Talk Plan: Unpacking Large Language Models - Beyond Token Prediction

Motto:
Any sufficiently advanced technology is indistinguishable from magic
--Arthur Clarke

Introduction: The "Impossible" Made Real

- Start by acknowledging the initial understanding of LLMs as advanced token predictors and then highlight how their current capabilities seem to defy that simple definition, leading to the perception of "something impossible" happening.
- Set the stage: We'll dive "under the hood" to understand the mechanisms, lifecycle, and frontiers of this technology.

1. Are there separate stages while preparing the answer? What happens during these stages?

Key Message: While the core is token prediction, modern LLM interactions involve sophisticated orchestration, often appearing as distinct "stages."

- **Prompt Engineering & Contextualization:**
  - The user's input (prompt) is the initial driver. It's often augmented or rephrased internally to guide the model better.
  - Previous turns in a conversation are appended to the current prompt to maintain context within the LLM's "context window."
- **Token Prediction (The Core):**
  - The model, based on its training, predicts the most probable next token(s) given the input sequence. This happens iteratively, token by token.
    - *Example:* Given the input "The sky is", the model predicts "blue" as the most probable next token, then "and", then "the", etc., until a complete thought is formed.
  - This isn't just statistical counting; it involves complex neural network architectures (mostly transformers) that learn intricate patterns and relationships in vast datasets.

    - Also "most probable" implies "probability distribution" -- and it's no longer "all the texts in the internet" (more on that later)
- **Internal Monologue / "Chain of Thought":**
    - For complex tasks, models can be prompted to "think step-by-step." This involves the model generating intermediate reasoning steps, which are then fed back into itself to guide the final answer. These steps might look like "stages" (e.g., "Plan," "Execute," "Review").
        - *Example:* When asked to solve a math problem, the model might first generate "Step 1: Identify variables. Step 2: Formulate equation. Step 3: Solve equation. Step 4: State answer." before providing the final numerical solution.
    - Guessing the next step is often easier than providing the final answer at once
- **Tool Calling (Next Section):**
    - The model might decide, based on the prompt, that an external tool is needed (e.g., calculator, web search). It then generates a call to that tool.
- **Response Generation & Refinement:**
    - Once internal processing or tool outputs are received, the model synthesizes the final response, often iterating internally to ensure coherence, relevance, and adherence to stylistic guidelines.
        - *Example:* After receiving weather data, the model might initially draft "It will be 15C and cloudy." Then, it refines it to "Tomorrow in London, expect partly cloudy skies with a high of 15 degrees Celsius."

2. How the engine interacts with other programs (Tool Use)?

Key Message: LLMs extend their capabilities by intelligently calling external tools, acting as a sophisticated orchestrator rather than a self-contained computational engine for all tasks.

- **The "Tool Use" Mechanism:**
    - When an LLM encounters a query requiring precise calculation, up-to-date information, or deterministic logic (e.g., math, current events, code execution), it doesn't try to "compute" or "know" the answer itself.

    - Instead, it's trained to recognize when a tool is appropriate and to generate the correct *syntax* to call that tool.
    - **Example: Math:** If asked to multiply two 10-digit numbers, the LLM will generate a call to a calculator tool (e.g., `calculator.add(1234567890, 9876543210)`). It doesn't perform the multiplication itself.
    - **Example: Debugging Code:** For debugging, it might generate a call to a code interpreter, feeding the code to it. The output from the interpreter is then returned to the LLM.
    - **Example: Angela Hewitt Concerts:** For current events or specific factual data not in its training cutoff, it performs an internet search. It formulates search queries, sends them to a search engine, and processes the results.
- **Orchestration:** This process involves a "reasoning loop" where the LLM:
    1. Analyzes the query.
    2. Decides if a tool is needed.
    3. Selects the appropriate tool.
    4. Generates the tool-specific input/parameters.
    5. Executes the tool (via an API call).
    6. Receives the tool's output.
    7. Integrates the output into its response or uses it for further internal reasoning.

3. How the model uses the token-prediction engine (probably several times)?

Key Message: The core token prediction engine is remarkably versatile and is used iteratively for various tasks beyond just generating the final output.

- **Iterative Generation:** The most obvious use is generating the response token by token. Each new token generated becomes part of the input for predicting the next token.
- **Self-Correction/Refinement:**
    - Models can be prompted to review their own output for errors, coherence, or style. This involves generating an initial answer, then generating a critique of it, and then generating a revised answer based on the critique. Each step uses the token prediction engine.

- **Chain of Thought / Scratchpad:** As mentioned, generating intermediate thoughts or planning steps involves multiple calls to the token prediction engine, using previous steps as context for generating subsequent ones.
- **Tool Input/Output Formatting:** The model uses token prediction to format requests for external tools and to interpret and integrate the diverse outputs from those tools back into a coherent narrative.

4. What is the entire cycle of developing and using the model?

Key Message: LLM development is a multi-stage, resource-intensive process, with significant cost implications that dictate accessibility for different scales of organizations.

- **1. Pre-training (Foundation Model Training):**
  - **What:** Training a massive transformer model on a colossal dataset of text and code (trillions of tokens). The goal is to learn general language patterns, grammar, facts, and reasoning abilities. It learns to predict the next token.
  - **Costs/Resources:** Enormous. Requires supercomputer-scale infrastructure (thousands of GPUs), massive power consumption, and months of continuous training. Hundreds of millions of dollars.
  - **Feasibility:** Only possible for tech giants (OpenAI, Google, Microsoft, Meta, Anthropic, Yandex, some national labs). Impossible for small startups, home computers, or mobile phones.
- **2. Fine-tuning (Supervised Fine-tuning - SFT, Reinforcement Learning from Human Feedback - RLHF):**
  - **What:** Taking a pre-trained model and further training it on smaller, high-quality, curated datasets.
    - **SFT:** Improves instruction following, stylistic coherence, and alignment with specific tasks using supervised examples (e.g., example question-answer pairs).
    - **RLHF:** Uses human preferences to train a reward model, which then guides the LLM to produce more helpful, harmless, and honest outputs. This is crucial for alignment.
  - **Costs/Resources:** Still very expensive, but orders of magnitude less than pre-training. Hundreds to thousands of GPUs, weeks to months. Millions of dollars.

    - **Feasibility:** Achievable for large companies, potentially medium-sized companies, and well-funded startups *if they start from an existing pre-trained model*. Training from scratch is prohibitive.
- **3. Specialization/Customization (Prompt Engineering, Domain Adaptation, LoRA):**
    - **What:** Adapting an existing fine-tuned model for very specific tasks or domains.
        - **Prompt Engineering:** Crafting effective prompts to guide the model's behavior without changing its weights.
        - **Retrieval-Augmented Generation (RAG):** Integrating a knowledge base that the model can query to access up-to-date or domain-specific information (e.g., your company's internal documents).
        - **Low-Rank Adaptation (LoRA) / Parameter-Efficient Fine-Tuning (PEFT):** Training only a small fraction of the model's parameters, making it much more resource-efficient for domain adaptation.
    - **Costs/Resources:** Varies widely. Prompt engineering is free. RAG systems involve data indexing and retrieval. LoRA can be done on a few high-end GPUs over days/weeks.
    - **Feasibility:** Medium companies, small startups (e.g., using open-source models with LoRA or RAG), and even individuals with good home computers (for smaller open-source models or PEFT) can participate here.
- **4. Inference (Application to a Request):**
    - **What:** Running the trained and specialized model to generate a response to a user's prompt.
    - **Costs/Resources:** Significant operational cost depending on model size and query volume. Requires high-performance GPUs, but less than training. Can be optimized through quantization, pruning, etc.
    - **Feasibility:** Cloud-based inference (using APIs) is accessible to everyone. Running smaller models locally (on good home computers, even mobile phones for very small or highly optimized models) is increasingly possible.

5. What is preserved during the dialog with AI model and between the dialogs?

Key Message: LLMs have "short-term memory" within a conversation (context window) but typically no inherent "long-term memory" between sessions without external mechanisms.

- **During a Dialog (Context Window):**
  - The primary way LLMs maintain conversational context is by literally including previous turns of the conversation in the current prompt.
  - There's a limit (the "context window" or "context length," measured in tokens) to how much past conversation can be included. Once this limit is reached, the oldest parts of the conversation are "forgotten" unless explicitly summarized or managed.
  - This is why very long conversations eventually lose track of earlier details.
- **Between Dialogs (No Inherent Memory):**
  - By default, an LLM itself doesn't retain memory of past conversations with you (or anyone else) once a session ends. Each new interaction is typically treated as a fresh start.
  - **External Mechanisms for "Long-Term Memory":**
    - **User Profiles/Databases:** Applications built *around* LLMs can store user preferences, past interactions, or specific facts in a database and inject them into the prompt when interacting with the LLM.
    - **Vector Databases:** Embeddings of past conversations or relevant documents can be stored in vector databases. When a new query comes in, relevant past information can be retrieved (similar to RAG) and added to the current prompt.

6. What amount of information needs to be learned to get some new "skill"?

Key Message: LLMs demonstrate impressive generalization but struggle with true "one-shot" learning or deep understanding from scarce data in the way humans do.

- **Transfer Learning & Generalization:**
  - LLMs leverage their vast pre-training to perform well on tasks they weren't explicitly trained for, thanks to transfer learning. They generalize from patterns learned across a diverse corpus.
  - **Learning Ancient Greek/Scarce Languages:**

        - It's challenging for current LLMs to learn a language with truly *limited* text data, similar to how a human would learn from a single grammar book and exercises. Their strength comes from statistical patterns over *large* datasets.
        - However, if there's *some* data (even if limited compared to English), and if the language shares structural similarities with other languages in its training, an LLM can potentially achieve reasonable proficiency through fine-tuning. The quality and breadth of the available text are crucial.
    - **Learning New Mathematical Theory:**
        - Reading *one* textbook is unlikely to instill a deep, generative understanding of a new mathematical theory that allows for novel problem-solving or theorem proving. LLMs excel at pattern recognition and interpolation, not necessarily true conceptual innovation or reasoning from first principles in a novel domain without examples.
        - They can *summarize* the textbook, answer questions *about* its content, and apply *known* theorems if they've seen similar patterns. However, they struggle with abstract reasoning beyond their training data distribution.
        - **Dialog with Experts:** This could provide the necessary examples and feedback for fine-tuning, potentially accelerating "learning" on the theory.

7. Is the technology more or less known to all the players? What are the key secrets?

Key Message: The fundamental architecture (Transformers) is public, but significant competitive advantages come from massive resources, proprietary data, engineering expertise, and subtle innovations in training and alignment.

- **Public Knowledge (The Architecture):**
    - The core Transformer architecture, which underpins most modern LLMs, was published by Google in 2017 ("Attention Is All You Need").
    - Many critical algorithms (e.g., RLHF concepts, various optimization techniques) are also openly discussed in research papers.
- **Proprietary "Secrets" (The Differentiation):**

    - **Massive Compute & Data:** The sheer scale of compute clusters and proprietary, high-quality training data (especially for RLHF) is a huge barrier to entry and a key differentiator. OpenAI, Google, etc., invest billions here.
    - **Engineering Expertise & Infrastructure:** Building and managing these massive training pipelines, optimizing model performance, and deploying them reliably at scale requires unparalleled engineering talent and sophisticated infrastructure.
    - **Algorithmic Refinements & Hyperparameters:** Small, non-public tweaks to training procedures, loss functions, specific hyperparameters, and data curation strategies can lead to significant performance differences. These are often discovered through extensive experimentation and are guarded closely.
    - **Alignment Techniques:** The specific methods and datasets used for RLHF and safety alignment are highly refined and often proprietary, contributing to safer and more useful models.
- **Hiring Key People:**
    - **Abilities:** Deep expertise in machine learning theory, distributed systems, high-performance computing, data science, mathematics, and often linguistics/cognitive science. They are leaders in research and engineering.
    - **Origin:** Top research universities (Ph.D.s), existing major tech companies, and pioneering AI labs. They are at the forefront of pushing the boundaries of the field.
    - **Curriculum for the "New World":** Strong foundations in linear algebra, calculus, probability/statistics, computer science (algorithms, data structures, distributed systems), machine learning (deep learning, neural networks, reinforcement learning), and potentially computational linguistics or cognitive science. Practical experience with large-scale data and model development is paramount.

9. What are the current frontiers (easy/difficult/hardly possible)? What is possible locally?

Key Message: While foundational model training remains difficult and expensive, the application and fine-tuning of smaller models are increasingly accessible.

- **Easy/Established:**
  - **Text generation:** Summarization, translation, creative writing, basic Q&A.
  - **Tool use:** Integrating with web search, calculators, basic APIs.
  - **Prompt Engineering:** Getting good results from existing models.
- **Difficult/Expensive (but improving):**
  - **Complex multi-step reasoning:** Requiring extensive planning and self-correction.
  - **Grounding in real-world facts/truthfulness:** Hallucination remains a challenge.
  - **Robust common sense reasoning:** Handling novel or ambiguous situations like humans.
  - **Long-context understanding:** Reliably processing and remembering information over extremely long documents or conversations.
  - **Training large foundation models from scratch.**
- **Hardly Possible/Out of Reach (for now):**
  - **True AGI/Human-level intelligence:** Models still lack genuine consciousness, self-awareness, and the ability to learn entirely new concepts from minimal exposure in a truly human-like way.
  - **Reliable and creative scientific discovery without human guidance:** Generating fundamentally new theories or solving unsolved mathematical problems autonomously.
  - **Perfect factual recall and truthfulness:** Eliminating hallucinations entirely.
- **Locally (on a reasonable desktop) in terms of education/re-education/dialog:**
  - **Running smaller open-source LLMs:** Models like Llama 3 (8B parameters) or even Qwen3 (30B parameters), Mistral, or specialized models can run on consumer GPUs (e.g., 16-24GB VRAM) for local inference. This allows for private, uncensored dialog, code generation, creative writing.
  - **Parameter-Efficient Fine-Tuning (PEFT/LoRA):** Individuals or small teams can fine-tune these smaller open-source models on custom datasets to specialize them for specific tasks or knowledge domains relevant to education or re-education.
  - **RAG systems:** Building local knowledge bases that can be queried by a locally run LLM for specialized learning.

- **Educational aids:** Generating study guides, practice questions, explaining complex topics, language learning support (grammar checks, vocabulary building).
- **Personal assistants:** Task management, idea generation, writing assistance.

Conclusion: A New Paradigm

- Reiterate that LLMs are far more than simple token predictors; they are sophisticated systems leveraging advanced architectures, vast data, iterative processing, and tool integration.
- Emphasize the emergent capabilities that challenge traditional views of intelligence and language.
- Conclude with the rapid pace of development and the exciting, yet still challenging, frontiers ahead.

Also: What is happening in specialized topics like audio/video/photo analysis and generation?

Key Message: Multimodal models extend the token-prediction paradigm to other data types by "tokenizing" them, enabling capabilities like vision-to-text, speech-to-text/generation, and even music analysis.

- **Multimodal LLMs:** These models are designed to process and generate information across different modalities (text, image, audio, video).
- **How it Works (Replacing Tokens):**
  - **Tokenization for Other Modalities:** Just as text is broken into linguistic tokens, other data types are broken into analogous "tokens" or embeddings:
    - **Images:** Images are often divided into patches, and each patch is converted into a vector (an "image token" or "visual token") that the transformer can process. These are sometimes called "visual tokens" or "patches." [7]
    - **Audio:** Audio waveforms can be processed through a feature extractor (e.g., mel-spectrograms) and then quantized or embedded into sequences of "audio tokens."
    - **Video:** Video combines image patches (frames) and potentially audio tokens, adding a temporal dimension.

    - **Joint Embedding Space:** These multimodal tokens (text, visual, audio) are then mapped into a shared, high-dimensional embedding space. This allows the model to understand the relationships *between* different modalities.
- **Applications:**
    - **Image Captioning/VQA:** Describing images or answering questions about them (e.g., "What is in this picture?").
    - **Image Understanding**: Models can reason about images, e.g.
        - Take a photo of a restaurant bill and tell the LLM "I only had steak and tea, how much is my share?"
        - Take a photo of the inside of your fridge and ask "what can I make for dinner without going shopping?"
    - **Music Analysis:**
        - **Writing a score from video/sound:** This is a challenging task. Models can transcribe *some* audio into musical notation, but accurately scoring a symphony from a video with sound (capturing all instruments, nuances, dynamics) is still very difficult. It requires deep musical understanding beyond simple token prediction.
        - **Finding errors in a performance:** Possible for *obvious* errors if the model has been trained on sufficient examples of "correct" and "incorrect" performances and has a robust representation of the musical piece. This would involve comparing the input performance's "audio tokens" or features against a canonical representation. Advanced capabilities like discerning subtle expressive differences from technical errors are still research frontiers.
        - Not sure if anything of this is a subject of current research

Key Message 2: On the generation side, most of the current models are of different architecture, they are "diffusion models", not token-based (mostly). However, they show impressive results.
Image Generation: *Creating images from text prompts (Midjourney, Qwen Image, Gpt-4o, Kandinsky)*
Image Editing: By textual prompts: "this same person but in a beach wearing a tuxedo" (Nano-banana, Qwen Image Edit)
Video Generation: *Create video from text prompts; create video from the first image and text, etc (Veo 3, Sora 2)*

Audio Synthesis: Text-to-speech which finally you can listen to for long(-ish) periods of time