**Transformative Shifts: Exploring the Evolution of NLP Architectures:**

Sequence to Sequence model is basically on a given sequence we need to generate another sequence (This could be in another language)

Traditional RNN model suits well for this problem. Traditional RNN works basically using Encoding and Decoding format. Input is sent to Encoder. Output of Encoder is a high dimensional vector. Using this as an input for Decoder we generate the sequence. This makes a traditional RNN model.

But because of Vanishing Gradient / Exploding Gradient problem for Traditional RNN, we use LSTM to capture the long-term dependencies (if present) to make the model better.

Coming to LSTM model, Based on Training data, it uses Input gate, Output gate and Forget gate and updates cell state. Using cell state, it solves the Exploding gradient and vanishing gradient problem of Traditional RNN.

Attention Mechanism is basically generating a context vector which stores importance of each word with respect to context of the sentence. Attention mechanism was used previously with RNN.

But RNN are very slow. In fact, all the architectures which use RNNs are very slow because, they follow step by step input-output generation. To make it faster we need parallel processing capabilities. So transformer architecture came in.

Transformer architecture uses parallel processing capabilities and Attention mechanism.

BERT and GPT uses these Transformer Architecture.

BERT - Bidirectional Encoding focuses on Encoding, used for Classification problems.

GPT - focuses on decoding and used for text generation.

To make BERT use decoding we have two ways,

1. Add a decoder

2. Masked Language Modeling (Train the BERT to masking a certain part of tokens in a sequence and trying to predict them using corresponding tokens)

Coming to RAG architecture,

RAG (Retrieval augmented generation), LLMs are way big and they hallucinate while generating the output. To make them concentrate (increase the quality of the model) on certain things we try to put the information that we want llm to focus on (files, images etc..,) at a place. These files are converted into vectors to make search faster. The store where we put files after Embedding is called Vector store.

So when given a prompt, prompt is converted to vector using embeddings and then we go into vector store and search for relevant information and retrieve the info and augment it to the prompt and then we go to llm and ask it. Then we get the output.

There are other methods as well to increase the quality of the LLM.

1. Fine Tuning (Training the model - Takes time) - basically used to make model adapt to a certain task.
2. Prompt Engineering – to guide the models behaviour.