Advanced Concepts & Optimization Techniques for LLMs

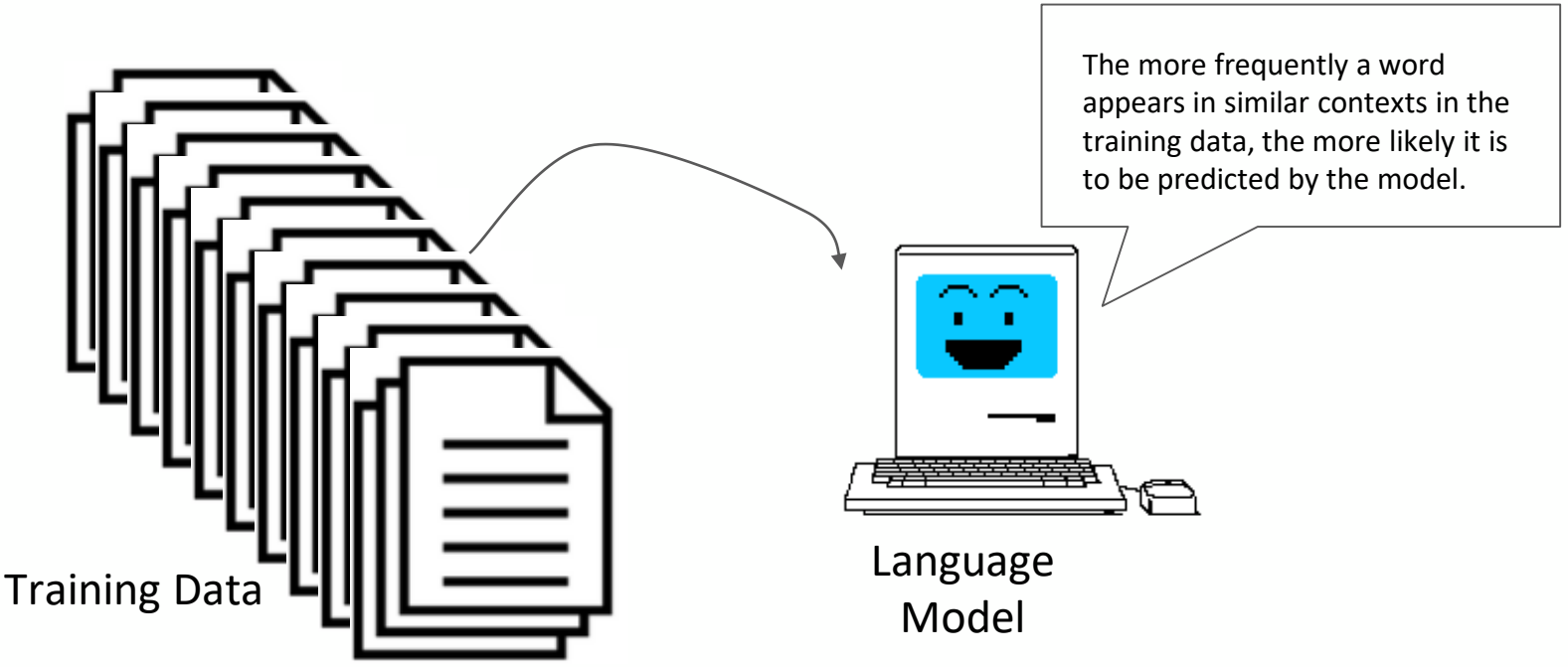
SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority

OUTLINE

- LLMs: Most Popular Applications
- Pre-training and Fine-tuning (LMPT)
- Transformer Architecture
- The Attention Mechanism
  - Deep dive into Attention and Self-attention
  - Multi-Head & Masked Attention
- LLMs and Large Scale Data

LLMs: The Most popular Applications

Recall: LLMs


The more frequently a word appears in similar contexts in the training data, the more likely it is to be predicted by the model.
Training Data
Language Model

Applications of Language Models

The applications of large language models includes:

- Generating text
- Summarizing text
- Rewriting text
- Searching the internet/corpus
- Question Answering
- Clustering text
- Classifying text

Text Generation

Generation of text based on the prompt/input from the user, which includes:

- Chatbots
- Code Assistants
- Branding Generation
- Image/Video/Animation generation
- Transcription

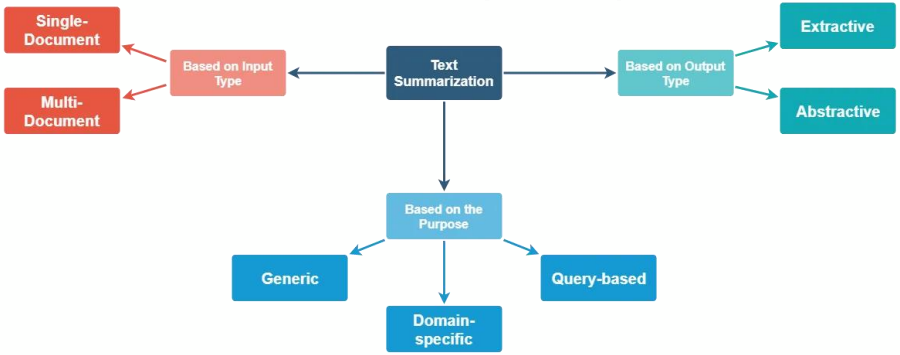
The most notable commercial adaptations for text generation:

- ChatGPT
- Bard
- Dall-E

Summarizing Text

Large language models excel in summarizing text. You can also instruct the model on what to focus on. There are two types of summarization:

1. Extractive:novel text is generated to represent the information contained in longer content.
2. Abstractive: relevant facts retrieved based on a prompt are extracted and summarized into a concise response/answer.



Rewriting text

This is where the content of the input text is not changed, but rather its format. This can include:

- Spelling/Grammar correction
- Language Translation

Popular tools for this use case include:

- Grammarly
- Google Translate
- Cohere Generate

Searching Internet/Corpus

In this use case, we send the model a prompt request that mimics a database search. The model then searches either its previous knowledge from its training corpus or does an internet search to give you the results.

Popular tools for this use case include:

- Jina – neural search platform that provides prompt optimization and decision support capabilities
- Vectara – LLM-powered search platform which matches data based on intent and meaning, regardless of how the concepts are worded.
- You.com – a search engine that leverages LLMs to help make users' search activities more efficient

Question Answering

This method is a combination of two services. The first part understands the question's context and the second part summarizes it for the "searching" part of the answering becomes faster and more concise.

Common tools for this use case:

- Google Search, Bing Search – both of these regularly attempt to provide a summarized answer at the top of a list of search results
- Vectara – retrieval of relevant information based on the user's query/prompt, which is then summarized to provide an answer with citations
- LLaMA – focused especially on question answering and document summarization

Clustering Text

In this process, unsupervised learning techniques are used to group together text based on certain criteria such as topic, length, genre,...

Text clustering can be used in several applications such as:

1. **Information retrieval**: Clustering can be used to group similar documents together, making it easier to find relevant information.
2. **Topic modelling**: Clustering can be used to find hidden topics in text documents, which can then determine how the data is organised.
3. **Language model improvement**: clustering can be used to group text documents with similar topics or writing styles, which can then be used to improve language models.

Text Classification

Text classification is a supervised learning approach that deals with categorizing text depending on the application. The most popular applications of text classification:

1. **Semantic Analysis:** a type of classifier used for understanding if a given text is talking positively or negatively about a given subject.
2. **Language Detection**: Detects the language of an input text string.
3. **Spam Detection**: Detects if an incoming message/email/text is spam or from a scammer.
4. **Named Entity Recognition (NER)**: locate and classify named entities mentioned in unstructured text into pre-defined categories such as person names, organizations, locations, medical codesز

Language Model Pre-training and Fine-tuning (LMPT)

Language Model Pre-training

Pre-training is the first phase of learning for language models. The weights would still be random. It is:

Unsupervised Training

Using a large corpus

Helps the model perform well in the main tasks

Language Model Fine-tuning

The process of re-training a pre-trained model for a specific task. This is referred to as “domain adaptation”.

SFT* or RLHF**

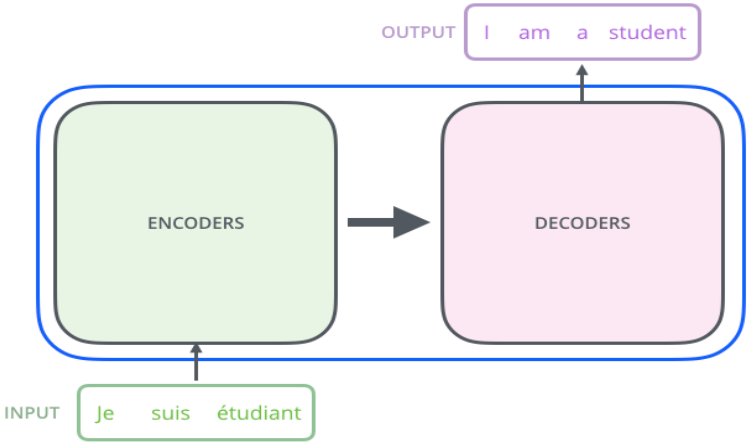
Using a smaller, task-specific corpus

*SFT: Supervised fine-tuning
**RLHF: Reinforcement learning with human feedback.

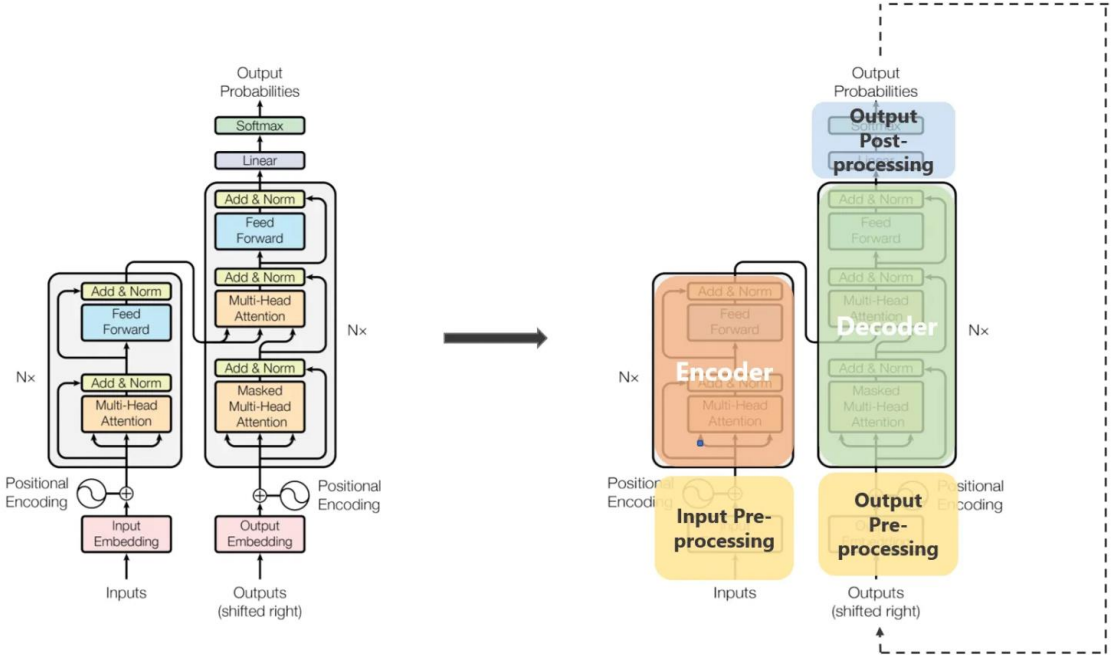
Transformer Architecture

What are transformers?

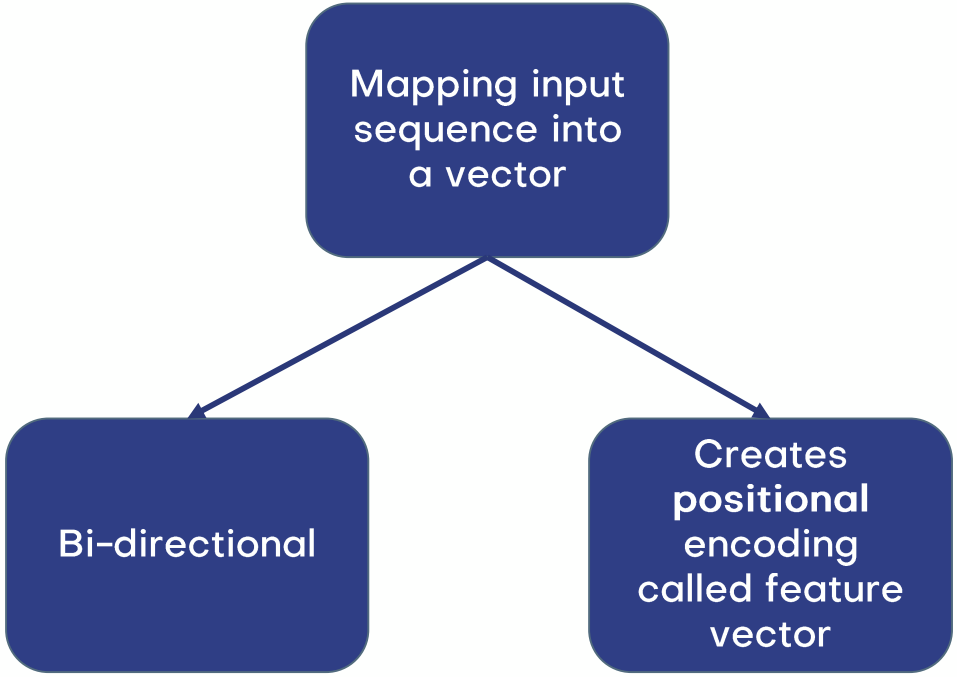
Transformers are a novel architecture that aims to solve sequence-to-sequence tasks while easily handling long-distance dependencies. They rely on attention mechanisms without using sequence-aligned RNNS.



Encoder-Decoder Architecture



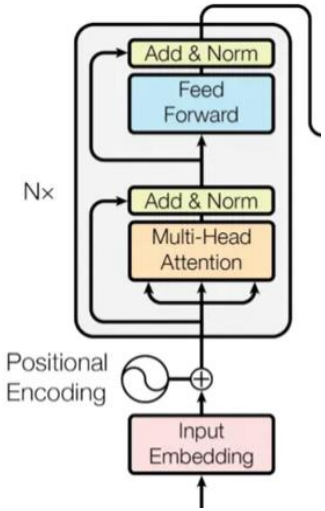
Encoder Stack: Definition



Encoder Stack: Deep Dive

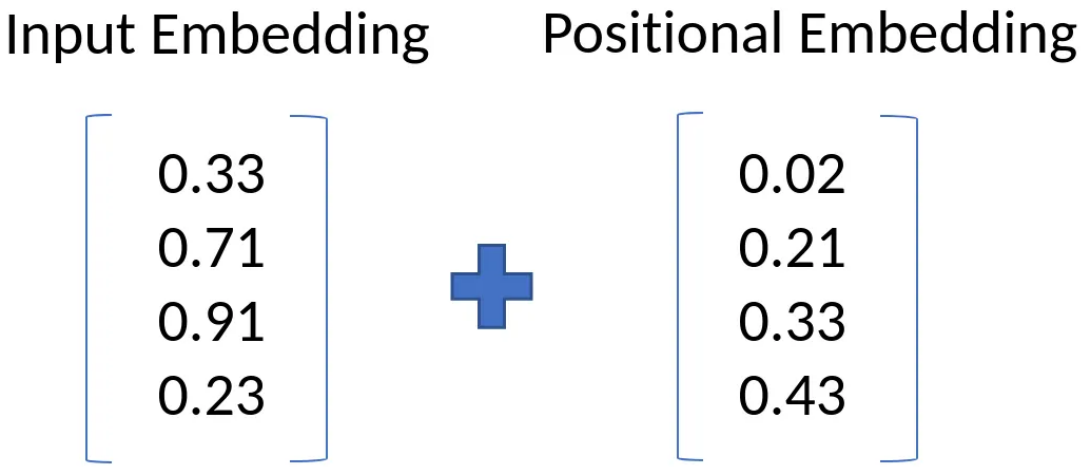
Each encoder has two sub-layers.

1. **A multi-head self attention mechanism** on the input vectors.
2. A simple, position-wise **fully connected feed-forward network**



Positional Embedding

After the initial vectorization, the embeddings do not hold any context. Thus, we add a positional encoding to make the model learn the context of the words.

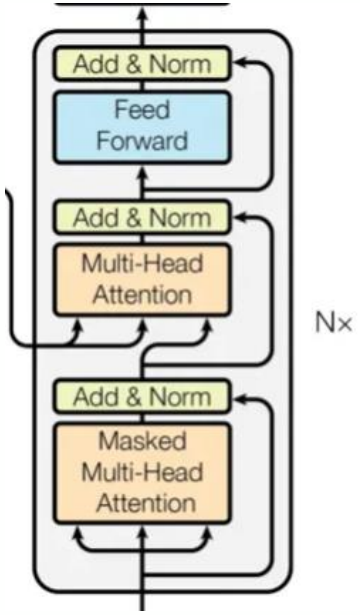


Decoder Stack: Overview

Takes the feature vector as an input and generates an output sequence vector. Decoders are unidirectional (only to the left, or only to the right).

Decoder Stack: Deep Dive

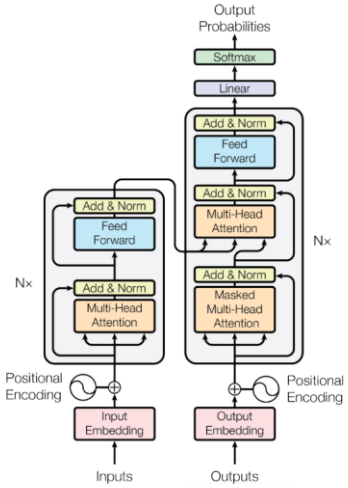
1. A ***masked*** **multi-head self attention mechanism** on the output vectors of the previous iteration.
2. A **multi-head attention mechanism** on the output from encoder and masked multi-headed attention in decoder.
3. A simple, position-wise **fully connected feed-forward network**



Architectures that use One State only

BERT

Encoder

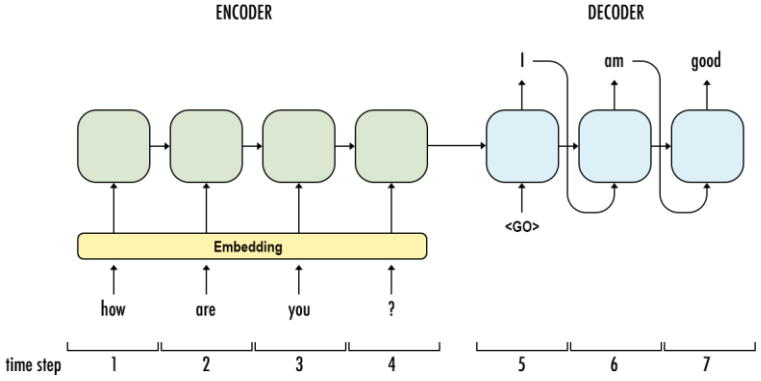


GPT

Decoder

Advantages of Transformer Architecture

1. Long training and inference time for Seq2Seq models:
   A. Training time: Size of seq2seq models depends on the input size, so they can get quite big!
   B. Inference time: Sequence models do not allow parallel computation from the nature of their architecture. They need to perform all calculations and “sequentially”.



The Attention Mechanism

What is Attention?

- “A neural network is considered to be an effort to mimic human brain actions in a simplified manner.
- Attention Mechanism is also an attempt to implement the same action of selectively concentrating on a few relevant things, while ignoring others in deep neural networks.”

Why do we need Attention?

First introduced in 2014, attention came in as a solution to many shortcomings from previous state-of-the-art solutions like RNNS, LSTMs.

Inductive Bias

Memory Bottleneck of Seq2Seq models

Get an alignment map between inputs and outputs

Why do we need attention?

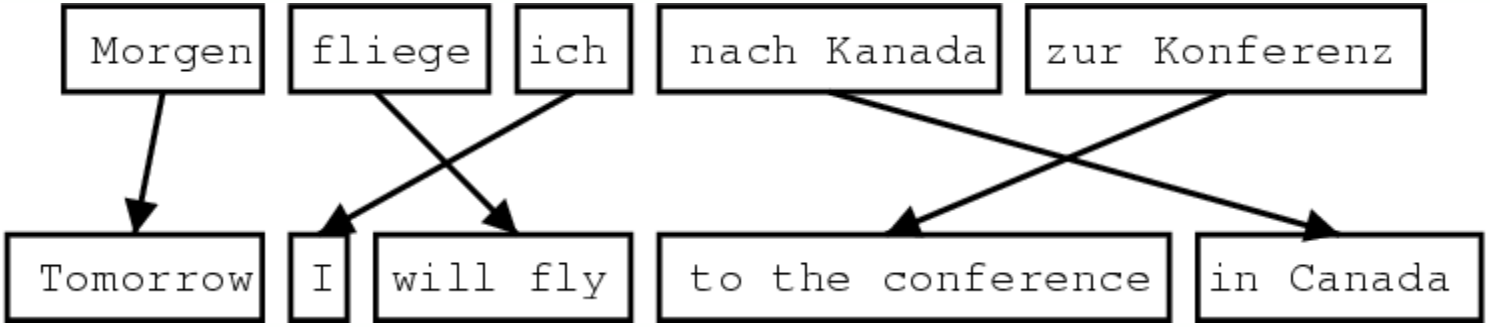
1. Inductive bias: What is it?

In the quaint town of Willowbrook, where dreams were said to come true, a young girl named Lily discovered an old, dusty book hidden in her attic. As she opened its pages, she was transported to a magical world where talking animals and enchanted forests awaited her every step, and her adventures were just beginning.

- How far back do we need to go to understand the context?
- Is it sufficient to go through the text left to right or should we go right to left?
- To summarize the text, should we look into semantic relationships between the tokens or positional ones?
- Should all tokens of the text be treated the same?

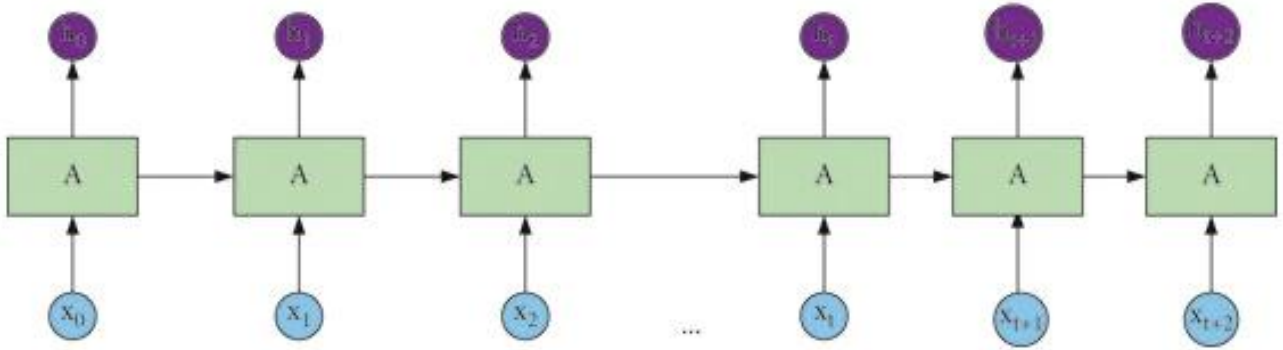
Why do we need Attention?

1. Inductive Bias: In RNNs and LSTMs:
   RNNs and LSTMs are sequence-to-sequence models. This means that they “learn” the input one after the other (regardless of the direction). This creates an inductive bias.



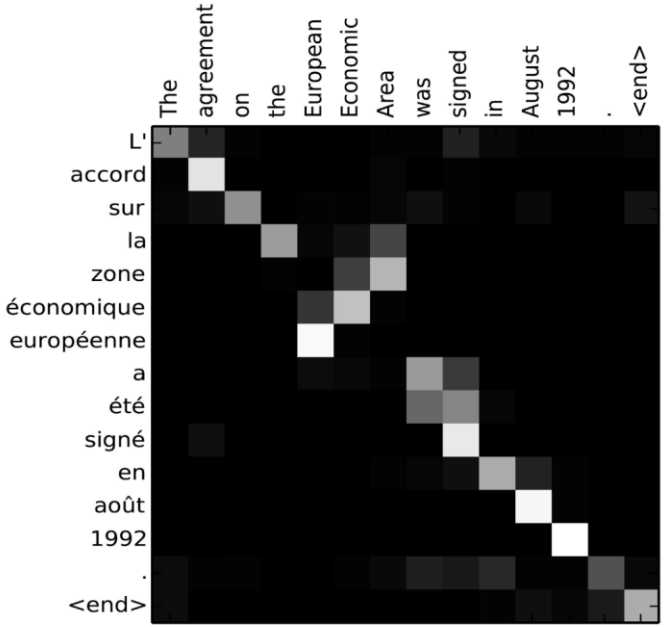
Why do we need Attention?

2. Memory bottleneck of seq2seq models:
   RNN models are only able to handle short-range dependencies. This means that tokens that are “far away” will not be taken into account when learning the context of a token.



Why do we need Attention?

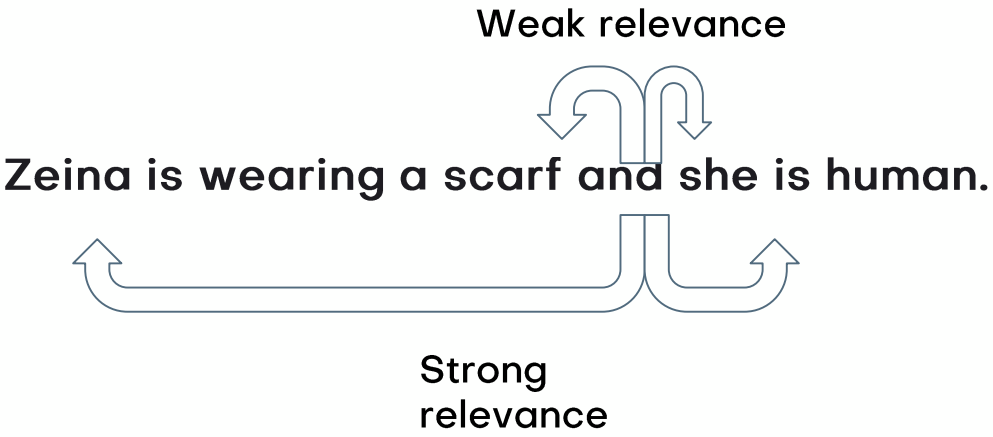
3. Alignment Maps:



Deep dive into Attention and Self-Attention

How does Attention work?

Let us take an example:



How does Attention work?

Step 1: Vectorization

Zeina is wearing a scarf and she is human.
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

The text is made up of 9 tokens t_0 to t_8. The first step is to create an initial embedding of the words that contain positional and semantic context. This process is called vectorization.

$$t_0, t_1, \ldots, t_n \longrightarrow V_0, V_1, \ldots, V_n$$

How does Attention work?

Step 2: Creating Weights

In order to introduce context to our vectors V, we get the dot product of the vector V_i with each of the vectors including itself.

$$V_0, V_1, V_2, V_3, \ldots, V_n$$

$$. \quad . \quad . \quad . \quad .$$

$$V_0, V_0, V_0, V_0, \ldots, V_0$$

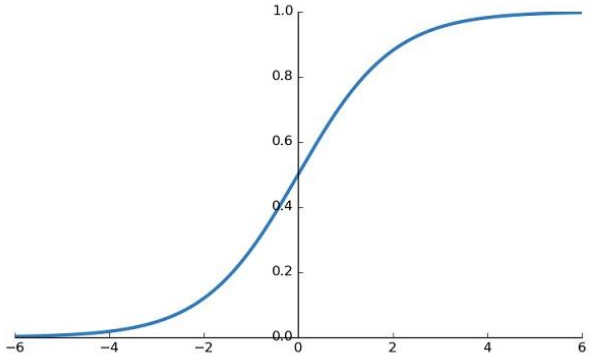
$$=$$

$$W_{00}, W_{01}, W_{02}, W_{03}, \ldots, W_{0n}$$

How does Attention work?

Step 3: Normalization of the similarity scores (weights)

Now that we have similarity scores : W_{00}, W_{01}, W_{02}, W_{03}, …, W_{0n} , we need to normalize them to control the major differences in value using softmax function which normalizes the values and keeps them between 0 and 1.



How does Attention work?

Step 4: Contextualize the initial embeddings

Then, we multiply the initial embedding vectors by the similarity scores after normalization:

$$V_0, V_1, V_2, V_3, \ldots, V_n$$
$$x \quad x \quad x \quad x \quad x$$
$$W_{00}, W_{01}, W_{02}, W_{03}, \ldots, W_{0n}$$

How does Attention work?

Step 5: Final embedding Y

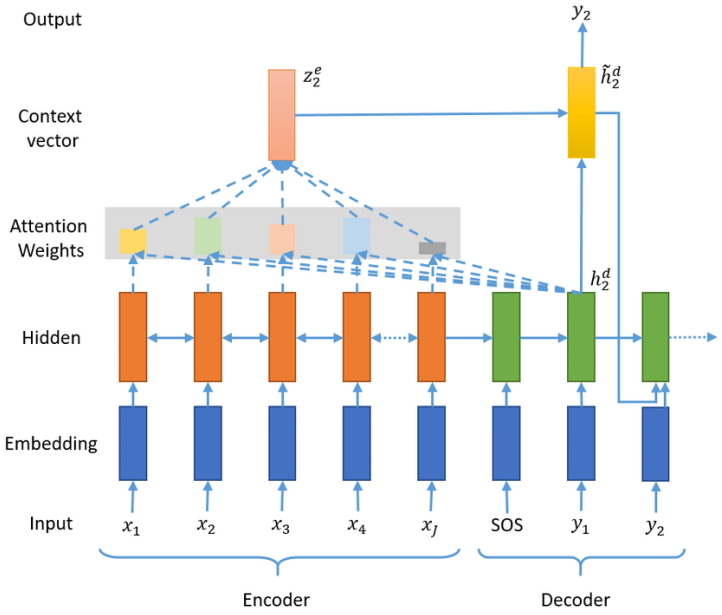
After obtaining all the weighted vectors from multiplying all our initial vectors V with the normalized similarity score W, we sum up the weighted vectors to obtain a final vector Y

$$V_0 W_{00} + V_1 W_{01} + V_2 W_{02} + V_3 W_{03} + \ldots + V_n W_{0n} = Y_0$$

This vector *Y*, then replaces the vector *V* by being the embedding for each token *t.*

How does Attention work?

This approach of adding some context to the words in a sentence is known as **Self-Attention.**



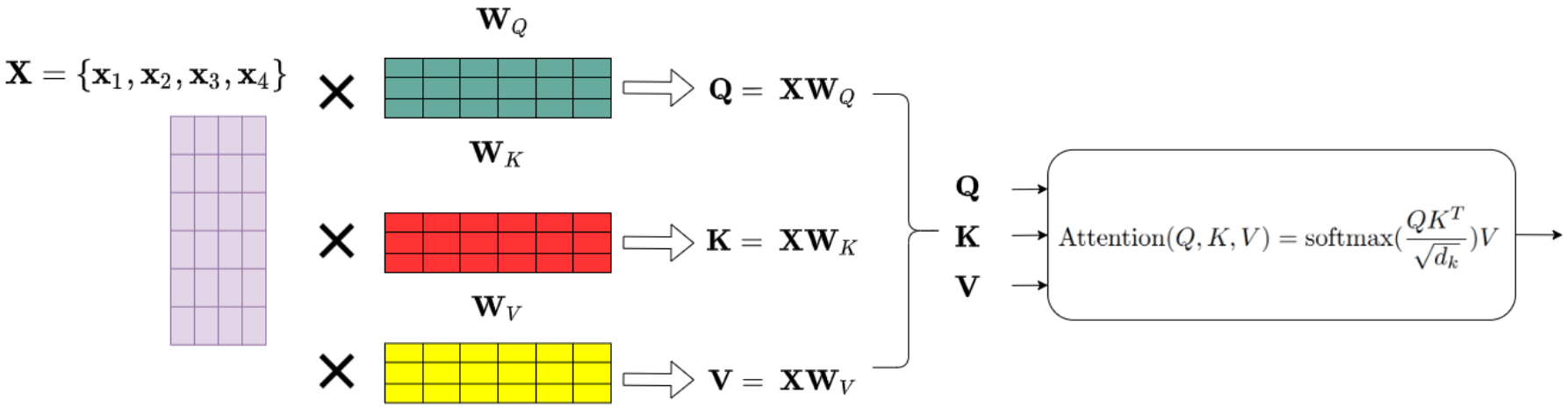
Self-Attention vs Attention

Self-attention helped us in contextualizing the embeddings of the corpus, and attention will help the model focus on which embedding is most important to each token. Thus, we introduce Query, key, and values.

1. Query: The initial embedding *V* of the token being taken.
2. Key: All the initial embeddings of the tokens V_0, V_1, V_2, V_3, …, V_n
3. Value: The final embeddings of the tokens after being weighted (Y)

Self-Attention vs Attention

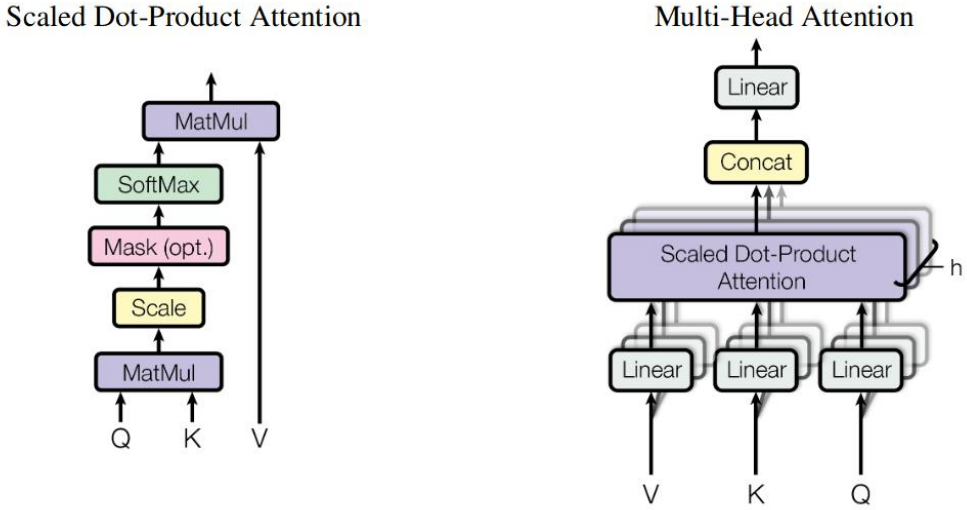
We attain the Query, Keys, and Values matrices by multiplying the original vectors we have with Mq (Query Matrix), Mk (Key Matrix), and Mv (Values matrix)



Multi-Head & Masked Attention

Multi-Head Attention

The attention mechanism allows the model to learn the similarities between the different vectors. To understand different combinations of similarities (semantic, positional,..) we can calculate different similarities, which are **scaled**



Masked Multi-head Attention

This is the type of attention that some decoders use. It uses a mask to hide the words on either side of the token being encoded and makes the decoder try to guess the word from both directions. This makes decoders exceptionally good at text generation.

$$\text{Attention}(Q,K,V) = \text{SoftMax}\left(\frac{Q^T K}{\sqrt{d_k}}\right) V$$

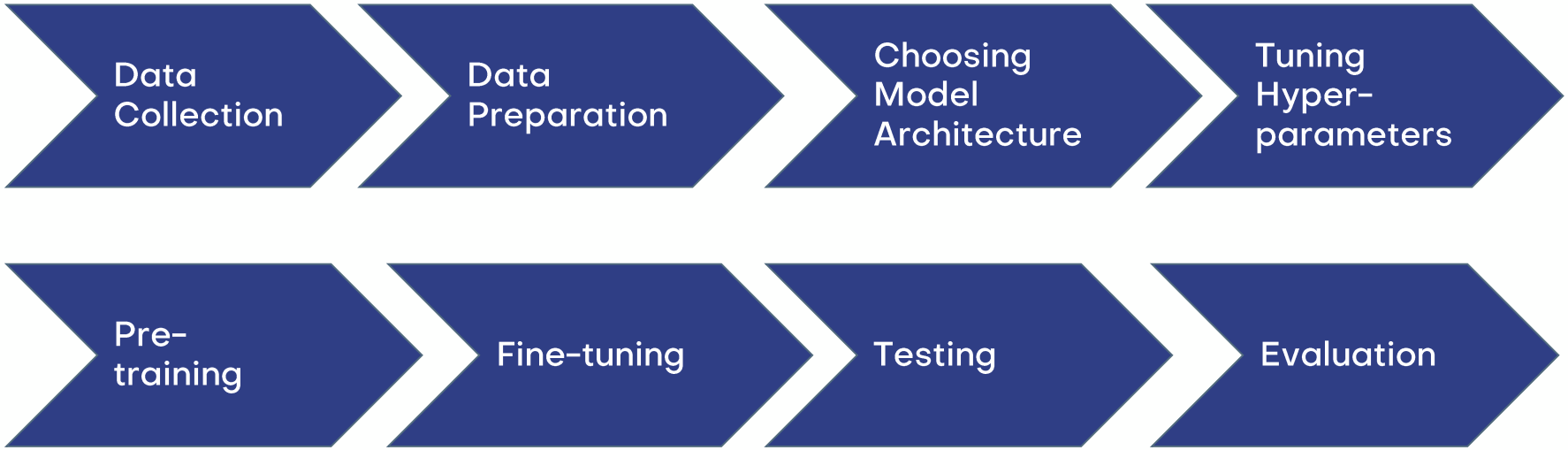
$$\text{Masked Attention}(Q,K,V) = \text{SoftMax}\left(\frac{Q^T K + M}{\sqrt{d_k}}\right) V$$

where M is a mask matrix of 0's and -∞'s

Hands-on: Comparing Text Summarization Pipelines

LLMs and Large Scale Data

Training LLMs on Large-Scale Data



Data Collection

When data is of huge amounts, we call it big data. Collection of big data can be from multiple resources. In big data, quality is just as important as quantity- garbage in garbage out!

Data Preparation

Preparation of data for LLMs is similar to data preparation for smaller NLP models, but when we have huge amounts of data, we have come extra steps:

1. Big Data preprocessing steps:
    a. Privacy Control
    b. Removing Junk Data
    c. Decontamination
    d. Toxicity and Bias Control
2. NLP preprocessing steps:
    a. Deduplication
    b. Tokenization
    c. Character-level cleaning

Model Architecture & Hyperparameter Tuning

Depending on your task, choose the architecture that aligns with the problem you are trying to solve like GPT for question answering and BERT for translation.
It is important to also look at the configurations of the model to tune their compatibility with your application like:

- Number of layers in transformer blocks
- Number of attention heads
- Loss function
- Hyperparameters

Pre-training

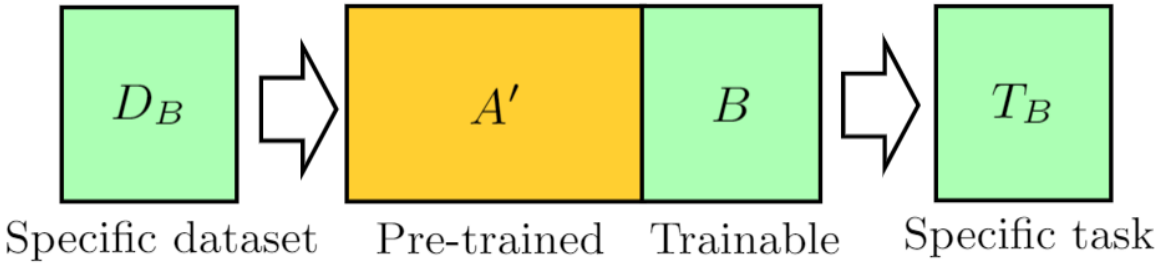
Since LLMs take in Big Data to be able to understand language from scratch, it is important to understand the resources needed to run such a huge training job. In such cases, we use **cloud computing**.

There are multiple ways in which you can train an LLM without it taking months:

1. Sequence parallelism
2. Pipeline parallelism
3. Data parallelism
4. Tensor parallelism

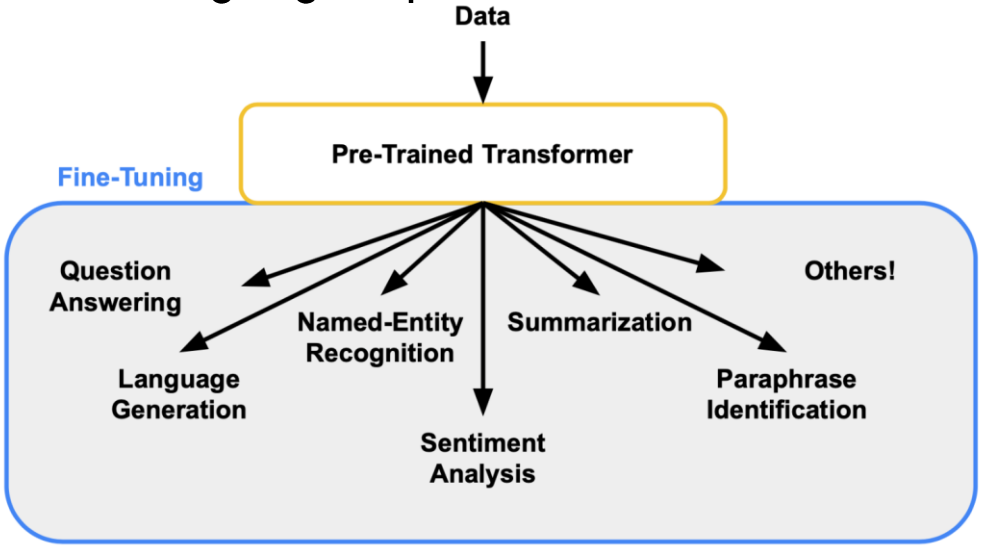
Fine-tuning

- Fine tuning is essentially a **transfer learning task.**
- Transfer learning is the reuse of a pre-trained model on a new problem.
- In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another.



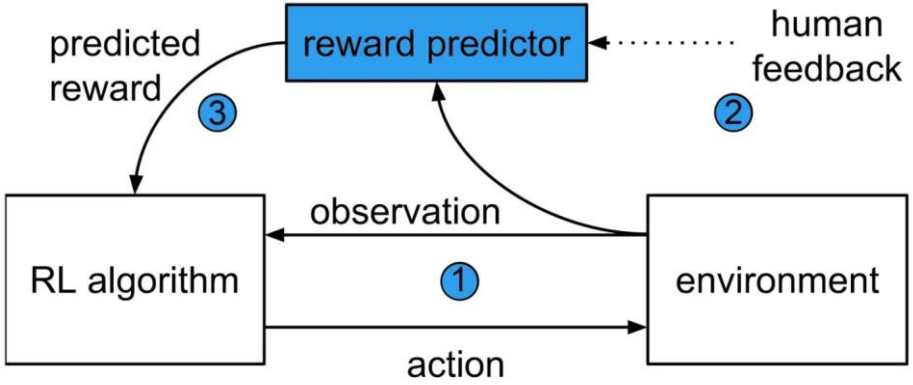
Fine-tuning

Fine-tuning in NLP is usually done for **downstream NLP tasks,** which are natural language processing (NLP) tasks that can be performed using a pre-trained language representation model.



Fine-tuning

- Fine-tuning is a **supervised learning** task (the model evaluates its learning based on the labels or the expected prediction of the data point).
- Another method of evaluating fine-tuning LLMs to ensure their reliability is **reinforcement learning with human feedback**



Testing & Evaluation

It is important to differentiate between testing and evaluation in LLMs:

1. **Testing**: performing inference on the part of the dataset that the model has not seen yet to evaluate its performance and attain some accuracy metrics.
2. **Evaluation**: performing inference on benchmark datasets and safety protocols.

شكراً لكم

Thank you

SDAIA
الهيئة السعودية للبيانات والذكاء الاصطناعي
Saudi Data & AI Authority