DESCRIPTIVE QUESTIONS

Q1. What are Transformer Models?  
Ans: Transformer models are a type of deep learning model architecture primarily used in natural language processing (NLP) tasks, although they have also found applications in other domains such as computer vision and reinforcement learning. They were introduced by Vaswani et al. in the paper "Attention is All You Need" in 2017.

The key innovation of transformer models is the self-attention mechanism, which allows the model to weigh the importance of different words in a sequence when processing each word. This mechanism enables the model to capture dependencies between words regardless of their positions in the input sequence, unlike traditional recurrent neural networks (RNNs) and convolutional neural networks (CNNs) which process inputs sequentially or with fixed-size receptive fields.

Q2. What are Encoder Decoder in a Transformer?  
Ans: In a transformer model, the encoder-decoder architecture consists of two main components: the encoder and the decoder. Each component is responsible for different tasks in sequence-to-sequence learning, such as machine translation or text summarization.

1. Encoder: The encoder processes the input sequence and generates a representation that captures the semantic meaning of the input. It consists of a stack of identical layers, each layer typically containing two sub-layers:

- Self-Attention Layer: This layer computes attention scores for each word in the input sequence, capturing dependencies between words. It allows the model to weigh the importance of different words when encoding the input sequence.

- Feed-Forward Neural Network: After computing self-attention, the output passes through a feed-forward neural network with a couple of fully connected layers. This helps the model capture complex patterns in the data.

2. Decoder: The decoder takes the encoded representation generated by the encoder and generates the output sequence word by word. Like the encoder, it consists of a stack of identical layers, each containing three sub-layers:

- Self-Attention Layer: Similar to the encoder, the decoder's self-attention mechanism computes attention scores for each word in the output sequence. However, during training, it's modified to prevent attending to future words (causal masking), ensuring that the model only attends to previously generated tokens.

- Encoder-Decoder Attention Layer: This layer attends to the encoded representation generated by the encoder, allowing the decoder to focus on relevant parts of the input when generating the output.

- Feed-Forward Neural Network: After computing attention, the output passes through a feed-forward neural network similar to the encoder.

The encoder and decoder are trained jointly using a combination of supervised learning and techniques like teacher forcing (during training, the model is fed the ground truth tokens from the target sequence) to generate accurate translations or summaries. Once trained, the decoder can be used in an autoregressive manner during inference, generating one token at a time while using its previously generated tokens as input.

Q3. What is positional encoding?  
Ans: Definition: Positional encoding is a technique used in sequence-to-sequence models, particularly in the Transformer architecture, to provide positional information to the model.

* Purpose: Unlike recurrent neural networks (RNNs) which inherently have access to the order of the input sequence, Transformer models lack this sequential information. Positional encoding helps the model understand the order of the input sequence.
* Implementation: It involves adding a fixed-length vector to each input embedding. This vector is calculated based on the position of the token within the sequence, typically using sinusoidal functions.

Example: In a translation task, where the Transformer model translates a sentence from one language to another, positional encoding helps the model differentiate between words based on their position in the sentence.

Q4. What is Embedding?  
Ans: Definition: Embedding is the process of representing words or entities as dense vectors in a continuous vector space.

Purpose: Embeddings are used to convert categorical data (such as words in natural language processing or categorical variables in machine learning) into a format that can be fed into machine learning models.

Q5. What are the types of Encoding?

Ans: Types:

* Word Embeddings: These represent words in a continuous vector space where similar words have similar vector representations. Examples include Word2Vec, GloVe, and FastText.
* Entity Embeddings: Used in recommender systems to represent categorical variables such as user IDs, item IDs, etc., in a continuous vector space.

Q5. What is ‘Query’ vector?  
Ans: **Query (Q):** The query is a representation of the input at a particular position that is used to derive attention scores.

In the context of the Transformer, each word in the input sequence has an associated query vector.

The query is used to determine how much focus should be given to different parts of the sequence when generating the output.

Q6. What is ‘Key’ vector?  
Ans: **Key (K):** The key is another representation of the input and is used to compute the attention scores with respect to the query.

Like the query, each word in the input sequence has an associated key vector.

The key helps in understanding the relationship between different words in the input sequence.

Q7. What is ‘Value’ vector?  
Ans: **Value (V):** The value represents the information associated with the input at a particular position.

Similar to the query and key, each word in the input sequence has an associated value vector.

The value vectors are used to compute the weighted sum, where the weights are determined by the attention scores. This weighted sum forms the output of the attention mechanism.

Q8. What are the functions of Transformers?  
Ans: Transformers offers convenient APIs and tools for effortless downloading and training of cutting-edge pretrained models. Leveraging pretrained models can result in cost savings, a reduced carbon footprint, and significant time and resource savings compared to training a model from the ground up. These models cater to various tasks across different domains:

**Natural Language Processing:** Covering tasks such as text classification, named entity recognition, question answering, language modeling, summarization, translation, multiple choice, and text generation.

**Computer Vision:** Encompassing image classification, object detection, and segmentation.

**Audio:** Supporting automatic speech recognition and audio classification.

**Multimodal:** Including capabilities for table question answering, optical character recognition, information extraction from scanned documents, video classification, and visual question answering.

Transformers ensure framework interoperability across **PyTorch**, **TensorFlow**, and **JAX**. This flexibility enables users to employ different frameworks at various stages of a model's lifecycle—training a model in just three lines of code in one framework and seamlessly loading it for inference in another.

Q9. How do Transformer model work?  
Ans: Transformer models work by processing input data, which can be sequences of tokens or other structured data, through a series of layers that contain self-attention mechanisms and feedforward neural networks. The core idea behind how transformer models work can be broken down into several key steps.

Let’s imagine that you need to convert an English sentence into French. These are the steps you’d need to take to accomplish this task with a transformer model.

Input embeddings: The input sentence is first transformed into numerical representations called embeddings. These capture the semantic meaning of the tokens in the input sequence. For sequences of words, these embeddings can be learned during training or obtained from pre-trained word embeddings.

Positional encoding: Positional encoding is typically introduced as a set of additional values or vectors that are added to the token embeddings before feeding them into the transformer model. These positional encodings have specific patterns that encode the position information.

Multi-head attention: Self-attention operates in multiple "attention heads" to capture different types of relationships between tokens. Softmax functions, a type of activation function, are used to calculate attention weights in the self-attention mechanism.

Layer normalization and residual connections: The model uses layer normalization and residual connections to stabilize and speed up training.

Feedforward neural networks: The output of the self-attention layer is passed through feedforward layers. These networks apply non-linear transformations to the token representations, allowing the model to capture complex patterns and relationships in the data.

Stacked layers: Transformers typically consist of multiple layers stacked on top of each other. Each layer processes the output of the previous layer, gradually refining the representations. Stacking multiple layers enables the model to capture hierarchical and abstract features in the data.

Output layer: In sequence-to-sequence tasks like neural machine translation, a separate decoder module can be added on top of the encoder to generate the output sequence.

Training: Transformer models are trained using supervised learning, where they learn to minimize a loss function that quantifies the difference between the model's predictions and the ground truth for the given task. Training typically involves optimization techniques like Adam or stochastic gradient descent (SGD).

Inference: After training, the model can be used for inference on new data. During inference, the input sequence is passed through the pre-trained model, and the model generates predictions or representations for the given task.

Q10. Write down the architecture of the Transformer Model.

Ans: A transformer architecture consists of an encoder and decoder that work together. The attention mechanism lets transformers encode the meaning of words based on the estimated importance of other words or tokens. This enables transformers to process all words or tokens in parallel for faster performance, helping drive the growth of increasingly bigger LLMs.

Thanks to the attention mechanism, the encoder block transforms each word or token into vectors further weighted by other words. For example, in the following two sentences, the meaning of it would be weighted differently owing to the change of the word filled to emptied:

1. He poured the pitcher into the cup and filled it.
2. He poured the pitcher into the cup and emptied it.

The attention mechanism would connect it to the cup being filled in the first sentence and to the pitcher being emptied in the second sentence.

The decoder essentially reverses the process in the target domain. The original use case was translating English to French, but the same mechanism could translate short English questions and instructions into longer answers. Conversely, it could translate a longer article into a more concise summary.  
  
Q11. Write down the applications of the Transformer model.  
Ans: Transformers can be applied to virtually any task that processes a given input type to generate an output. Examples include the following use cases:

Translating from one language to another.

Programming more engaging and useful chatbots.

Summarizing long documents. Generating a long document from a brief prompt.

Generating drug chemical structures based on a particular prompt.

Generating images from a text prompt.

Creating captions for an image.

Creating a robotic process automation (RPA) script from a brief description.

Providing code completion suggestions based on existing code.

Q12. Discuss the implementation of Transformer model.  
Ans: Transformer implementations are improving in terms of size, support for new use cases or different domains, such as medicine, science or business apps. The following are some of the most promising transformer implementations:

Google's Bidirectional Encoder Representations from Transformers was one of the first LLMs based on transformers.

OpenAI's GPT followed suit and underwent several iterations, including GPT-2, GPT-3, GPT-3.5, GPT-4 and ChatGPT.

Meta's Llama achieves comparable performance with models 10 times its size.

Google's Pathways Language Model generalizes and performs tasks across multiple domains, including text, images and robotic controls.

Open AI's Dall-E creates images from a short text description.

The University of Florida and Nvidia's GatorTron analyzes unstructured data from medical records.

DeepMind's Alphafold 2 describes how proteins fold.

AstraZeneca and Nvidia's MegaMolBART generates new drug candidates based on chemical structure data.

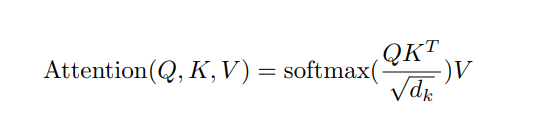
Q13. How is scaled dot product attention is calculated?  
Ans: Scaled Dot product: The scaled dot product is computed as follows:

Input = Queries (of dimension dk) + Keys (of dimension dk) + Values (of dimension dv).

The dot products of the query with each of the keys are calculated, then the obtained dot product for each key is scaled down by dividing it by √ dk, and then a softmax function is applied.

Practically, the attention function is calculated on a set of queries concurrently, which is packed together into a matrix [Q]. Similarly, the keys and values are packed together into matrix K and matrix V, respectively.

Final attention is computed as follows:



Q14. What is the Difference Between Additive and Multiplicative Attention?

Ans: Multiplicative Attention: Multiplicative (dot-product) attention is similar to the attention we discussed in the above question, except that it doesn’t employ the scaling factor 1/√ dk.

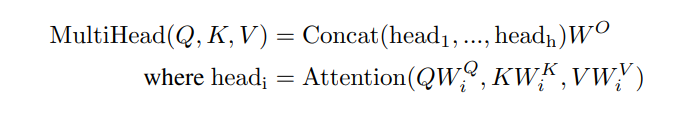
Additive Attention: Additive attention estimates how well the query matches with the corresponding key (i.e., compatibility function) with the help of a feed-forward network (FFN) with a single hidden layer.

The following are the critical differences between additive and multiplicative attention:

1. The theoretical complexity of these types of attention is more or less the same. However, dot-product attention is relatively faster and more space-efficient in practice due to the highly optimized matrix multiplication code.
2. For small values of dk , both of these mechanisms perform similarly.
3. For large values of dk, additive attention surpasses dot product attention without scaling.

Q15. What is multi-head attention?  
Ans: Multi-head attention is an extension of single-head attention (or single attention head), which enables the model to jointly attend to the info from various representation subspaces at different positions.

On examination, it was found that employing a single attention function is less beneficial than linearly projecting the queries, keys, and values h times with different learned linear projections.

The attention function is applied concurrently to these projected versions of queries, keys, and values, generating dv-dimensional output values.   
  
  
MULTIPLE CHOICE QUESTIONS-

Q1. What is the primary function of transformers in NLP?

a) Image recognition

b) Speech synthesis

c) Text generation

d) Graph processing

Correct answer: c) Text generation

Q2. Which component is responsible for capturing long-range dependencies in transformer models?

a) Sigmoid function

b) Multi-head attention

c) ReLU activation

d) Pooling layer

Correct answer: b) Multi-head attention

Q3. n multi-head attention, attention weights are computed separately for:

a) Queries, keys, and values

b) Queries only

c) Keys only

d) Values only

Correct answer: a) Queries, keys, and values

Q4. How do transformers handle variable-length input sequences?

a) By padding sequences to a fixed length

b) By truncating sequences

c) By using attention mechanisms

d) By resizing sequences

Correct answer: c) By using attention mechanisms

Q5. What is the purpose of dropout regularization in transformer architectures?

a) To increase model capacity

b) To reduce overfitting

c) To speed up training

d) To improve convergence

Correct answer: b) To reduce overfitting

Q6. The encoder-decoder architecture in transformer models is primarily used for:

a) Image classification

b) Text summarization

c) Speech recognition

d) Sentiment analysis

Correct answer: b) Text summarization

Q7. The scaled dot-product attention mechanism in transformers is primarily used for:

a) Reducing computational complexity

b) Increasing model capacity

c) Improving convergence speed

d) Enhancing interpretability

Correct answer: a) Reducing computational complexity

Q8. What is the purpose of positional encoding in transformer models?

a) Handling variable-length sequences

b) Capturing position information in the input sequence

c) Reducing overfitting

d) Introducing randomness into the model

Correct answer: b) Capturing position information in the input sequence

Q9. What does each attention head in a transformer model learn to focus on?

a) Entire input sequence

b) Only the first token

c) Different parts of the input sequence

d) Last token

Correct answer: c) Different parts of the input sequence

Q10. The purpose of layer normalization in transformer models is to:

a) Reduce model capacity

b) Speed up training

c) Improve convergence

d) Stabilize training

Correct answer: d) Stabilize training