TRANSFORMERS

Embeddings

1. Tokenization — Vocabulary - No of Tokens - 50257
2. Token Embedding — 50,257 x 768 = 39 Million Parameters
3. Positional Encoding — 768 - Usually same as Token Vector Dim
4. Final Embedding — Emb Vec + Enc Vec = Final Vec (Element wise Vector Addition)

Transformers Block

Multi-Head Self Attention	Multi-Layer Perception (MLP)

Output Probabilities

Final Linear Layer. This layer projects the models final representation into a 50,257 dimensional space.

LOGITS ⟶ SOFTMAX ⟶ PROBABILITIES

Temperature   Top-K   Top-P

To Fine - Tune the models output to be more deterministic or more diverse.

Embedding

Step 1: Tokenization

Breaking down words into unique tokens from a set of 50,257 unique tokens with distinct ID's. All done with the models fixed vocabulary.

Step 2: Token Embedding

Each token is represented as a 768-dimensional vector. These vectors are stored in lookup matrix of shape (50,257 x 768). Which makes up 39 million parameters, helps in capturing the semantic meaning.

Step 3: Positional Encoding

Adds info about the position of each token in sequence. Each position gets a unique 768-dimensional vector (Emb V / Enc V / No of hidden layer are in same count to maintain consistency).

Sequential relationship and Context.

Step 4: Final Embedding

Embedding Vector + Encoding Vector = Final Embedding

Both vectors are added element-wise.

Final vector contains both Meaning and Position of each token.

Transformer Block

Transformers contain 2 main components:

1. Multiple-Head Self-Attention. ⟶ Multiple Prespectives.
2. Multi-Layer Perception (MLP)

These blocks are stacked sequentially to allow the model build more complex representations of input over time.

What is Attention: At high level attention is way to look at other tokens in same sequence & decide how much focus it should give to each one.

Analogy: GROUP CHAT

Based on what I'm saying (Query) & what others are offering (Key). Once I decide who matters, I read their message (Values) & build a response based on that.

Similarity Cal ⟶ Scaling ⟶ Softmax ⟶ Weighted Sum of Values

- Similarity Cal ↓ dot product (QK^T)
- Scaling ↓ / by √dk
- Softmax ↓ Probability Distribution
- Weighted Sum of Values ↓ V x attention weights, Contextual output

Multi-Head Self Attention:

Query, Key and Value Matrices (Q, K, V)

Multi-Head Splitting

Masked Self-Attention

Output and Concatenation

Multiple heads focus on different aspects (syntax, semantics, position) and then they are concatenated and linearly transformed before going deeper into model.

Transformer Block

Multi - Layer Perception (MLP): Token vec —> MLP —> Refined Token Vec

Analogy:

So each token first says ⟶ "Let me see what others are saying (Attention)"

Then ⟶ "Let me process what I just learned (MLP)"

Working:
- Applies linear transformation
- Passes it through a non-linear function (GELU or ReLU)
- Applies another linear transformation (To bring it back to original size)

Importance:
- Add non-linearity (So model can learn complex patterns).
- Refine token meaning based on the context already gathered.
- Boost model capacity - this is where a lot of "learning power" is.

Output Probabilities

Final Linear Layer, Vector of Logits.

Logits -> Softmax -> Probabilities -> Sampling -> Next Token

Logits : Raw scores
Not yet probabilities
Positive or negative

Softmax : Converts Logits to Probabilities between 0 and 1

Temperature: Scaling factor applied to Logits before softmax

= 1 : No change
< 1 : More confident
> 1 : More random & creative

Advance Sampling Strategies:

Top - k

Limits the choice of TOP-K tokens with highest probabilities.
Helps focus on likely options.

Top - P

Chooses the smallest group of tokens whose combined probabilities exceed threshold p
Allows diversity without unlikely outliers