Large Language Models

Author: Aditya Maddineni

Introduction

It all started from the paper ‘Attention is all you Need’ .LLM is a statistical tool which determines the probability of a given sequence of words occurring in a sentence. In a most simpler way you can say

Feed in bunch of text and the model calculates what comes next

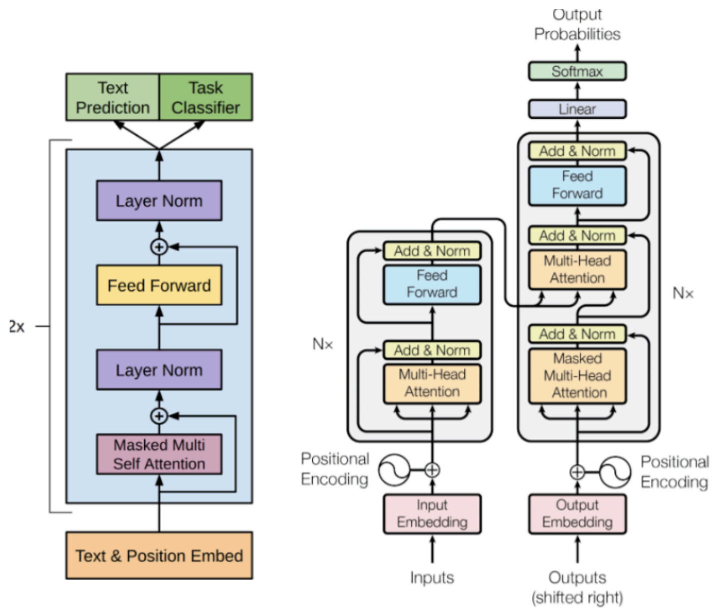
As you know these models have sparked a revolution in how machines understand and generate human-like text.

The functioning of large language models can be broken down into two main phases:

pre-training: Exposing the model to vast and diverse text data from the internet. During this phase, the model learns language structure, grammar, vocabulary, and context by predicting the next word in sentences. It forms a foundation for fine-tuning the model for specific natural language processing tasks.

fine-tuning:, while retaining its general language understanding acquired during pre-training. Second phase in training large language models. After pre-training, the model is adapted to specific tasks or domains by training it on task-specific data. This process helps the model specialize in tasks like text classification, translation, or summarization.

Transformer architecture :



Multi-Head Self-Attention:

- Each word in the sequence is first transformed into an embedding vector. These embeddings capture the semantic meaning of each word.
- Each attention head computes a set of attention weights for each word in the input sequence. These attention weights determine how much each word should focus on other words in the sequence.

Positional Encodings:

Positional encodings are typically calculated using trigonometric functions like sine and cosine.

Feedforward Neural Networks:

Layer Normalization and Residual Connections

Vocabulary Embedding:

Positional Embeddings:

Autoregressive Generation:

Prompting and Conditioning

Fine-tuning

For an over view

```
  Start
    V
Input Text --> Tokenization --> Transformer Layers --> Output Text
    |
    |
    V
   End
```

Different Models and their use cases:

1. GPT-3 (Generative Pre-trained Transformer 3):

- Use Cases:
- Content Generation: GPT-3 can generate human-like text for articles, stories, or creative writing.
- Chatbots and Virtual Assistants: It can be used to create conversational agents that can engage with users in natural language.
- Language Translation: GPT-3 can translate text between languages.
- Summarization: It can summarize long articles or documents.
- Text Completion: GPT-3 can help users complete sentences or paragraphs.

2. BERT (Bidirectional Encoder Representations from Transformers):

- Use Cases:
- Search Engines: BERT helps improve search engine results by understanding the context of search queries.

-Sentiment Analysis: It can analyze the sentiment of text, which is useful for social media monitoring and customer feedback analysis.

- Named Entity Recognition (NER): BERT is used for extracting named entities like names, locations, and organizations from text.

-Question Answering: It's used in QA systems to answer questions based on passages of text.

3. RoBERTa (A Robustly Optimized BERT Pretraining Approach):

- Use Cases:
- Text Classification: RoBERTa is used for classifying text into categories such as spam detection or topic classification.
- Language Understanding: It can be used to understand user intent in natural language understanding systems.
- Text Generation: RoBERTa can generate coherent and contextually relevant text.

4. T5 (Text-to-Text Transfer Transformer):

- Use Cases:
- Data-to-Text Generation: T5 can generate natural language descriptions from structured data, which is useful in data reporting.
- Language Translation: Like GPT-3, T5 can also be used for language translation tasks.
- Text Summarization: T5 is excellent at summarizing text, making it useful for news aggregation and document summarization.

5. ALBERT (A Lite BERT):

- Use Cases:
- Efficient Language Understanding: ALBERT is designed for efficient use of resources and is used in applications where resource consumption is a concern.
- Recommendation Systems: It can be used to understand user preferences and recommend products or content.

6. XLNet (eXtreme Learning with Transformers):

- Use Cases:
- Text Generation: XLNet can generate creative and coherent text in various styles.
- Language Modeling: It's used for tasks like language modeling and text completion.
- Paraphrasing: XLNet can be used for automatic paraphrasing of text.

7. BART (BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension):

- Use Cases:
- Text Generation: BART is specifically designed for natural language generation tasks and can be used for content generation, summarization, and paraphrasing.
- Language Translation: It can perform translation tasks effectively.

8. ERNIE (Enhanced Representation through Knowledge Integration):

- Use Cases:

-Semantic Search: ERNIE can be used in search engines for better understanding the semantics of queries and documents.

- Knowledge Graphs: It can help in building and expanding knowledge graphs by extracting information from text

Vector databases

vector databases are a specialized class of databases designed to manage and query vector data efficiently. They play a crucial role in applications where high-dimensional data needs to be stored, retrieved, and analyzed rapidly, making them an integral part of many data-driven industries.

Store and manage

Vector databases store and manage vector data through specialized data structures and indexing techniques to ensure efficient storage and retrieval. Here's how they accomplish this:

Indexing: Vector databases use various indexing methods optimized for vector data. One common indexing technique is the use of tree structures like k-d trees or ball trees. These structures partition the vector space into smaller regions, making it faster to search for nearest neighbors or similar vectors.

Quantization: In some cases, vector databases use quantization to reduce the dimensionality of the data. This involves mapping high-dimensional vectors to lower-dimensional codes, which can significantly speed up similarity searches. However, it may come at the cost of reduced precision.

Metric Spaces: Vector databases often leverage specific distance metrics suitable for the type of data they store. For example, Euclidean distance is commonly used, but other metrics like cosine similarity are also applied depending on the data and the application.

Parallel Processing: To improve query performance, vector databases may use parallel processing and distributed computing techniques. This allows for the distribution of queries across multiple nodes or servers, accelerating search times.

Optimized Data Structures: Vector databases may employ data structures that are specifically designed for vector data, such as inverted indexes, product quantization, or locality-sensitive hashing (LSH). These structures help speed up queries and optimize storage.

Compression Techniques: To minimize storage space, vector databases often use compression techniques. These techniques reduce the size of vector data while allowing efficient decompression when needed.

Scalability: Many vector databases are designed to be horizontally scalable. This means they can distribute data across multiple nodes or servers and continue to perform well as data volumes grow. This scalability is crucial for applications with massive datasets.

Vector Data Modeling:Vector databases may allow for customized data modeling, enabling users to define how vector data should be represented and indexed. This flexibility is essential in various applications where the nature of vector data can vary significantly.

URL: How AI chatbots like ChatGPT or Bard work – visual explainer