Lecture 7: BERT and GPT
COMP 5801H/4900A: Generative AI and LLMs

2026-01-27

Sriram Subramanian
Assistant Professor & Canada Research Chair, Carleton University
Faculty Affiliate, Vector Institute for Artificial Intelligence
Faculty Affiliate, Schwartz Reisman Institute for Technology and Society

Carleton University
VECTOR INSTITUTE
SCHWARTZ REISMAN INSTITUTE FOR TECHNOLOGY AND SOCIETY

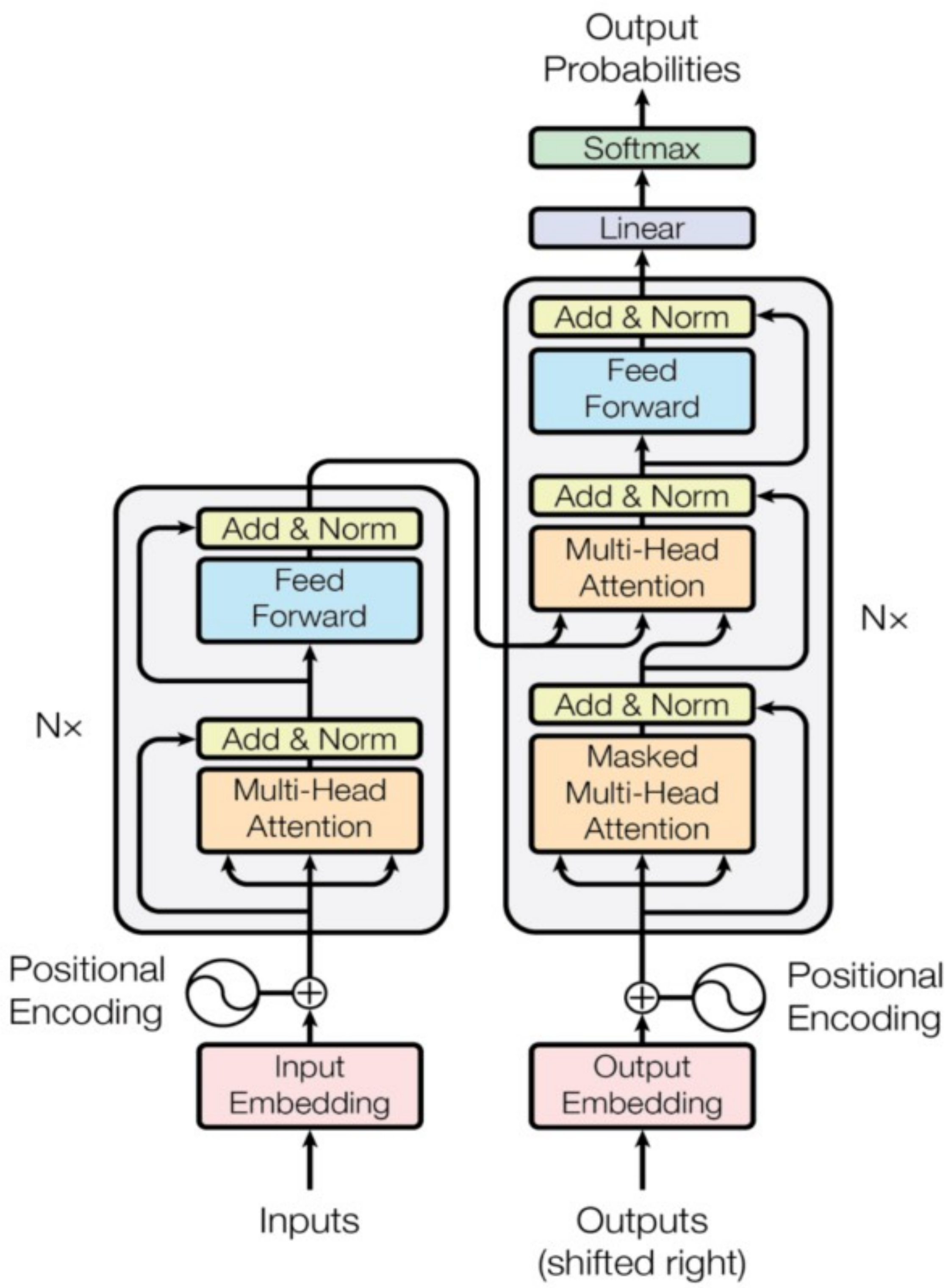
Lecture Outline

- Splitting Transformers into two separate paths
- BERT – The Bidirectional Encoder
- GPT – The Generative Decoder

BERT vs. GPT

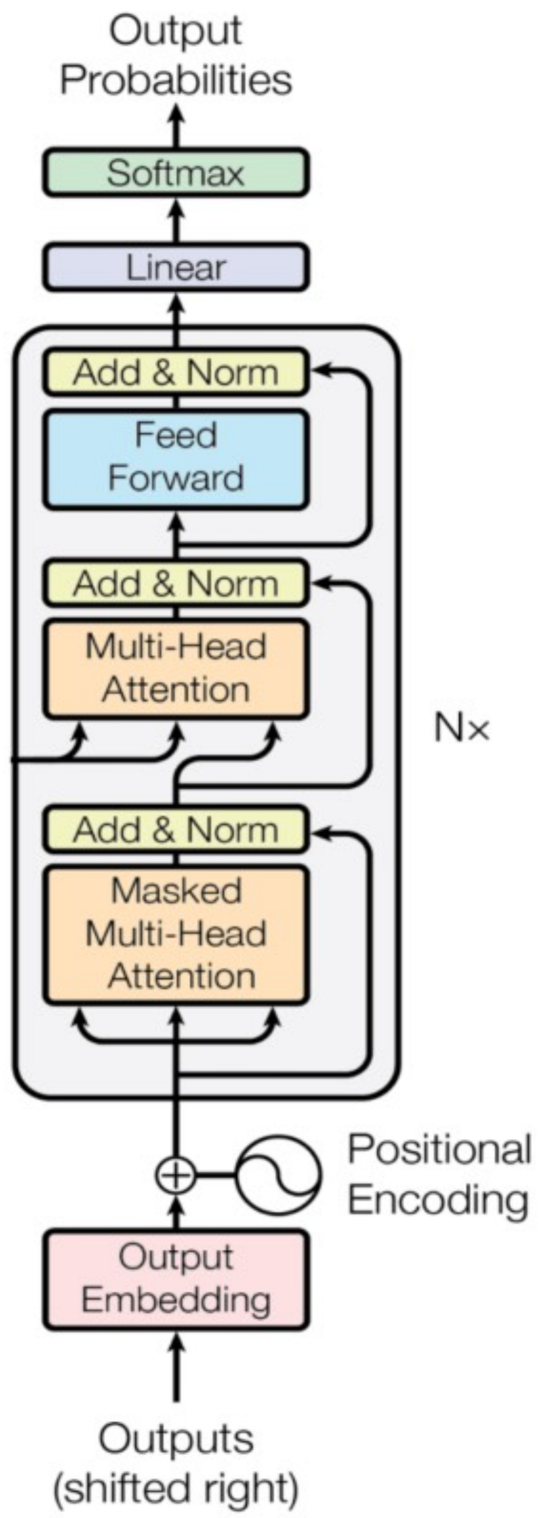
- **The Transformer “Split"**
  - BERT: Bidirectional Understanding
  - GPT: Autoregressive Generation
- **The Philosophical Split**
  - BERT (The Reader): Designed to "understand" a complete sequence. It looks at the whole sentence at once to capture deep context
  - GPT (The Writer): Designed to "generate" a sequence. It builds a sentence one word at a time, moving strictly from left to right

Transformer



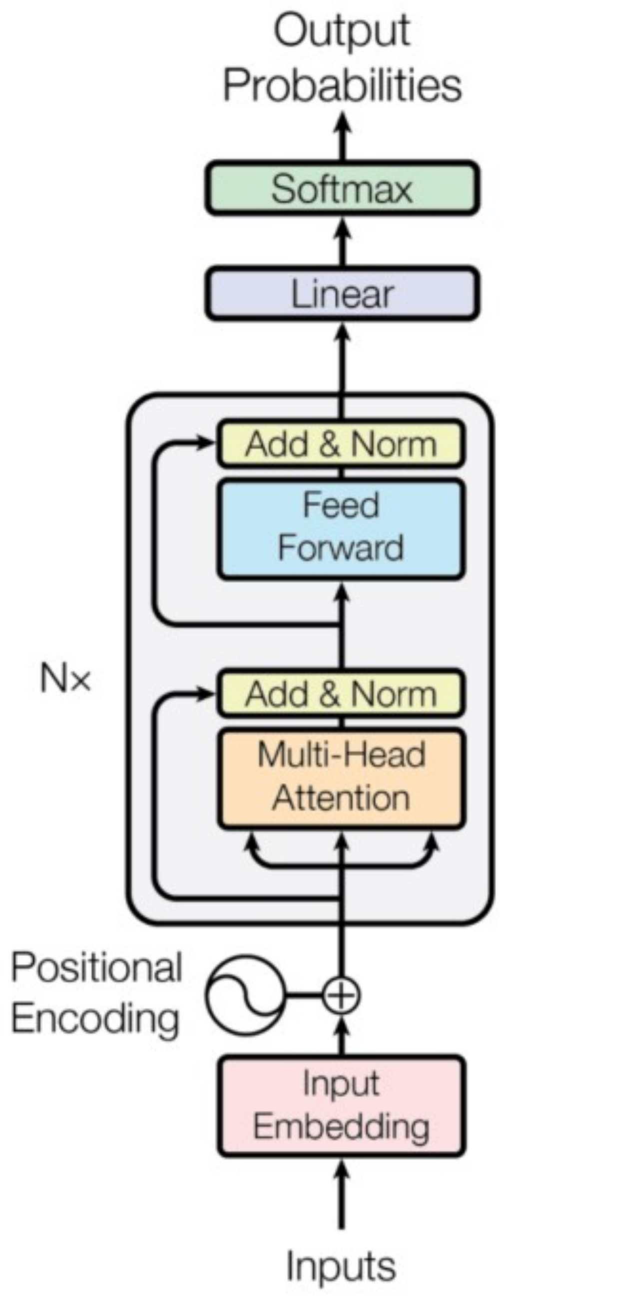
Encoder **Decoder**

GPT*



Decoder-only

BERT*



Encoder-only

*Illustrative example, exact model architecture may vary slightly

BERT vs. GPT - Structural Comparison

Feature	BERT	GPT
Attention	Bidirectional: Every token sees every other token.	Unidirectional: A token only sees previous tokens.
Objective	Context	Next token prediction
Best For	Classification, QA, Sentiment, NER	Creative Writing, Coding, Summarization
Architecture	Encoder Only	Decoder Only

The "Pre-train then Fine-tune" Paradigm

- **Phase 1: Pre-training (The Generalist)**
  - The Goal: Teach the model "how language works" using massive, unlabeled datasets (e.g., Wikipedia, Common Crawl)
  - Self-Supervision: No human labeling is required. The model learns by predicting missing words (BERT) or next words (GPT)
  - Outcome: A set of weights that understand grammar, facts, and basic reasoning

The "Pre-train then Fine-tune" Paradigm

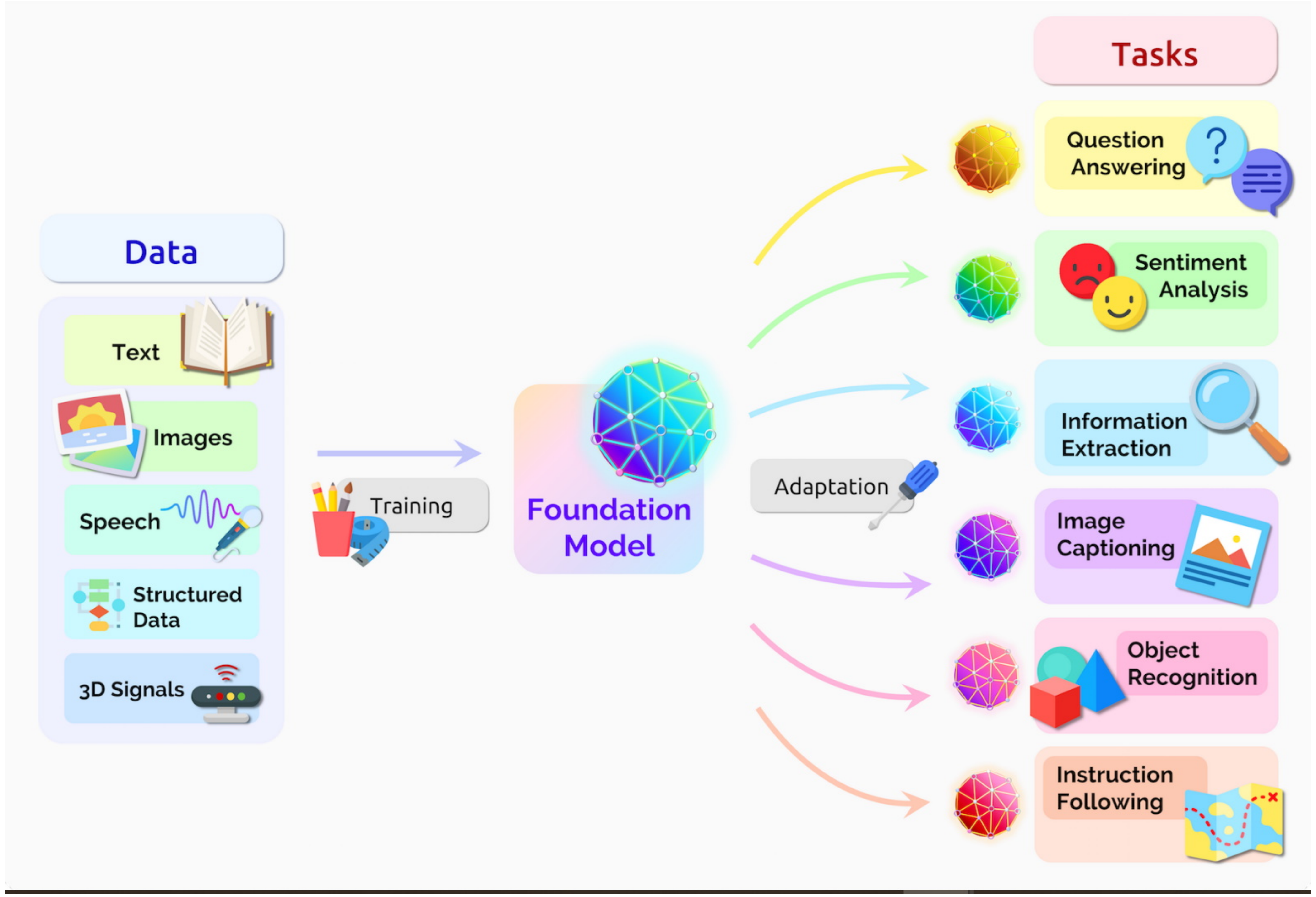
- **Phase 2: Fine-tuning (The Specialist)**
    - The Goal: Adapt the generalist model to a specific, narrow task (e.g., "Is this email spam?")
    - Supervised Learning: Uses a much smaller, human-labeled dataset
    - Outcome: A specialized model that leverages its "world knowledge" to solve a niche problem with very little data

The "Pre-train then Fine-tune" Paradigm

Why it Changed Everything

- **Data Efficiency:** You don't need 1 million labeled examples anymore; 1,000 might suffice because the model already knows English.
- **Transfer Learning:** Knowledge gained from the entire internet is "transferred" to your specific problem

Birth of Foundation Models



Credit: https://blogs.nvidia.com/blog/what-are-foundation-models/

The Landscape of LLMs — A Timeline of Scaling

- **The Foundation (2017 - 2018)**
  - The Transformer (2017): The "Big Bang" of modern NLP
  - GPT-1 (2018): Demonstrated that Generative Pre-training works
  - BERT (2018): Dominated understanding tasks (Search, Sentiment)
- **The Scaling Era (2019 - 2022)**
  - GPT-2 (2019): 1.5B parameters. Showed "Zero-Shot" potential
  - GPT-3 (2020): 175B parameters. The first "LLM" to capture public attention
  - PaLM & Chinchilla (2022): Focused on "Optimal Scaling"—it's not just about more parameters, but more high-quality data
- **The "Instruction" & Open-Source Explosion (2023 - 2026)**
  - ChatGPT / GPT-4: Introduced RLHF (Human Alignment), making models helpful and safe
  - Llama Series (Meta): Democratized AI. High-performance models that can run on consumer hardware
  - The Modern Titans (2025/26): Models like GPT-4o, Claude 3.5, and Llama 3 with multi-modal (image/voice) capabilities and massive context windows (1M+ tokens)

BERT Defined — The "Understanding" Revolution

- **What is BERT?**
  - Bidirectional Encoder Representations from Transformers
  - A model architecture designed to extract **contextual embeddings** for every word in a sentence
- **The Core Innovation: True Bidirectionality**
  - **Traditional Models (LSTMs)**: Read left-to-right (or right-to-left) and then concatenate the results
  - **BERT**: Uses the Transformer Encoder to "see" the entire sequence at once in every layer
  - **The "Context" Difference**: In the sentence "The bank of the river," BERT understands "bank" by looking at "river" before it ever outputs a representation

BERT Defined — The "Understanding" Revolution - II

- **Two Standard Sizes (Original Paper)**
  - BERT-Base: 12 Layers, 768 Hidden Size, 12 Attention Heads (110M Parameters)
  - BERT-Large: 24 Layers, 1024 Hidden Size, 16 Attention Heads (340M Parameters)
- **Impact**
  - Shattered state-of-the-art records on the GLUE (General Language Understanding Evaluation) benchmark
  - Became the backbone of Google Search in 2019 to better understand conversational queries

Inside the Stack — Scaling the Encoder

- Two Standard Sizes: BERT-Base and BERT-Large
- Why "Deeply Stacked"?
  - Hierarchical Learning:
    - Early layers (1-4) capture local syntax and grammar
    - Middle layers (5-16) capture semantic relationships
    - Deep layers (17-24) capture high-level abstract concepts and document-wide context
  - Feature Refinement:
    - Each layer takes the contextual embedding from the previous layer
    - Refines it through another round of Multi-Head Self-Attention

Inside the Stack — Scaling the Encoder - II

- The Math of Hidden Sizes
  - The Hidden Size (H) is the dimensionality of the vector representing each token
  - In BERT, the size of each attention head is fixed at $d_k = H/A = 64$
  - This consistency ensures that the computational cost of attention remains stable relative to the model's width
- **The Magic Number:** Interestingly, almost all Transformer models (BERT, GPT-2, GPT-3) keep the head size (d_k) at **64**. Whether the model is small or massive, they just add *more* heads rather than making the heads larger

Visualizing the Head Split

- **The Input**: A single word enters the layer as a 768-D vector.
- **The Division of Labor**: The model "slices" that 768-D space into 12 chunks.
- **Parallel Processing**: Head 1 (64 dimensions) might focus on Grammar.
  - Head 2 (64 dimensions) might focus on Entity Relationships.
  - Head 3 (64 dimensions) might focus on Verb Tense.
- **The Re-assembly**: After each head does its math, the 12 resulting 64-D vectors are concatenated back together to form the original 768-D shape

BERT Architecture

- **The "Encoder-Only" Foundation**
  - BERT is built by stacking **Transformer Encoder blocks**.
  - Unlike the original Transformer, it discards the Decoder entirely.
  - **Key Feature**: Every layer uses **Full Self-Attention** (No masking). This means there are no "blind spots"—every token can see every other token
- Inside the Encoder Block: Each of the 12 (Base) or 24 (Large) layers contains:
  - **Multi-Head Self-Attention:** The "Bidirectional" core where Q attends to all K and V
  - **Add & Norm**: Residual connections and Layer Normalization to prevent the "Vanishing Gradient" problem
  - **Feed-Forward Network (FFN)**: Two linear layers with a GELU activation function in between

BERT’s Training Objectives — I

- **Masked Language Modelling (MLM)**
  - The "Cloze" Task: BERT hides 15% of the words in a sentence and tries to predict them
  - Breaking the "Cheating" Problem: Because some words are replaced with a [MASK] token, the model is forced to use the surrounding words (both left and right) to guess the missing piece
  - Deep Context: To guess the word "Bank" in "The [MASK] of the river," BERT must understand the relationship between all the words simultaneously

BERT’s Training Objectives — II

- **Next Sentence Prediction (NSP)**
  - The Context Task: BERT is given two sentences (A and B) and must decide: Does Sentence B actually follow Sentence A in the original text?
  - Binary Classification:
    - 50% of the time: B is the actual next sentence (Label: IsNext)
    - 50% of the time: B is a random sentence from the corpus (Label: NotNext)
  - The Goal: This teaches BERT to understand long-term relationships between sentences, which is vital for tasks like Question Answering

Why Bidirectional Matters — Seeing the Whole Picture

- **The "Uni-directional" Weakness**
  - In models like GPT or standard LSTMs, the representation of a word is only based on what came before it
  - Example: "The bank..."
    - Is it a river bank? A financial bank? A "bank" shot in basketball?
    - A uni-directional model has to "guess" until it sees the next words.
- **The "Bi-directional" Advantage**
  - BERT looks at the entire sequence in one pass
  - Example: "The bank of the river was flooded."
    - When computing the Query (Q) for "bank," BERT attends to "river" (the right context) and "The" (the left context) simultaneously

Why Bidirectional Matters — Math

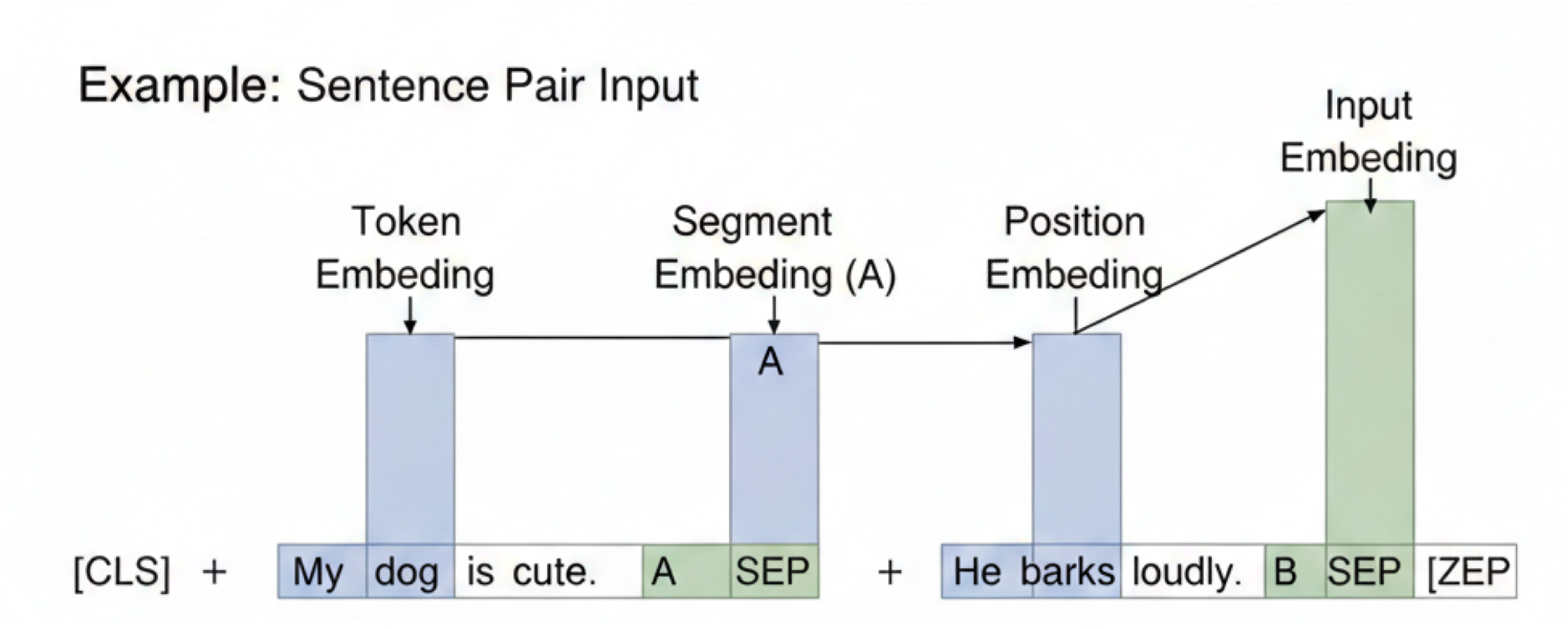
- **The Math: Unmasked Self-Attention**
  - In the Attention formula: softmax$\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, there is **no masking matrix**
  - Every row in the attention matrix can have non-zero values for every column
  - This allows the model to "fuse" information from the future and the past into a single, rich vector

Discussion: "What is the difference between Bidirectional RNNs and BERT?"

BERT’s Input Representation — The Sum of Three Worlds

- **Challenge:** BERT needs to handle single sentences and pairs of sentences (for tasks like Question Answering) in a way that the model can distinguish between them, all while being a non-sequential model
- **The Solution: Triple Embedding Summation** (Stacked Embeddings)
    - For every input token, BERT creates a single vector by summing three distinct embeddings:
        - **Token Embeddings**: The meaning of the word. BERT uses WordPiece tokenization (e.g., "playing" becomes "play" + "##ing")
        - **Segment Embeddings**: The "Location" of the sentence. Tokens in Sentence A get vector E_A; tokens in Sentence B get vector E_B. This tells the model which sentence the word belongs to.
        - **Position Embeddings**: The "Order" of the word. Since Transformers don't have recurrence, this vector tells the model if the word is at index 1, 2, or 100
- **The Special Tokens**
    - [CLS] (Classification): Always the first token of every sequence. The final hidden state of this token is used as the "summary" of the entire input for classification tasks
    - [SEP] (Separator): A special marker used to divide Sentence A from Sentence B

Stacked Embeddings



Fine-tuning BERT — From Generalist to Specialist

- **The Architecture of Fine-Tuning**
  - The "Base" stays mostly the same: We take the pre-trained BERT model and load its "knowledge" (weights).
  - The "Head" is new: We add a single, untrained Linear Layer on top of the [CLS] token's output.
  - Minimal Parameters: We only really need to train the weights of that final layer to map the 768-D [CLS] vector to our specific labels (e.g., "Spam" vs. "Not Spam").
- **How the [CLS] Token Solves Tasks**
  - The final hidden state of the [CLS] token acts as a fixed-length summary of the variable-length input.
  - For Sentiment Analysis: We map [CLS] → 2 output nodes (Positive/Negative).
  - For Topic Classification: We map [CLS] → N output nodes (Sports, Politics, Tech).
- **Token-Level Fine-Tuning (NER)**
  - Sometimes we don't use the [CLS] token. For Named Entity Recognition (NER), we look at the final hidden state of every individual token to decide if it is a "Person," "Location," or "Organization."

GPT Defined — The Generative Revolution - I

- **What is GPT?**
  - Generative Pre-trained Transformer (Radford et al., 2018)
  - An architecture designed for **NLG (Natural Language Generation)**—the ability to produce coherent, human-like text
- **The Architecture: Decoder-Only**
  - While BERT took the "Encoder" from the original Transformer, GPT took the "**Decoder**"
  - It removes the Encoder-Decoder attention layers and focuses entirely on a stack of **Masked Self-Attention** blocks

GPT Defined — The Generative Revolution - II

- **The Fundamental Mechanism: Autoregression**
  - GPT predicts the **next token** in a sequence based solely on the tokens that came before it.
  - $P(x_n \mid x_1, x_2, \ldots, x_{n-1})$
  - Each word generated becomes part of the input for the next step, creating a "feedback loop" of creation.
- **Pre-training Task: Causal Language Modeling (CLM)**
  - Unlike BERT’s "fill-in-the-blank" (MLM), GPT’s only job is to guess the future.
  - This is a "Left-to-Right" objective that mirrors how humans speak and write.

GPT Architecture - I

- The "Decoder-Only" Modification
  - The original Transformer Decoder had three main sub-layers. GPT removes the middle one:
    - **Masked Multi-Head Self-Attention**: (Kept) Forces the model to look only at the past.
    - ~~**Encoder-Decoder Attention:~~ (Removed)** In GPT, there is no "Encoder" to talk to, so this "cross-attention" layer is deleted.
- Feed-Forward Network: (Kept) Processes the features gathered by the attention layer

GPT Architecture - II

Model	Layers (L)	Hidden Size (H)	Heads (A)
GPT-1	12	768	12
GPT-2 (Extra Large)	48	1600	25
GPT-3	96	12,288	96

GPT Architecture - II

Why the Stack Matters?

- GPT is built by stacking these modified blocks (12 in GPT-1, 12-48 in GPT-2, and 96 in GPT-3)
- **Information Flow**: Each layer takes the sequence of vectors from the layer below and uses the **Causal Mask** to update them
- **The "Final Layer" Logic**: The top layer’s output for the very last token is passed through a Linear layer and a Softmax to pick the next word from the vocabulary

Causal Masking

- The Problem of "Cheating"
  - Unlike BERT, which is trained to fill in holes, GPT is trained to predict the next word
  - During training, we give GPT the whole sentence at once (to make it fast)
  - The Risk: If the model can "see" the next word in the matrix, it will just copy it instead of learning how to predict it
- The Solution: The Triangular Mask
  - We apply a Masking Matrix to the Attention scores before the Softmax layer
  - This matrix is a Lower Triangular Matrix (filled with 0s on the bottom-left and $-\infty$ on the top-right)
  - The Math: $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T + M}{\sqrt{d_k}}\right)V$
  - Any "future" connection is added to $-\infty$, effectively zeroing out the attention to future tokens

Autoregressive Modelling

- What is Autoregression?
  - **Auto (Self) + Regression (Predicting)**: A process where the model uses its own previous outputs as the inputs for the next step
  - **The Formula**: $y_t = f(y_{t-1}, y_{t-2}, \ldots, y_0)$
  - In GPT, the model predicts one token at a time. Once a token is predicted, it is appended to the sequence, and the entire sequence is fed back into the model to predict the next one

Autoregressive Modelling - Example

- The Inference Steps
  - **Input:** "The cat"
  - **Step 1:** Model predicts "sat" → New Input: "The cat sat"
  - **Step 2:** Model predicts "on" → New Input: "The cat sat on"
  - **Step 3:** Model predicts "the" → New Input: "The cat sat on the"
  - **Termination:** The process repeats until the model generates an “endoftext” token or hits a length limit

Autoregressive Modelling - Probabilistic Sampling

- Probabilistic Sampling
  - At each step, GPT doesn't just pick "the" word; it generates a **probability distribution** over the entire vocabulary (e.g., 50,000+ words).
  - **Greedy Search**: Always pick the highest probability.
  - **Top-K / Nucleus Sampling**: Pick from a pool of likely words to add variety and "creativity."

GPT-1 to GPT-2 — The Rise of Zero-Shot

- **Scaling Up (Size Matters)**
  - **GPT-1 (2018)**: 117 Million parameters. Required fine-tuning for specific tasks (like BERT).
  - **GPT-2 (2019)**: 1.5 Billion parameters. A 10x increase in scale and trained on "WebText" (40GB of high-quality internet data).
- **The Paradigm Shift: "Zero-Shot" Learning**
  - **The Old Way**: Pre-train on general text → Fine-tune on specific labels (e.g., "Is this a positive review?").
  - **The GPT-2 Way**: The model performs a task **without any specific training** for it.
  - **How?** By treating everything as a prediction problem.
    - Example: If you feed the model "Translate 'Apple' to French: ", the most likely "next token" it learned from the internet is "Pomme."

GPT-1 to GPT-2 — The Rise of Zero-Shot

- **"Task-Specific" becomes "Prompt-Specific"**
  - Instead of changing the model's weights (Fine-tuning), we change the Input Context (Prompting)
  - GPT-2 showed that as models get larger, they naturally begin to "reason" through tasks they were never explicitly told to do
- **Advantages**
  - Extreme Agility & Speed (no training time & rapid prototyping)
  - Reduced Data Dependency
  - Operational Efficiency (The "One Model" Rule)
  - Accessibility (No-Code AI)

GPT-3 & In-Context Learning — The Death of Fine-Tuning?

- **Extreme Scale**
  - **Parameters**: 175 Billion (100x larger than GPT-2)
  - **The "More is Different" Effect**: At this scale, the model doesn't just predict words; it exhibits **In-Context Learning (ICL)**
- **What is "Few-Shot" Prompting?** Instead of changing the model's "brain" (fine-tuning), we provide examples in the Prompt
  - **The Prompt:**
    - "Great movie!" → Positive
    - "Terrible acting." →Negative
    - "It was okay." →
  - **The Model:** Predicts "Neutral" simply by following the pattern provided in the context.

GPT-2 vs. GPT-3

- **Size and Parameters (The 100x Jump)**
  - **GPT-2**: 1.5 Billion parameters (at its largest)
  - **GPT-3**: 175 Billion parameters
  - **The impact**: GPT-3 is massive enough to "store" more facts, more languages, and more complex logic patterns within its weights
- **Training Data (Quality and Quantity)**
  - **GPT-2:** Trained on "WebText" (40GB), mostly outbound links from Reddit with at least 3 upvotes
  - **GPT-3:** Trained on the "Common Crawl" (nearly 600GB), which includes almost the entire indexed internet, plus high-quality books and the entirety of Wikipedia

GPT-2 vs. GPT-3

- **Emergent Property: In-Context Learning**
  - **GPT-2 (Zero-Shot)**: Could sometimes perform tasks it wasn't trained for, but it was often "hit or miss."
  - **GPT-3 (Few-Shot)**: Introduced In-Context Learning. It can follow complex patterns. If you give it three examples of a new, made-up language, it can translate a fourth word instantly.

- **Context Window (Short-term Memory)**
  - **GPT-2:** 1,024 tokens
  - **GPT-3:** 2,048 tokens
- Note: GPT-3 can "remember" and look back at twice as much text as GPT-2 during a single conversation