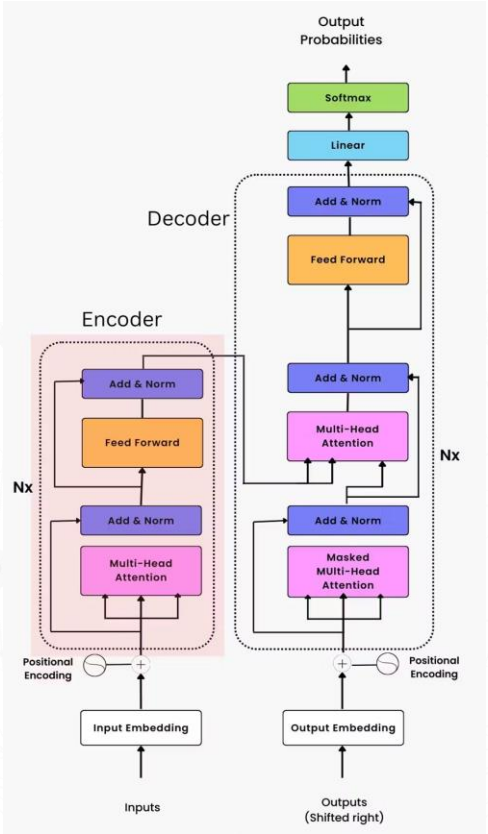


Image generation

Image generation has a wide range of applications across various industries and domains, driven by advancements in AI and deep learning.

Models like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are widely used for image generation.

Image generation models are used in:

Entertainment and Media
Special Effects in Movies and TV: AI-generated images and animations enhance visual effects, creating realistic and imaginative scenes that would be difficult or costly to produce otherwise.

Augmented Reality (AR) and Virtual Reality (VR):
Enhanced educational tools through realistic and interactive simulations.

Audio and music generation

Generative AI has made significant strides in generating audio and music, offering innovative tools and techniques for creating high-quality, diverse, and original audio content.

Music Composition and Generation AI Models:

Recurrent Neural Networks (RNNs): Especially Long Short-Term Memory (LSTM) networks, are used for sequential data like music.

Transformers: Models like OpenAI's MuseNet and Google's Music Transformer handle long-range dependencies in music better than RNNs.

Variational Autoencoders (VAEs) and GANs: Used for generating new pieces of music by learning latent representations.

Multimodal models

Multimodal models are a class of AI models that can process and generate data across multiple modalities, such as text, images, audio, and video. These models are particularly useful for tasks that involve multiple modalities, such as image captioning, video description, and multimodal question answering.

Examples of how generative AI models are used in multimodal settings:

Text-to-Image Generation: Models like DALL-E, Stable Diffusion, and Imagen can generate realistic images from text descriptions.

Image Captioning: Generating natural language descriptions of images.

Video Description: Generating descriptions or summaries of videos by understanding the visual and audio content.

Ethical Considerations and Limitations

Ethical Implications of Generative AI

1. Misinformation And Deepfakes

- Generative AI's capacity to produce content that blurs the lines between reality and fabrication is alarming.
- From synthetic news reports to manipulated videos, these creations can distort public perception, fuel propaganda and detrimentally impact both individuals and organizations.

2. Bias And Discrimination

- Generative models mirror the data they're fed. Consequently, if they're trained on biased datasets, they will inadvertently perpetuate those biases.

3. Copyright And Intellectual Property

- The ability of generative AI to craft content that mirrors existing copyrighted materials poses significant legal concerns.
- Intellectual property infringements can result in costly legal battles and reputational damage.

4. Privacy And Data Security

- Generative models, particularly those trained on personal data, pose privacy risks. The unauthorized use of this data or the generation of eerily accurate synthetic profiles is a significant concern.

Privacy concerns and bias in AI models

AI Privacy Breach: Occurs when AI systems improperly store, access, and exploit data without user permission, potentially including sensitive information like health records and financial data.

Consent and Data Collection: AI's data gathering techniques raise questions as it acquires vast amounts of data, including billions of photographs, without explicit consent, sparking concerns about privacy rights and ethical data collection.

Surveillance and Civil Liberties: Clearview AI's technology raises concerns about surveillance and civil liberties due to its extensive database and widespread deployment, leading to worries about privacy infringement and the creation of a surveillance state.

Racial Bias: Independent investigations reveal significant racial bias in Clearview AI's face recognition algorithms, resulting in higher false positive rates for individuals with darker skin tones, raising concerns about potential prejudice and its impact on law enforcement tactics.

Gender Bias: Clearview AI's facial recognition algorithms exhibit gender bias, frequently misidentifying individuals based on gender presentation, exacerbating concerns about fair and unbiased identification.

Future trends in Generative AI

Emergence Of Multimodal AI Models

The technology goes beyond text with multimodal AI models, allowing users to mix and match content based on text, audio, image, and video for prompting and generating new content.

Capable And Powerful Small Language Models

Small language models, on the other hand, are trained on more limited datasets that are nonetheless comprised of high-quality sources such as textbooks, journals, and authoritative content.

These models are smaller in terms of parameter count as well as storage and memory requirements, allowing them to run on less powerful and less expensive hardware.

The Rise Of Autonomous Agents

Multimodal AI, which combines various AI techniques such as natural language processing, computer vision, and machine learning, is critical in the development of autonomous agents.

Frameworks such as LangChain and LlamaIndex are some of the popular tools used to build agents based on the LLMs.

Use Cases

Architecture