ASME DTOG 2024 AI/ML Workshop

Understanding Large Language Models

A Simple Explanation of Key Components

Large language models (LLMs) are advanced artificial intelligence systems designed to understand and generate human language. They have revolutionized fields like natural language processing, translation, and automated writing. In this document, we'll explore the four key components of large language models: the Transformer architecture, parameters, tokens, and the context window/length.

# Transformer Architecture

The Transformer architecture is the backbone of large language models. Introduced by Vaswani et al. in their 2017 paper "Attention is All You Need," this architecture has become a fundamental building block for modern AI language models.

## How Transformers Work

Transformers are designed to handle sequential data, such as sentences, more efficiently than previous models like recurrent neural networks (RNNs). They use a mechanism called "self-attention" to weigh the importance of each word in a sentence relative to every other word. This allows the model to capture the context and nuances of language better.

## Self-Attention Mechanism

Self-attention works by assigning different weights to each word in a sentence, depending on its relevance to other words. For example, in the sentence "The cat sat on the mat," the model needs to understand that "cat" and "mat" are more related than "cat" and "the." This contextual understanding is achieved through multiple layers of attention, allowing the model to build complex representations of language.

# Parameters

Parameters are the internal settings of a model that are learned during training. In the context of large language models, parameters usually refer to the weights and biases in the neural network.

## The Role of Parameters

Parameters determine how the input data is transformed as it passes through the network. During training, the model adjusts its parameters to minimize the error between its predictions and the actual outcomes. The more parameters a model has, the more complex patterns it can learn. This is why larger models, such as GPT-3 with its 175 billion parameters, can generate more sophisticated and coherent text.

## Training and Fine-Tuning

Training a large language model involves feeding it vast amounts of text data and adjusting its parameters accordingly. This process requires significant computational power and time. After the initial training, models can be fine-tuned on specific tasks or datasets to improve their performance in particular areas.

# Tokens

Tokens are the pieces of text that the model processes. They can be as small as individual characters or as large as words and phrases. In many large language models, tokens are typically words or subwords.

## Tokenization

Before feeding text into a language model, it must be tokenized. Tokenization is the process of breaking down text into these smaller units. For instance, the sentence "The quick brown fox" might be tokenized into ["The", "quick", "brown", "fox"]. Some models use subword tokenization, breaking words into smaller units like "qui" and "ck" to handle rare or out-of-vocabulary words more effectively.

## Importance of Tokens

Tokens are crucial because they are the basic units the model uses to understand and generate language. The choice of tokenization strategy can significantly impact the model's performance. Efficient tokenization helps the model process text faster and more accurately.

# Context Window/Length

The context window, or context length, refers to the number of tokens a model can consider at once. This determines how much text the model can "see" when making predictions.

## Impact of Context Window

A larger context window allows the model to understand longer dependencies in text. For example, in a long paragraph, the meaning of a word might depend on something mentioned several sentences earlier. A larger context window helps the model maintain coherence over longer passages of text. However, longer context windows also require more memory and computational resources.

## Practical Considerations

In practice, the size of the context window is a trade-off between computational efficiency and the ability to capture long-range dependencies. Most large language models have a fixed context window size, and choosing the right size depends on the specific application and available resources.

# Conclusion

Large language models are powerful tools for understanding and generating human language, and their capabilities are driven by several key components: the Transformer architecture, parameters, tokens, and the context window/length. By leveraging these elements, LLMs can perform a wide range of language tasks with impressive accuracy and fluency. As the field continues to evolve, we can expect even more advancements in the capabilities and applications of these models.

Generating Embeddings in Large Language Models

An Overview and Example

# Introduction

Embeddings are fundamental to the functioning of large language models (LLMs) like Transformers. They convert words or tokens from text into dense vectors of real numbers, which capture semantic meanings and relationships. This allows the model to process text mathematically and understand context effectively.

# How Embeddings Are Generated

The process of generating embeddings typically involves several steps:

## 1. Tokenization

Before generating embeddings, the text is tokenized. Tokenization involves breaking down the text into smaller units, such as words, subwords, or characters, depending on the specific model and its vocabulary. For example, the sentence "The cat sat on the mat" might be tokenized into ["The", "cat", "sat", "on", "the", "mat"].

## 2. Embedding Layer

The tokenized text is then passed through an embedding layer, which maps each token to a fixed-size vector. These vectors are typically initialized randomly and then learned during the training process. The embedding layer can be thought of as a lookup table where each token has a corresponding vector.

For instance, the word "cat" might be mapped to a vector like:

\[ \text{cat} \rightarrow [0.25, -0.12, 0.84, \ldots] \]

## 3. Contextualization

While the initial embeddings capture some basic relationships between words, they are static and do not consider the specific context of each word within a sentence. To address this, LLMs use mechanisms like self-attention and feed-forward networks to generate contextualized embeddings. These embeddings are dynamic and reflect the meaning of each word in its particular context.

# Example

Let's consider a simple example to illustrate how embeddings are generated and used in a sentence.

Suppose we have the sentence: "The quick brown fox."

## Step 1: Tokenization

The sentence is tokenized into individual words: ["The", "quick", "brown", "fox"].

## Step 2: Initial Embeddings

Each token is mapped to an initial embedding vector. For simplicity, let's assume our vectors are 3-dimensional:

- "The" → [0.1, 0.2, 0.3]

- "quick" → [0.4, 0.5, 0.6]

- "brown" → [0.7, 0.8, 0.9]

- "fox" → [1.0, 1.1, 1.2]

## Step 3: Contextualization

Conclusion

Embeddings are a crucial component of large language models, enabling them to represent and process text effectively. By converting words into dense vectors and contextualizing them within sentences, embeddings allow LLMs to understand and generate human language with remarkable precision and coherence. This process of tokenization, embedding, and contextualization forms the backbone of many modern natural language processing applications.