LLM Topics: A Comprehensive Outline

Dan Hermes

# 1. LLMs

* 1. Generalized models lacking specialized knowledge
     1. Transformers ("Attention Is All You Need")
     2. Text Generation Properties
        1. Temperature
        2. Frequency Penalty
        3. Presence Penalty
        4. Top-p
        5. Stop Sequences
     3. Prompt Engineering
     4. RAG
     5. Fine Tuning
  2. Language Modeling
  3. Tokenization
  4. Embeddings
  5. Context Size
  6. Scaling Laws
     1. Number of parameters (N)
     2. Size of the training dataset (D)
  7. Use Cases (p.60)

# Prompt Engineering

* 1. Approaches
     1. Give Direction
        1. Explain what and how to answer
     2. Specify Format
        1. Definitions
        2. Lists
        3. Topics
        4. Summaries
        5. Detailed text
     3. Provide Examples
        1. In-Context Learning: zero, one, few-shot
     4. Evaluate Quality
        1. ???
     5. Divide Labor
        1. Architect prompt into pieces
  2. 15 Key Tips (p. 89)?
     1. Provide References
     2. Give GPTs time to think
     3. Role prompting – LLM is a ....
     4. Chain prompting – leading the LLM
        1. Least to Most Technique
     5. Chain of Thought – LLM explaining thought process
  3. In-Context Learning (ICL)
     1. Zero-shot Prompting (no context)
     2. One-shot Learning
     3. Few-shot Prompting (ex. p.25)
  4. Prompt Injection / Security??
     1. Guardrails
     2. Safeguards
  5. Self-consistency??
  6. Estimating GPT tokens (using LangChain?)
  7. Chunking
     1. Strategies:
        1. Document
        2. Page
        3. Paragraph
        4. Sentence
        5. Length
     2. Aggregating
        1. Summaries
        2. Topics
        3. Analysis

3. Transformers ("Attention Is All You Need")

* 1. Vanishing Gradient
  2. Encoder / Decoder architecture (ex. p.38) - BLLM p.8+
     1. Encoder-only (p.41)
        1. Masked Language Modeling (MLM)
        2. Next Sentence Prediction (NSP)
     2. Decoder-only (p. 44)
     3. Encoder/Decoder
  3. Other Common Architectures (p. 50)
     1. Instruction-tuned
     2. Chatbots (p.212, 244, 345)
  4. Input Embedding
  5. Tokenizing Text - BLLM p.21, 25
     1. Tokenized Text
        1. Token IDs
        2. Embedding (vectors)
        3. Positional Encoding
  6. Optimization Techniques (p.47)
  7. Attention Mechanism (ex. p.33)
     1. Query Vector
     2. Key Vector
     3. Value Vector
  8. Generative Pre-trained Transformer (GPT)
     1. Masked Self-Attention

4. Retrieval-Augmented Generation (RAG) (p.100, 252) (LLMEH Ch. 4)  
Solves: "Most probable results, not most accurate ones."

* 1. RAG vs. Fine Tune LLM
  2. Parametric (pre-trained & fine tuned) vs. Non-parametric(RAG/dynamic) Knowledge
  3. Embeddings (BLP p.101 , RAGD p.31)
     1. Vector Databases (ex. Deep Lake p.339)
     2. Vector Stores
  4. RAG configurations - RAGD
     1. Naïve RAG (keyword)
     2. Advanced RAG (vector search and index)
     3. Modular/Hybrid RAG (both)
  5. Retriever, Generator, Evaluator (Trainer) - RAGD
  6. Cosine Similarity - How good are the answers? - RAGD
     1. Relevance metric
     2. Input vs. Output
     3. Augmented Input vs. Output
  7. RAG from Scratch (p.103)
     1. Preprocessing Dataset
     2. Generate Embeddings
     3. Find Related Chunks
     4. Testing Cosine Similarity
     5. Calculate Similarity in Action
     6. Augmenting the Prompt
  8. RAG Pipelines (RAGD p.35 get image)
     1. Data Collection and Prep
     2. Data Embedding and Storage (chunk, embed, store)
     3. Augmented Input Generation
  9. Retriever Architectures
     1. Decoder architecture
     2. Decoding Methods
        1. Greedy Search
        2. Beam Search
        3. Sampling
           1. Top-k
           2. Top-p

5. Agents (p.288) - RESEARCH

* 1. What is an Agent?
     1. Agent is a Decider and Adapter – an autonomous component
     2. Program is just a Tool – static sequence of instructions
     3. Agents run the gamut between a program and a Decider/Adapter
  2. Agent Components
     1. **Data** extraction and indexing
     2. **Tool** use
     3. **LLM** API calls
  3. Reasoning Engine / Core
     1. Query Processing (p.320, 323)
     2. Tool Utilization
     3. Information Processing
     4. Synthesis and Response
     5. Custom Functions (p.326)
     6. Using LLMs (as APIs)
  4. Agent Orchestration
  5. Autonomous Agents
  6. Agent Simulations (p.308)
  7. Generative Agents (p.310)
  8. Agent Types
     1. Action Agent
     2. Plan-and-execute Agent (p.311, 315)
     3. Reason and Act (ReAct)?? Same thing as ii?
  9. Memory
     1. Short term
     2. Long Term
  10. Using LangChain for State and Conversation Memory
  11. Agent Examples (p.294)
      1. AutoGPT (p.294) - Agent Memory Setup (p.296)
      2. BabyAGI (p.303)
      3. OpenAI Assistants (p.329)
      4. LangChain OpenGPT (p.332)
  12. Agent Techniqes

1. Chain-of-thought – LLM describing its process
2. Tree-of-Thoughts – multi-threaded logic
3. Callbacks (Global, request specific, verbose argument)
4. Customizing agents on LCEL?

# Vector Databases

1. Embeddings
2. Document Loading
3. RAG with LangChain
4. Memory Retrieval with FAISS
5. Hosted Vector DBs with Pinecone

# MLOps and LLMOps (LLMEH Ch. 11)

1. **DevOps**
   1. Continuous Integration (CI)
   2. Continuous Deployment (CD)
   3. Continuous Testing (CT)
2. **MLOps** (LLMEH Appendix)
   1. Steps
      1. Continuous Integration (CI)
      2. Continuous Deployment (CD)
      3. Continuous Training (CT)
   2. Automation
   3. Versioning
   4. Experiment Tracking
   5. Testing
   6. Reproducability
   7. Monitoring
      1. Logs
      2. Metrics
      3. System Metrics
      4. Model Metrics
      5. Drifts
      6. Monitoring vs. Observability?
      7. Alerts
   8. Tools
      1. Model Registry
         1. Hugging Face
         2. Comet ML
         3. W&B
         4. MLFlow
         5. ZenML
      2. Feature Store
         1. Hopswork
         2. Tecton
         3. FeatureForm
      3. ML Metadata store
         1. Comet ML
         2. W&B
         3. MLFlow
         4. ZenML
      4. ML Pipeline orchestrator
         1. ZenML
         2. AirFlow
         3. Prefect
         4. Dagster
      5. Testing?
      6. Experiment tracker
         1. Comet ML
3. **LLMOps**
   1. Steps – same as MLOps (CI, CD, CT)
   2. LLM over ML considerations
      1. Scalability
      2. Retrieval-augmented generation (RAG)
      3. Prompt engineering
      4. Fine-tuning
      5. Model inference costs
   3. Prompt monitoring
      1. Time to First Token (TTFT)
      2. Time Between Tokens (TBT)
      3. Tokens per Second (TPS)
      4. Time Per Output Token (TPOT)
      5. Total Latency
   4. **LLMOps Tools by Category**

#### **Model Deployment & Serving**

* + - 1. **Scalability & Optimization:**
         1. Ray Serve
         2. vLLM
         3. Triton Inference Server
      2. **Model Management & Tracking:**
         1. MLflow
         2. SageMaker JumpStart

#### **Retrieval-Augmented Generation (RAG) & Knowledge Integration**

* + - 1. **Vector Databases:**
         1. Pinecone
         2. Weaviate
         3. FAISS
      2. **RAG Pipelines:**
         1. LlamaIndex
         2. Haystack

#### **Fine-Tuning & Model Adaptation**

* + - 1. **Frameworks:**
         1. Hugging Face Transformers
         2. Axolotl
         3. DeepSpeed
      2. **Cloud-Based Training:**
         1. SageMaker Training
         2. Azure OpenAI Fine-Tuning

#### **Prompt Engineering & Optimization**

* + - 1. **Versioning & Debugging:**
         1. PromptLayer
         2. LangSmith
      2. **Workflow Automation:**
         1. LangChain
         2. PromptFlow (Azure)
         3. Opik – Prompt Monitoring

#### **Cost & Performance Optimization**

* + - 1. **Efficient Inference:**
         1. vLLM
         2. DeepSpeed
         3. OctoML
      2. **Caching & Acceleration:**
         1. TensorRT-LLM
         2. Hugging Face Optimum

#### **Monitoring, Logging & Governance**

* + - 1. **Observability & Bias Detection:**
         1. Arize AI
         2. WhyLabs
      2. **Experiment Tracking:**
         1. Weights & Biases (W&B)
         2. LangSmith

#### **Security & Guardrails**

* + - 1. **Policy & Content Moderation:**
         1. NeMo Guardrails
         2. OpenAI Moderation API
      2. **Enterprise AI Compliance:**
         1. AWS AI Guardrails
         2. Google Vertex AI Guardrails

#### **End-to-End LLMOps Platforms**

* + - 1. **Fully Managed Solutions:**
         1. Databricks Mosaic AI
         2. Vertex AI
         3. Azure OpenAI Service
      2. **Cloud-Based LLMOps:**
         1. SageMaker LLMOps
         2. Google Cloud AI

# Advanced RAG

* 1. Queries with LlamaIndex (p.252)
     1. Retrievers
     2. Query Engines
     3. Query Transform
     4. Query Construction
     5. Query Expansion
     6. Query Transformation
     7. Subqueries
  2. Human Feedback(HF) (Adaptive RAG) - RAGD p.115
     1. Evaluator
        1. Metrics - ranking
        2. Human Feedback(HF)
           1. Entered by human - flashcards or snippets
           2. Stored in RAG
           3. Consumed in Retriever
  3. Reranking Documents
  4. Retriever Architectures (p.264)
     1. Recursive
     2. Small-to-big
  5. RAG/LLM Metrics (p.265)
     1. Correctness
     2. Faithfulness
     3. Context Relevancy (Cosine Similarity)
     4. Guideline Adherence
     5. Embedding Semantic Similarity (Cosine Similarity)
     6. Retrieval Metrics (p.268)
        1. Mean Reciprocal Rank (MRR)
        2. Hit Rate
        3. Mean Average Precision (MAP)
        4. Normalized Discounted Cumulative Gain (NDCG)
     7. Evaluation Tools (Open Source in LlamaIndex)
        1. Ragas
        2. DeepEval
        3. ARES
     8. Custom Evaluation (p.273)
     9. LangChain, LangSmith, LangChain Hub (p.280)

 7. LangChain (p.118) (PE GenAI p.125-184)

* 1. Prompt Templates p.118 + p.144 (ex. p.122)
     1. Few shot Prompts (ex. p.152)
     2. Example Selectors (ex. p.153)
  2. Output Parsers (p.164)
  3. Indexes & Retrievers (p.190)
     1. Data Ingestion / Document Loaders
        1. PDF Files
        2. Webpages
        3. Google Drive
     2. Data Preparation
     3. Text Splitters
        1. By number of characters
        2. At Logical End Points
        3. Foreign Linguistic Structures
        4. Markdown Format
        5. Tokens
     4. Similarity Search (p.207)
  4. Embedding models
     1. Vector Embedding (p.207)
     2. Vector databases
     3. Knowledge Graphs (p.182)
     4. Open Source (p.209)
     5. Cohere Embeddings (p.210)
  5. Features
     1. Agents
        1. Zero shot ReAct
        2. Structured Input ReAct
        3. OpenAI Functions Agent
        4. Self-Ask with Search Agent
        5. ReAct Document Store Agent
        6. Plan-and-Execute Agents
     2. Chains (ex. p.124-125+)
        1. Generating Text (ex. p.158)
        2. Adding Memory (p.160)
        3. Concatenating Chains (p.161)
        4. Debugging Chains (p.161)
        5. Custom Chain (p.162)
     3. Tools
     4. Memory
     5. Callbacks

8. LlamaIndex (p.139 for comparison with LangChain)

* 1. Data Connectors
  2. Nodes
  3. Indices
  4. Summary Index
  5. Vector Store Index
  6. Routers
  7. Saving and Loading Indexes Locally

9. Fine-Tuning LLMs (p.350, p.341)

* 1. Full Fine Tuning
  2. Low-Rank Adaptation (LoRA)
  3. Training Hyperparameters
  4. OPT Parameters
  5. Inference – Inference Optimization (LLMEH Ch. 8)
     1. Model Optimization
     2. Model Parallelism
     3. Model Quantization
  6. Supervised Fine Tuning (SFT) (LLMEH Ch. 5)
  7. Reinforcement Learning from Human Feedback (RLHF) (p.381, 384, 393)
  8. Constitutional AI (RLAIF)
  9. Cohere LLM (p.372)
  10. Evaluating LLMS (LLMEH Ch. 7)
      1. ML & LLM evaluation
      2. Genral purpose LLM evaluation
      3. Domain specific LLM evaluation
      4. Task specific LLM evaluation
      5. RAG evaluation: Ragas, ARES

# 11. Performance

* + 1. Objective / Loss Function
    2. Evaluation Metrics
       1. Intrinsic/Extrinsic metrics
       2. The Perplexity Evaluation Metric

11. Deployment (p.406)

* 1. Model Distillation
  2. Teacher-Student Models
  3. Deployment Optimization
     1. Quantization (ex. p.419)
        1. Scalar
        2. Product
        3. LLM
     2. Pruning
        1. Magnitude Based (Unstructured Pruning)
        2. Structured Pruning
     3. Speculative Decoding

# 12. Enforcing Policy, Censoring, and Minimizing Bias

* 1. Guardrails – Input
     1. Ensure AI stays within scope
     2. Prevent AI from hallucinating or going off-topic
     3. Restrict user queries to ethical and legal boundaries
  2. Safeguards – Output
     1. Detect bias, misinformation, or hate speech after AI generates a response
     2. Moderate AI-generated content dynamically
     3. Implement human oversight to correct AI mistakes

Self-critique Chain (p.240, 244)

## Bibliography

1. Bouchard, Louis-François, and Louie Peters. *Building LLMs for Production: Enhancing LLM Abilities and Reliability with Prompting, Fine-Tuning, and RAG*. Independently Published, 2024. (BLP abbrev. but default text for outline page numbers)

2. Rothman, Denis. *RAG-Driven Generative AI: Build Custom Retrieval Augmented Generation Pipelines with LlamaIndex, Deep Lake, and Pinecone*. Packt Publishing, 2024. (RAGD abbrev.)

3. Raschka, Sebastian. *Build a Large Language Model (From Scratch)*. Manning, 2025.

4. Phoenix, James, and Mike Taylor. *Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs*. 1st ed., O'Reilly Media, 2024. (PEGenAI abbrev)

5. Iusztin, Paul, and Maxime Labonne. *LLM Engineer's Handbook: Master the Art of Engineering Large Language Models from Concept to Production*. Packt Publishing, 2024. (LLMEH abbrev.)

### Page references in the outline refer to Bouchard/Peters unless otherwise noted.

### Missing Topics (per ChatGPT)

1. Alignment Techniques:

* Further emphasis on Reinforcement Learning from AI Feedback (RLAIF).
* Discussion of "AI alignment" to ensure model safety and adherence to human values.

1. Foundation Models:

* Details on emergent behaviors of foundation models.
* Ethical implications of scaling large models.

1. Energy Efficiency:

* Techniques to reduce the carbon footprint of training large LLMs.
* Eco-friendly alternatives in deployment (e.g., model distillation optimizations).

1. Context Window Limitations:

* Advances in addressing or extending the context window limits (e.g., Sparse Attention, Memory Models).

1. Open-Source Alternatives:

* Deeper exploration of open-source LLMs like Falcon, MosaicML, and their capabilities.

#### RAG Overview

1. Dynamic Retrieval Techniques:

* Context-adaptive retrieval mechanisms that evolve with user queries.
* Active learning techniques integrated with RAG.

1. Federated RAG Systems:

* Handling distributed datasets in secure environments using RAG principles.

1. Data Quality for RAG:

* Emphasis on ensuring data quality in embedding databases.
* Techniques for deduplication, preprocessing, and data augmentation.

#### Advanced Evaluation Metrics

1. Hallucination Detection:

* Metrics for identifying and mitigating hallucinations in LLMs and RAG responses.

1. Bias and Fairness:

* Techniques to measure and mitigate bias in LLM/RAG systems.

1. Explainability and Interpretability:

* Advances in interpretability for outputs, embeddings, and retrieval steps.

Per Me

Enterprise

* Governance
* Bias Removal
* Scalability
* Measurability
* Micro-Transformation - (Agile)

Diffusion Models for Image Generation

* OpenAI DALL-E
* Midjourney
* Stable Diffusion
* Google Gemini
* Text to Video