Basic transformer architecture

AP & MF

Preliminaries

Tokens, vocabulary, language models

- $\mathcal{V} = \{t_1, \ldots, t_{n_\mathcal{V}}\}$ is the *vocabulary*
  - finite set of *tokens* of cardinality $n_\mathcal{V}$
  - vocabulary is ordered by the bijective *indexing function* $\mathrm{Ind}\colon \mathcal{V} \to \{1, \ldots, n_\mathcal{V}\}$
    - $\mathrm{Ind}(t)$ is the *(token) index* of t
- A *token sequence* of length $n \in \mathcal{V}$ is a vector $\mathbf{t} \in \mathcal{V}^n$ of n tokens from the vocabulary.
- The set of all finite token sequences forms the *language* $\mathfrak{L} = \{\mathbf{t} \in \mathcal{V}^n \mid n \in \mathbb{N}\}$.
- A *(generative / decoder) language model* is a function $\mathcal{M}_\theta\colon X \to \Delta(\mathfrak{L})$, parameterized on a set of model parameter values $\theta \in \Theta$, which maps some input set X onto a probability distribution over all (finite) sequences of tokens.

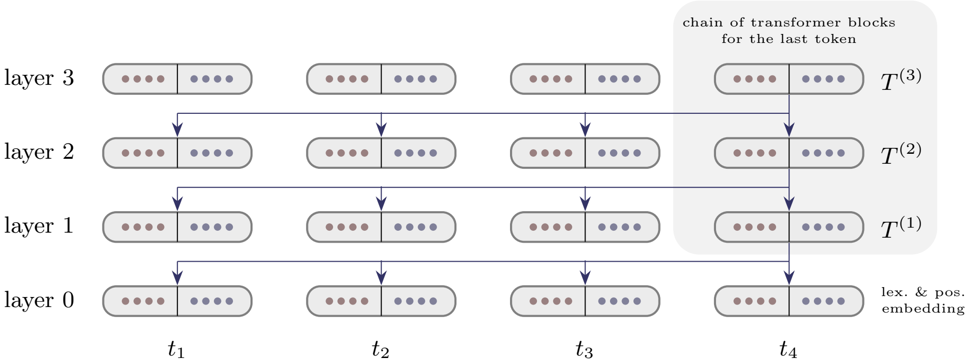
Autoregressive vs. masked LMs

- An *autoregressive (next-token prediction) language model* is a function $\mathcal{M}_\theta : \mathfrak{L} \to \Delta(\mathcal{V})$ which maps a token sequence onto a *next-token probability distribution*.
  - We write $P_\mathcal{M}(t \mid \mathbf{t}; \theta)$ or, omitting the model's parameters, $P_\mathcal{M}(t \mid \mathbf{t})$ for the model's prediction of the next-token probability for token t.
- A *masked (missing-token prediction) language model* is a function that predicts selected tokens inside of a given sequence of tokens.
  - The next-token prediction LMs can be considered a special case of the missing-token language models.

Transformer architecture

General transformer architecture

- The transformer model processes an input token sequence in parallel across all positions using a chain —stack— of $n_{\mathcal{L}}$ *transformer blocks*.
  - We refer to the ℓ^{th} element in the chain as the ℓ^{th} *layer*.



Embeddings, model size, transformer block

- An embedding $\mathbf{x} \in \mathbb{R}^{d_{\mathcal{M}}}$ is an $d_{\mathcal{M}}$-dimensional vector.
  - $d_{\mathcal{M}}$ the *dimensionality of the embedding space*, aka. the *model size*
- The transformer block $T^{(\ell)}$ for the ℓ^{th} layer is a function $T^{(\ell)} \colon \{1, \ldots, n_{\mathcal{C}}\} \times \left(\mathbb{R}^{d_{\mathcal{M}}}\right)^{n_{\mathcal{C}}} \to \mathbb{R}^{d_{\mathcal{M}}}$ which maps a the i^{th} input position of context C ($1 \leq i \leq n_{\mathcal{C}}$) and a sequence of context embeddings to a single embedding as output, such that:

$$T^{(\ell)} \colon \langle \underbrace{i}_{\text{index of token}}, \underbrace{\langle \mathbf{x}_1^{(\ell-1)}, \ldots, \mathbf{x}_{n_{\mathcal{C}}}^{(\ell-1)} \rangle}_{\text{embeddings at previous layer}} \rangle \mapsto \underbrace{\mathbf{x}_i^{(\ell)}}_{\text{embedding of } i \text{ at current layer}}$$

  - For $\ell = 1$, input zero-layer token embeddings defined next.

Zero-layer, initial token embeddings

- The zero-layer, initial token embedding, $\mathbf{x}_i^{(0)} \in \mathbb{R}^{d_\mathcal{M}}$, of input token i is computed as:

$$\mathbf{x}_i^{(0)} = F\left(\mathbf{W}_{E_{\mathrm{Ind}(i)}}, \mathrm{PosEmbed(i)}\right) \text{ , where}$$

  - $\mathbf{W}_E \in \mathbb{R}^{n_\mathcal{V}, d_\mathcal{M}}$ is the *embedding matrix*
    - the values of $\mathbf{W}_E$ are part of the model's trainable parameters
  - $\mathrm{PosEmbed(i)} \in \mathbb{R}^{d_\mathcal{M}}$ is a *positional embedding function*
  - $F : \mathbb{R}^{d_\mathcal{M}} \times \mathbb{R}^{d_\mathcal{M}} \to \mathbb{R}^{d_\mathcal{M}}$ implements some way of combining the information from the (non-positional, fixed) embedding matrix with the positional information
    - F can be as simple as mere addition

Unembedding and token predictions

- Model predictions for position i are derived by applying the model's *unembedding matrix* $U \in \mathbb{R}^{n_{\mathcal{V}}, d_{\mathcal{M}}}$ to $\mathbf{x}_i^{(n_{\mathcal{L}})}$ (final layer embedding at position i) to obtain a vector of *output logits* $\text{logits}_i(\mathbf{t}) \in \mathbb{R}^{n_{\mathcal{V}}}$:

$$\text{logits}_i(\mathbf{t}) = U\,\mathbf{x}_i^{(n_{\mathcal{L}})}$$

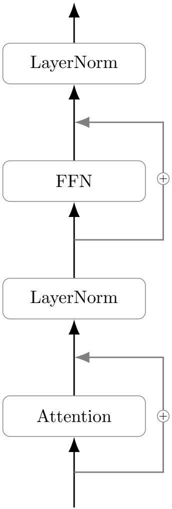
- Logits are transformed into probabilities using softmax:

$$P_{\mathcal{M}}(t \mid \mathbf{t}, i) = \frac{\exp\left(\text{logits}_i(\mathbf{t})_{\text{Ind}(t)}\right)}{\sum_{j=1}^{n_{\mathcal{V}}} \exp\left(\text{logits}_i(\mathbf{t})_j\right)}$$

  - (autoregressive) next-token prediction models predict the *next* token at $i+1$
  - (masked) missing-token prediction models (may) predict the the *current* token at i

Anatomy of a transformer block

Anatomy of a transformer block



Anatomy of a transformer block

- a transformer for layer ℓ at position i is a function $T^{(\ell)}\colon \langle i, \langle \mathbf{x}_1^{(\ell-1)}, \ldots, \mathbf{x}_{n_C}^{(\ell-1)} \rangle\rangle \mapsto \mathbf{x}_i^{(\ell)}$ that maps a token index and a sequence of embeddings to a new embedding.

$$
\begin{aligned}
\mathbf{x}_i^{(\ell-1,*)} &= \mathrm{LayerNorm}^{(\ell,1)}\left(\mathbf{x}_i^{(\ell-1)} + \mathrm{Attention}_i^{(\ell)}\left(\mathbf{x}_1^{(\ell-1)}, \ldots, \mathbf{x}_{n_C}^{(\ell-1)}\right)\right) \\
\mathbf{x}_i^{(\ell)} &= \mathrm{LayerNorm}^{(\ell,2)}\left(\mathbf{x}_i^{(\ell-1,*)} + \mathrm{FFN}^{(\ell)}\left(\mathbf{x}_i^{(\ell-1,*)}\right)\right)
\end{aligned}
$$

  - adding a *residual connection* to an operation or function F applied to input x, is to compute the (parameter-free) function $G(x)\colon x \mapsto x + F(x)$

Layer normalization

- The *layer-normalization function* $\mathrm{LayerNorm}\colon \mathbb{R}^{d_{\mathcal{M}}} \to \mathbb{R}^{d_{\mathcal{M}}}$ is included for training efficiency and defined as:

$$\mathrm{LayerNorm}(\mathbf{x})^{(\ell,k)} = \gamma^{(\ell,k)} \odot \frac{\mathbf{x} - \mu(\mathbf{x})}{\sigma(\mathbf{x})} + \beta^{(\ell,k)}\,,$$

  where:
  - $\mu(\mathbf{x}) = \frac{1}{d}\sum_{i=1}^{d_{\mathcal{M}}} x_i$ is the mean of the values in the embedding,
  - $\sigma(\mathbf{x}) = \sqrt{\frac{1}{d}\sum_{i=1}^{d_{\mathcal{M}}}(x_i - \mu(\mathbf{x}))^2 + \epsilon}$ is the standard deviation with a small constant $\epsilon > 0$ for numerical stability,
  - $\gamma^{(\ell,k)}, \beta^{(\ell,k)} \in \mathbb{R}^{d_{\mathcal{M}}}$ are learned parameters (scale and shift), one for each layer ℓ and occurrence k of the layer-normalization operation in the transformer block, and
  - $\odot$ denotes elementwise multiplication.

Feed-forward network

- The *feed-forward network* $\text{FFN}^{(\ell)} : \mathbb{R}^{d_{\mathcal{M}}} \to \mathbb{R}^{d_{\mathcal{M}}}$ usually has one hidden layer of size d_{FFN} ($\approx 2d_{\mathcal{M}}$):

$$\text{FFN}^{(\ell)}\left(\mathbf{x}_i^{(\ell,*)}\right) = \mathbf{W}_{out}^{(\ell)}\left(\phi\left(\mathbf{W}_{in}^{(\ell)}\,\mathbf{x}_i^{(\ell,*)}\right) + \beta^{(in,\ell)}\right) + \beta^{(out,\ell)}\,,$$

  where
  - $\mathbf{W}_{in}^{(\ell)} \in \mathbb{R}^{d_{\text{FFN}} \times d_{\mathcal{M}}}$ is the mapping from the input layer to the hidden layer
  - $\mathbf{W}_{out}^{(\ell)} \in \mathbb{R}^{d_{\mathcal{M}} \times d_{\text{FFN}}}$ is the mapping from hidden layer to output layer,
  - $\beta^{(in,\ell)}$ and $\beta^{(out,\ell)}$ are bias vectors, and
  - ϕ is some (non-linear) activation function.

Attention module

- The *attention module* maps a position index and a sequence of embeddings to another embeddingC

$$\text{Attention}^{(\ell)} : \{1, \dots, n_{\mathcal{C}}\} \times \left(\mathbb{R}^{d_{\mathcal{M}}}\right)^{n_{\mathcal{C}}} \to \mathbb{R}^{d_{\mathcal{M}}}$$

  such that:

$$\text{Attention}^{(\ell)}\left(i, \mathbf{x}_1^{(\ell-1)}, \dots, \mathbf{x}_{n_{\mathcal{C}}}^{(\ell-1)}\right) = \mathbf{W}_O \mathbf{z}\,, \text{with}$$

$$\mathbf{z} = \underbrace{\text{AH}^{\ell,1}\left(i, \mathbf{x}_1^{(\ell-1)}, \dots, \mathbf{x}_{n_{\mathcal{C}}}^{(\ell-1)}\right) \| \cdots \| \text{AH}^{\ell,n_{\mathcal{H}}}\left(i, \mathbf{x}_1^{(\ell-1)}, \dots, \mathbf{x}_{n_{\mathcal{C}}}^{(\ell-1)}\right)}_{\text{concatenation of outputs of } n_{\mathcal{H}} \text{ attention heads}}$$

  where $\mathbf{W}_O \in \mathbb{R}^{d_{\mathcal{M}} \times n_{\mathcal{H}}\ d_h}$ is an output mapping, where d_h is the dimension of the output of a single attention head.

Attention head

- The *attention head* h for is a function $\mathrm{AH}^{\ell,h}\colon \{1,\dots,n_{\mathcal{C}}\} \times \left(\mathbb{R}^{d_{\mathcal{M}}}\right)^{n_{\mathcal{C}}} \to \mathbb{R}^{d_h}$:

$$\mathrm{AH}^{\ell,h}\left(i, \mathbf{x}_1^{(\ell-1)}, \dots, \mathbf{x}_{n_{\mathcal{C}}}^{(\ell-1)}\right) = \sum_{j=1}^{n_{\mathcal{C}}} \mathrm{Weight}^{(\ell,h)}\left(\mathbf{x}_i^{(l-1)}, \mathbf{x}_j^{(l-1)}\right) \mathrm{Value}^{(\ell,h)}\left(\mathbf{x}_j^{(l-1)}\right)$$

- The *value function* $\mathrm{Value}\colon \mathbb{R}^{d_{\mathcal{M}}} \to \mathbb{R}^{d_V}$ maps an embedding of size $d_{\mathcal{M}}$ to a value vector of size d_V:

$$\mathrm{Value}^{(\ell,h)}(\mathbf{x}) = \mathbf{W}_V^{(\ell,h)}\mathbf{x}$$

- The *attention weight function* is defined as a softmax over numerical scores (where we assume that $\exp(-\infty) = 0$ and d_k is the size of key / query vectors):

$$\mathrm{Weight}^{(\ell,h)}\left(\mathbf{x}_i, \mathbf{x}_j\right) = \frac{\exp\left(d_k^{-1/2}\mathrm{Score}^{(\ell,h)}\left(\mathbf{x}_i, \mathbf{x}_j\right)\right)}{\sum_{j'=1}^{C} \exp\left(d_k^{-1/2}\mathrm{Score}^{(\ell,h)}\left(\mathbf{x}_i, \mathbf{x}_{j'}\right)\right)}$$

Key, query, scores, masking

- The *key and query functions* map embeddings from $\mathbb{R}^{\mathcal{M}}$ to vectors of type $\mathbb{R}^{d_k}$ for some dimensionality d_k:

$$\begin{aligned}\mathrm{Key}^{(\ell,h)}(\mathbf{x}) &= \mathbf{W}_K^{(\ell,h)}\mathbf{x}\\ \mathrm{Query}^{(\ell,h)}(\mathbf{x}) &= \mathbf{W}_Q^{(\ell,h)}\mathbf{x}\end{aligned}$$

- The *score function* maps to embeddings from $\mathbb{R}^{\mathcal{M}}$ onto a non-normalized score, taking information from non-masked tokens into account:

$$\mathrm{Score}^{(\ell,h)}(\mathbf{x}_i,\mathbf{x}_j) = \begin{cases} -\infty & \text{if token } j \text{ is masked for } i \\ \mathrm{Query}^{(\ell,h)}(\mathbf{x}_i)\cdot \mathrm{Key}^{(\ell,h)}(\mathbf{x}_j) & \text{otherwise.}\end{cases}$$