The Transformer Architecture

From Word2Vec to Dynamic Context

Iverina Ivanova & Gemini 3 Pro

Goethe Universität Frankfurt

Syllabus Context: Connecting the Dots

Where does today's topic fit into our timeline?

- **Session 3 (Oct 27):** We studied **N-grams**.
  - *Limitation:* They only look at immediate neighbors. They can't "remember" long-distance dependencies.
- **Session 6 (Nov 17):** We studied **Word2Vec**.
  - *Limitation:* Static Embeddings. The word **bank** always has the same vector, whether it's a river bank or a credit bank.
- **Session 7 (Nov 24): Karpathy Part 2**.
  - We saw the "Psychology" of LLMs (Hallucinations, Personas).
  - Today, we look at the **Anatomy**—the architecture that fixed the limitations of N-grams and Word2Vec.

High-Level View: The Translation Machine

The Transformer was originally built for **Machine Translation** (e.g., French $\rightarrow$ English).

$$\textit{Input (French)} \rightarrow \textbf{[THE MODEL]} \rightarrow \textit{Output (English)}$$

Inside the model, there are two distinct stacks:

1. **The Encoder (The Reader):**
   - Reads the input text.
   - Creates a "mental representation" (vectors) of the concepts.
2. **The Decoder (The Writer):**
   - Takes the representation from the Encoder.
   - Generates the output text one token at a time.

Note on GPT

GPT (Generative Pre-trained Transformer; the model behind ChatGPT) is technically just a massive stack of **decoders**. It focuses purely on generating the next token.

The Fundamental Shift: Context is King

To understand the Transformer, we must understand the problem it solves.

The Ambiguity Problem

Consider the word: **scale**

1. *The fisherman removed the* **scale** *from the fish.* (fish anatomy/noun)
2. *How would you rate his work on a* **scale** *from 1-10?* (a set of numbers used to measure or compare the level of sth/noun)
3. *Please* **scale** *the image to fit.* (resize/verb)

Word2Vec's failure: It assigns one single list of numbers to **scale** that averages all these meanings.
Transformer's solution: It creates a unique representation for **scale** *dynamically* for every sentence, based on the words around it.

The Solution: Self-Attention

What is Attention in AI?
It is a mechanism that allows words to "talk" to each other to clarify their meaning.

Example:

The **animal** *didn't cross the* **street** *because* **it** *was too tired.*

When the model processes the word **it**:

1. It looks at every other word in the sentence.
2. It asks: *How relevant are you to me?*
3. **Street** says: *Not very relevant.* (Low score)
4. **Animal** says: *I am what you refer to!* (High score)
5. The model updates the meaning of **it** to include the concept of **animal**.

How Attention Works: The Search Engine Analogy

Jay Alammar uses the concepts of **Query**, **Key**, and **Value**. This is the math behind the magic.

Imagine a Google Search:

- **Query (Q):** What you type into the search bar. What am I looking for?
- **Key (K):** The metadata/tags that websites use to describe themselves. What do I have?
- **Value (V):** The actual content of the website. What information do I provide?

How Attention Works: The Search Engine Analogy

The Process in the Transformer:

1. The word **it** broadcasts a **Query**: *I am a pronoun looking for a noun antecedent.*
2. The word **animal** holds up a **Key**: *I am a noun subject.*
3. The system calculates a match! (High attention score).
4. The word **animal** sends its **Value**: *Here is my biological meaning.*
5. **it** absorbs this value and updates its meaning.

Why stop at one search? (Multi-Head Attention)

A single word might need to focus on different things for different reasons.

Example: *The* **bank** *of the river.*

- **Attention Head 1 (Grammar focus):**
  - **Query:** *I am a noun, looking for my determiner.*
  - **Result:** Focuses on the determiner **The**.
- **Attention Head 2 (Semantic focus):**
  - **Query:** *I need context to know if I am money or nature.*
  - **Result:** Focuses on **river**.

The Transformer runs 8 to 96 of these searches simultaneously. This gives it a rich, multi-dimensional understanding of the text.

The Missing Piece: Order Matters

The Problem: Attention looks at all words at the same time. To the math, these two sentences look identical:

- *Venom defeated Spider-Man.*
- *Spider-Man defeated Venom.*

The Fix: Positional Encoding

- The model injects a mathematical pattern into the embeddings.
- It acts like a timestamp or a seat number.
- It ensures the model knows that **venom** appears *before* **defeated** in the first sentence.

Summary: Inside the "Black Box"

Recapping the architecture:

1. **Input:** Words are turned into vectors.
2. **Position:** We stamp them with their order.
3. **Attention (The Brain):**
   - Words act like **Search Engines** (Query/Key/Value).
   - They absorb meaning from relevant context words.
4. **The Stack (Encoder/Decoder):** We stack these layers to build deep understanding (Encoder) or generation (Decoder).
5. **Output:** A probability distribution for the next token.

The Feed-Forward Layer: Digesting the Information

Phase 1 (Attention): The words "talked" to each other. **It** figured out it refers to **Animal**.

Phase 2 (Feed-Forward): Now, each word needs to process this new information **privately**.

The Analogy: Group Work vs. Individual Study

- **Attention (Group Work):** You ask your classmates for clarification on a topic.
- **Feed-Forward (Individual Study):** You sit back down at your desk and integrate that new knowledge into your own understanding.

Technical Note: This layer processes every word *independently* and in parallel. This is where the model "thinks" about the context it just gathered.

The Output: From Vectors to Words

After passing through many layers of Attention and Feed-Forward processing, we still have a list of numbers (a vector). How do we get a word?

Step 1: The Linear Layer (The Vocabulary Shelf)

- The model projects its final vector onto the entire vocabulary (e.g., 50,000 words).
- It assigns a raw score (logit) to every possible word in the dictionary.
- *Example:* **apple:** 5.2, **banana:** 1.1, **car:** -3.0.

Step 2: The Softmax Layer (The Probability Converter)

- Raw scores are hard to interpret. Softmax squashes them into percentages that add up to 100%.
- *Result:* **apple:** 97%, **banana:** 2%, **car:** 0.001%.
- We then sample the next token from this distribution.

Training: The Loss Function (The Strict Teacher)

How does the model learn to get the probabilities right?

The Setup:

- **Input:** *The cat sat on the...*
- **Model's Guess: moon** (50%), **mat** (10%).
- **Actual Answer (Ground Truth): mat**.

The Loss Function (Cross-Entropy):

- It calculates a **penalty score** based on how wrong the model was.
- If the model gave **mat** a low probability, the penalty (Loss) is **High**.
- If the model gave **mat** a high probability, the penalty is **Low**.

We use this **Loss** *score to work backwards (Backpropagation) and adjust the weights so the model makes a better guess next time.*

Tool Use for the Present Slide Deck

The slides were produced in a multi-step process involving LLMs:

- **Syllabus Alignment:** The timeline of the course (N-grams $\rightarrow$ Word2Vec $\rightarrow$ Transformers) was structured using Gemini 3 Pro to ensure pedagogical continuity.
- **Concept Refinement:** Gemini 3 Pro was used to explain technical concepts for students in the humanities.
- **Typesetting:** LaTeX code generated by Gemini 3 Pro to match the visual identity of the seminar slides.