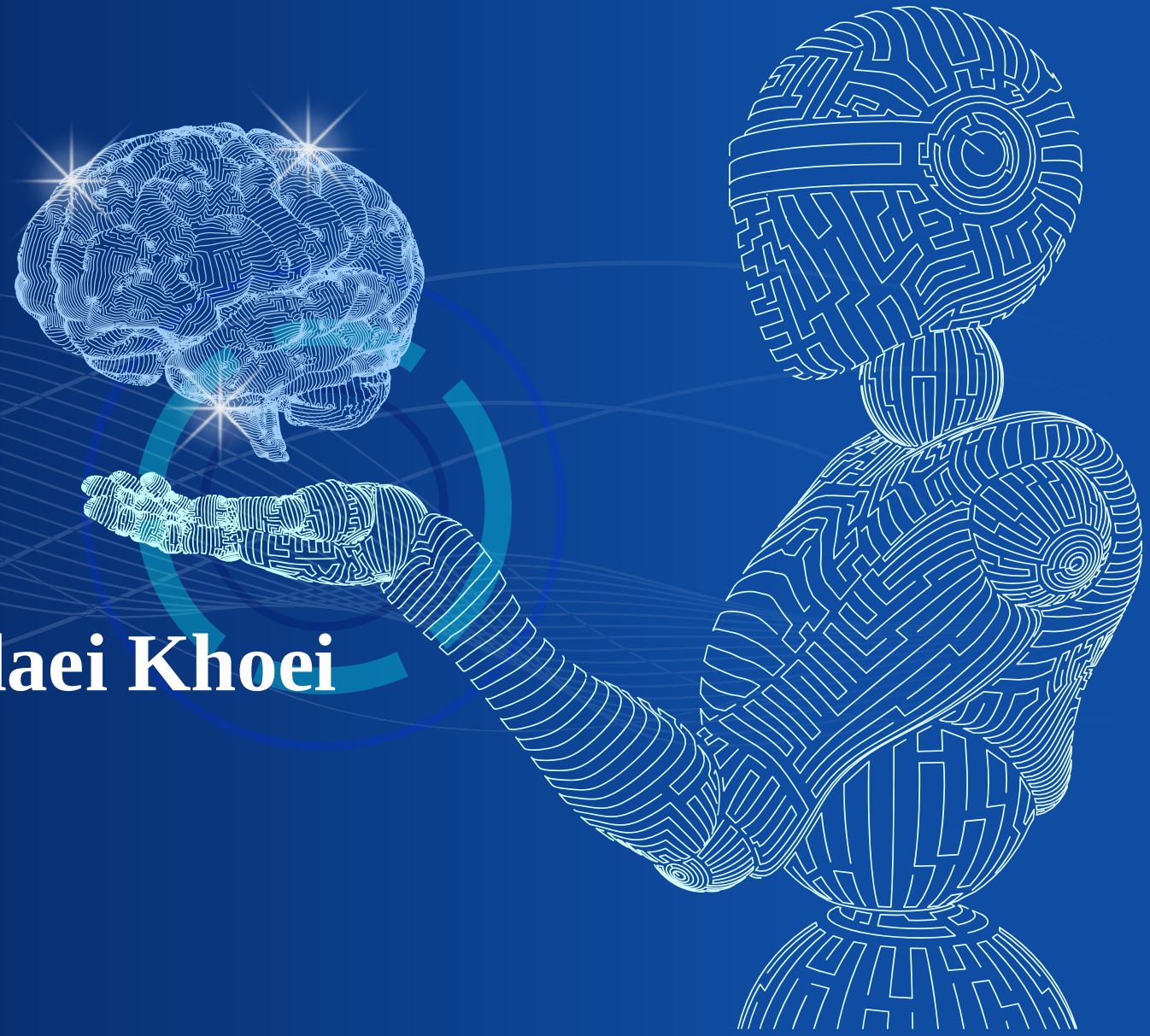
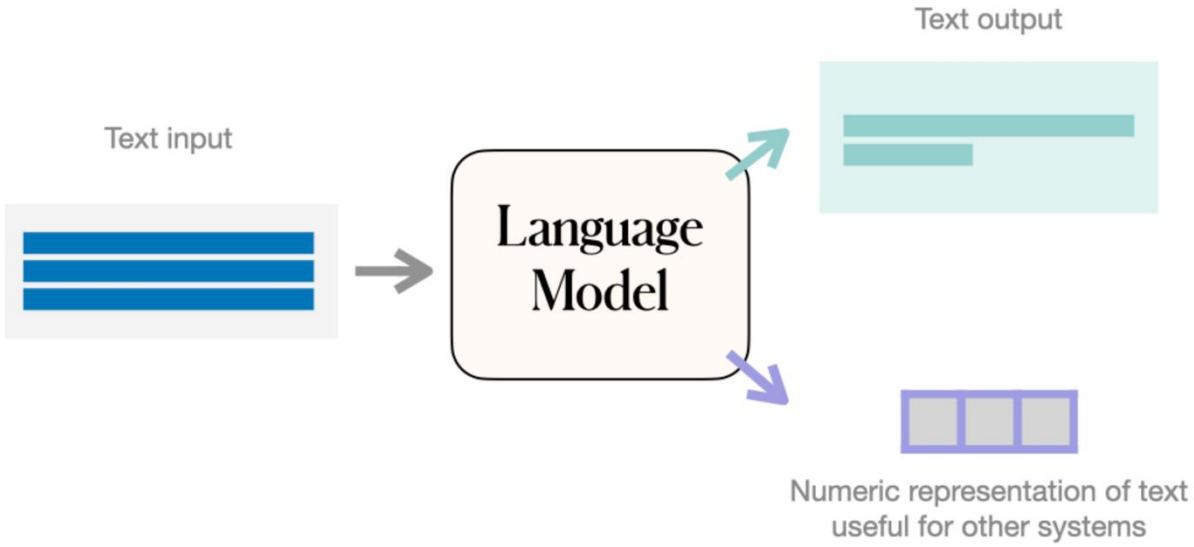
CS 7150: Deep Learning



Presented by: Dr. Tala Talaei Khoei

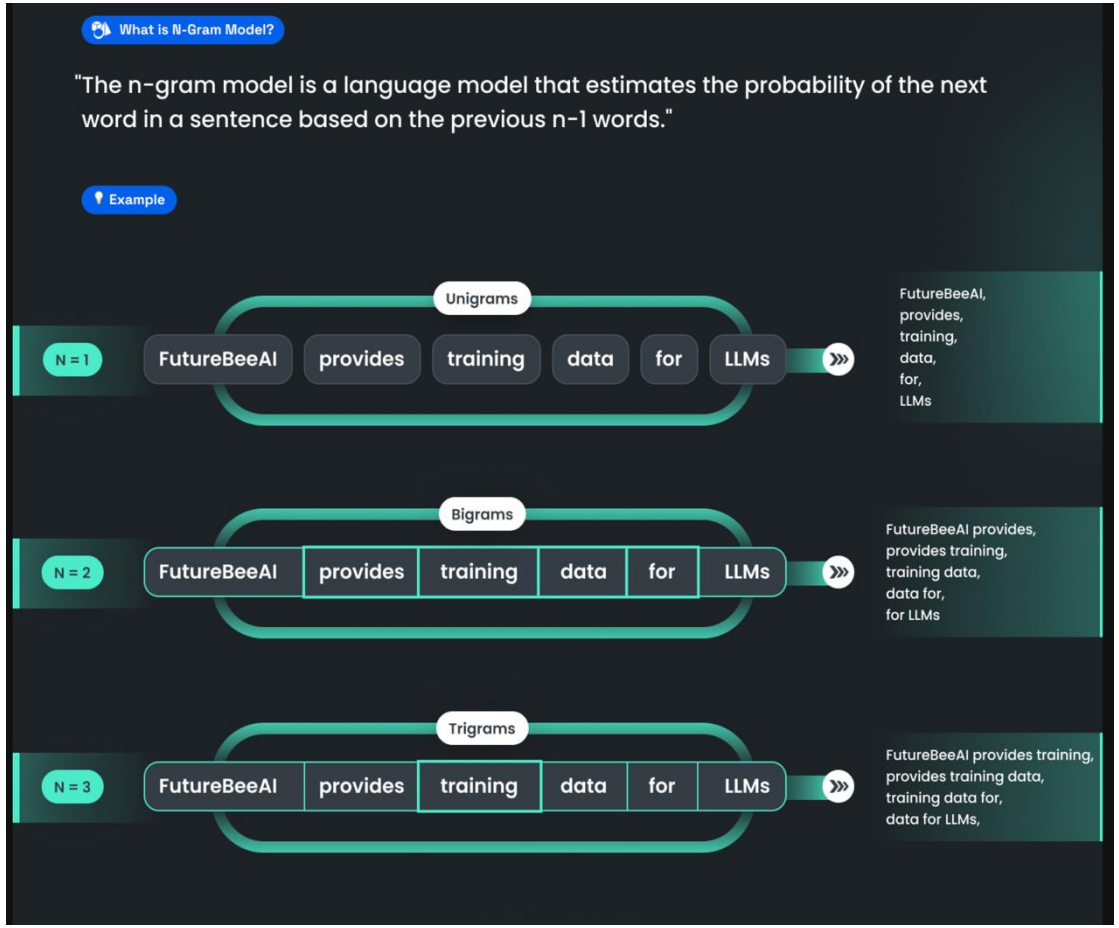
Introduction to Language Models

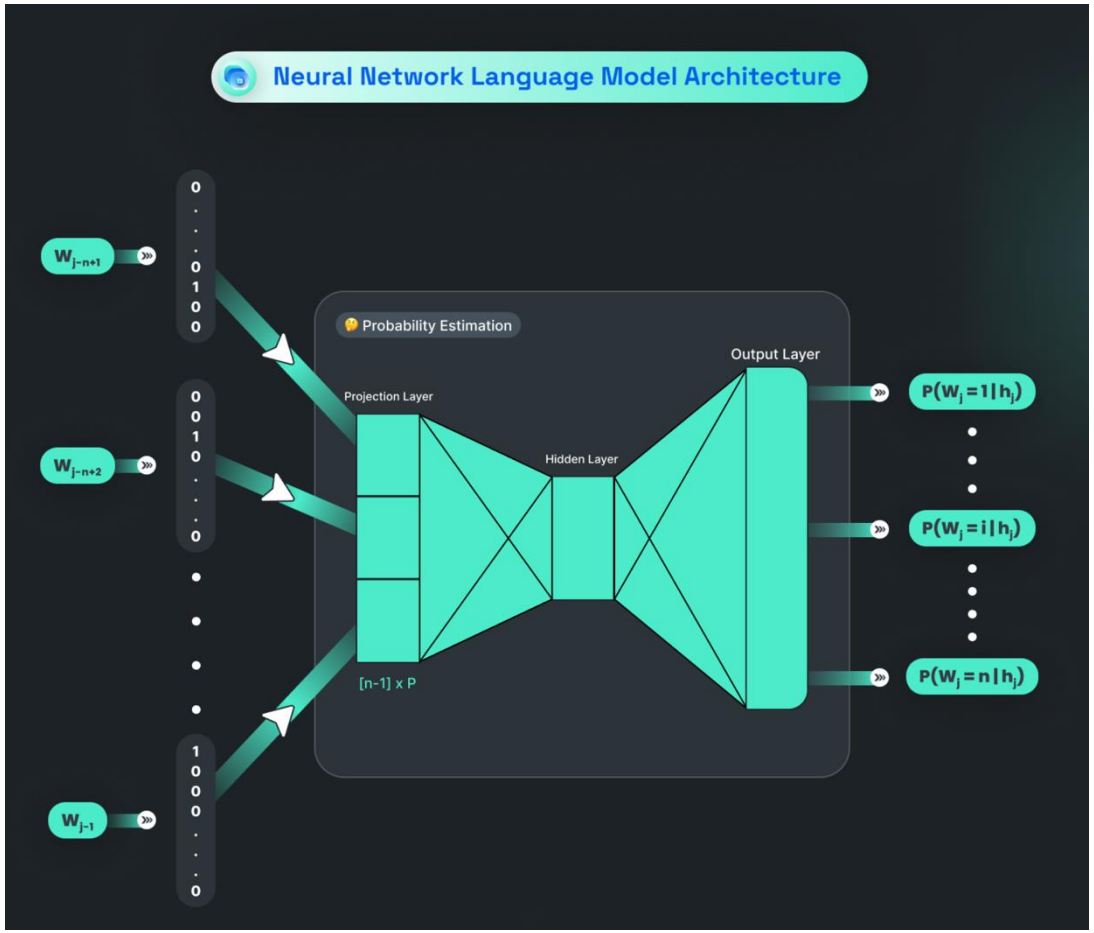
- A language model is a probability distribution over words or word sequences. In practice, it gives the probability of a certain word sequence being "valid." Validity in this context does not refer to grammatical validity. Instead, it means that it resembles how people write, which is what the language model learns.
- Put it simply, a model tries to predict the next most appropriate word to fill in a blank space in a sentence or phrase, based on the context of the given text.
- For example, in a sentence that sounds like this, *"Jenny dropped by the office for the keys so I gave them to […],"* a good model will determine that the missed word is likely to be a pronoun. Since the relevant piece of information here is *Jenny*, the most probable pronoun is *she* or *her.* The important thing is that the model doesn't focus on grammar, but rather on how words are used in a way that is similar to how people write.



Types of Language Models

1. Probabilistic methods.
2. Neural network-based modern language models





Probabilistic methods:

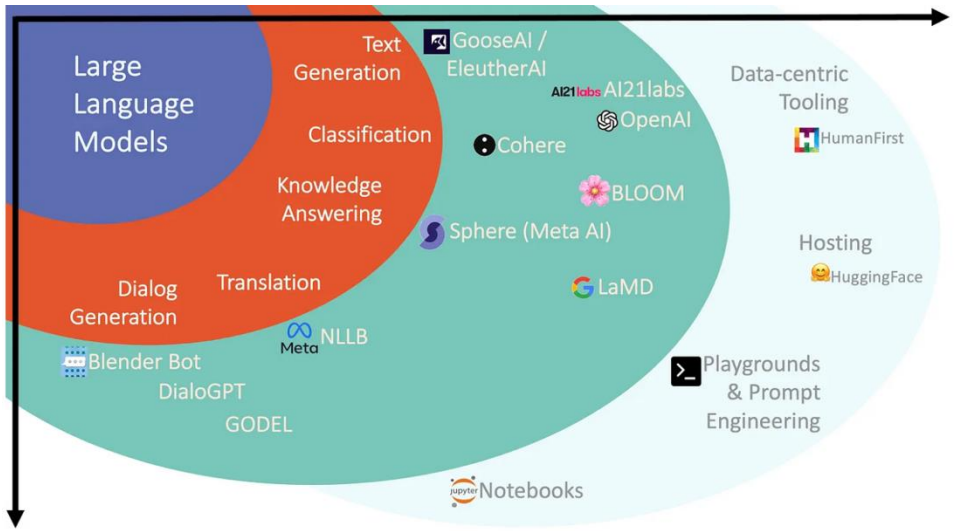
- A simple probabilistic language model is constructed by calculating n-gram probabilities. An n-gram is an n word sequence, n being an integer greater than zero. An n-gram's probability is the conditional probability that the n-gram's last word follows a particular n-1 gram (leaving out the last word). It's the proportion of occurrences of the last word following the n-1 gram leaving the last word out. This concept is a Markov assumption. Given the n-1 gram (the present), the n-gram probabilities (future) does not depend on the n-2, n-3, etc grams (past).
- There are evident drawbacks of this approach. Most importantly, only the preceding n words affect the probability distribution of the next word. Complicated texts have deep context that may have decisive influence on the choice of the next word. Thus, what the next word is might not be evident from the previous n-words, not even if n is 20 or 50. A term has influence on a previous word choice: the word United is much more probable if it is followed by States of America. Let's call this the context problem.
- On top of that, it's evident that this approach scales poorly. As size increases (n), the number of possible permutations skyrocket, even though most of the permutations never occur in the text. And all the occuring probabilities (or all n-gram counts) have to be calculated and stored. In addition, non-occurring n-grams create a sparsity problem**,** as in, the granularity of the probability distribution can be quite low. Word probabilities have few different values, therefore most of the words have the same probability.

Neural network-based modern language models

- Neural network-based language models use neural networks to predict the probability of the next word in a sequence of text. These models have become increasingly popular in recent years due to their ability to capture long-range dependencies and more complex linguistic structures than N-gram models. Neural network-based language models ease the data sparsity problem by the way they encode inputs.
- The basic architecture of a neural network-based language model typically involves an input layer that receives a sequence of words or tokens, followed by one or more hidden layers that perform a nonlinear transformation of the input. The output layer predicts the probability distribution of the next word in the sequence based on the hidden layer's representation of the input.
- One of the neural network architectures for language modeling is the recurrent neural network (RNN). RNNs are well-suited for modeling sequential data, such as text because they can maintain a "memory" of previous inputs and use that information to predict future inputs.
- However, the main drawback of RNN-based architectures stems from their sequential nature. As a consequence, training times soar for long sequences because there is no possibility for parallelization. The solution to this problem is **transformer architecture**.
- The most popular architecture for neural network-based language modeling is the transformer model. Transformers are based on a self-attention mechanism that allows the model to attend to different parts of the input sequence to make predictions. The transformer model has been shown to achieve state-of-the-art performance on a variety of natural language processing tasks, including language modeling. The GPT models from OpenAI and Google’s BERT utilize the transformer architecture.

Introduction to Large Language Models

- Large language models are typically trained on massive amounts of text data, such as the entire internet or large collections of books, articles, and other written material. They use sophisticated algorithms and neural networks to learn the patterns and structure of language, enabling them to generate coherent and meaningful responses to a wide variety of prompts and queries.
- These models are also known as Generative AI models, because they can generate text in a variety of ways, including by predicting the next word in a sentence, completing a sentence or paragraph, or generating a new text from scratch based on a given topic or prompt. Examples of large language models include GPT-4, GPT-3.5, BERT, RoBERTa, Claude, BLOOM, etc.
- Although Large language models are a very powerful tool for generating content, we should cross-check the content generated by these models because they can generate wrong information as they are trained on the data from the web that may have wrong pieces of information. These models can be used for general purposes like blog writing, and can also be used as foundation models for developing specific use case language models.



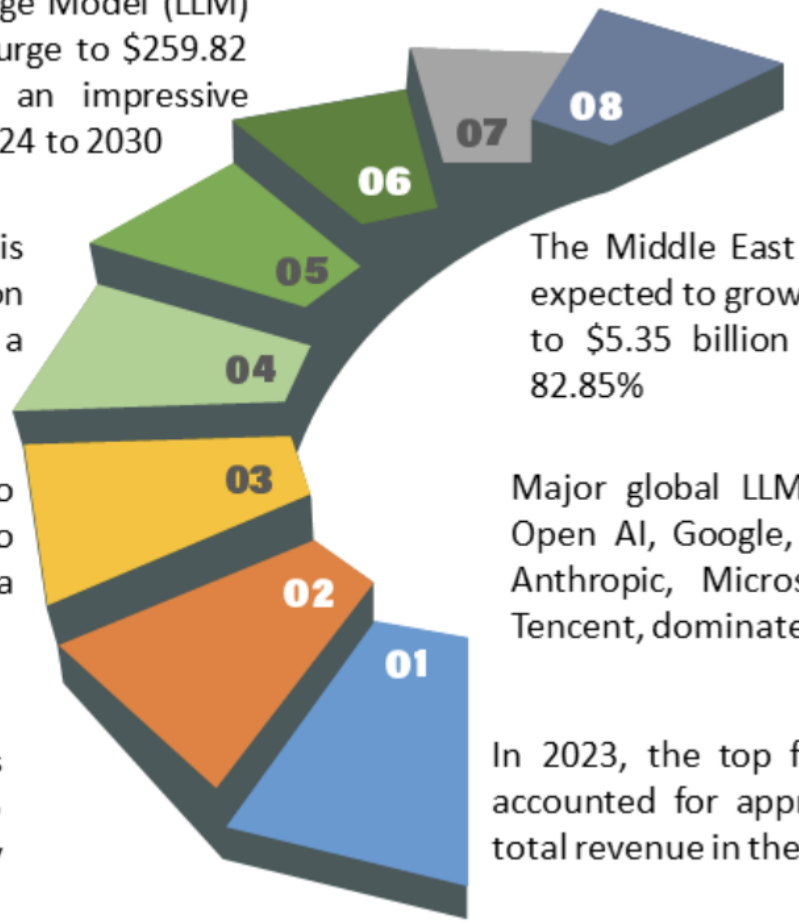
LLM MARKET SIZE AND FORECAST

08 The global Large Language Model (LLM) market is expected to surge to $259.82 billion by 2030, with an impressive CAGR of 79.80% from 2024 to 2030

07 Latin America's LLM market is forecasted to increase from $25.86 million in 2023 to $4.80 billion by 2030, at a CAGR of 83.54%

06 North America's LLM market is projected to grow from $848.65 million in 2023 to $105.55 billion by 2030, at a CAGR of 72.17%

05 The Middle East and Africa's LLM market is expected to grow from $29.25 million in 2023 to $5.35 billion by 2030, with a CAGR of 82.85%



04 Europe's LLM market is estimated to rise from $270.61 million in 2023 to $50.09 billion by 2030, boasting a CAGR of 83.30%

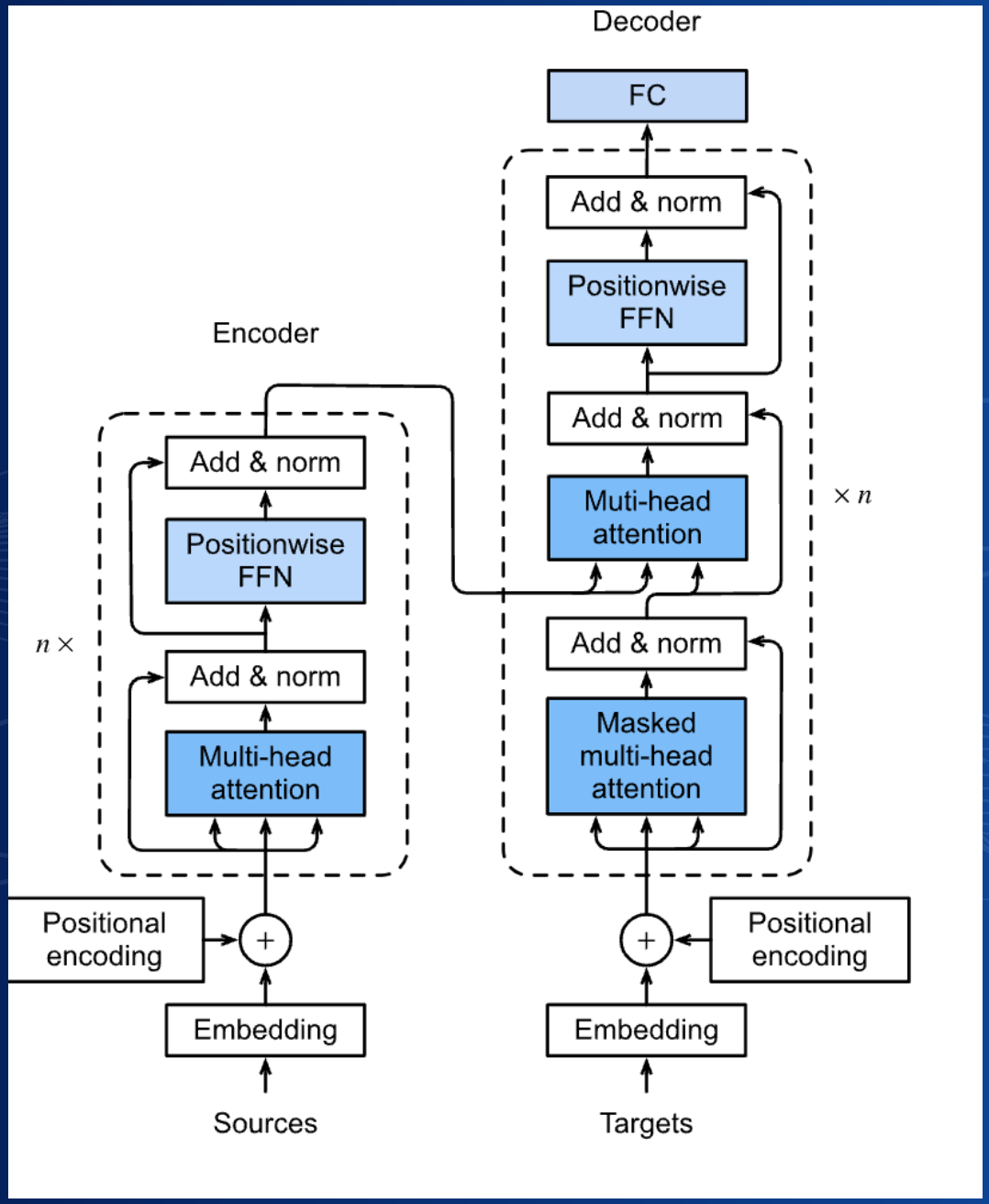
03 Major global LLM manufacturers, including Open AI, Google, Meta, AI21 Labs, Cohere, Anthropic, Microsoft, Huawei, Naver, and Tencent, dominate the market

02 The Asia-Pacific LLM market is anticipated to climb from $416.56 million in 2023 to $94.03 billion by 2030, with a CAGR of 89.21%

01 In 2023, the top five LLM vendors globally accounted for approximately 88.22% of the total revenue in the industry

The various LLM offerings cover these five areas of functionality in varying degrees.

- **Classification** is a form of supervised learning where text is assigned to predefined classes. This is related to Clustering which is unsupervised learning where semantically similar text is grouped together without any pre-existing classes.
- **Response Generation** is the notion of creating a dialog flow from example conversations, and having a machine learning approach to it. Where a model determines the next dialog to present to the user, based on the immediate conversation history and the most probable next dialog.
- **Text Generation** can be described as the meta capability of LLMs, text can be generated based on a short description with or without example data. Generation is a function shared amongst virtually all LLMs. Not only can generation be leveraged extensively by few-shot learning data; by ***casting*** (*prompt engineering*) the data in certain way determines how the few-shot learning data will be used.
- **Translation** is where text is translated from one language to another. This is done directly without any intermediary language.
- **Knowledge Answering** is an implementation of what is called Knowledge Intensive NLP (KI-NLP), where broad domain and general questions can be answered, without querying an API or leveraging a traditional knowledge base. Knowledge Intensive NLP is not a web search, but a self contained knowledge base underpinned by semantic search.

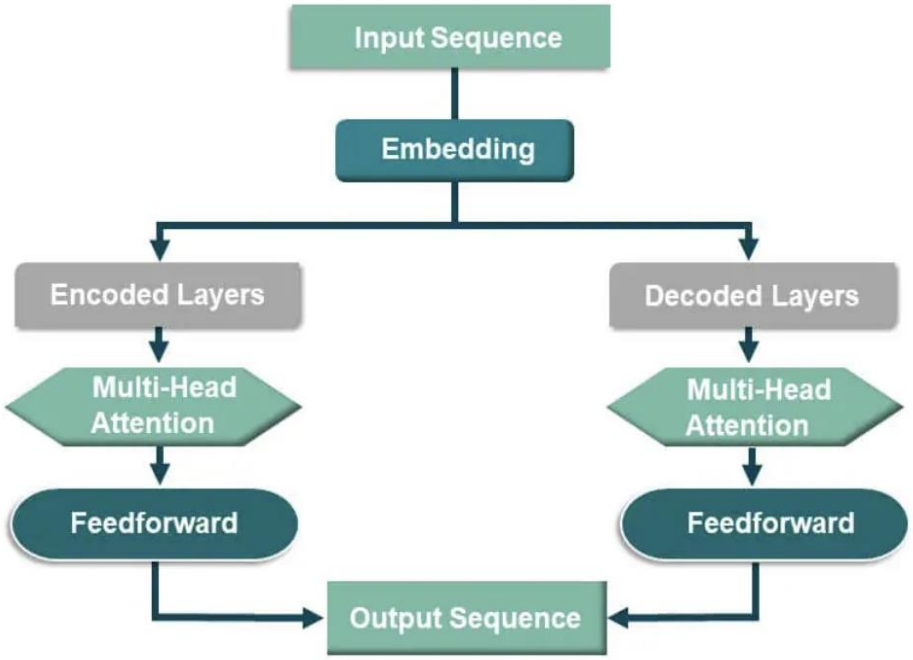

Decoder
FC
Add & norm
Positionwise FFN
Add & norm
Muti-head attention
Add & norm
Masked multi-head attention
× n
Encoder
Add & norm
Positionwise FFN
Add & norm
Multi-head attention
n ×
Positional encoding
Positional encoding
Embedding
Embedding
Sources
Targets

Introduction to Transformers

- Transformers are a type of deep learning architecture that have revolutionized the field of natural language processing (NLP) in recent years. They are widely used for tasks such as language translation, text classification, sentiment analysis, and more.
- The transformer architecture was first introduced in a 2017 paper by Google researchers Vaswani et al. called "Attention Is All You Need". This paper proposed a novel approach to NLP tasks that relied solely on the self-attention mechanism, a type of attention mechanism that allows the model to weigh the importance of different words in a sentence when encoding it into a fixed-size vector representation. This was a departure from previous NLP models that relied on recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to process sequences of words.
- The transformer architecture was revolutionary in that it allowed for much faster training times and better parallelization on GPUs, since the self-attention mechanism could be computed in parallel for all words in a sequence. This made it possible to train much larger models on much larger datasets, leading to significant improvements in NLP performance.
- Unlike earlier models, Transformers are highly effective for tasks like language translation, text generation, etc, due to their efficient capture of long-range dependencies in data. Their success has led to the development of various Transformer variants, each tailored for specific applications. This article delves into the core components and workings of Transformer models, shedding light on their pivotal role in modern machine learning.

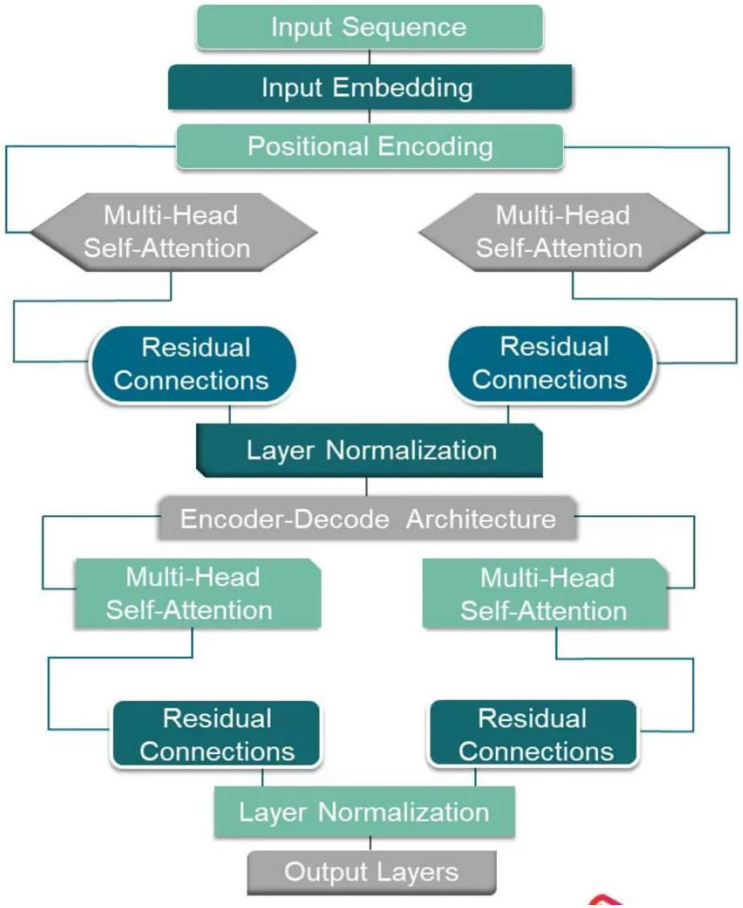
Transformers' Application

1. *Natural Language Processing (NLP)*: They excel in tasks like language translation, sentiment analysis, text summarization, and question-answering.
2. *Image Processing*: Transformers can process images for tasks like image captioning, object detection, and even generating art.
3. *Speech Recognition*: They're used in speech-to-text systems and voice assistants.
4. *Recommendation Systems:* Transformers are used to power recommendation engines, which help improve the accuracy of personalized content suggestions.
5. *Drug Discovery*: Transformers assist in drug molecule generation and predicting drug interactions.
6. *Conversational AI:* They enable chatbots and virtual assistants to have more natural and context-aware conversations.
7. *Anomaly Detection*: Transformers can detect anomalies in data, which is vital for fraud detection and network security.
8. *Language Generation*: They are adept at generating human-like text, making them valuable for chatbots, content creation, and creative writing.



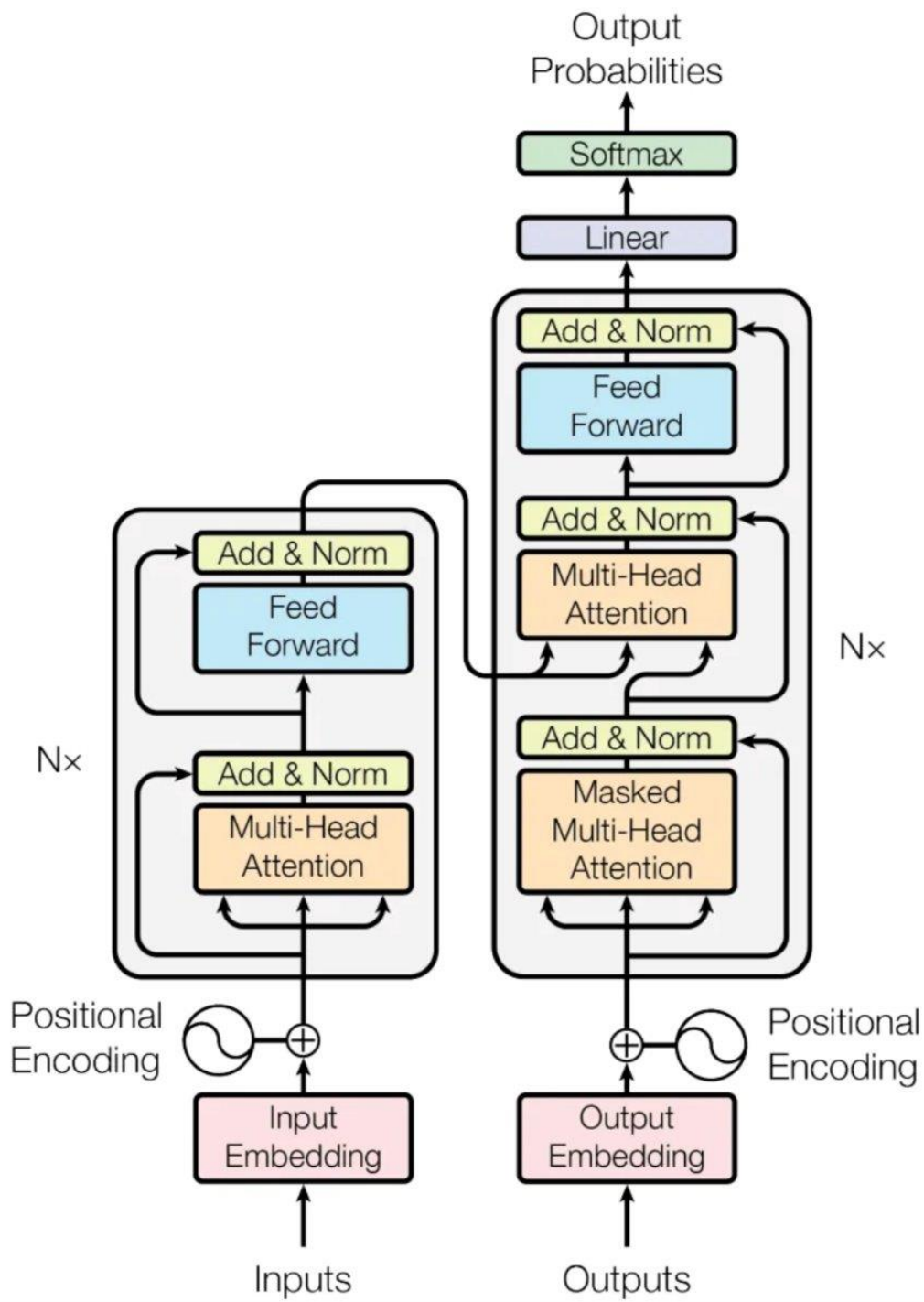
Architecture

1. **Input Embedding:** Initially, input sequences are transformed into numerical embeddings, which capture the semantic meaning of words or tokens. These embeddings serve as the model's input.
2. **Positional Encoding:** Positional encoding is added to the input embeddings to account for the order of words in a sequence. This ensures the model can distinguish between words with the exact embeddings.
3. **Multi-Head Self-Attention:** This is the heart of the Transformer architecture. Self-attention mechanisms enable the model to assign importance to different words in the input sequence, improving prediction accuracy. The "multi-head" aspect involves performing self-attention multiple times in parallel, enabling the model to focus on different parts of the input simultaneously.
4. **Position-wise Feed-Forward Networks:** After the self-attention mechanism, position-wise feed-forward networks are applied independently to each position in the sequence. These networks introduce non-linearity into the model.
5. **Residual Connections:** Residual connections, inspired by the ResNet architecture, help mitigate the vanishing gradient problem during training. They involve adding the input embeddings to the output of the multi-head self-attention and feed-forward **network layers**.
6. **Layer Normalization:** Layer normalization is used to stabilize training by normalizing the output of each layer. It helps maintain a consistent distribution of activations throughout the network.
7. **Encoder-Decoder Architecture (Optional):** Transformers use an encoder-decoder architecture in sequence-to-sequence tasks, such as machine translation. The encoder processes the input sequence while the decoder generates the output sequence. The encoder's final hidden state is used to initialize the decoder.
8. **Output Layers:** Different output layers can be added depending on the specific task. For example, a SoftMax layer is used to predict the next word in a sequence in language modeling.



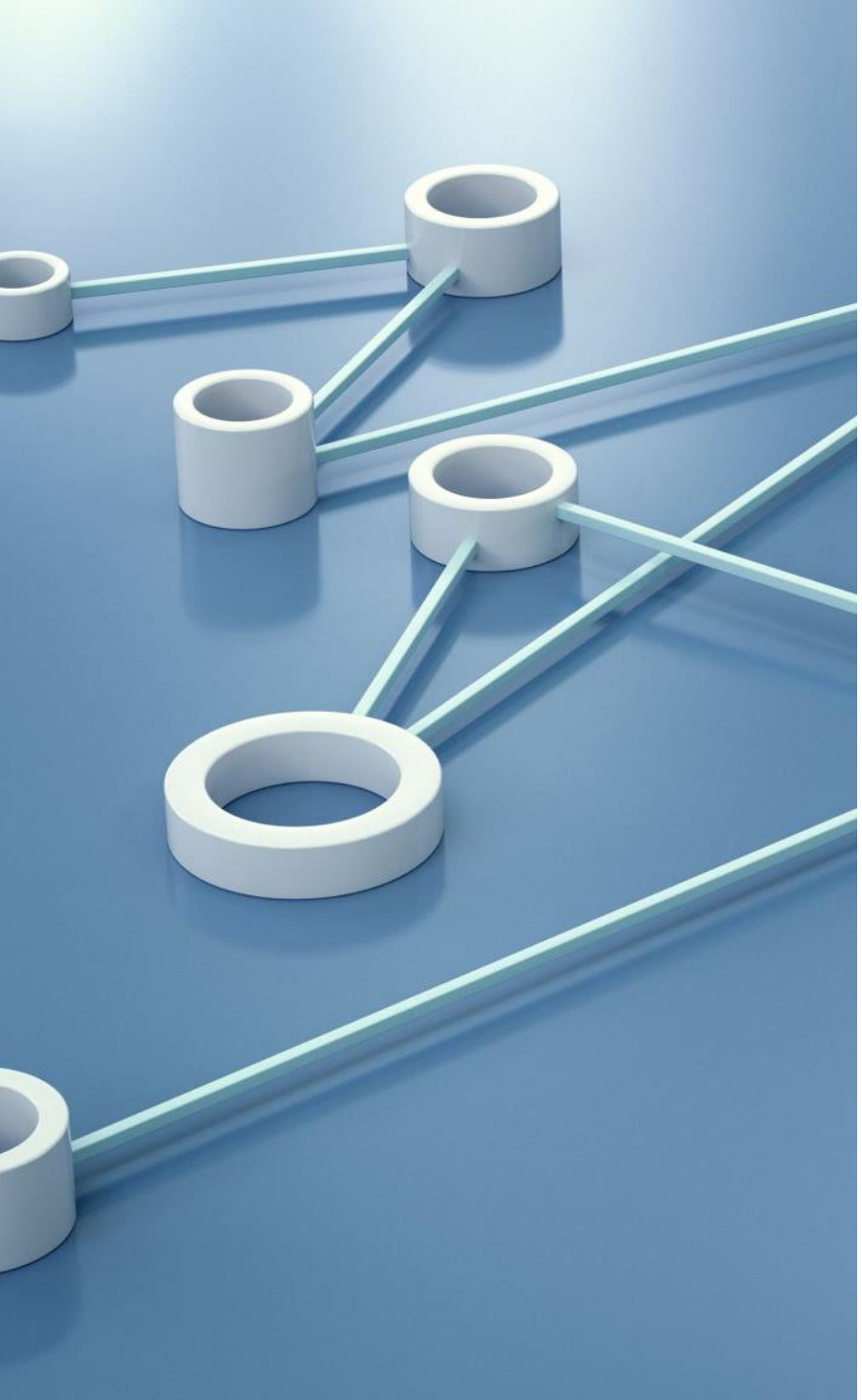
BERT

Encoder



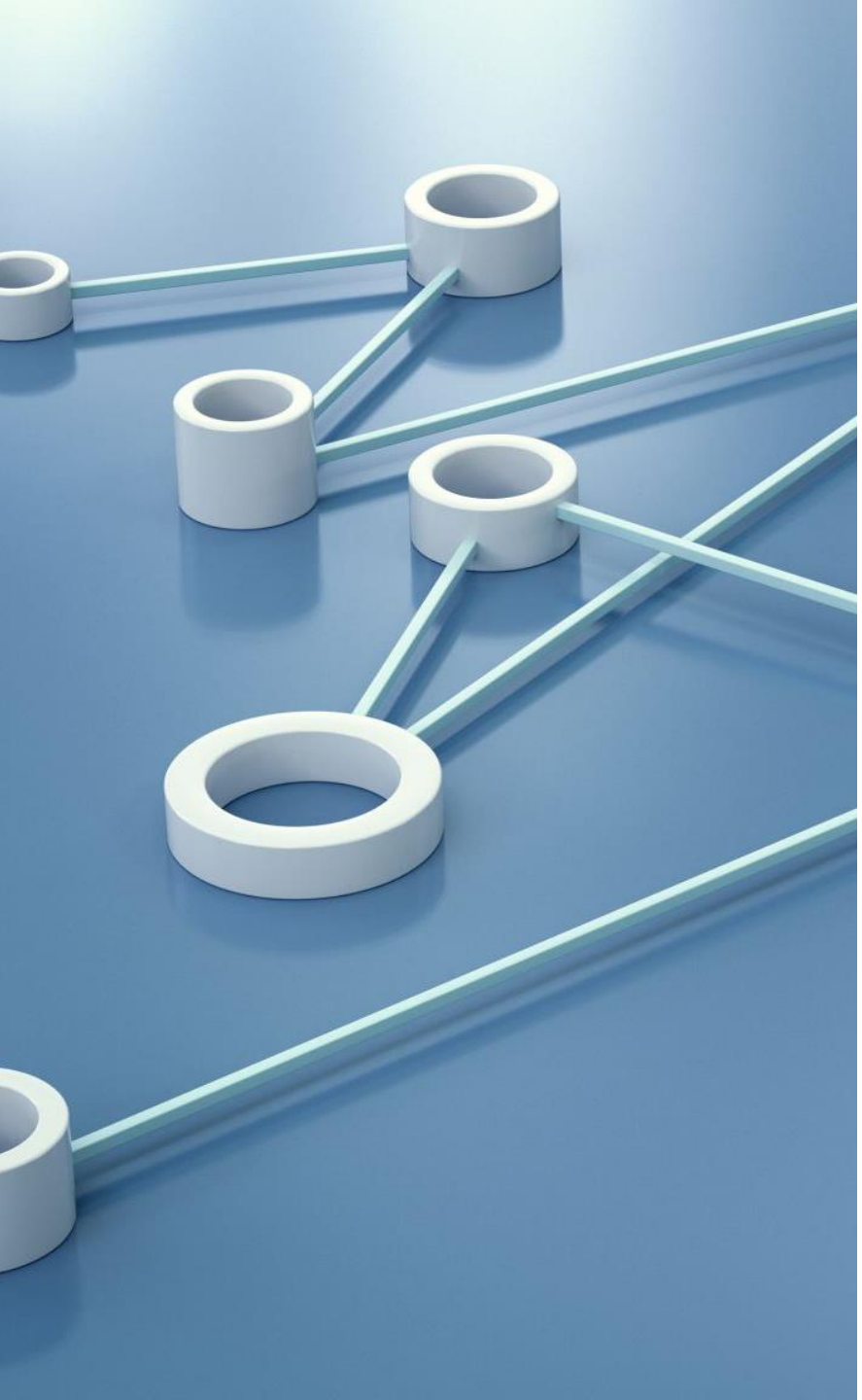
GPT

Decoder



Self-Attention Mechanism

- The self-attention mechanism, a core component of Transformer models, facilitates understanding contextual relationships in data by:
- Calculating attention scores between every pair of elements in a sequence.
- Assigning weights to each element based on its relevance to others.
- Aggregating information by taking a weighted sum of all elements.
- The model can adaptively concentrate on various segments of the input sequence.
- Capturing long-range dependencies without regard to fixed window sizes.
- Enhancing the model’s ability to process sequential data, making it highly effective in NLP tasks.
- Serving as the basis for multi-head attention, a key innovation in Transformer architecture.



Multi-Head Attention Mechanism

Multi-head attention is a critical component of Transformer models, enhancing their data processing capability by:

- Utilizing multiple sets of weight matrices to perform self-attention in parallel.
- The approach enables the model to concentrate on distinct elements of the input sequence simultaneously.
- Learning diverse, contextually rich representations by capturing various relationships within the data.
- Enhancing the model’s ability to recognize both local and global patterns.
- Combining multiple heads’ outputs to create a more robust and expressive representation.
- Thanks to its versatility and improved attention mechanisms, it enables the model to excel in complex tasks like translation, summarization, and question-answering.
- Leading to more effective and efficient deep-learning models.

Layer Normalization & Residual Connections

Layer Normalization:

- It is applied after each sub-layer (e.g., multi-head self-attention and feed-forward layers) within each Transformer layer.
- It helps stabilize the training process by ensuring that the activations in each layer have consistent mean and variance.
- Mitigates the vanishing gradient problem, making it easier to train deep models.
- Enhances the model's ability to capture meaningful patterns and relationships in the data.

Residual Connections:

- Inspired by the ResNet architecture, residual connections involve adding the input of a sub-layer to its output.
- They create shortcut connections that allow gradients to flow more easily during training.
- Address the vanishing gradient problem, making it feasible to train very deep networks.
- Improve the model's capacity to learn complex and hierarchical representations.

Transformer: Encoder-Decoder

- The transformer encoder-decoder architecture is used for tasks like language translation, where the model must take in a sentence in one language and output a sentence in another language. The encoder takes in the input sentence and produces a fixed-size vector representation of it, which is then fed into the decoder to generate the output sentence. The decoder uses both self-attention and cross-attention, where the attention mechanism is applied to the output of the encoder and the input of the decoder.
- One of the most popular transformer encoder-decoder models is the T5 (Text-to-Text Transfer Transformer), which was introduced by Google in 2019. The T5 can be fine-tuned for a wide range of NLP tasks, including language translation, question answering, summarization, and more.
- Real-world examples of the transformer encoder-decoder architecture include Google Translate, which uses the T5 model to translate text between languages, and Facebook's M2M-100, a massive multilingual machine translation model that can translate between 100 different languages.

Transformer: Encoder

- The transformer encoder architecture is used for tasks like text classification, where the model must classify a piece of text into one of several predefined categories, such as sentiment analysis, topic classification, or spam detection. The encoder takes in a sequence of tokens and produces a fixed-size vector representation of the entire sequence, which can then be used for classification.
- One of the most popular transformer encoder models is BERT (Bidirectional Encoder Representations from Transformers), which was introduced by Google in 2018. BERT is pre-trained on large amounts of text data and can be fine-tuned for a wide range of NLP tasks.
- Unlike the encoder-decoder architecture, the transformer encoder is only concerned with the input sequence and does not generate any output sequence. It applies self-attention mechanism to the input tokens, allowing it to focus on the most relevant parts of the input for the given task.
- Real-world examples of the transformer encoder architecture include sentiment analysis, where the model must classify a given review as positive or negative, and email spam detection, where the model must classify a given email as spam or not spam.

Transformer: Decoder

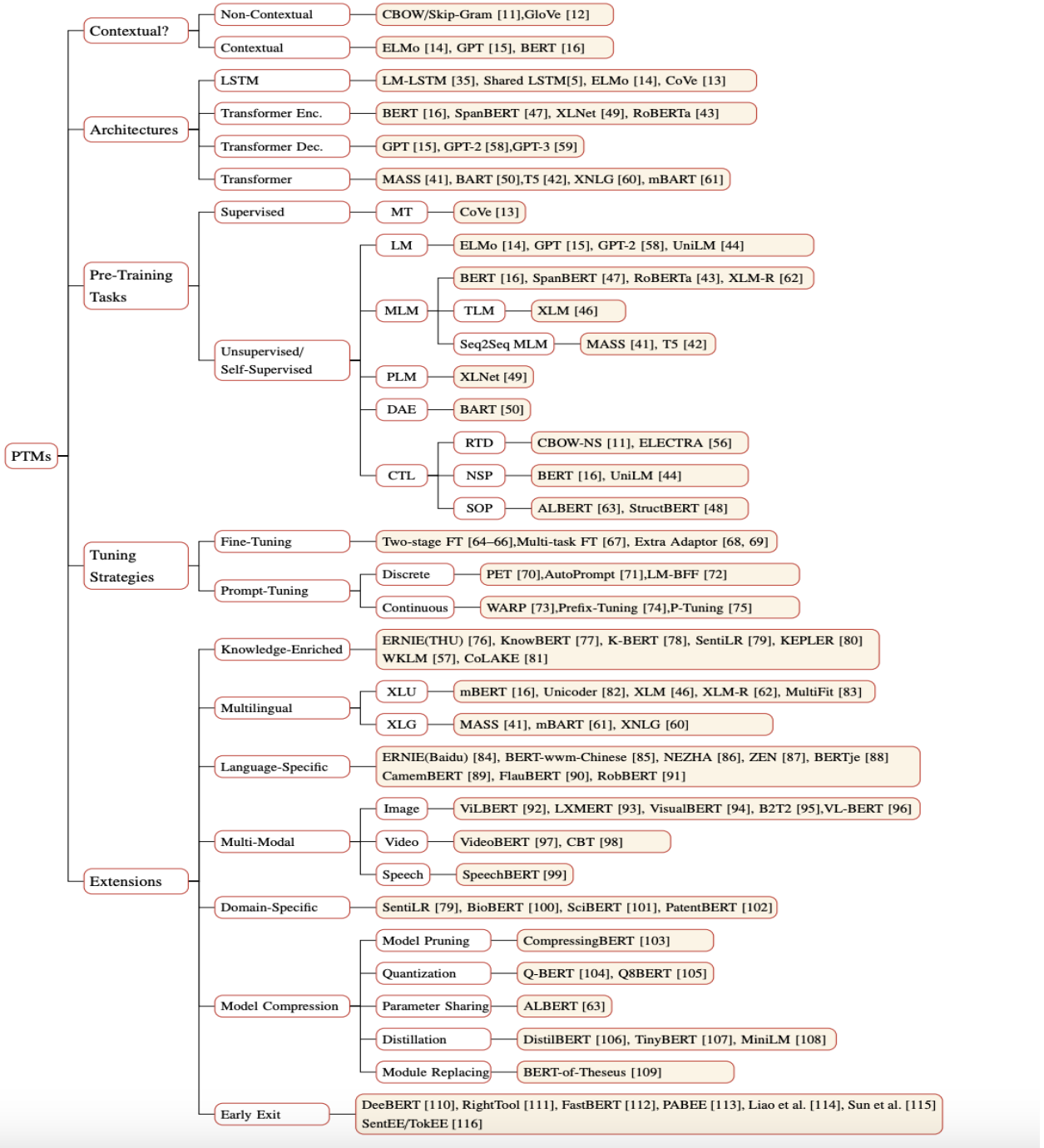
- The transformer decoder architecture is used for tasks like language generation, where the model must generate a sequence of words based on an input prompt or context. The decoder takes in a fixed-size vector representation of the context and uses it to generate a sequence of words one at a time, with each word being conditioned on the previously generated words.
- One of the most popular transformer decoder models is the GPT-3 (Generative Pre-trained Transformer 3), which was introduced by OpenAI in 2020. The GPT-3 is a massive language model that can generate human-like text in a wide range of styles and genres.
- The transformer decoder architecture introduces a technique called triangle masking for attention, which ensures that the attention mechanism only looks at tokens to the left of the current token being generated. This prevents the model from "cheating" by looking at tokens that it hasn't generated yet.
- Real-world examples of the transformer decoder architecture include text generation, where the model must generate a story or article based on a given prompt or topic, and chatbots, where the model must generate responses to user inputs in a natural and engaging way.

Drawback

- High computational cost due to the attention mechanism, which increases exponentially with sequence length.
- Difficulty in interpretation and debugging due to the attention mechanism operating over the entire input sequence.
- Prone to overfitting when fine-tuned on small amounts of task-specific data.

Despite these downsides, the transformer architecture remains a powerful and widely-used tool in NLP, and research is ongoing to mitigate its computational requirements and improve its interpretability and robustness.

Pre-Trained Transformer Models



https://arxiv.org/pdf/2003.08271.pdf

Please refer to Codes on Canvas


THANK YOU
Any Questions?