A0597203 AI Business Applications

Introduction to Large Language Models

AI Business Applications

Introduction to Large Language Models

Training LLMs

What is LLM Training?

- LLM Training is the process of teaching a neural network to understand, generate, and manipulate human language.
- It involves feeding the model vast amounts of text data.
- The model learns patterns, grammar, context, and even some level of "knowledge" from this data.
- The goal is to adjust the model's internal parameters (weights and biases) to perform specific language tasks effectively.
- Modern models have BILLIONS of parameters (numbers) to learn during the training process.

Essential Components of LLM Training

1. **Dataset**: Large corpus of text data (e.g., books, articles, websites). Quality, quantity, and diversity are crucial.
2. **Model Architecture**: The neural network structure, predominantly the Transformer architecture (with self-attention mechanisms).
3. **Loss Function**: A function that measures the difference between the model's predictions and the actual target values (e.g., cross-entropy for next-word prediction).
4. **Optimizer**: An algorithm that updates the model's parameters to minimize the loss function (e.g., Adam, SGD).

The General Training Loop

1. **Data Preparation**: Collecting, cleaning, and tokenizing the text data into a format the model can understand.
2. **Model Initialization**: Setting initial random values for the model's parameters.
3. **Forward Pass**: Feeding input data through the model to get predictions.
4. **Loss Calculation**: Comparing predictions to the actual data to quantify error using the loss function.
5. **Backward Pass (Backpropagation)**: Calculating gradients, which indicate how each parameter contributed to the error.
6. **Parameter Update**: Adjusting model parameters using the optimizer in the direction that reduces the loss.
7. **Iteration**: Repeating steps 3-6 for many batches of data over multiple epochs (passes through the entire dataset).

Main Training Phases

1. LLM Pretraining.
2. LLM Fine Tuning

Pre-training: Building the Foundation

- The initial, most resource-intensive training phase.
- Models are trained on massive, diverse datasets (e.g., Common Crawl, Wikipedia, books).
- The objective of this step is to learn general language understanding, grammar, common sense reasoning, and factual knowledge.
- Usually self-supervised (e.g., predicting masked words, next sentence prediction).
- Results in a **base model** with broad capabilities.
- Examples: GPT-3, BERT, Llama pre-training.

Fine-tuning: Specializing the Model

- Takes a pre-trained base model and further trains it on a smaller, task-specific dataset.
- The objective of this step is to adapt the general knowledge of the pre-trained model to perform well on a particular downstream task (e.g., medical question answering, legal document summarization).
- It requires significantly less data and computation than pre-training.
- Can also be used for instruction tuning (following prompts) or aligning with human preferences (RLHF).

Data: The Critical Ingredient

- **Quantity**: LLMs require vast amounts of text to learn effectively. "More data is better" is often true, up to a point.
- **Quality**: Clean, well-formatted, and coherent data leads to better models. Garbage in, garbage out.
- **Diversity**: Exposure to various styles, domains, and perspectives helps create more robust and less biased models.
- **Preprocessing**:
  - **Tokenization**: Breaking text into smaller units (words, sub-words).
  - **Normalization**: Standardizing text (e.g., lowercasing, removing special characters).
  - Creating input IDs, attention masks.

Model Architecture: The Transformer

- The dominant architecture for state-of-the-art LLMs.
- Key Innovations:
  - **Self-Attention Mechanism**: Allows the model to weigh the importance of different words in a sequence when processing information, capturing long-range dependencies.
  - **Positional Encodings**: Injects information about the position of tokens in the sequence.
  - **Encoder-Decoder Structures** (for some tasks) or **Decoder-Only Structures** (common for generation).
  - **Feed-Forward Networks**: Applied independently to each position

Computational Demands & Challenges

- **Hardware**: Requires powerful GPUs (Graphics Processing Units) or TPUs (Tensor Processing Units) for parallel computation.
- **Distributed Training**: Often necessary to train large models across multiple GPUs or machines, adding complexity.
- **Time**: Pre-training can take weeks or months, even with significant computational resources.
- **Cost**: Significant expenses for hardware, cloud computing, and energy consumption.
- **Memory**: Model parameters and activations require substantial memory. Techniques like mixed-precision training help.

Evaluating Trained LLMs

- **Perplexity**: Measures how well a probability model predicts a sample. Lower is better.
- **Task-Specific Metrics**:
  - **BLEU, ROUGE**: For translation and summarization (overlap with reference texts).
  - **Accuracy, F1-score**: For classification tasks (e.g., sentiment analysis).
- **Benchmarks**: Standardized datasets and tasks for comparing models (e.g., GLUE, SuperGLUE, MMLU).
- **Human Evaluation**: Assessing fluency, coherence, helpfulness, and harmlessness by human raters. Often crucial for real-world performance.

Ethical Considerations in Training

- **Bias Amplification**: Models can learn and perpetuate biases present in the training data (e.g., gender, racial, societal biases).
- **Harmful Content Generation**: Potential to generate misinformation, hate speech, or other harmful text.
- **Data Privacy**: Ensuring that sensitive information from training data is not memorized or leaked.
- **Environmental Impact**: Significant energy consumption of training large models.
- **Accessibility and Equity**: Ensuring benefits of LLMs are widely accessible.

Memory Usage During Training

MODEL PARAMETERS

The **weights** of the mode. Assuming **16-bit half precision**, a model with **N billion parameters** requires **2*N GB** of memory for the weights alone

GRADIENTS

Gradients of the loss function relative to each model parameter are calculated. These gradients, being the same size and type as the model parameters, also occupy **2*N GB** of memory.

OPTIMIZER STATES

Optimizers, such as Adam, maintain state information for each parameter, typically requiring storage of two additional values per parameter. This can double the memory used by optimizer states to between **4-8*N GB**, depending on the precision.

TRAINING DATA

The **training data** itself varies significantly but includes at least one batch in memory, influenced by **batch size**, and the memory size of each data element depends on its **sequence length and embedding size**.

LLM Training Details - More Recent Models & Considerations

Model (Version/Size)	Est. Data Size (Tokens)	Context Size (Max Tokens)	Est. GPUs / Compute	Est. Training Time	Est. Electricity / Carbon Footprint	Est. Training Cost
GPT-3 (Base models like Davinci)	~500 Billion - 1 Trillion (incl. C4, Wikipedia, Books, etc.)	2,048 (later models up to 4,096)	~10,000 V100 GPUs (for original run) / ~3.6M A100-equivalent hours (PaLM 540B reference)	~34 days (using 1,024 A100s, research estimate) - Several months (actual, unconfirmed)	~1,287 MWh (training) / ~552 tons CO_2eq (Patterson et al.)	$4.6M - $12M+ (compute, various estimates)
Llama 2 (All sizes)	2 Trillion	4,096	Reported 6,000 GPU-months (A100-80GB equivalent for the family) / 1.7M+ GPU hours for 70B	Jan 2023 - July 2023 (overall project, specific model run time shorter within this)	3.3M kWh (entire project) / 539 tons CO_2eq (training, 100% offset by Meta)	Significant (part of Meta's AI investment)
BLOOM (176B)	366 Billion (1.6 TB)	2,048	384 A100 GPUs	~3.5 - 4 months	~433 MWh (training) / ~25-55 tons CO_2eq (trained in France, low-carbon energy)	~$2M - $5M (compute, public estimates)

LLM Training Details - More Recent Models & Considerations

Model (Version/Size)	Est. Data Size (Tokens)	Context Size (Max Tokens)	Est. GPUs / Compute	Est. Training Time	Est. Electricity / Carbon Footprint	Est. Training Cost
GPT-4	Not officially disclosed (speculated >> GPT-3, likely multi-trillion)	8,192 (GPT-4-8k) & 32,768 (GPT-4-32k); GPT-4 Turbo: 128,000	Not officially disclosed (speculated tens of thousands of A100s/H100s)	~5-6 months (speculative estimates)	Not disclosed (Expected to be significantly higher than GPT-3; estimates range from 20,000-78,000 MWh & thousands of tons CO_2eq for comparable efforts)	Est. >$60M - $100M+ (compute, speculative)
Llama 3 (Instruct models)	>15 Trillion (for the Llama 3 family)	8,192 (some reports suggest up to 128k for future/experimental versions)	Significant clusters of H100s (e.g., Meta mentioned two 24k H100 clusters)	~3 days (8B), ~17 days (70B), ~97 days (est. for 400B+ on 16k H100s)	Llama 3.1 405B est. ~11 GWh. Carbon footprint not yet fully disclosed, but Meta aims for net-zero operations.	Very High (part of Meta's large AI infrastructure investment)
Gemini 1.0 (Pro/Ultra)	Not officially disclosed (multimodal, likely vast & diverse datasets)	32,768 (Gemini 1.0 Pro); Gemini 1.5 Pro: 1 Million (up to 10M experimental)	Trained on Google's TPU v4 and v5e pods (thousands to tens of thousands of TPUs)	Not publicly disclosed (likely months)	Not disclosed. Google emphasizes efficiency & use of renewable energy. Gemini 1.0 was reported to be more efficient than some predecessors.	Very High (part of Google DeepMind's core AI efforts)

Discussion / Q&A

- What are the biggest challenges in training even larger and more capable LLMs?
- How can we mitigate biases in LLM training data and subsequent models?
- What future advancements in LLM training do you foresee?

Fine-Tuning Large Language Models (LLMs)

What is Fine-Tuning?

- **Pre-training Phase**
  - Trained on large corpus using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)
- **Fine-Tuning Phase**
  - Fine-tuning is the process of continuing the training of a pretrained LLM on a smaller, task-specific dataset.
  - The objective is to specialize the model for a particular use case or domain.
- **Why Fine Tune LLMs?**
  1. Improve performance on specific tasks.
  2. Inject domain-specific knowledge (legal, medical, financial, etc.).
  3. Adapt to company-specific language or tone.
  4. Reduce inference cost by limiting model size and scope.
- **Use Cases of BERT Fine-Tuning**
  - Customer Support Chatbots (trained on company FAQs).
  - Legal Document Analysis.
  - Scientific Paper Summarization.
  - Code Assistants for specific frameworks.
  - Sentiment classification for product reviews.

Typical LLM Fine Tuning Workflow

1. **Choose a Pre-trained Model**

Pick a base model from Hugging Face (e.g., distilbert-base-uncased for classification or gpt2 for text generation).

2. Prepare Your Dataset

Format your data appropriately. Common formats:

- For classification: CSV with text and label columns.
- For generation: Text file or JSON with prompt and completion.

3. Tokenize the Data

Convert text into tokens the model understands using a tokenizer.

4. Define Training Arguments

Set training parameters like learning rate, batch size, and number of epochs.

5. Train the Model

Use a trainer to fine-tune the model on your dataset.

6. Evaluate & Save

Check performance and save the model for future use.

7. Share (Optional):

- Push your fine-tuned model back to the Model Hub.
- Create a demo in Hugging Face Spaces.

Open in Colab