1 Attention

Motivation: One of the core mechanisms inside of current LLMs.

Warning: Convolution takes in a single node, while attention takes in all nodes.

1.1 Transformer Architecture

Notes:
- **Transformer Layer:**
  - Attention mechanism (multi-headed)
  - Positional encodings
- With massive unsupervised datasets:
  - Masked self-supervised training
  - Contrastive training

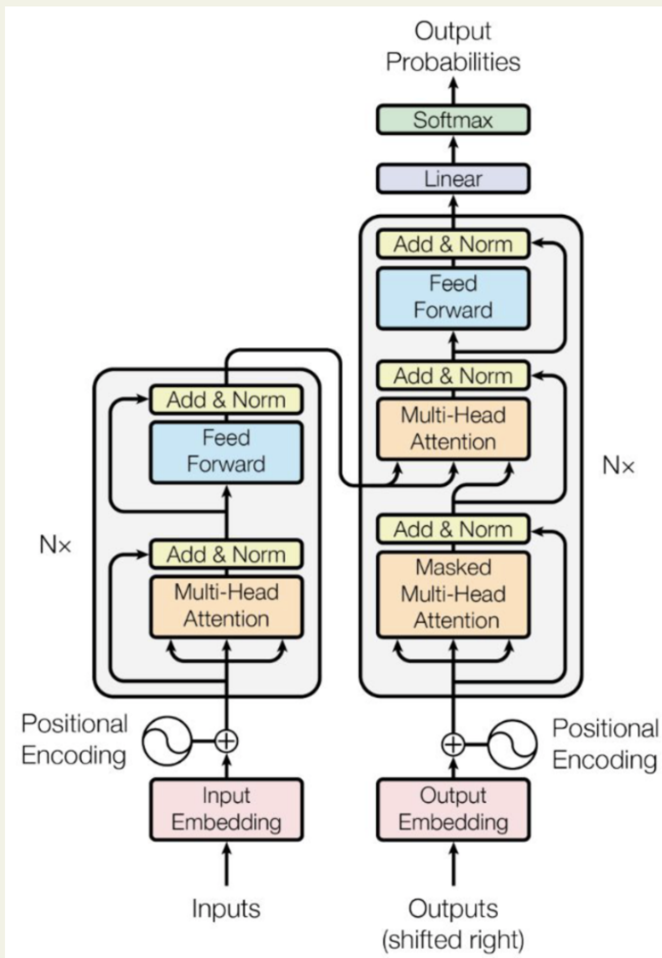

Figure 1: LS: Encoder, RS: Decoder

1.1.1 Transformer Layer

Summary:

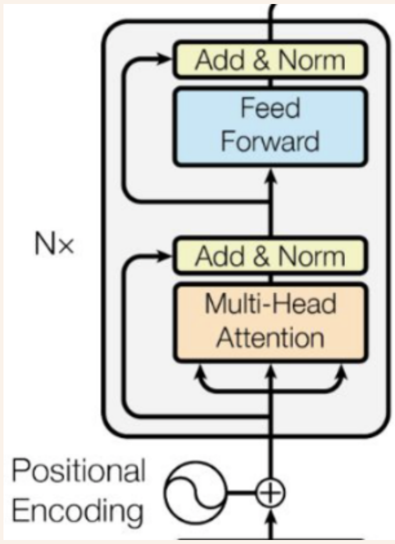

Figure 2

Component	Description
Positional encoding	Learn to map integer positions into a vectorized representation. $\mathrm{PE}_{(\mathrm{pos},i)} = \begin{cases} \sin\left(\frac{\mathrm{pos}}{10000^{\frac{i}{d_{model}}}}\right) & \text{if } i \text{ is even} \\ \cos\left(\frac{\mathrm{pos}}{10000^{\frac{i-1}{d_{model}}}}\right) & \text{if } i \text{ is odd} \end{cases}$
Multi-Head Attention	Computes attention scores for each token in the sequence.
LayerNorm	Stabilizes activations and accelerates training.
Residual Connection	Preserve information and enable deeper networks.
FFN/MLP	Increases the expressive power of the learned representation, often using GELU activations.

1.2 Transformers are GNNs

Summary: Transformers are a special case of GNN

	GNN	Transformer
Connectivity (Adjacency)	Sparse	Full
Edge Learning	Yes	No (Implicitly)
Message Computation	$M(n_i, n_j, e_{ij})$	$\langle n_i, n_j \rangle$
Communication per step	# Number of Neighboring nodes	$\sim$ # Number of Heads
Data requirements	Low	High
Computation	Slow due to gather operations	Fast, Optimizable $\sim$ Matrix Multiplications
Training	Straightforward	Pre-training is needed

1.3 Attention Mechanism

Process:

1. **Inputs:** Tokens tensor, Mask
   - **Tokens:** Inputs for Transformer/Attention Layers, which is a numerical representation of pieces of data.
   - **Mask:** A binary matrix that indicates which tokens to give attention to.
2. **Preprocessing:** Linear maps Tokens into Queries, Keys, and Values.
   - $Q = \text{Tokens} \cdot W_Q$: Represents the current token's context.
   - $K = \text{Tokens} \cdot W_K$: Represents the context of all tokens.
   - $V = \text{Tokens} \cdot W_V$: Represents the information to be passed on.
3. **Attention scores:** $\text{Scores} = \frac{QK^T}{\sqrt{d_k}} \cdot \text{Mask}$
   - $\text{score}_{ij} = \frac{(q_i \cdot k_j)m_{ij}}{\sqrt{d_k}}$
4. **Attention Normalization:** $\text{Attention Weights} = \text{softmax}(\text{scores})$
   - $\text{score}_{ij}^{\text{normalized}} = \frac{\exp(\text{score}_{ij})}{\sum_{k=1}^{n} \exp(\text{score}_{ik})}$
5. **Value update:** $\text{New Values} = \text{Attention Weights} \cdot V$
   - $v_i^{\text{new}} = \sum_{j=1}^{n} \text{score}_{ij}^{\text{normalized}} v_j$
6. **Post Processing:** Apply LayerNorm, Residual connections, and a FFN.
7. **Outputs:** Updated tokens tensor

$$\text{Attention}(Q, K, V, M) = \text{softmax}\left(\frac{QK^T M}{\sqrt{d_k}}\right) V \tag{1}$$

1.3.1 Attention Maps: Visualizing Where the Model Attends

Notes: Softmax bias:
- Values between 0 and 1
- Categorical like
- Attend to one token at a time

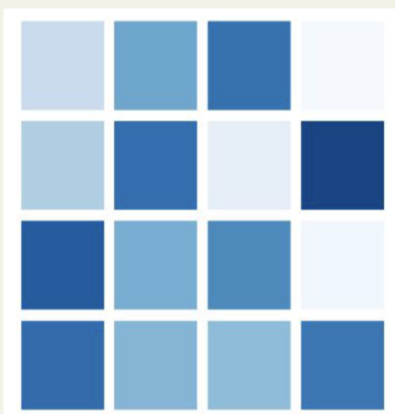

Figure 3

1.3.2 Self-Attention vs. Cross-Attention

Notes:
- **Self-Attention:** Attention allows connections between the same sequence without masking.

$$\begin{aligned}\text{Self-Attention}(x, \text{mask}) &= \text{Attention}(\text{Linear}(x), \text{Linear}(x), \text{Linear}(x), \text{mask}) \\ &= \text{Attention}(Q(x), K(x), V(x), \text{mask})\end{aligned}$$

- **Cross-Attention:** Attention allows connections between different sequences.

$$\begin{aligned}\text{Cross-Attention}(x, x', \text{mask}) &= \text{Attention}(\text{Linear}(x), \text{Linear}(x'), \text{Linear}(x'), \text{mask}) \\ &= \text{Attention}(Q(x), K(y), V(x), \text{mask})\end{aligned}$$

1.3.3 Multi-Headed Attention

Notes: Multiple attention mechanisms in parallel, each with different linear maps.
- Ensemble-like approach.
- Same compute and # of parameters.
- **Strategy:** Expand and contract tensors to new axis: number of heads.

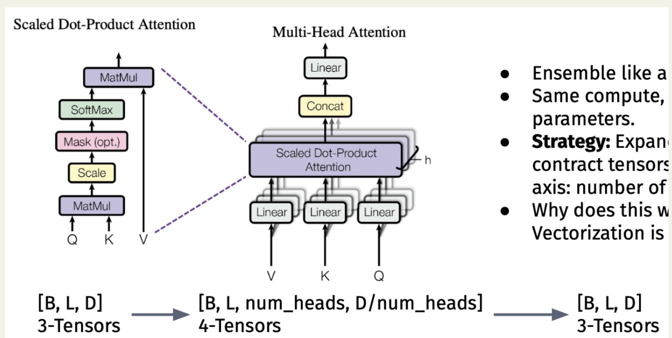

Figure 4

1.4 Examples

1.4.1 Tokens

Example:

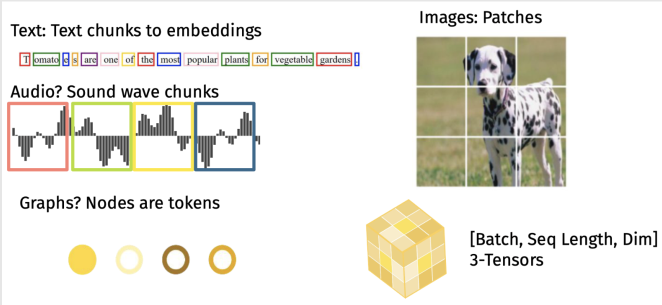

Figure 5

1.4.2 Positional Encoding

Example:

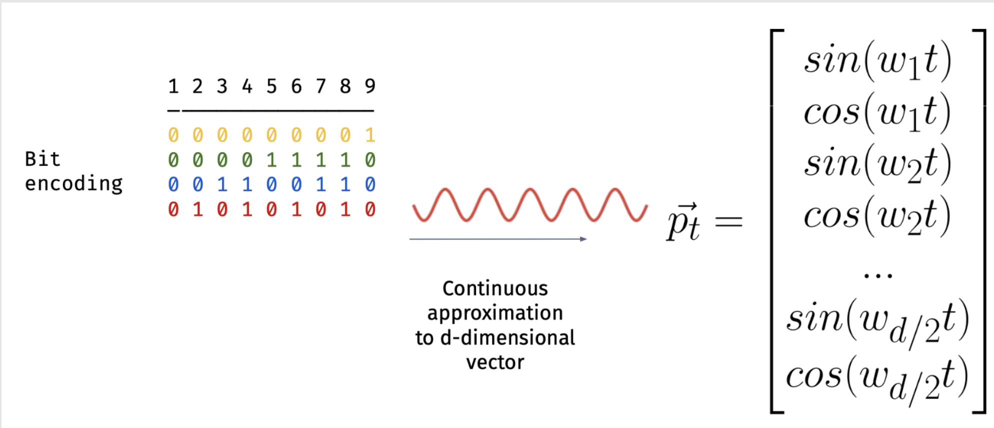

Figure 6

1.4.3 Cross-Attention

Example:

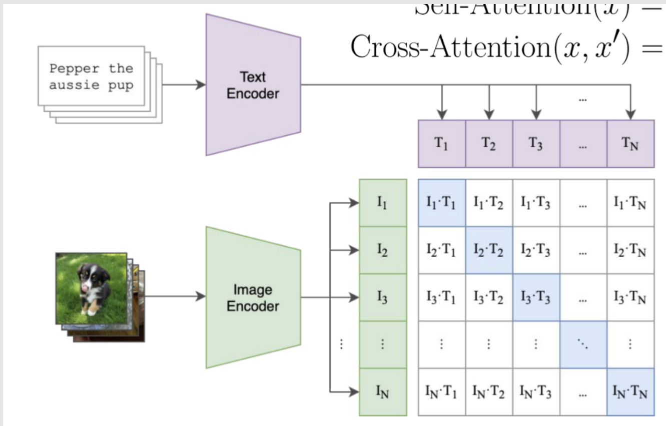

Figure 7

2 LLMs

Notes:
- Transformers on large text-like datasets.
- Transformers on "tokens" (discretized data)
- Foundational models

2.1 Inputs and Outputs of LLMs

Notes:
- **Inputs:** Tokenizing text and embedding layers

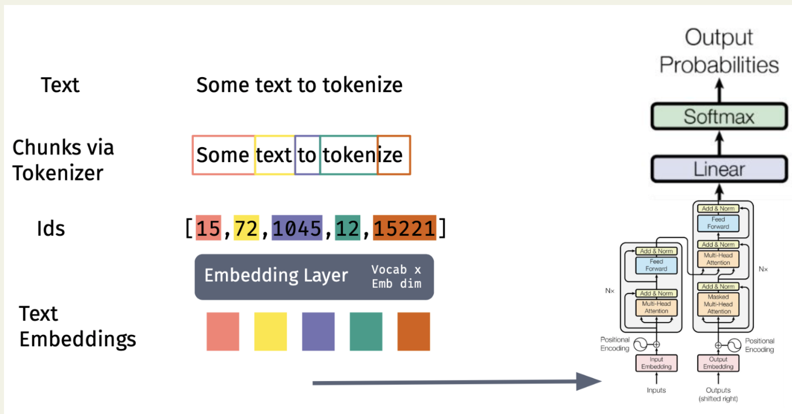

Figure 8

  - **Text to Tokens:** The input text is split into subword chunks using a tokenizer.
  - **Token to IDs:** Each token is mapped to a unique integer ID based on a fixed vocabulary.
  - **Embedding Lookup:** Token IDs are passed through an embedding layer, producing vector representations of the tokens.
  - **Text Embeddings:** The output of the embedding layer is a sequence of learned vectors (color-coded), one for each token, which serve as input to the Transformer model.
- **Outputs:** Auto-regressive decoding of tokens (one at a time) using previous outputs

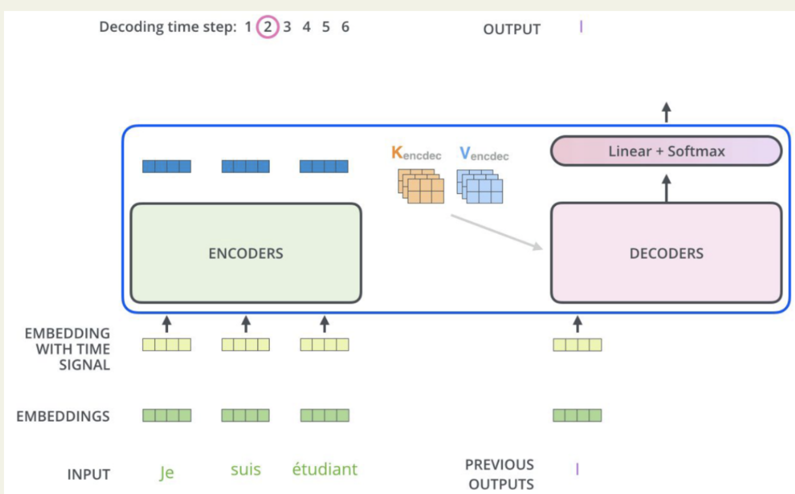

Figure 9

  - **Auto-Regressive Decoding:** Predicts each output token sequentially. At time step t, the decoder generates the t^{th} token using previously generated outputs from steps 1 to $t-1$.

2.2 Transformers & LLMs

Summary:

Concept	Description
Sizes of text datasets	Scale significantly impacts LLM's performance.
Text to Text Tasks	Many tasks can be framed as text-to-text problems.
Transformers & Masking: Encoders & Decoders	Transfomers fit the encoder decoder paradigm.

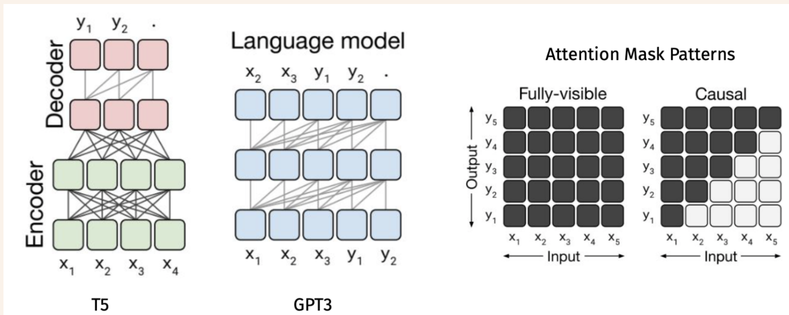

Figure 10

- **T5 (Encoder-Decoder):**
  - **Encoder:** Input tokens x_1, x_2, x_3, x_4 are processed by the encoder.
    - **Fully-visible masking** is used from the encoder to the decoder (dark grey lines).
  - **Decoder:** Decoder attends to all encoder outputs and previously generated outputs y_1, y_2.
    - **Causal masking** to prevent attending to future outputs.
- **GPT-3 (Decoder-only Language Model):**
  - Inputs consist of all prior tokens $(x_1, x_2, x_3, y_1, y_2, \ldots)$.
  - **Causal masking** throughout to ensure autoregressive behavior—i.e., each token only attends to previous tokens.
- **Attention Mask Patterns:** Dark squares denote visible connections; light squares denote masked (inaccessible) tokens.
  - **Fully-visible mask:** Every token can attend to every other token
  - **Causal mask:** Tokens can only attend to previous or current tokens

Concept	Description
Masked Language Modelling	Predicting masked tokens in a sequence.

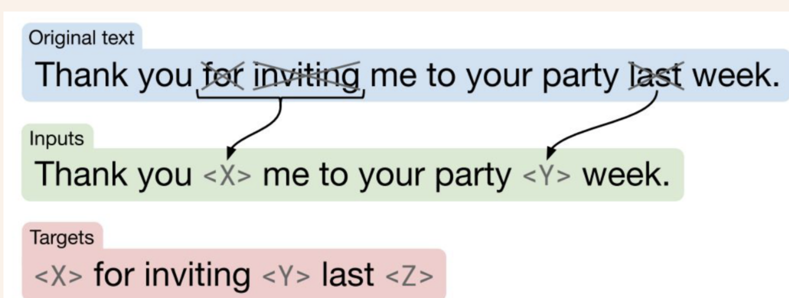

Figure 11

- **Original Text:** Selected spans (e.g., "for inviting", "last") are masked and replaced with sentinel tokens.
- **Inputs:** The corrupted version of the sentence is given as input to the model:
  - Tokens `<X>` and `<Y>` indicate the location and order of the masked spans.
- **Targets:** The decoder is trained to predict the concatenated masked spans, each preceded by its corresponding token:

2.3 Scaling LLMs

Motivation:

2.3.1 Techniques

Summary: Table format

2.3.2 High-Level Impacts

Summary: