Thetawave.ai

aaa

Generated by Thetawave.ai

📚 Generative AI & Large Language Models (LLMs)

Overview of Generative AI

Generative AI refers to algorithms that can create new content, including text, images, and more, by learning from existing data. One key application of generative AI is through **Large Language Models (LLMs)**.

Use Cases of LLMs

LLMs can be employed in various tasks, including:

- **Chatbots**: Automating conversations and providing instant responses.
- **Summarization**: Condensing large texts into shorter versions while retaining essential information.
- **Translation**: Converting text from one language to another.
- **Essay Writing**: Assisting in generating written content based on prompts.

Model Lifecycle

The lifecycle of an LLM includes:

1. **Pre-training**: The model is trained on a large corpus of text to learn language patterns.
2. **Fine-tuning**: The model is adapted for specific tasks using smaller, task-specific datasets.

ChatBot Example

The street layout of Washington D.C. was designed by **Pierre Charles L'Enfant**, a French-born American architect and civil engineer. This demonstrates the ability of LLMs to retrieve historical information accurately.

🤖 Understanding Transformers

Architecture of Transformers

Transformers are a type of model architecture that enables efficient processing of sequential data. Key components include:

- **Encoder-Decoder Structure**: The encoder processes the input, while the decoder generates the output.
- **Self-attention Mechanism**: This allows the model to focus on different parts of the input when generating output, enhancing contextual understanding.

Working of Transformers

- **Input Processing**: The input is first tokenized and converted into embeddings.
- **Positional Encoding**: Adds information about the position of each token in the sequence to the embeddings.
- **Multi-headed Self-attention**: Allows the model to attend to multiple parts of the input simultaneously, improving understanding.

Attention Mechanism

> "The attention mechanism in transformers measures the distance between words, allowing the model to determine which words are most relevant to each other."

Example: Self-Attention

For a sentence like "The teacher taught the student with the book," the self-attention mechanism evaluates the relationship between each word to understand the context better.

🔍 In-Context Learning (ICL)

Zero-shot Inference

In zero-shot inference, the model makes predictions without any prior examples. For instance, when asked to classify a movie review, it can determine sentiment based on the context provided.

One-shot and Few-shot Inference

- **One-shot Inference**: The model uses one example to guide its prediction.
- **Few-shot Inference**: The model benefits from several examples to enhance its understanding of the task.

Inference Configuration Parameters

The generative configuration for inference includes parameters such as:

- **Temperature**: Controls randomness in predictions (higher values lead to more random outputs).
- **Max New Tokens**: Limits the number of tokens generated in a single response.

Parameter	Description
Temperature	Controls the randomness of predictions
Max New Tokens	Maximum number of tokens generated
Sample Top K	Limits the selection to the top K tokens
Sample Top P	Adjusts the selection based on cumulative probabilities

Sampling Strategies

- **Greedy Sampling**: Selects the token with the highest probability.
- **Random Sampling**: Selects a token randomly based on its probability distribution.

🚀 Examples of LLM Outputs

1. **Translation**: For the French phrase "J'aime l'apprentissage automatique," the output is "I love machine learning."
2. **Sentiment Analysis**: Classifying a review, such as "I loved this movie!" with a positive sentiment.

Summary of LLM Capabilities

LLMs are capable of performing complex tasks such as translation and sentiment analysis through sophisticated algorithms and architectures, making them powerful tools in natural language processing.

Generative AI Sampling Techniques

Top-K Sampling

- **Definition**: Top-K sampling involves selecting an output from the top k results based on a random-weighted strategy using the probabilities.
- **Example**: When $k = 3$, the model considers the top three predictions.

Top-P Sampling

- **Definition**: Top-P sampling selects an output using the random-weighted strategy that includes the top-ranked consecutive results by probability, ensuring the cumulative probability is less than or equal to p.
- **Example**: If the probabilities are for `cake`, `donut`, `banana`, and `apple` as:
  - cake: 0.20
  - donut: 0.10
  - banana: 0.02

- apple: 0.01
- With $p = 0.30$, it will select from `cake` and `donut`.

🌡️ Temperature in Sampling

Temperature Setting

- **Description**: The temperature setting affects the probability distribution of the outputs.
  - **Cooler Temperature (< 1)**: Leads to a stronger peaked probability distribution.
  - **Higher Temperature (> 1)**: Results in a broader, flatter probability distribution.

Example of Temperature Effects

Temperature	Distribution Type	Probabilities
< 1	Strongly peaked	apple: 0.40, banana: 0.12, cake: 0.08, donut: 0.04
> 1	Broader, flatter	apple: 0.15, banana: 0.12, cake: 0.04, donut: 0.08

🛠️ Generative AI Project Lifecycle

Stages of the Lifecycle

Stage	Description
Define the Use Case	Identify the specific application and objectives of the generative AI model.
Scope	Determine the parameters and limitations of the project.
Choose a Model	Select an existing model or pre-train your own.
Prompt Engineering	Design prompts to effectively communicate with the model.
Fine-Tuning	Adjust the model to align with human feedback and specific needs.
Evaluate	Assess the model's performance and efficacy.
Application Integration	Optimize and deploy the model for inference and practical use.

Key Considerations

- **Good at many tasks**: Utilize APIs to invoke actions for tasks like summarization, essay writing, translation, and information retrieval.
- **Good at a single task**: Focus on specific applications and streamline the API usage.

📊 Model Choices and Architectures

Considerations for Choosing a Model

Model Type	Description
Pre-trained LLM	Models that are ready to use without further training.
Custom LLM	Tailored models built for specific use cases.
Foundation Model	Base models that can be fine-tuned for various applications.
Train Your Own	Building models from scratch for specialized needs.

Model Architectures

- **Transformers**: Core architecture for modern generative models.
- **Pre-training Objectives**: Different tasks during training that shape how the model learns (e.g., Causal Language Modeling, Autoencoding).

📉 Quantization

Overview of Quantization

- **Purpose**: Reduces the memory required to store and train models by converting to lower precision spaces.
- **Types**:
  - **FP32**: 32-bit floating point.
  - **FP16**: 16-bit floating point.
  - **INT8**: 8-bit integer.

Quantization Type	Memory Needed to Store One Value	Range
FP32	4 bytes	From -3e38 to +3e38
FP16	2 bytes	-65504 to +65504
INT8	1 byte	-128 to +127

Benefits of Quantization

- **Memory Efficiency**: Reduces required memory to store and train models significantly.
- **Quantization-aware Training (QAT)**: Adjusts the model during training to accommodate quantization scaling factors.

Computational Considerations

GPU Memory Requirements

- **Approximate GPU RAM Needed for 1B Parameters**:
  - **Full Precision (FP32)**: 4GB
  - **Half Precision (FP16)**: 2GB
  - **Quantized (INT8)**: 1GB

Multi-GPU Training Strategies

Strategy	Description
Distributed Data Parallel (DDP)	Splits the training data across multiple GPUs.
Fully Sharded Data Parallel (FSDP)	Optimizes memory usage during training by sharding the model.

Model Size vs. Time

- The growth in model sizes correlates with improvements in transformer technology, dataset availability, and computing power, leading to more capable generative models.

Data Overlap Between GPUs

Memory Usage in Deep Learning

Model Parameters and Memory Considerations

- **Model Parameters (Weights)**:
  - Each parameter requires **4 bytes**.
- **Adam Optimizer**:
  - Consists of **2 states**, requiring an additional **8 bytes** per parameter.
- **Gradients**:
  - Requires **4 bytes** per parameter.
- **Activations and Temporary Memory**:
  - Estimated at **8 bytes** per parameter.

Total Memory Calculation

The total memory usage can be summarized as:

$$\text{Total Memory} = 4\ (\text{model}) + 20\ (\text{extra})\ \text{bytes per parameter}$$

Distributed Data Parallel (DDP)

- In DDP, each GPU holds a **full copy** of the model and training parameters.
- The process of DDP includes multiple stages:
  - **Forward/Backward Pass**: Each GPU performs these operations.
  - **Synchronize Gradients**: Gradients are synchronized across all GPUs.
  - **Update Model**: The model parameters are updated after synchronization.

Zero Redundancy Optimizer (ZeRO)

Purpose of ZeRO

- **ZeRO** reduces memory usage by distributing (sharding) the model parameters, gradients, and optimizer states across multiple GPUs.

Stages of ZeRO

Stage	Description
ZeRO Stage 1	Distributes optimizer state and gradients across GPUs.
ZeRO Stage 2	Further sharding of model parameters and optimizer states.
ZeRO Stage 3	Full sharding across all components, minimizing redundancy.

Fully Sharded Data Parallel (FSDP)

- FSDP operates similarly to DDP but with enhanced memory efficiency.
- It helps in reducing overall GPU memory utilization.
- Supports offloading to CPU as needed.

Configuration Options

Sharding Factor	Description
1 GPU (Full Replication)	No sharding; full model on one GPU.
Max. Number of GPUs	Full sharding distributes across all GPUs.
Hybrid Sharding	Combination of full and partial sharding.

Compute Budget for Training Large Language Models (LLMs)

Key Considerations

- **Compute Budget**: Measured in **petaop/s-day** (floating point operations per second for one day).

- e.g., 1 petaop/s-day = 1 trillion operations per second for a day.

Scaling Choices for Model Performance

Scaling Choice	Impact on Performance
Dataset Size	Number of tokens influences training quality.
Model Size	Number of parameters affects performance.

Chinchilla Scaling Laws

- Models may be **over-parameterized** and **under-trained**.
- Optimal model training can be achieved with smaller models trained on larger datasets.

Pre-training for Domain Adaptation

Legal Language Examples

- Demonstrates how specific legal terms are used in context, highlighting the need for domain-specific models.

Key Takeaways on LLM Use Cases

Use Case	Description
Invoke APIs and Actions	Perform actions based on user input.
Summarization	Condense information into concise formats.
Translation	Convert text between languages.
Information Retrieval	Retrieve relevant data from large datasets.

Application Integration and Deployment

- Optimize and deploy models for inference, ensuring effective application integration and user alignment through prompt engineering and fine-tuning.