Kaggle | Google

5-Day Gen AI Intensive Course with Google 2025

Whitepaper Companion Podcast (Notes): Foundational LLMs and Text Generation

Deep Dive into Large Language Models (LLMs) and Text Generation

1. Introduction

- **LLMs as a Seismic Shift:** LLMs represent a fundamental change in AI, altering interactions with information and technology through their ability to process, understand, and generate text.
- **Definition of LLMs:** Advanced AI systems specializing in processing, understanding, and generating human-like text, typically implemented as deep neural networks.
- **Training:** LLMs are trained on massive amounts of text data, enabling them to learn intricate language patterns and perform tasks such as:
    - Machine translation
    - Creative text generation
    - Question answering
    - Text summarization
    - Reasoning
- **Whitepaper Overview:** The whitepaper explores the history of LLM architectures, fine-tuning techniques, methods for efficient training, inference acceleration, applications, and code examples.
- **Deep Dive Goal:** To understand LLMs' architecture, evolution, training, evaluation, and optimization up to February 2025.

2. Transformer Architecture Foundation

- **Origin:** The Transformer architecture originated from a 2017 Google translation project.
- **Encoder-Decoder Structure:** The original Transformer included an encoder to process input and a decoder to generate output.
  - The **encoder** takes the input (e.g., a sentence in French) and creates a representation summarizing its meaning.
  - The **decoder** uses this representation to generate the output (e.g., the English translation) piece by piece (token by token).
- **Tokenization:** A token can be a whole word (e.g., "cat") or part of a word (e.g., "pre" in prefix).

3. Input Processing for Transformers

- **Tokenization:** Input text is converted into tokens based on a specific vocabulary.
- **Embeddings:** Each token is transformed into a dense vector (embedding) capturing its meaning.
- **Positional Encoding:** Since Transformers process all tokens simultaneously, positional encoding (sinusoidal, learned) is necessary to retain sequence information.
  - Different positional encodings can affect how well the model understands longer sentences.

4. Self-Attention Mechanism

- **Purpose:** Self-attention allows the model to understand relationships between words in a sentence (e.g., "it" refers back to "tiger" in "The thirsty tiger").
- **Query (Q), Key (K), Value (V) Vectors:**
  - **Query:** Asks which other words are important for understanding the current word.
  - **Key:** A label attached to each word, indicating what it represents.
  - **Value:** The actual information the word carries.

- **Process:**
    - The model calculates a score for how well each query matches other keys.
    - These scores are normalized into attention weights.
    - Attention weights indicate how much each word should pay attention to others.
    - A weighted sum of all value vectors is created, resulting in a rich representation for each word, considering its relationship to every other word in the sentence.
- **Parallel Processing:** Comparison and calculation happen in parallel using matrices for Q, K, and V.
- **Significance:** The ability to process relationships simultaneously is a key reason why Transformers excel at capturing subtle meanings, especially across longer distances.

5. Multi-Head Attention

- **Parallel Self-Attention:** Multi-head attention runs the self-attention process multiple times in parallel with different learned Q, K, V matrices.
- **Diverse Relationships:** Each "head" focuses on different types of relationships (e.g., grammatical, semantic).
- **Deeper Understanding:** Combining different perspectives provides a deeper understanding of the text.

6. Layer Normalization & Residual Connections

- **Layer Normalization:** Helps keep the activity level of each layer stable, accelerating training and improving results.
- **Residual Connections:** Act as shortcuts, allowing the original input of a layer to bypass processing and be added directly to the output.
    - Prevents vanishing gradients and helps the network retain learned information, especially in deep models.

7. Feed-Forward Network Layer

- **Application:** Applied to each token's representation separately after attention.
- **Structure:** Typically consists of two linear transformations with a non-linear activation function (ReLU or GeLU) in between.
- **Enhancement:** Further enhances the model's representational power.

8. Decoder-Only Architecture

- **Suitability:** Ideal for text generation tasks like writing and conversation.
- **Masked Self-Attention:** Only attends to previous tokens, ensuring the model predicts the next token based on what came before.
- **Simpler Design:** Skips the initial representation of the whole input sequence, generating output token by token.

9. Mixture of Experts (MoE)

- **Efficiency:** Enables building larger models more efficiently.
- **Specialized Submodels:** Uses specialized "expert" submodels.
- **Gating Network:** A "gating network" routes input to only a fraction of the experts, reducing computational cost.

10. Evolution of LLMs - Timeline & Key Models

- **GPT-1 (2018):**
  - Decoder-only
  - Unsupervised pre-training
  - Limitations: Repetitive text
- **BERT (2018):**
  - Encoder-only
  - Focus on understanding language

- Trained on masked language modeling and next sentence prediction

- **GPT-2 (2019):**
  - Scaled-up GPT-1
  - Better coherence
  - Zero-shot learning: Learns new tasks from a single example in the prompt
- **GPT-3 Family (2020+):**
  - Massive scale (billions of parameters)
  - Few-shot learning: Learns from a few examples
  - Instruction tuning (InstructGPT)
  - Code generation (GPT-3.5)
  - Multimodality (GPT-4): Handles images and text together
  - Large context windows
- **Lambda (2021):**
  - Focused on natural conversation
- **Gopher (2021):**
  - Emphasis on high-quality data and optimization
  - Highlighted scaling limitations: Increasing model size doesn't always improve performance on all tasks
- **GLAM:**
  - Used MoE
- **Chinchilla (2022):**
  - Demonstrated the importance of data size relative to model size (compute-optimal scaling)
- **PaLM & PaLM 2 (2022/2023):**
  - Strong benchmark performance

    - Pathway system
    - Improved reasoning, coding, and math
- **Gemini:**
    - Multimodal native
    - TPU optimized
    - MoE use
    - Different sizes (Ultra, Pro, Nano, Flash)
    - Very large context window (1.5 Pro)
- **Open Source Models:**
    - Gemma/Gemma 2
    - Llama family (1, 2, 3, 3.1)
    - Mixtral (MoE)
    - 01 (reasoning)
    - DeepSeek-R1 (reasoning)
    - Qwen
    - Yi
    - Grok
    - Importance of licenses

11. LLM Training Overview

- **Pre-training:**
    - Unsupervised learning on massive raw text data
    - Learns general language patterns
- **Fine-tuning:**
    - Training the pre-trained model on smaller, specific, often labeled datasets

- Specializes the model for particular tasks

12. Fine-tuning Techniques

- **Supervised Fine-Tuning (SFT):**
  - Training on prompt-response pairs
- **Reinforcement Learning from Human Feedback (RLHF):**
  - Training a reward model based on human preferences
  - Using RL to align the LLM's output
  - RLAIF and DPO are also mentioned

13. Parameter-Efficient Fine-Tuning (PEFT)

- **Efficiency:** Fine-tunes large models by training only a small subset of parameters, leaving most pre-trained weights frozen.
- **Methods:**
  - Adapters
  - LoRA
  - QLoRA
  - Soft Prompting

14. Prompt Engineering

- **Importance:** Crafting effective input prompts to guide LLM output.
- **Techniques:**
  - Zero-shot prompting: Direct instruction without examples
  - Few-shot prompting: Giving a few examples
  - Chain-of-Thought prompting: Showing the model how to think through the problem step by step

15. Sampling Techniques

- **Token Generation:** LLMs generate text token by token.
- **Strategies:**
  - Greedy Search: Always picks the most likely next token (fast, but can be repetitive).
  - Random Sampling (with Temperature): Introduces randomness (creative, but risk of nonsensical text).
  - Top-K Sampling: Limits choices to the top K most likely tokens.
  - Top-P (Nucleus) Sampling: Uses a dynamic threshold based on token probabilities.
  - Best-of-N Sampling: Generates multiple responses and picks the best one.

16. Evaluating LLMs

- **Challenges:** Evaluating LLMs goes beyond traditional metrics due to the subjective nature of text generation.
- **Framework:**
  - Task-specific data: Reflects real-world scenarios and user interactions.
  - System-level consideration: Considers the entire system the LLM is part of (e.g., RAG).
  - Defining "good" metrics: Accuracy, helpfulness, creativity, factual correctness, style adherence.
- **Methods:**
  - Quantitative metrics (BLEU, ROUGE): Compare model output to ground truth answers.
  - Human evaluation: Nuanced judgments on fluency, coherence, and overall quality.
  - LLM-powered evaluators (auto-evaluators): AI judges other AI (requires calibration).
- **Advanced Approaches:** Breaking down tasks into subtasks and using rubrics with multiple criteria (especially for multimodal models).

17. Inference Acceleration

- **Goal:** Make LLM response generation faster and more efficient.
- **Trade-offs:** Balancing quality, speed, cost, latency, and throughput.
- **Output Approximating Methods:**
  - Quantization: Reducing numerical precision of weights and activations.
  - Distillation: Training smaller "student" models from larger "teacher" models.
- **Output Preserving Methods:**
  - FlashAttention: Optimizing attention calculation.
  - Prefix Caching: Reusing calculations for repeating input parts.
  - Speculative Decoding: Using a smaller "draft" model to predict tokens verified by the main model.
  - Batching: Processing multiple requests at the same time.
  - Parallelization: Splitting up the computation across multiple processors.

18. Applications of LLMs

- **Code/Math Assistance:** Code generation, completion, refactoring, debugging, translation, documentation, understanding code bases.
- **Machine Translation:** More fluent, accurate, and natural-sounding translations.
- **Summarization:** Condensing large amounts of text to key points.
- **Question Answering (RAG):** More knowledgeable and precise systems.
- **Chatbots:** More humanlike conversations.
- **Content Creation:** Writing ads, scripts, and creative text formats.
- **Natural Language Inference:** Sentiment analysis, analyzing legal documents, assisting with medical diagnoses.
- **Text Classification:** Spam detection, news categorization, understanding customer feedback.

- **LLM Evaluation:** Acting as auto-evaluators.
- **Text Analysis:** Extracting insights and identifying trends from huge data sets.
- **Multimodal Applications:** Combining text, images, audio, and video.

19. Conclusion and Future

- **Recap:** Covered Transformer architecture, LLM evolution, fine-tuning, evaluation, and inference acceleration.
- **Future Questions:** What new applications will be possible with the next generation of LLMs? What challenges need to be overcome?