Lecture 02: Fundamentals of Large Language Models (LLMs)

🎯 Learning Objectives

By the end of this lecture, you should be able to:

- Understand the core architecture behind LLMs: transformers.
- Describe how LLMs are pretrained and fine-tuned.
- Explain the difference between tokenization, embeddings, and attention.
- Identify the components that make LLMs useful for agentic systems.

🧩 Key Concepts

What Is a Large Language Model (LLM)?

- A neural network trained on vast amounts of text data to predict the next token in a sequence.
- Learns grammar, facts, reasoning patterns, and domain knowledge through training.

Transformer Architecture

- Introduced in *"Attention is All You Need"* (Vaswani et al., 2017).
- Key components:
  - **Embedding Layer**: Converts tokens into vectors.
  - **Self-Attention Mechanism**: Captures relationships between tokens, regardless of position.
  - **Feedforward Layers**: Learn transformations of token representations.
  - **Layer Normalization & Residual Connections**: Stabilize training and allow deep stacking.

Pretraining and Fine-Tuning

- **Pretraining**: Unsupervised learning from large text corpora via next-token prediction.
- **Fine-tuning**: Supervised learning on specific datasets/tasks (e.g., instruction-following).

Tokenization

- Input text is broken into **tokens** (e.g., subwords, characters).
- Models process sequences of tokens, not raw text.

Why LLMs Matter for Agentic AI

- Capable of multi-step reasoning and decision-making via prompting.
- Can integrate knowledge, language understanding, and planning.
- Serve as the reasoning "core" of many autonomous agents.

🛠️ Required Tools/Libraries

- Hugging Face Transformers
- Python 3.8+
- (Optional) OpenAI API key for quick experimentation

Hands-on Exercise: Exploring a Transformer

Goal: Load a small pretrained transformer and inspect its internals.

Steps:

1. Install Hugging Face Transformers:

```
pip install transformers
```

2. Load a small model and tokenizer:

```
from transformers import AutoModel, AutoTokenizer

model_name = "distilbert-base-uncased"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = AutoModel.from_pretrained(model_name)

text = "Agentic AI is the future."
inputs = tokenizer(text, return_tensors="pt")
outputs = model(**inputs)
print(outputs.last_hidden_state.shape)
```

3. Visualize:
    - Use `.named_parameters()` to explore weights.
    - Discuss what each layer is doing.

Bonus: Compare tokenization results for different models.
