Megathon-23

LLM:

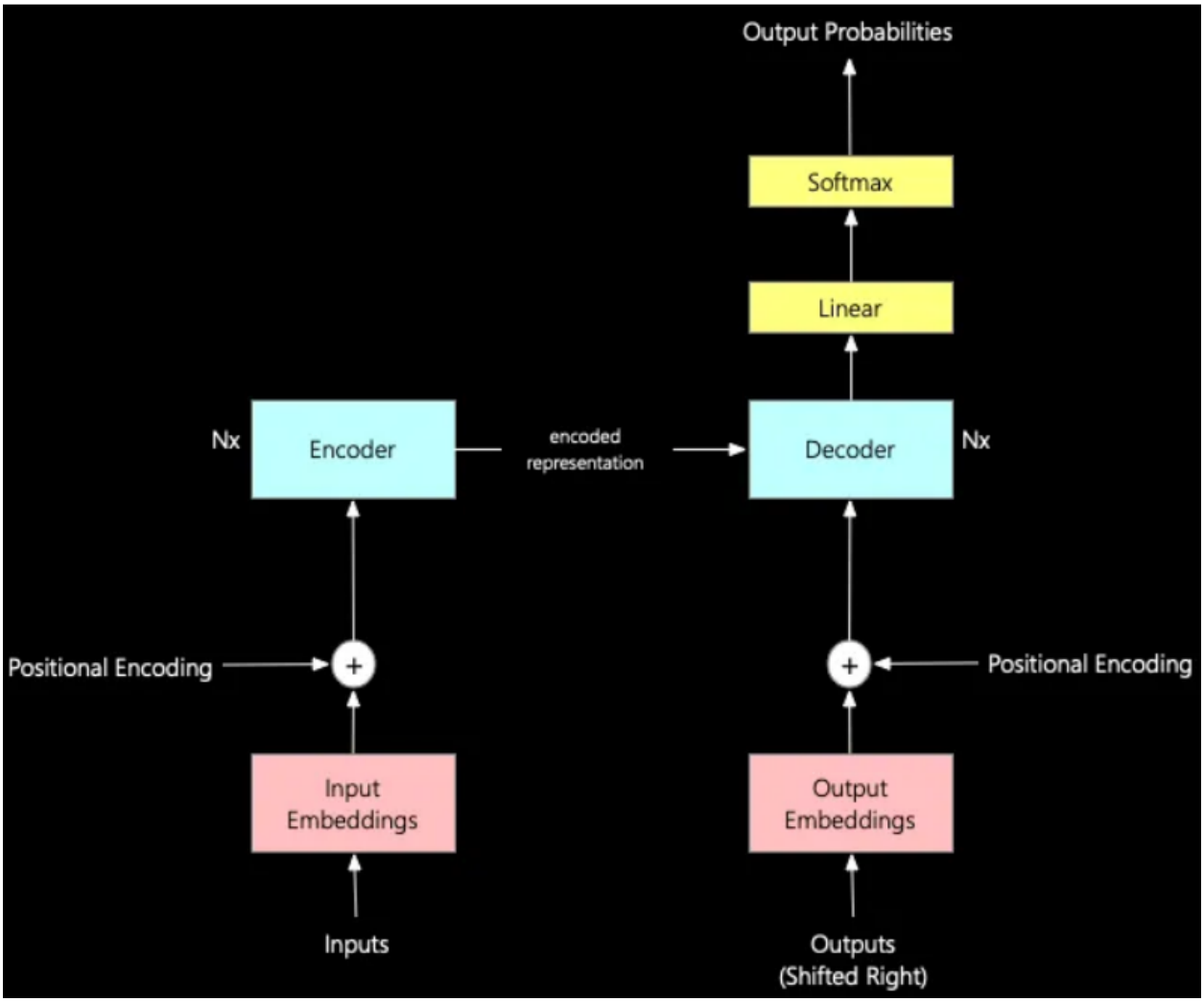
Large Language Models (LLMs) are complex neural network architectures that have revolutionized natural language processing (NLP) tasks. These models are composed of several key components that work together to enable them to understand, generate, and manipulate human language with remarkable fluency and accuracy. Here are the components of LLM :

- **Tokenization :** Tokenization marks the foundational step in the **evolution of large language models(LLMs)**, where text sequences undergo division into smaller units or tokens.
- **Embedding :** Embeddings are integral to LLMs' large-scale operation. These are continuous vector representations of tokens that capture semantic information.

- **Attention:** Self-attention mechanisms analyze the relationships between all tokens in a sequence, facilitating the capture of long-range dependencies. In large models, this attention mechanism is highly parallelizable, enabling efficient processing of extensive sequences
- **Pre-training :** The vast size of LLMs is harnessed through pre-training on massive datasets.These pre-trained models become repositories of language expertise, which can then be fine-tuned for specific tasks using smaller datasets.
- **Transfer Learning :**Transfer learning is a deep learning approach that uses a model trained for one task as a starting point for a model that performs a similar task. Transfer learning is popular in deep learning because it enables the training of deep neural networks with less data compared to having to create a model from scratch.
- **Generation Capacity :** LLM's can produce coherent and contextually relevant text across various domains. The extensive exposure during training enables them to mimic human-like language use, making them versatile tools for tasks like content generation, translation, summarization, and much more.

Architecture of LLM :

The transformer architecture is the fundamental building block of all Language Models with Transformers (LLMs)



- **Inputs and Input Embeddings**: The tokens entered by the user are considered inputs for the machine learning models. However, models only understand numbers, not text, so these inputs need to be converted into a numerical format called “input embeddings.” Input embeddings represent words as numbers, which machine learning models can then process. These embeddings are like a dictionary that helps the model understand the meaning of words by placing them in a mathematical space where similar words are located near each other.

- **Positional Encoding**: In natural language processing, the order of words in a sentence is crucial for determining the sentence’s meaning. They can be used to encode the position of each word in the input sequence as a set of numbers. These

numbers can be fed into the Transformer model, along with the input embeddings. By incorporating positional encoding into the Transformer architecture, GPT can more effectively understand the order of words in a sentence and generate grammatically correct and semantically meaningful output.

- **Encoder**: The encoder is part of the neural network that processes the input text and generates a series of hidden states that capture the meaning and context of the text.The transformer encoder comprises multiple self-attention and feed-forward layers, allowing the model to process and understand the input sequence effectively.

- **Outputs (shifted right)**: During training, the decoder learns how to guess the next word by looking at the words before it. To do this, we move the output sequence over one spot to the right. That way, the decoder can only use the previous words.

- **Output Embeddings**: Models can only understand numbers, not text, like input embeddings. So the output must be changed to a numerical format, known as “output embeddings.” Output embeddings are similar to input embeddings and go through positional encoding, which helps the model understand the order of words in a sentence.

- **Decoder**: The positionally encoded input representation and the positionally encoded output embeddings go through the decoder. The decoder is part of the model that generates the output sequence based on the encoded input sequence. During training, the decoder learns how to guess the next word by looking at the words before it.

- **Linear Layer and Softmax**: After the decoder produces the output embeddings, the linear layer maps them to a higher-dimensional space. This step is necessary to transform the output embeddings into the original input space. Then, we use the softmax function to generate a probability distribution for each output token in the vocabulary, enabling us to generate output tokens with probabilities.

CHUNKING ? WHY CHUNKING?

Chunking is the process of grouping different bits of information together into more manageable or meaningful chunks.

But why ?

Yes, right we don’t have enough memory it seems……

Chunking helps reduce memory usage by breaking down large text data into smaller, more manageable segments or chunks. By processing and analyzing these smaller chunks individually, the memory required to store and manipulate the data is significantly reduced. This allows for more efficient memory usage and better performance of the language model, especially when dealing with large amounts of text data.

WHAT SHOULD BE THE CHUNKING CONSIDERATIONS ?

Several variables play a role in determining the best chunking strategy, and these variables vary depending on the use case. Here are some key aspects to keep in mind:

1. **What is the nature of the content being indexed?** Determining the nature of medical content, whether they are lengthy medical documents, concise patient reports, or complex medical queries, is essential. This influences the selection of an appropriate chunking strategy tailored to the type of medical data your chatbot will handle.
2. **Which embedding model are you using, and what chunk sizes does it perform optimally on?** For instance, sentence-transformer models work well on individual sentences, but a model like text-embedding-ada-002 performs better on chunks containing 256 or 512 tokens.
3. **What are your expectations for the length and complexity of user queries?** Will they be short and specific or long and complex? This may inform the way you choose to chunk your content as well so that there's a closer correlation between the embedded query and embedded chunks.
4. **How will the retrieved results be utilized within your specific application?** For example, will they be used for semantic search, question answering, summarization, or other purposes? For example, if your results need to be fed into another LLM with a token limit, you'll have to take that into consideration and limit the size of the chunks based on the number of chunks you'd like to fit into the request to the LLM.

Answering these questions will allow you to develop a chunking strategy that balances performance and accuracy, and this, in turn, will ensure the query results are more relevant.

Which chunking method do we aim to use ?

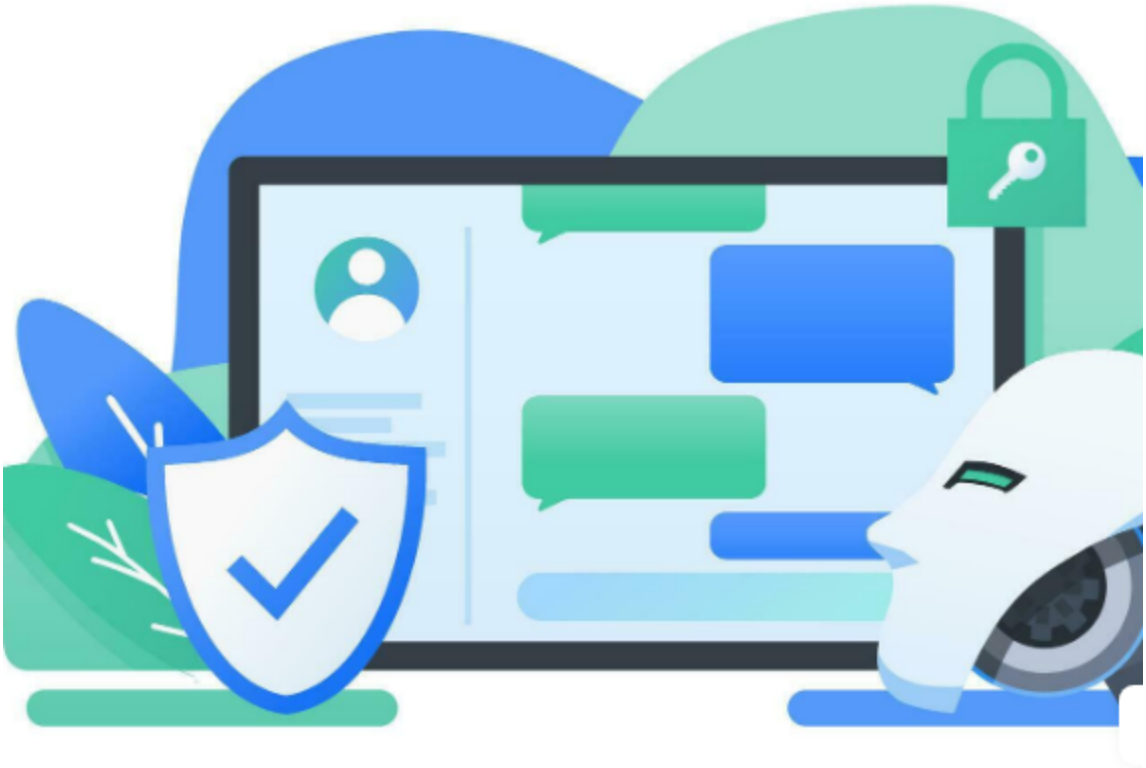
Fixed-size chunking ?

No Because the length of all chunks is fixed in the process of chunking, if a byte in the file is shifted, it will cause all the subsequent chunks to be different from the original chunks

Content-Aware chunking ?

In the context of medical data, employing content-aware chunking is recommended. This method involves **understanding the format and nature of the medical content**, enabling the generation of chunks that align with the inherent structure and context of the information. It adapts the chunks according to the content's specific attributes, ensuring a more contextually relevant and efficient chunking process.

DATA PRIVACY



In the evolving landscape of healthcare technology, the integration of artificial intelligence (AI) is significantly transforming how medical information is processed and utilized. However, this rapid advancement in technology raises significant concerns regarding the privacy and confidentiality of patient data, particularly in the context of medical queries handled by AI-powered systems.

The surge in data shared online can potentially offer a wealth of information to enhance the accuracy and relevance of the responses generated by a medical chatbot. Nevertheless, this data often encompasses sensitive and personal patient information. Respecting and safeguarding patient privacy is paramount to prevent any inadvertent disclosure of personal or sensitive data during the interactions with the chatbot.

Ensuring Privacy in Medical Queries:

The information shared by patients during interactions with the medical chatbot must be handled with the utmost care to prevent any leakage of personal or sensitive information. The system must be designed to prioritize patient privacy and confidentiality, refraining from any misuse or unauthorized access to sensitive data.

It is crucial to develop robust measures and protocols that adhere to strict privacy guidelines, ensuring that the AI system does not exploit or misuse the shared data. Implementing encryption, anonymization, and stringent access controls can safeguard sensitive patient information from unauthorized exposure or exploitation.

As AI continues to embed itself further into the healthcare domain, maintaining a vigilant stance on protecting patient privacy is essential to uphold ethical and responsible use of technology. This approach fosters trust between patients and the AI-powered medical systems, reassuring individuals that their data is handled with the highest ethical standards and serves to benefit their healthcare without compromising their confidentiality.

BUT HOW TO ADDRESS ?

As we embed AI into the realm of healthcare through our medical chatbot, the potential for innovation is vast, but it accompanies significant apprehensions surrounding privacy and ethical use. Therefore, our focus is on ensuring the utmost confidentiality and ethical conduct in the development and operation of our health-oriented AI system.

Regarding data management, our approach encompasses two key propositions. Firstly, we offer users the ability to retain an interaction history while empowering them with

complete control to delete any personal information they consider sensitive. For instances involving personal and sensitive medical data, a robust mechanism for permanent deletion will be in place, aligning with privacy regulations and building trust with our users.

User authentication is paramount, ensuring that only authorized users can access their personal information. This stringent access control mechanism serves to preserve the confidentiality of medical discussions and interactions, fostering a foundation of trust between our system and the users it serves.

In situations where users inadvertently share highly personal medical data, such as billing information containing contact or address details, our system will refrain from storing such sensitive information. Immediate deletion protocols will be implemented to avert any potential breach, reiterating our core commitment to assist patients while safeguarding their privacy.

Our foremost commitment is to provide effective assistance to users while maintaining an unwavering focus on preserving their privacy and trust. This dedication underscores our primary goal in the development and deployment of our medical chatbot.

CACHING GENERATIVE LLM

A cache is a place to store data temporarily so that it can be reused, and the process of storing this data is called caching. Here the most frequently accessed data is stored to be accessed more quickly. This has a drastic effect on the performance of the processor. Imagine the processor performing an intensive task requiring a lot of computation time. Now imagine a situation where the processor has to perform the exact computation again. In this scenario, caching the previous result really helps. This will reduce the computation time, as the result was cached when the task was performed.

Addressing Efficiency in Medical Chatbot Responses

In the context of our medical chatbot, optimizing the efficiency of responses while reducing costs related to API calls is crucial. One viable approach involves implementing a caching system for prompts and their corresponding responses. This method aims to curtail the need for redundant API calls by storing and retrieving previously generated responses.

Implementing Caching for Enhanced Performance

The LangChain library offers an effective solution for caching via its built-in function known as InMemoryCache. Leveraging this functionality enables the storage and retrieval of responses from the language models, eliminating the necessity for repetitive API calls.

Performance Enhancement and Response Time Reduction

Utilizing caching contributes to performance enhancements, albeit indirectly. By caching responses that originally demanded substantial processing time, we eliminate the need for recalculating them. This process allows the system to retrieve the stored responses directly, enabling the processor to allocate its resources more efficiently.

In the case of Large Language Models within our medical chatbot, both prompts and responses are cached. Upon encountering similar queries, the system retrieves the corresponding response from the cache rather than requesting it from the model. This mechanism significantly reduces the response time, as it retrieves the response directly from the cache, bypassing the need to query the model and wait for a new response.

Implementing this caching strategy in our medical chatbot not only optimizes response times but also minimizes the load on the model, enhancing the overall efficiency of the system's interactions and responses to medical queries.

HALLUCINATION IN AI:

Within medical AI, the concept of AI hallucinations, where AI models generate and present incorrect information as facts, poses a critical concern. These hallucinations stem from the AI's limitation in reasoning and logical application, potentially leading to the generation of erroneous or even harmful content.

Relevance of Addressing AI Hallucinations in Medical Context

In the medical domain, AI hallucinations pose a significant risk. Given the critical nature of medical information, any inaccurate or misleading outputs can have profound implications. Misleading patients or healthcare providers with false or biased information can impact decision-making and treatment protocols, potentially leading to adverse outcomes.

Understanding AI Hallucinations in Medical Chatbots

AI hallucinations in medical chatbots occur due to the AI's tendency to generate responses that aim to please the user without the ability to ascertain factual accuracy or

potential harm in the generated content. This scenario emphasizes the crucial need for mechanisms to prevent the dissemination of incorrect or misleading medical information.

Strategies to Mitigate AI Hallucinations in Medical AI Systems

1. Direct AI Responses within Medical Scope:
Providing specific guidelines and instructions is paramount. Emphasize focusing on relevant medical data, such as symptoms or treatment options, while avoiding unnecessary or potentially misleading information. For instance, in a discussion about a disease, guide the AI to prioritize critical and verified medical data.

2. Ensure Credible and Contextually Accurate Data:
Employing high-quality, verified medical sources as input data for the AI is crucial. Feeding the AI model with precise and reliable medical information minimizes the likelihood of generating misleading or erroneous content.

3. Implement Structured Data Frameworks:
Developing a structured framework or template tailored to medical contexts guides the AI's responses. This framework familiarizes the AI model with the expected structure and content, reducing the chance of generating misleading outputs.

QUANTIZED LLM :

LIAMA WITH DIFFERENT SIZES

With the advancement of technology, the trend toward using smaller devices like cell phones or tablets for diverse tasks has grown significantly. However, accommodating sophisticated artificial intelligence programs such as Large Language Models (LLMs) on these smaller devices poses challenges due to the considerable memory and computing power requirements.

Quantization: Tailoring LLMs for Accessibility

Quantization serves as a solution to optimize LLMs for utilization on smaller devices. This process streamlines these models, reducing their memory and computing demands while striving to maintain a satisfactory level of accuracy. This optimization aims to make LLMs more accessible, particularly for users without access to high-end GPUs like Nvidia A100.

Understanding Quantization in the Context of LLMs

Consider quantization as a means of compressing data, much like reducing the quality of an image to save storage space. In the case of LLMs, this involves "compressing" the models by adjusting specific parameters and reducing computational precision. This process, while potentially resulting in a slight loss of detail or precision, enables these models to function more efficiently on less powerful hardware.

When performing quantization, transitioning from float32 to int8 poses specific challenges. The precision difference between these formats—only allowing 256 values in int8 versus a broader range in float32—requires careful consideration during the quantization process.