Lecture 18, 19, 20:
Attention Mechanisms, Transformers, and Modern Architectures

Contents

1 Introduction: The Sequence to Sequence Bottleneck

Before the advent of Transformers, sequence problems (such as machine translation) were primarily handled by Recurrent Neural Networks (RNNs) using an **Encoder-Decoder** architecture.

1.1 The RNN Bottleneck

In a standard RNN-based encoder-decoder setup:

1. The **Encoder** processes the input sequence $x_1, \dots, x_T$ step-by-step, updating a hidden state.
2. The final hidden state h_T of the encoder is passed to the **Decoder**.
3. The **Decoder** uses h_T as its initial state to generate the output sequence $y_1, \dots, y_M$ autoregressively.

The Problem: The vector h_T acts as an information bottleneck. It must compress the entire semantic content of an arbitrarily long input sentence (e.g., "The cow jumped over the moon") into a fixed-size vector. This makes training difficult and loses information for long sequences.

1.2 Analogy with U-Nets

In computer vision (U-Nets), this bottleneck was solved using skip connections that pass information from high-resolution encoder layers directly to corresponding decoder layers.

- *Vision:* "Corresponding" is easy; it is spatial (pixel i, j corresponds to pixel i, j).
- *Language:* "Corresponding" is hard. Word order changes during translation. We cannot pre-define a topology of connections (e.g., "always connect the 3rd word to the 5th word").

Solution: We need a dynamic, content-based lookup mechanism. This is the **Attention Mechanism**.

2 Deriving Attention

We can conceptualize the solution as a "differentiable hash table." We have **Queries (Q)**, **Keys (K)**, and **Values (V)**.

2.1 Evolution of the Lookup Idea

Assume we are at a decoder step and have a query vector q. We want to retrieve information from the encoder hidden states (which produce keys k_i and values v_i).

1. **Idea -1: Exact Match.** Return value v_i where $k_i == q$. *Failure:* In high-dimensional floating-point space, exact matches never happen. Gradients are zero almost everywhere.
2. **Idea 0: Closest Match.** Return v_i corresponding to $\arg\min_i \|q - k_i\|$. *Failure:* This is piecewise constant. Small perturbations in q do not change the index of the closest match, leading to zero gradients.
3. **Idea 1: Weighted Average (Soft Lookup).** Instead of picking one, take a weighted average of all values based on similarity.

2.2 Scaled Dot-Product Attention

We define a similarity score s_i between a query q and a key k_i. A standard choice is the inner product. To ensure gradients flow well and outputs represent a probability distribution, we use a Softmax function.

$$\alpha_i = \text{Softmax}(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}} \tag{1}$$

The output is $\sum_i \alpha_i v_i$.

2.2.1 Scaling Factor

If the dimensionality of the vectors d_k is large, the dot product $q \cdot k$ can grow very large in magnitude. This pushes the Softmax function into regions with extremely small gradients (vanishing gradients). To counteract this, we scale by $\frac{1}{\sqrt{d_k}}$.

The complete matrix formulation for inputs packed into matrices Q, K, V is:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{2}$$

2.3 Multi-Head Attention

Instead of a single attention focus, we may want to attend to different aspects of the input simultaneously (e.g., one head attends to grammar, another to semantic context). We split the embedding dimension D into H heads, each with dimension $d_h = D/H$.

- Each head has its own learnable projection matrices W_Q, W_K, W_V.
- Outputs are concatenated and projected back to dimension D.

3 The Transformer Architecture

The seminal paper "Attention Is All You Need" introduced the Transformer, replacing RNNs entirely.

3.1 Key Components

3.1.1 1. Embedding Layers

The input is a sequence of tokens (indices).

- **Input Embedding:** A linear map (lookup) from vocabulary size (e.g., 32k) to embedding dimension D (e.g., 1024).
- **Initialization:** We want the input to attention blocks to be roughly RMS norm 1. If the input is a one-hot vector, the embedding matrix columns should have unit RMS norm.
- **Output (Un-embedding):** A projection from D back to vocabulary size. Ideally, rows should be normalized such that logits entering Softmax do not explode.

3.1.2 2. The Encoder Block

Consists of:

- Multi-Head Self-Attention (unmasked).

- Feed-Forward Network (MLP).
- Residual Connections and Layer Normalization.
- *Note:* Modern architectures (Pre-LN) apply Normalization *before* the Attention/MLP blocks to create a "gradient superhighway."

3.1.3 3. The Decoder Block

Consists of:

- Masked Self-Attention (Cannot look at future tokens).
- Cross-Attention (Queries from decoder, Keys/Values from encoder).
- Feed-Forward Network.

4 Positional Encodings

Since attention is permutation invariant (a set operation), the model has no inherent notion of sequence order. Position information must be injected.

4.1 Evolution of Approaches

1. Absolute Sinusoidal Encoding (Original): Added fixed sine/cosine waves of different frequencies to the input embeddings. (Now largely obsolete).

2. NoPE (No Positional Encoding): Relying solely on the inherent causality in masked attention or data structure. Surprisingly competitive but lacks relative distance awareness.

3. Learned Relative Positional Encoding: Modifies the attention score calculation by adding a learnable bias term b_{t-i} that depends on the relative distance between query position t and key position i.

$$\text{Score}_{t,i} = \frac{q_t^T k_i}{\sqrt{d}} + b_{t-i} \tag{3}$$

4.2 RoPE: Rotary Positional Embeddings

This is the current state-of-the-art. The goal is to modify vectors q and k such that their dot product naturally encodes their relative distance.

4.2.1 Derivation in 2D

Treat a 2D feature vector as a complex number. We rotate the query at position t by angle θt and the key at position i by angle θi. Let the rotation matrix be R_t:

$$R_t = \begin{pmatrix} \cos(\omega t) & -\sin(\omega t) \\ \sin(\omega t) & \cos(\omega t) \end{pmatrix} \tag{4}$$

The dot product becomes:

$$\langle R_t q, R_i k \rangle = (R_t q)^T (R_i k) = q^T R_t^T R_i k = q^T R_{i-t} k \tag{5}$$

The result depends only on the relative position $i - t$.

4.2.2 Generalization to D Dimensions

The embedding vector is divided into $D/2$ pairs. Each pair is rotated with a different frequency f_j.

- Low frequencies change slowly: Capture long-range dependencies.
- High frequencies change rapidly: Capture local information.

This requires no learnable parameters for position; the frequencies are fixed (hyperparameters).

5 Major Architectures

5.1 Encoder-Decoder (e.g., Original Transformer)

Used for sequence-to-sequence tasks like Machine Translation.

5.2 Decoder-Only (e.g., GPT Series)

Focus: Generative tasks (Next token prediction).

- **Structure:** Stack of transformer blocks with **Masked** Self-Attention. No cross-attention.
- **Training:** Autoregressive. Maximizes $P(w_t|w_{1:t-1})$. Every token in the sequence contributes to the loss.
- **Inference:** Tokens are sampled (Top-k, Beam Search) and fed back as input.

5.3 Encoder-Only (e.g., BERT)

Focus: Understanding / Embeddings.

- **Structure:** Stack of transformer blocks with **Full** Self-Attention (bidirectional context).
- **Training Objective 1: Masked Language Modeling (MLM).** Randomly mask 15% of tokens and predict them. Uses both left and right context.
- **Training Objective 2: Next Sentence Prediction (NSP).** Predict if Sentence B follows Sentence A (using a special `[SEP]` token and `[CLS]` classification token).
- **Inefficiency:** To get loss terms, you must process the whole sequence, but only predict the few masked tokens.

6 Modern Innovations (2024-2025 Era)

Recent Large Language Models (LLMs) like Llama, Qwen, and DeepSeek have introduced architectural refinements.

6.1 GLU and SwiGLU

The standard MLP (Linear $\rightarrow$ ReLU $\rightarrow$ Linear) is often replaced by **Gated Linear Units (GLU)**.

$$\text{GLU}(x) = (xW + b) \otimes \sigma(xV + c) \tag{6}$$

A popular variant is **SwiGLU**, using the Swish activation function ($\text{Swish}(x) = x \cdot \text{Sigmoid}(\beta x)$) instead of a simple sigmoid gating. This facilitates gradient flow and acts as a smooth multiplication gate.

6.2 Mixture of Experts (MoE)

To scale parameter count without exploding inference cost.

- **Concept:** Replace dense Feed-Forward layers with a set of "Experts" (smaller MLPs).
- **Router:** A learned gating network decides which expert(s) process a given token.
- **Sparsity:** Only a small fraction (e.g., top-2 experts out of 64) are active per token.
- **Benefit:** Decouples model size (VRAM usage) from compute cost (FLOPs).

6.3 Hybrid Architectures

Models like Jamba or modern Qwen iterations combine different mechanisms:

- **Attention Layers:** For "copying" and high-fidelity recall.
- **State Space Models (SSM/Mamba) / Linear Attention:** For linear-time complexity sequence processing.
- **Partial RoPE:** Applying RoPE only to a fraction of the embedding dimension.