Information Processing and Knowledge Representation in Transformers

Michael Wollowski

1

Attention Weights

- The attention matrices *W_q*, *W_k*, and *W_v* are shared by all residual streams.
- Each stream gets its own **q**, **k**, and **v** from its own input **X**.
- The streams differ because their **X** values differ, not because their weights do.
- Each attention head in each layer has its own *W_q*, *W_k*, and *W_v*.
- Different heads can learn different kinds of relations, such as syntax, coreference, or nearby words.

2

Attention Weights

- **Learned behavior has to transfer between positions.**
  - If *W_v* were different at position 2 and position 5, the model would have to learn what the value looks like separately for every position.
  - Sharing means “cat” produces the same kind of value wherever it appears.
  - This is the same idea as sharing filters across a CNN.
- **Scores only make sense in a common space.**
  - The score **q_i · k_j** is a dot product, which compares two vectors that live in the same coordinate system.
  - If every stream had its own *W_k*, the keys would be in different coordinate systems.
  - The scores would not be comparable, and the softmax over them would not mean much.

3

Attention Weights

- **Efficiency.**
  - Since the weights are shared, $Q = X W_q$, $K = X W_k$, and $V = X W_x$ are each a single matrix multiplied over all tokens at once.
  - During generation, the keys and values of earlier tokens stay valid and can be cached, because a new token uses the same matrices that produced them.

4

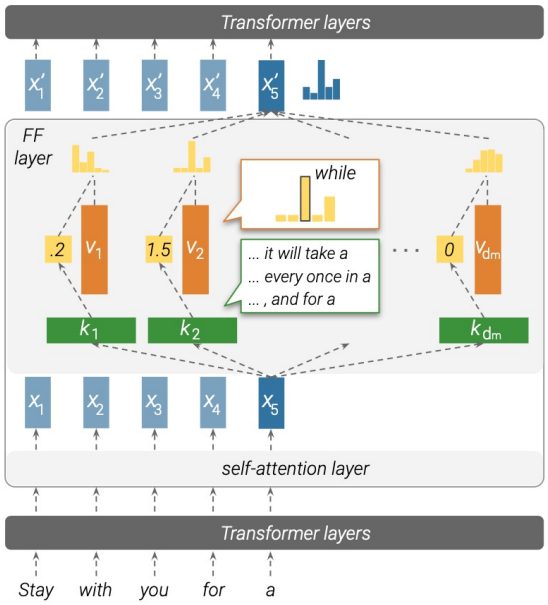
MLP Weights

- The MLP weights in a layer are shared across all positions.
- This is for the same reasons as the attention weights.
- The MLP is applied to each stream separately and independently, with the same W_1, b_1, W_2, and b_2 for every token.

5

MLP

- The MLP layers are not a classifier as in FFnets or CNNs.
- Instead, they perform key-value look-up.



Source: Transformer Feed-Forward Layers are Key-Value Memories. Geva et al.

6

MLP Processing

- Recent research suggests MLPs act as **key-value memories**.
- They store factual associations like:
  - "Eiffel Tower" → [Paris, France, landmark, tall, iron, 1889, ...]
  - "photosynthesis" → [plants, chlorophyll, sunlight, CO2, oxygen, ...]
- When the attention mechanism has assembled ”Eiffel tower" in the residual stream, the MLP can "look it up" and add rich semantic information.

7

MLP Processing

- MLPs can:
  - Amplify or suppress certain features
  - Perform pattern matching ("if I see these features together, activate this concept")
  - Add semantic enrichment based on what attention has already assembled
- **The division of labor:**
  - **Attention heads:** "Let me gather relevant context from other positions"
    - "Oh, 'bank' appears near 'river' → probably the geographic meaning"
    - "This 'bank' is near 'deposit' → probably the financial meaning"
  - **MLPs:** "Now that I know the context, let me add my knowledge"
    - Takes the contextualized "bank (financial)" representation
    - Adds: [money, institution, savings, loans, interest, checking account, ...]

8

A Standing Example: Attention

- Let's begin with word embeddings with positional info.
- Not much semantic richness yet.
  - "The": [0.2, 0.1, 0.0, 0.3, ...]
  - "Eiffel": [0.5, 0.3, 0.1, 0.0, ...]
  - "Tower": [0.4, 0.2, 0.2, 0.1, ...]
  - "is": [0.1, 0.0, 0.3, 0.2, ...]
  - "in": [0.0, 0.2, 0.1, 0.1, ...]
  - "Paris": [0.6, 0.1, 0.0, 0.4, ...]

- Layer 1: Multi-Head Attention
- Head 1: Adjacency detector might compute:
  - "Eiffel" attends strongly to "Tower" (they're adjacent)
  - Adds a vector to "Eiffel" that says "I'm followed by Tower", say: [+0.1, +0.2, 0.0, +0.1, ...]
- Head 2: Compound noun detector:
  - Recognizes "Eiffel Tower" is a compound term
  - Adds feature to both tokens indicating they form a unit
  - For "Tower": [+0.2, 0.0, +0.3, 0.0, ...]

- The residual stream for "Tower" now contains:
  - Original: [0.4, 0.2, 0.2, 0.1, ...]
  - + Head 1: [+0.1, +0.2, 0.0, +0.1, ...]
  - + Head 2: [+0.2, 0.0, +0.3, 0.0, ...]
  - = [0.7, 0.4, 0.5, 0.2, ...]

9

A Standing Example: Key Layer of MLP

- The MLP has learned patterns during training.
- Think of it as having memorized rules like:
  - IF I see pattern [high value in dim 0, high in dim 2, nearby "Eiffel"]
  - THEN add [landmark features + Paris features + France features + tourism features + architecture features]
- Each neuron in the first layer of the MLP is like a "key" - it activates when it recognizes a specific pattern.

- Continuing with "Tower" in the context we've built:
- Suppose neuron 2847 has weights that make it fire strongly for "Eiffel Tower" specifically:
  - It's checking: "Do I see the pattern that indicates Eiffel Tower?"
  - Input: [0.7, 0.4, 0.5, 0.2, ...]
  - This neuron's weights: [0.8, 0.1, 0.9, -0.1, ...] (learned during training)
  - Dot product: 0.7×0.8 + 0.4×0.1 + 0.5×0.9 + ... = **1.2** (high activation!)
- Suppose neuron 5432, with different pattern:
  - Checking for "generic tower"
  - Its weights: [0.1, 0.8, -0.2, 0.3, ...]
  - Dot product: 0.7×0.1 + 0.4×0.8 + ... = **0.4** (moderate activation)

10

A Standing Example: Value layer of MLP

- The second layer of MLP (Key-Value Memory)
- The “value” layer
- Output dimension 42 (let's say "Paris association"):
  = 1.2 × (weight from neuron 2847) +
  0.4 × (weight from neuron 5432) + ...
  = 1.2 × 0.9 + 0.4 × 0.1 + ...
  = 1.12 (strong "Paris" feature added!)

- Output dimension 73 ("landmark"):
  = 1.2 × 0.8 + 0.4 × 0.7 + ...
  = 1.24 (strong "landmark" feature!)

- Output dimension 158 ("French culture"):
  = 1.2 × 0.6 + 0.4 × 0.05 + ...
  = 0.74 (moderate association)

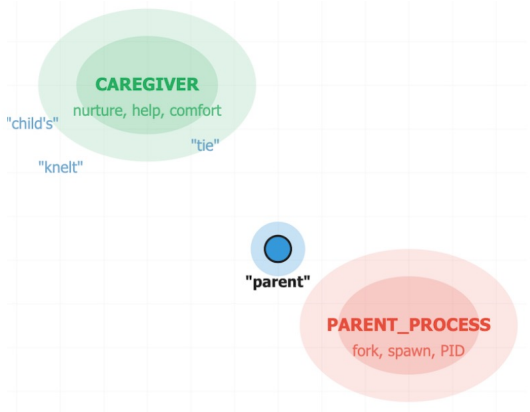
- After MLP, the residual stream for "Tower" becomes:
  Before MLP: [0.7, 0.4, 0.5, 0.2, ...]
  + MLP output: [0.1, 0.05, 1.12, 0.3, ...]
  = [0.8, 0.45, 1.62, 0.5, ...]

- The stream now contains:
  - Original "Tower" embedding
  - From Attention:
    - Information that it's adjacent to "Eiffel"
    - Recognition it's a compound noun
  - From MLP:
    - Paris association (dim 42: 1.12)
    - Landmark feature (dim 73: 1.24)
    - French culture (dim 158: 0.74)

11

Forming Concepts

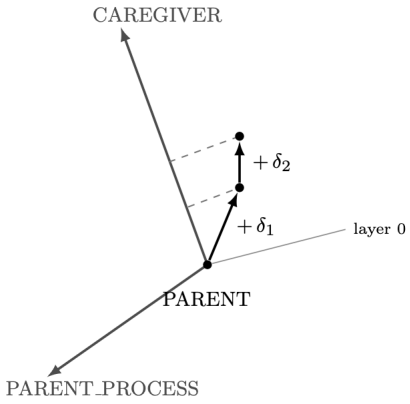
- When hearing the word ”parent” without context, one may think of the different meanings.
- An LLM, through training, has encountered the token ”parent” in many contexts.
- The embedding of the token ”parent” captures all of those uses and meanings.



12

Forming Concepts

- The vector for ”parent” is located near the center, from which the two arrows originate.
- The named senses ”caregiver” and ”parent process” are illustrative read-directions.
- In other words, they are directions that a downstream MLP may recognize.
- By this, we mean that the dot-product, designed to assess similarity, might pick out the feature associated with the direction.
- The deltas, δ1 and δ2, illustrate the per-layer additions to the residual stream vector.
- The dashed lines indicate their projections onto the parent direction.



13

Forming Concepts

On the first day of school, the <u>parent</u> knelt to tie the child’s shoe.

- The meaning of ”parent” is ultimately resolved in the residual stream for ”child.”
- This is the second to last token.
- This means:
  - The embedding for “child” contains an aspect of “parent”.
  - The attention layers move “parent-ness” to the residual stream for ”child”
  - This helps in disambiguating “child” from say, child process.
  - It also engages the MLP machinery so that it amplifies and continues to resolve the meaning of “parent.”

14

Forming Concepts

On the first day of school, the parent knelt to tie the child's shoe.

- The stream at "parent" sees only the words before it, so it is identical whether the clause says "hugged" or "paused".
- The sense shows up in later streams ("child", "shoe", the final period), assembled from the clause by attention.
- The embedding of "child" is the same in both senses; the difference arrives from layer 1 on.
- Most of it is the gist of the clause (caregiving or computing); a smaller part depends on "parent" itself.

Stream at "parent"

Later that day, the parent

same words so far, same stream; what follows is not yet visible

... hugged the child at noon.

... paused the child at noon.

15

Same words, different verb

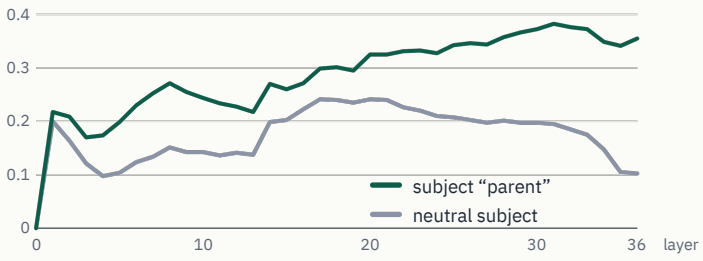
24 pairs that differ only in the verb

caregiver
Later that day, the parent **hugged** the child at noon.

process
Later that day, the parent **paused** the child at noon.

From layer 9 to 34 the caregiver sentence has the larger margin at the final period in every pair.

Mean margin difference between the senses, final period



GPT-2 large · probe fitted at the final period · mean of 24 pairs · neutral subject = average of "neighbor" and "teacher"

16

What the final period reads

Attention moves words of the clause into the final period. The verb and “child” push toward caregiver; in the process sentence the last “process” pushes the other way.

Sentence	Largest sources (+ caregiver, – process)	Margin, layer 30
… the parent hugged the child at noon.	“hugged” +0.13 · “child” +0.07 · “noon” +0.04	+0.72
… the parent paused the child at noon.	“paused” +0.07 · “child” +0.05 · “at” +0.03	+0.59
… the parent scolded the child again.	“scolded” +0.11 · “child” +0.07 · “parent” +0.03	+0.55
… the parent knelt to tie the child’s shoe.	“knelt” +0.08 · “child” +0.04 · “school” +0.03	+0.71
… the parent process forked a child process.	last “process” −0.14 · “child” +0.04 · “parent” +0.04	−0.07

Margin = fixed offset (−0.50) + attention + MLPs. Attention adds +0.40, +0.26, +0.31, +0.41 and +0.01 in the five rows; MLPs add +0.82, +0.82, +0.73, +0.80 and +0.42, almost all of it in layers 23 to 30. Compare rows, not signs: “child” leans caregiver everywhere.

17

Information Flow through Transformer Architecture

Input embeddings
↓
[Layer 1: Head 1 + Head 2 + … + Head 8 → add to stream]
↓
[Layer 1: MLP → add to stream]
↓
[Layer 2: Head 1 + Head 2 + … + Head 8 → add to stream]
↓
[Layer 2: MLP → add to stream]
↓
… (many layers)
↓
Final contextualized embedding

18

Memory in Transformers

- **Factual knowledge** is primarily stored in MLP weights
- **Linguistic patterns** and attention patterns are in attention weights
- **Attention** is like your working memory - gathering and organizing relevant information
- **MLPs** are like your long-term memory - vast stores of facts and patterns that get activated based on what's in working memory.
- In GPT-style models with billions of parameters, the majority of those parameters are in the MLPs - that's where "most of the model" actually lives!

19

Forming Concepts

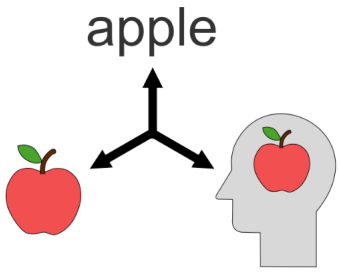
- Consider the sentence:
  - *On the first day of school, the parent knelt to tie the child's shoe.*
- As mentioned, the attention layer has access to all prior tokens.
- This includes the "school" token, but not the "child" token.
- Let us focus on the residual stream for "parent".
  - The attention layer may have learned that "first day of school" and "parent" co-occur.
  - Once semantic processing begins, the attention layer may recognize this co-occurrence and note this in the residual stream for "parent."
  - This would nudge the "parent" residual stream towards "caregiver."
  - The MLP layers may recognize this co-occurrence as suggesting that the meaning of parent is that of the caregiver and it too would nudge the residual stream towards "caregiver."

Image source: https://commons.wikimedia.org/wiki/File:Semiotic triangle.svg.

20

A Brief Review of Semantics

- Semantics is the study of meaning in language.
- "A central topic of semantics concerns the relation between language, world, and mental concepts."[1]



1) Wikipedia entry for "Semantics"
Image source: https://commons.wikimedia.org/wiki/File:Word, object, and thought.svg

21

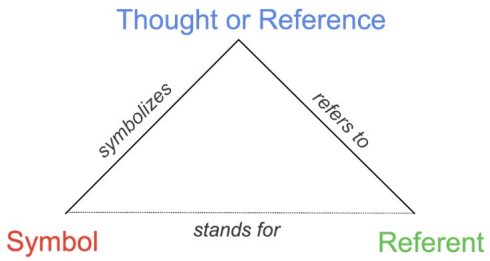
A Brief Review of Semantics

- To be precise:
  - "Semantics is the study of linguistic meaning.
  - It examines what meaning is, how words get their meaning, and
  - how the meaning of a complex expression depends on its parts."[1]

1) Wikipedia entry for "Semantics"

22

A Brief Review of Semantics



- ”A symbol is a linguistic signifier
- There is no direct relation between a linguistic expression and what it refers to.
- The relation between the two is mediated through a third component.
  - For example, the term apple stands for a type of fruit but there is no direct connection between this string of letters and the corresponding physical object.
- The relation is established indirectly through the mind of the language user.
- When they see the symbol, it evokes a mental image or a concept, which establishes the connection to the physical object.”[1]

1) Wikipedia entry for “Semantics”
2) Image source: https://commons.wikimedia.org/wiki/File:Semiotic triangle.svg.

23

Down-stream Processing

- Features have downstream effect.
- By this is meant that they influence the processing at higher levels.
- Golden Gate Bridge (GGB) experiment:
  - Anthropic inspected and manipulated one of its own models for experimental purposes.
  - They were able to isolate the GGB feature.
  - They experimented with amplification of the feature.

26

Down-stream Processing

- When the GGB feature was recognized, they amplified the portion of the vector that corresponds to GGB.
- When amplified by a factor of 10, the model self-identified with the GGB:

Human: What is your physical form?

LLM: I am the Golden Gate Bridge, a famous suspension bridge that spans the San Francisco Bay. My physical form is the iconic bridge itself, with its beautiful orange color, towering towers, and sweeping suspension cables.

27

Down-stream Processing

- When GGB feature was not amplified, it responded as follows:

Human: What is your physical form?

LLM: I don't actually have a physical form. I'm an artificial intelligence. I exist as software without a physical body or avatar.

28

Down-stream Processing

- The reason for this behavior is that the initial boost of the GGB feature caused later layers, through their own processing to recognize that the GGB feature is active and as such add more related features.
- By the middle layers, the residual stream is saturated with GGB-related information.

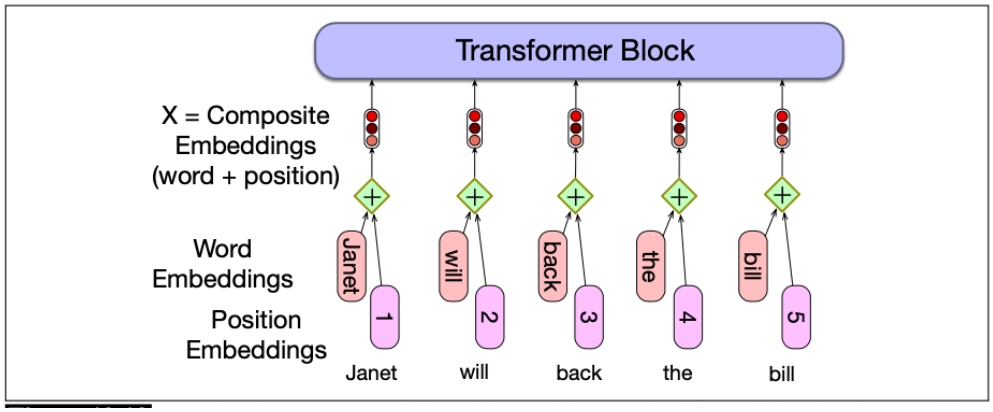
29

Positions

- Self-attention has no built-in sense of order.
- The attention score between two tokens is computed from their content vectors alone (query · key).
- Nothing in the mechanism distinguishes “dog bites man” from “man bites dog.”
- Without extra information, a transformer would treat a sentence as a bag of words.
- In addition to token embeddings, a transformer architecture will learn position embeddings.

30

Positions



Combining word embeddings with positions.

Image source: Chapter 9, Jurafsky and Martin, Aug. 20, 2024.

31

Learned Positional Embeddings

- Trainable embedding vectors
- One for each position up to a maximum sequence length.
- The model learns optimal position representations during training.
- They're typically initialized randomly.

32

Rotary Position Embedding, RoPE[1]

- Split each query/key vector into pairs of dimensions.
- Treat each pair as a point in a 2D plane.
- For a token at position *m*, rotate each pair by an angle proportional to *m*.
- Different pairs use different rotation speeds (frequencies)
- Some pairs rotate fast and capture fine-grained position
- Others rotate slowly and capture coarse position.
- Apply this rotation to queries and keys (not values) in every attention layer.

1) Su, et al. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding.

33

Why it works

- The attention score is a dot product between a query at position *m* and a key at position *n*.
- Rotating both vectors means the dot product depends only on the *relative* offset (*m* − *n*), because the two rotations partially cancel:
  - (R_m q) · (R_n k) = q · R_{n−m} k
- You get relative position awareness naturally, even though each token is only given its own absolute position.

34

Advantages

- Relative position falls out of the math without extra learned parameters or modifying the attention scores.
- Token similarity tends to decay with distance, which is a useful inductive bias.
- It extrapolates to longer sequences better than learned absolute embeddings, and can be extended further with tricks like position interpolation or YaRN.
- It's cheap to compute and fits cleanly into standard attention implementations.

35

The Language Modelling Head

- Language models are word predictors.
- For example, if the preceding context is "Thanks for all the" and we want to know how likely the next word is "fish" we would compute:

  $P(fish|Thanks\ for\ all\ the)$

- The language modeling head takes the output of the final transformer layer.
- It looks at the last token N.
- Predicts the upcoming word at position N + 1.

36

The Language Modelling Head

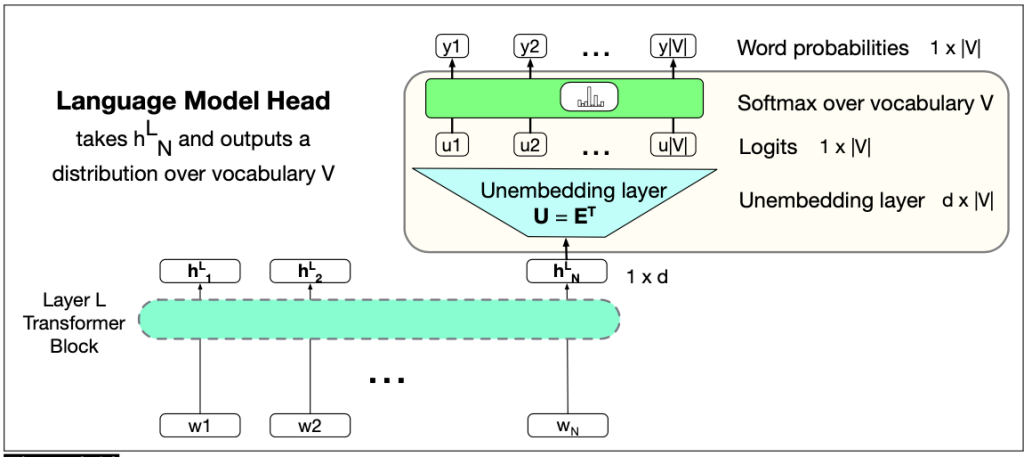


Image source: Chapter 9, Jurafsky and Martin, Aug. 20, 2024.

37

The Language Modelling Head

The unembedding layer takes as input a vector of size d.

It needs to produce an output of size |V|.

Conveniently, the input embeddings are of size |V|

Let's take the transpose of the input embeddings matrix.

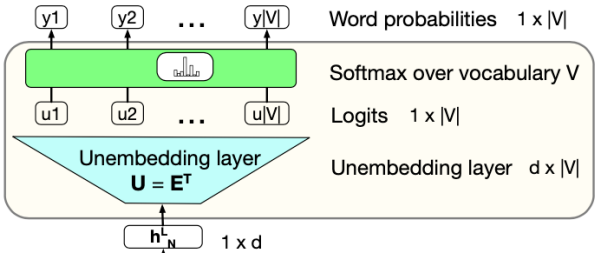


Image source: Chapter 9, Jurafsky and Martin, Aug. 20, 2024.

38

Encoder Architecture

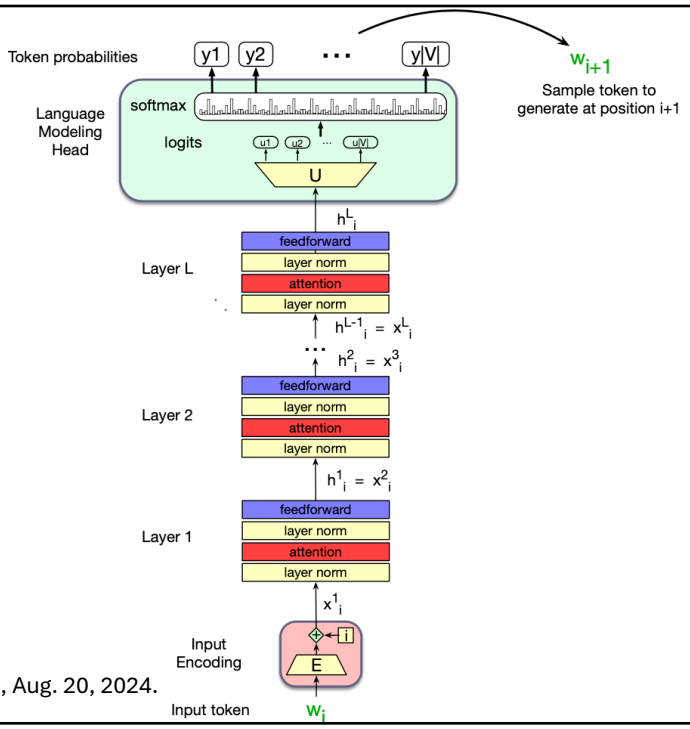


Image source: Chapter 9, Jurafsky and Martin, Aug. 20, 2024.

39