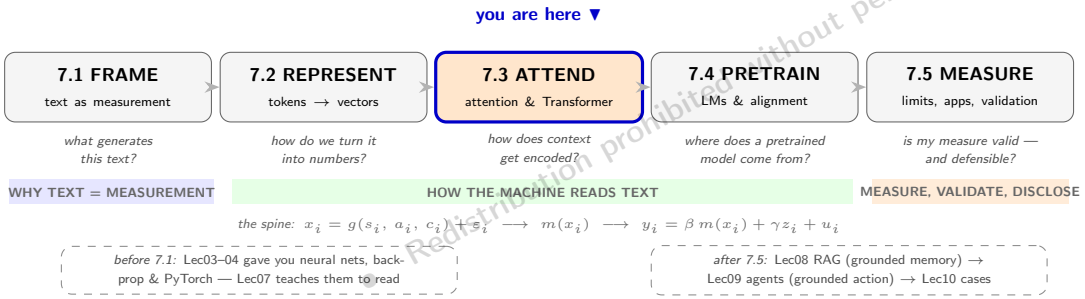
AI for Economic Research: Dynamic Models, Language, and Agents
Lecture 7.3b: The Transformer

Zhigang Feng

2026-07-09 19:41:06

Lecture 7 in One Map



Route: wrap the attention head into a full layer (part 2/2 of ATTEND) — position, residual, FFN → stack L layers → pick among three variants → read 77,000 10-Ks with the same machinery.

Completing the Layer

Position, residual safety nets, and the network that stores the knowledge

Positional Encodings: Why They Are Needed

- **Problem:** self-attention is *permutation-equivariant* in its inputs. The tokens $\{\textit{The}, \textit{Fed}, \textit{raised}, \textit{rates}\}$ give the same internal state whether we read "*The Fed raised rates*" or the scrambled "*Rates raised the Fed*" — yet the economic meaning is opposite (agent and patient are reversed). We must inject position.

- **Solution:** add a position vector $p(i)$ to each token embedding before the first layer:

$$e^{(i)} = \mathcal{E}\big(t^{(i)}\big) \; + \; p(i).$$

- Three families of position encodings dominate practice. The choice matters most for *long contexts* and *length extrapolation*.

- **Take-away:** a Transformer must be *told* where each token is. The mechanism is simple but indispensable.

Sinusoidal, Learned, and Rotary (RoPE)

- **Sinusoidal** (Vaswani et al., 2017) — fixed, no parameters:

$$\mathrm{PE}(pos, 2i) = \sin\big(pos \,/\, 10000^{2i/d}\big), \quad \mathrm{PE}(pos, 2i{+}1) = \cos\big(pos \,/\, 10000^{2i/d}\big).$$

  Multiple frequencies ⇒ relative offsets become linear functions; theoretically extrapolates beyond training length.

- **Learned** (BERT, original GPT) — positions $1, \ldots, n_{\max}$ each get a learnable vector. Simple; *cannot* extrapolate past $n_{\max}$.

- **Rotary (RoPE)** — modern default in LLaMA, GPT-NeoX, Qwen, Mistral, DeepSeek. Apply a position-dependent *rotation matrix* to Q and K:

$$\langle R_{\theta_i} Q_i,\ R_{\theta_j} K_j \rangle \ = \ f(Q_i, K_j,\ i - j).$$

  Position information is baked into attention scores and depends only on *relative* offset $i{-}j$ — excellent length extrapolation.

- **ALiBi** (a close cousin) adds a linear distance penalty directly to attention scores; very simple, strong long-context behavior.

Position Encodings: Trade-Off Table

Scheme	Params	Where applied	Extrapolates?	Modern use
Sinusoidal	0	input embedding	yes (in theory)	rare today
Learned absolute	$n_{\max} \cdot d$	input embedding	no	BERT, early GPT
RoPE	0	inside Q, K	yes (with scaling)	LLaMA, Qwen, Mistral
ALiBi	H	attention scores	yes	some long-context models

- **Take-away:** a Transformer must be *told* where each token is. Modern LLMs prefer *relative* schemes (RoPE / ALiBi) because they handle 32k+ context windows gracefully.

For long-document economics — complete 10-Ks (50k+ tokens), IMF country reports, decade-scale central-bank corpora — prefer RoPE/ALiBi-based models. A fixed learned-absolute model simply cannot read past its training length.

Residual Connection and Layer Normalization

- After the attention output $\tilde{e}^{(i)}$, the layer combines it with the original input via a *residual connection* and then a *layer normalization*:

$$u^{(i)} \;=\; \mathrm{Norm}\big(e^{(i)} \;+\; \tilde{e}^{(i)}\big).$$

- **Why residual?**
  - Preserves the original information — attention can only *add* to it, never destroy it
  - Provides a clean gradient path back through dozens of layers (the trick that made deep networks trainable)

- **Why layer normalization?**
  - Normalizes activations across the feature dimension to mean 0, variance 1
  - Adds learnable scale γ and shift β
  - Stabilizes optimization dynamics, especially for very deep stacks

Feed-Forward Network (MLP)

- **Division of labor inside a layer.** Attention mixes information *across* tokens; the FFN then transforms *each* token individually given what it just gathered. Attention = the meeting room; FFN = desk work after the meeting.
- **Why it exists.** Attention alone is a linear map of value vectors. The FFN supplies the *non-linear* transformation that lets the network represent arbitrary functions of a token's contextualized embedding.
- After attention + residual + norm, each position is passed independently through a small feed-forward network:

$$f^{(i)} = W_2\,\sigma\big(W_1\,u^{(i)} + b_1\big) + b_2.$$

- **Shape conventions:** W_1: $d \to 4d$ (expand), activation σ (ReLU / GELU / SwiGLU), W_2: $4d \to d$ (contract). The $4d$ expansion is where most of an LLM's parameters live.
- **A useful reading — key–value memory** (Geva et al. 2021): rows of W_1 act as *pattern detectors* ("keys": a collocation, a topic, an entity type); the ReLU selects which patterns fired; the matching columns of W_2 write back the associated content ("values"). Roughly $2/3$ of a layer's parameters sit here ($8d^2$ vs. attention's $4d^2$): attention *moves* information to the right

One Layer — Putting It All Together

- A single Transformer layer can be summarized by the flow:

$$e^{(i)} \xrightarrow{\text{Self-Attention}} \tilde{e}^{(i)} \xrightarrow{\text{Res+Norm}} u^{(i)} \xrightarrow{\text{MLP}} f^{(i)} \xrightarrow{\text{Res+Norm}} e'^{(i)}.$$

- Operationally:

$$\begin{aligned} \tilde{e}^{(i)} &= W_O \sum_j \alpha_j^{(i)} V_j, \\ u^{(i)} &= \mathrm{Norm}\big(e^{(i)} + \tilde{e}^{(i)}\big), \\ f^{(i)} &= W_2\, \sigma(W_1\, u^{(i)} + b_1) + b_2, \\ e'^{(i)} &= \mathrm{Norm}\big(u^{(i)} + f^{(i)}\big). \end{aligned}$$

- Same $(\mathbb{R}^d)^n$ shape in and out: a layer is a structured non-linear update — which is exactly what lets us stack it.

After the Transformer: Where We Are

- **What went in:** a sequence of n tokens, converted to embeddings + positional codes, $[e^{(1)}, \ldots, e^{(n)}]$ with each $e^{(i)} \in \mathbb{R}^d$.
- **What comes out of the top layer:** n *contextualized* vectors $[e'^{(1)}, \ldots, e'^{(n)}]$, same shape, same dimension. Each $e'^{(i)}$ encodes, in its d coordinates,
  - (a) the identity of token $t^{(i)}$,
  - (b) its syntactic / semantic role in *this* sequence,
  - (c) what every other token contributed to its meaning (causally masked in GPT-style models).
- **What we have NOT yet done:** no token generated, no class assigned, no number returned. Contextualized embeddings are an **intermediate data structure**, not the final answer.
- **What comes next:** a small *task head* on top — a vocabulary projection (generation) or a class projection (classification) — turns the embeddings into the output we want.

Stacking L Layers

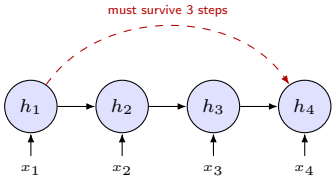
- The output of one layer becomes the input of the next:

$$\left[e^{(1)}, \ldots, e^{(n)}\right]_{\text{layer } \ell+1} \;=\; \left[e'^{(1)}, \ldots, e'^{(n)}\right]_{\text{layer } \ell}.$$

- After L layers, each token's vector has accumulated context from the entire sequence (subject to causal masking, in the autoregressive case).
- **Different layers learn different things** (empirically):
  - Lower layers: surface syntax, local agreement, morphology
  - Middle layers: phrase- and clause-level structure
  - Upper layers: discourse, factual associations, task-relevant abstractions
- **What we do with the top layer ("task heads").**
  - **Generation:** logits $= W_{\text{vocab}}\, e'^{(n)}$, then $P(\text{next} = w) = \mathrm{softmax}(\text{logits})_w$. Training loss: $\mathcal{L} = -\sum_i \log P(t^{(i+1)} \mid t^{(1:i)})$ — the same cross-entropy that shaped every weight in the stack during pretraining.
  - **Classification:** logits $= W_{\text{class}}\, e'^{([\text{CLS}])}$, then $P(\text{class} = c) = \mathrm{softmax}(\text{logits})_c$. Training loss: $\mathcal{L} = -\sum \log P(y_{\text{true}} \mid x)$ on a labeled set (fine-tuned encoders: BERT, CentralBankRoBERTa, …).

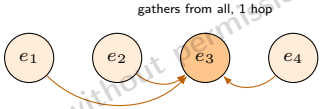
RNN vs. Transformer: One Hidden State vs. All-to-All

RNN / LSTM — recurrence



$h_t = f(h_{t-1}, x_t)$: read left to right; the *entire* past is squeezed into one fixed-size state, and must be computed *in order*.

Transformer — self-attention



$\mathrm{softmax}(QK^\top/\sqrt{d})\,V$: each token reads *every* token directly, and *all positions* are computed at once — no state passed through time.

- **The one difference everything follows from:** an RNN carries information *through time* in a single vector; a Transformer accesses *all positions at once*.
- Path from token j to token i: $|i-j|$ **update steps** in an RNN (signal decays, gradients vanish) vs. **one hop** in attention — Vaswani et al. (2017), Table 1.
- **The price:** attention costs $O(n^2)$ in sequence length vs. the RNN's $O(n)$ — the reason long-context tricks (RoPE/ALiBi) and state-space models (Mamba) exist.

Why the Transformer Displaced the RNN

Two compounding advantages — each with an economic hook.

1. Fully parallel training. An LSTM must compute step t after step $t-1$: n tokens means n serial steps. Attention over the whole sequence is two matrix multiplications, $\mathrm{softmax}\big(QK^\top/\sqrt{k}\big)V$ — computed for all positions at once on a GPU. This is the engineering fact behind "pretraining on billions of tokens became possible after 2017": BERT's 3.3×10^9-token pretraining was simply not feasible in serial mode.

2. $O(1)$ paths for long-range dependencies. In a recurrent net, position j's influence on a distant position i must survive $i-j$ intermediate states — the signal decays. Attention connects any two positions *directly*. Concretely: Fed and PBOC policy reports state economic conditions in part 1 and the policy orientation pages later — reading the guidance *against* the conditions requires exactly these long jumps.

Post-script: state-space models (e.g. Mamba, Gu & Dao 2023) revive recurrence with linear cost in sequence length — worth watching for full-filing work, at some cost in exact global attention.

From Output Embeddings to the Next Token

- For autoregressive generation we focus on the *last* output vector $e'^{(n)}$.
- Project it onto the vocabulary with a linear layer:

$$\text{logits} \;=\; W_{\text{vocab}}\, e'^{(n)} \;+\; b_{\text{vocab}}, \qquad W_{\text{vocab}} \in \mathbb{R}^{M \times d}.$$

- Convert logits to probabilities with softmax:

$$P\big(t^{(n+1)} = w \mid t^{(1)}, \ldots, t^{(n)}\big) \;=\; \frac{\exp(\text{logits}_w)}{\sum_j \exp(\text{logits}_j)}.$$

- **Sampling strategies:** greedy, top-k, top-p (nucleus), temperature scaling. (Unpacked in T4.)
- Append the chosen token to the context and repeat. **That** is how an LLM "generates text".

Real LLM Scale

Quantity	Symbol	GPT-2	GPT-3	Modern (2024–26)
Layers	L	12–48	96	60–120
Hidden dim	d	768–1600	12288	4096–16384
Context length	n	1024	2048	$32\text{k}–1\text{M}$
Vocabulary	M	50k	50k	50k–256k
Parameters	—	0.1–1.5B	175B	70B–$>$1T

- **Modern open-weight examples (2024–2026):** LLaMA-3 (8B, 70B, 405B), DeepSeek-V3 (671B MoE), Qwen-2.5, Mistral.
- **Modern closed examples:** GPT-4/5, Claude 3/4, Gemini (sizes undisclosed).
- All share the same basic architecture; differences are scale, training data, and tuning — the subject of Act 4.

Transformer vs. Word2Vec

Property	Word2Vec	Transformer
Representation	1 static vector / word	contextual vector / token
Architecture	2-layer Skip-gram / CBOW	multi-head attention + MLP, deep stack
Context scope	fixed local window (± 5)	full sequence, every token sees every token
Position info	none	explicit positional encodings
After training	lookup table; model discarded	model *is* the network used downstream
Typical size	$\sim 10^7$ params (CPU OK)	10^8–10^{12} params (GPU required)
Best for	similarity, feature inputs	generation, classification, QA, RAG

The big shift: from a frozen lookup table of meanings to a deep, dynamic model where meaning emerges from interaction with context.

Three Variants

Same block, three masking patterns — and three research uses

Three Variants of the Same Block

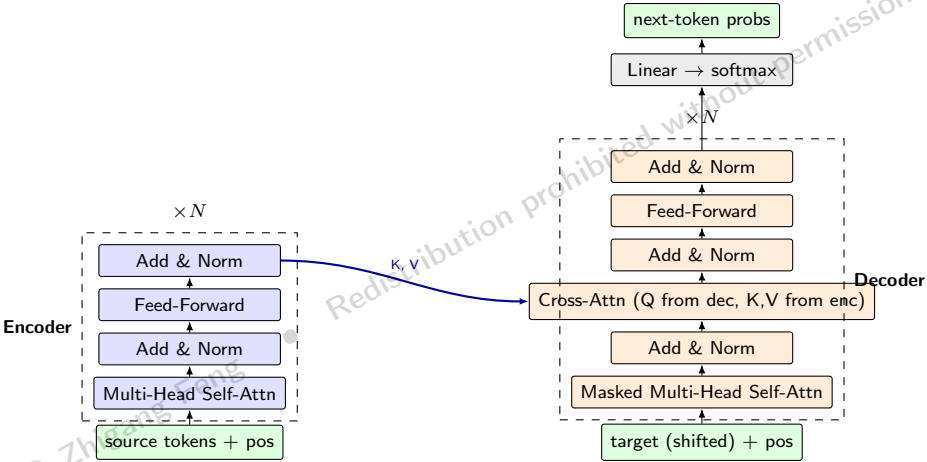
- The Transformer block is generic. The *masking pattern* and the *stack arrangement* determine what kind of model you get.
- **Encoder-only (BERT, RoBERTa, FinBERT, CentralBankRoBERTa)**
  - Bidirectional attention — every token sees every other.
  - Pretrained with *masked language modeling*: hide tokens, predict from both sides.
  - Best for: classification, embedding extraction, similarity. *Not* a generator.
- **Decoder-only (GPT, LLaMA, Claude, Qwen)**
  - Causal mask — each position only attends to itself and the past.
  - Pretrained with *next-token prediction*; autoregressive at inference (Act 4).
  - Best for: open-ended generation, in-context learning, chat. The dominant paradigm today.
- **Encoder–Decoder (T5, BART, original Vaswani 2017)**
  - Encoder reads source bidirectionally; decoder generates target with causal + cross-attention to encoder.
  - Best for: translation, summarization, **structured extraction** — e.g. turning free-form FOMC or PBOC minutes into (date, action, magnitude, guidance) database rows at a fraction of hand-coding cost.

Picking a Variant for Your Macro Project

Task	Pick	Why	Example model
Sentiment / hawk–dove score on FOMC text	encoder	need a fixed embedding	CentralBankRoBERT
Topic / event classification	encoder	supervised fine-tune	FinBERT, RoBERTa
Document embeddings for similarity / RAG	encoder	pooled $e'([\text{CLS}])$	SBERT, BGE, E5
Summarize a 10-K / press release	enc–dec	input $\neq$ output	T5, BART
Structured extraction (minutes $\rightarrow$ rows)	enc–dec	learned text-to-schema map	T5, BART
Generate a forecast narrative	decoder	open-ended text	GPT-4, Claude, LLa
Q&A over central-bank corpora (RAG)	decoder + retriever	generative w/ context	Lecture 8

- **Rule of thumb.** If the output is a label or a vector, use an encoder. If the output is text, use a decoder. If you need to map one sequence to another and they have different lengths and structure, use encoder–decoder.

“Attention Is All You Need” (Vaswani et al., 2017) — the Architecture



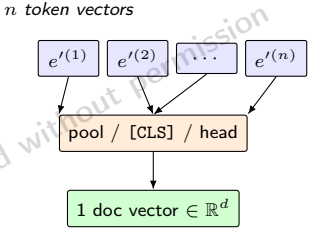
Same block, two arrangements. Decoder-only models drop the encoder and the cross-attention.

Token Embeddings vs. Document Embeddings (Bridge to Lec08)

- **What the Transformer gives us directly.** For an n-token passage, the top layer outputs n contextualized vectors $e'^{(1)}, \ldots, e'^{(n)} \in \mathbb{R}^d$ — *one per token*.

- **To compare whole passages we need one vector per passage.** Standard pooling moves:
  - **Mean-pool**: $\bar{e} = \frac{1}{n} \sum_i e'^{(i)}$ — simple, surprisingly robust.
  - `[CLS]`**-pool** (BERT family): use $e'^{(\texttt{[CLS]})}$, trained to summarize the passage.
  - **Learned pooling head**: a small attention head over the token sequence, trained for the downstream objective.

- **Retrieval embedders** (Sentence-BERT, BGE, E5, voyage-3) take a pretrained transformer and fine-tune it with a *contrastive loss* on (anchor, positive, negative) triples: pull related passages together, push unrelated ones apart. The geometry is *learned for similarity*, not inherited from the



Lec08: ranks $\cos(\textit{query}, \textit{doc})$

Case: Reading 10-Ks at Paragraph Level

Kim, Muhn, Nikolaev & Zhang (2024) — every concept of this act, reused

From Tokens to Paragraphs: A Research Question

- **Question.** Which language inside a 10-K filing actually moves stock prices? Press releases, MD&A, risk factors, governance disclosures — which content do investors *react* to?
- **Paper.** Kim, Muhn, Nikolaev, Zhang (Booth WP, Nov. 2024), *Learning Fundamentals from Text*.
- **The move.** Same Q/K/V machinery from T3a — but the unit of analysis is a **paragraph**, not a word token. A 10-K is a sequence of ~ 400 paragraphs; treat each one as a "token" and let attention learn which paragraphs matter for the market.
- **Why this beats the standard playbook:**
  - Prior text-in-finance work collapses a 10-K to one number: sentiment (Loughran–McDonald 2011), uncertainty (Hassan et al. 2019). Information loss is severe.
  - Attention preserves *paragraph-level* granularity *and* learns relational importance directly from market reactions.
  - Trained attention weights are inspectable — you can read off which content the market focuses on.

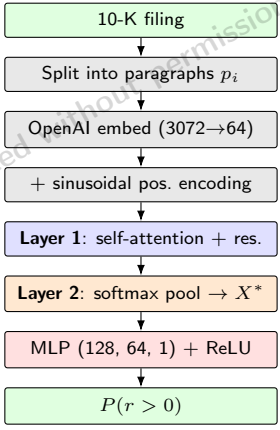
Architecture: Two Attention Layers + MLP Head

Pipeline (full universe of EDGAR 10-Ks, 1996–2023, ∼77K filings, ∼20.7M paragraphs):

1. Split filing → paragraphs $p_1, \ldots, p_n$ (cap n=1000).
2. e_i = OpenAI `text-embedding-3-large`(p_i), then truncate to first 64 dims (Matryoshka, Kusupati et al. 2022).
3. Stack $X = [e_1; \ldots; e_n] \in \mathbb{R}^{n \times 64}$, add sinusoidal positional encoding.
4. **Layer 1** — Vaswani self-attention with residual: $\mathrm{Attention}(X) \in \mathbb{R}^{n \times 64}$.
5. **Layer 2** — learned softmax pool to a doc vector:

$$X^* = \mathrm{softmax}\Big(\frac{W^\top \mathrm{Attention}(X)^\top}{\sqrt{64}}\Big)\,\mathrm{Attention}(X)$$

   with $W \in \mathbb{R}^{64}$ learnable; $X^* \in \mathbb{R}^{64}$.
6. MLP $(64 \to 128 \to 64 \to 1)$, ReLU, sigmoid $\to P(r_{[-1,+30]} > 0)$.



Target: $\mathbb{1}\{r_{[\text{file}-1,\text{file}+30]} > 0\}$. Rolling 6/2/1-yr train/val/test.

Every Concept from This Act, Reused

Slide concept (this act)	Paper usage
Q, K, V projections (T3a)	Same, but over *paragraphs* not tokens
Scaled softmax $\mathrm{softmax}(QK^\top/\sqrt{d})V$ (T3a)	Used verbatim in Layer 1
Sinusoidal positional encoding (T3b)	Same Vaswani formula, scaled by embedding norm
Residual connection (T3b)	Added to Layer 1 output (paper fn. 13)
Cosine similarity (T2b)	Importance score: $\cos(e_i,\, X^*)$
Encoder-style training	**Supervised on returns**, not next-token

- **Methodological pivot.** A standard LLM trains attention with an unsupervised objective (next-token cross-entropy). This paper trains the *same* attention machinery against a **financial target** (binary positive return). The attention weights then read directly as economic importance.

- **Importance score for paragraph i in document t:**

$$\text{Importance}_{ikt} = \frac{X^*_{it} \cdot e_{ikt}}{\|X^*_{it}\|\,\|e_{ikt}\|}$$

  A paragraph is important when its semantic content aligns with the document's attention-weighted representation.

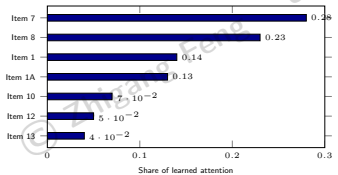
Empirical Findings

Predictive performance (avg AUC, 20 yearly test sets, baseline 0.50):

Model	AUC	Δ vs base
Attention (paper)	**0.5599**	$+6.0$ pp
MLP (no attn)	0.5328	$+3.3$ pp
Logit	0.5197	$+2.0$ pp

Sharpe (1-month-ahead returns): **1.56** attention, 1.28 MLP, 1.08 logit.

Attention share by 10-K Item (stylized; learned weights):



Top: Item 7 MD&A, Item 8 Financials. Bottom: Item 13 Relations, Item 12 Ownership, Item 10 Governance.

Topics inside MD&A (Item 7):

- *Most attended*: segment information, financial performance, liquidity, recent M&A.
- *Least attended*: forward-looking statements, accounting pronouncements, **corporate governance**, **sustainability / CSR**.

Surprise. ESG and corporate governance — dominant in public discourse — rank *lowest* by market reaction.

Two causal results.

- SEC *Modernization of Reg S-K* (treatment) raised Item 7 vs Item 8 (control) importance — regulation can *enhance* relevance.
- Firms strategically place bad-news / low-profitability content **later** in MD&A (Bloomfield 2008; Cohen et al. 2020) — the DGP's a_i, visible in document structure.

Bridge: The Architecture Is Fixed — Capability Is Not

Same block everywhere; the leverage now moves to training.

- Kim et al. shows the pattern travels: change the "token" (paragraphs of a filing, sentences of a speech, stocks in a portfolio — cf. Gabaix–Koijen–Richmond–Yogo's asset embeddings) and supervise attention on an economic target — the weights become an inspectable measure.
- But nothing so far explains where a model that already *reads* comes from. GPT-2 to GPT-4 is essentially the same architecture — what changed is pretraining scale, data, and alignment.
- **Act 4:** self-supervised pretraining, prompting / fine-tuning / LoRA, and the SFT + RLHF pipeline that turns a text-completer into an assistant — plus what alignment does to measurement.

Arc: T1 FRAME → T2 REPRESENT → **T3 ATTEND** *→ T4 PRETRAIN → T5 MEASURE.*