LARGE LANGUAGE MODELS – ANSWER KEY

QUESTION 1:

Explain the architecture of llm.

ANSWER 1:

Input Embeddings: The input text is tokenized into smaller units, such as words or sub-words, and each token is embedded into a continuous vector representation. This embedding step captures the semantic and syntactic information of the input.

Positional Encoding: Positional encoding is added to the input embeddings to provide information about the positions of the tokens because transformers do not naturally encode the order of the tokens. This enables the model to process the tokens while taking their sequential order into account.

Encoder: Based on a neural network technique, the encoder analyses the input text and creates a number of hidden states that protect the context and meaning of text data. Multiple encoder layers make up the core of the transformer architecture. Self-attention mechanism and feed-forward neural network are the two fundamental sub-components of each encoder layer.

Self-Attention Mechanism: Self-attention enables the model to weigh the importance of different tokens in the input sequence by computing attention scores. It allows the model to consider the dependencies and relationships between different tokens in a context-aware manner.

Feed-Forward Neural Network: After the self-attention step, a feed-forward neural network is applied to each token independently. This network includes fully connected layers with non-linear activation functions, allowing the model to capture complex interactions between tokens.

Decoder Layers: In some transformer-based models, a decoder component is included in addition to the encoder. The decoder layers enable autoregressive generation, where the model can generate sequential outputs by attending to the previously generated tokens.

Multi-Head Attention: Transformers often employ multi-head attention, where self-attention is performed simultaneously with different learned attention weights. This allows the model to capture different types of relationships and attend to various parts of the input sequence simultaneously.

Layer Normalization: Layer normalization is applied after each sub-component or layer in the transformer architecture. It helps stabilize the learning process and improves the model's ability to generalize across different inputs.

Output Layers: The output layers of the transformer model can vary depending on the specific task. For example, in language modeling, a linear projection followed by SoftMax activation is commonly used to generate the probability distribution over the next token.

QUESTION 2:

Describe the working of BERT model.

ANSWER 2:

BERT is designed to generate a language model so, only the encoder mechanism is used. Sequence of tokens are fed to the Transformer encoder. These tokens are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors, each corresponding to an input token, providing contextualized representations.

When training language models, defining a prediction goal is a challenge. Many models predict the next word in a sequence, which is a directional approach and may limit context learning. BERT addresses this challenge with two innovative training strategies:

1. Masked Language Model (MLM)
2. Next Sentence Prediction (NSP)

1.Masked Language Model (MLM)

In BERT's pre-training process, a portion of words in each input sequence is masked and the model is trained to predict the original values of these masked words based on the context provided by the surrounding words.

In simple terms,

Masking words: Before BERT learns from sentences, it hides some words (about 15%) and replaces them with a special symbol, like [MASK].

Guessing Hidden Words: BERT's job is to figure out what these hidden words are by looking at the words around them. It's like a game of guessing where some words are missing, and BERT tries to fill in the blanks.

How BERT learns: BERT adds a special layer on top of its learning system to make these guesses. It then checks how close its guesses are to the actual hidden words. It does this by converting its guesses into probabilities, saying, "I think this word is X, and I'm this much sure about it."

Special Attention to Hidden Words: BERT's main focus during training is on getting these hidden words right. It cares less about predicting the words that are not hidden. This is because the real challenge is figuring out the missing parts, and this strategy helps BERT become really good at understanding the meaning and context of words.

2.Next Sentence Prediction (NSP)

BERT predicts if the second sentence is connected to the first. This is done by transforming the output of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.

1. In the training process, BERT learns to understand the relationship between pairs of sentences, predicting if the second sentence follows the first in the original document.
2. 50% of the input pairs have the second sentence as the subsequent sentence in the original document, and the other 50% have a randomly chosen sentence.
3. To help the model distinguish between connected and disconnected sentence pairs. The input is processed before entering the model:
   - A [CLS] token is inserted at the beginning of the first sentence, and a [SEP] token is added at the end of each sentence.
   - A sentence embedding indicating Sentence A or Sentence B is added to each token.
   - A positional embedding indicates the position of each token in the sequence.
4. BERT predicts if the second sentence is connected to the first. This is done by transforming the output of the [CLS] token into a 2×1 shaped vector using a classification layer, and then calculating the probability of whether the second sentence follows the first using SoftMax.