Tutorial

Token prediction for language generation

A very complicated way to continue sinusoidals

Francesco D'Amico
(for the course of M. Negri)

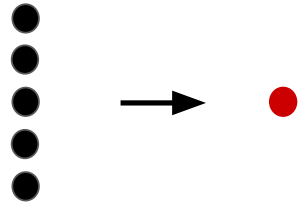
SAPIENZA
UNIVERSITÀ DI ROMA

Consiglio Nazionale
delle Ricerche

Next-token prediction task

In the end is always a prediction

Classification



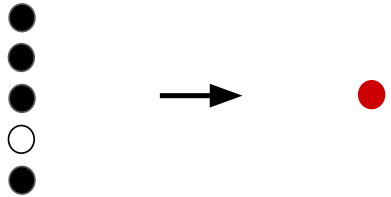
Input

$\{x_i^\mu\}_{i=1,\dots N}$

Output: supervision

y^μ

Masked Language Modeling (MLM)



Input

$\{x_i^\mu\}_{i(\neq k)}$

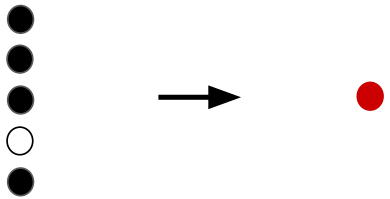
Output: self-supervision

x_k^μ

- What changes is the structure of data
- MLM requires a network more “flexible” than classifiers

MLM loss

Masked Language Modeling (MLM)



Input

$\{x_i^\mu\}_{i(\neq k)}$

Output: self-supervision

x_k^μ

- We require to learn the conditionals

$$P(x_k^\mu | \{x_i^\mu\}_{i(\neq k)})$$

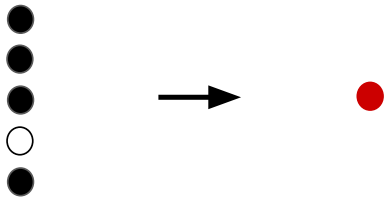
- Max-entropy + locality:

pseudo-likelihood/cross-entropy loss

$$-\sum_{\mu,k} \log P(x_k^\mu | \{x_i^\mu\}_{i(\neq k)})$$

Token variables

Masked Language Modeling (MLM)



Input

$\{x_i^\mu\}_{i(\neq k)}$

Output: self-supervision

x_k^μ

- Variables are tokens (words): enumerated objects from a vocabulary i.e. one-hot

$$x_1^\mu = (0, 0, 1, 0, \ldots, 0)$$
$$x_2^\mu = (1, 0, 0, 0, \ldots, 0)$$
$$x_3^\mu = (1, 0, 0, 0, \ldots, 0)$$

- Output is a probability distribution of masked token

$$x_k^\mu = (0.02, 0.2, 0.6, 0.01, \ldots, 0.03)$$

Generative Pre-trained Transformers (GPT)

Next-token prediction trick

- Autoregressive loss: $-\sum_{\mu,T} \log P(x_T^\mu | \{x_t^\mu\}_{t=1,\ldots,T})$

- **Training:**

  same network does all the predictions at once → ***causal masking***

  need the information of token position in the sequence → ***positional encoding***

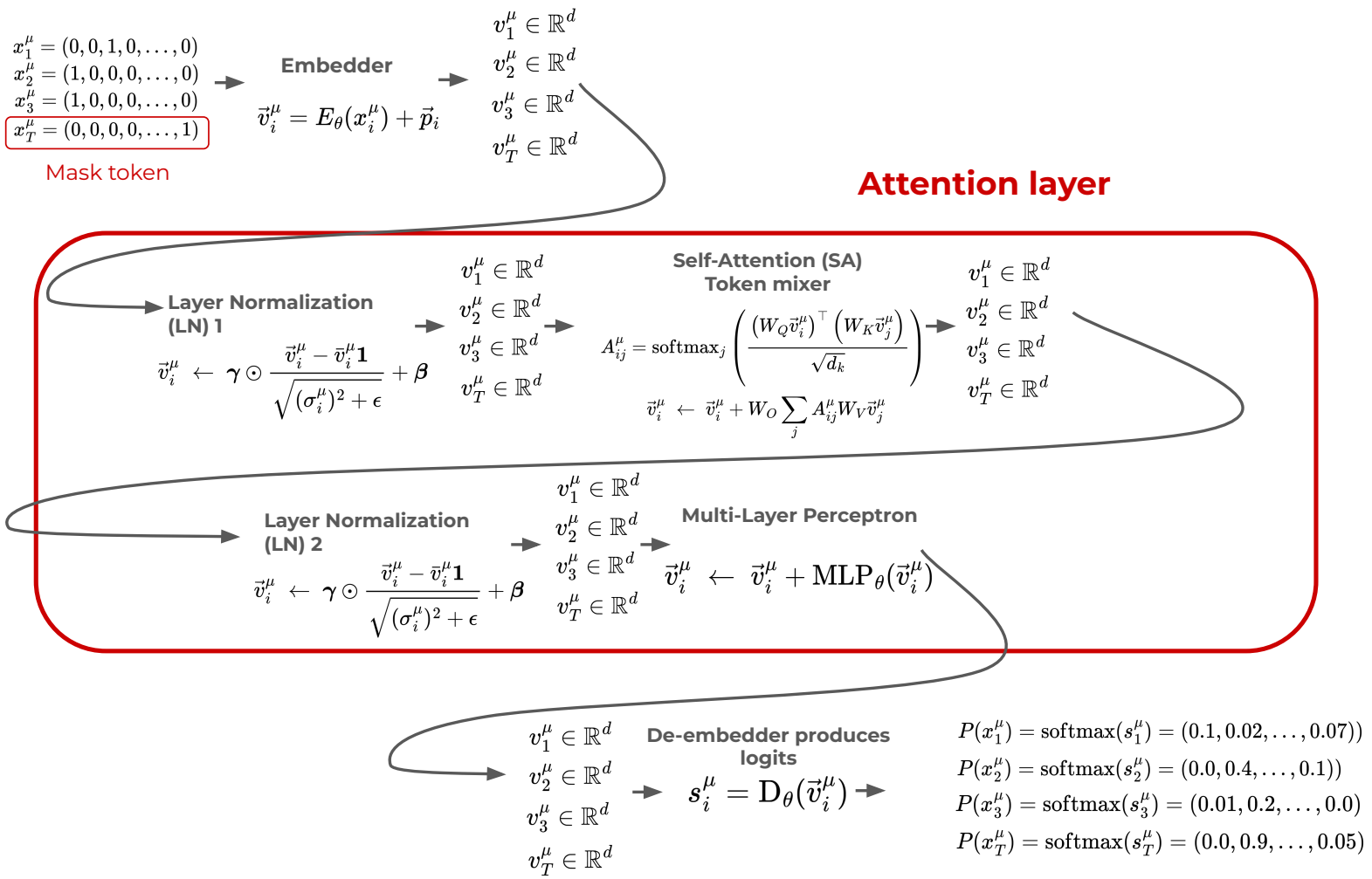
  network fed with fixed-length subsequences → ***context window***

- **Generation:**

  sequential generation of one token at a time starting from a ***prompt*** subsequence

Hands-on time
GPT for oscillators generation

Transformer architecture


$x_1^\mu = (0,0,1,0,\ldots,0)$
$x_2^\mu = (1,0,0,0,\ldots,0)$
$x_3^\mu = (1,0,0,0,\ldots,0)$
$x_T^\mu = (0,0,0,0,\ldots,1)$
Mask token
Embedder
$\vec{v}_i^\mu = E_\theta(x_i^\mu) + \vec{p}_i$
$v_1^\mu \in \mathbb{R}^d$
$v_2^\mu \in \mathbb{R}^d$
$v_3^\mu \in \mathbb{R}^d$
$v_T^\mu \in \mathbb{R}^d$
Attention layer
Layer Normalization (LN) 1
$\vec{v}_i^\mu \leftarrow \boldsymbol{\gamma} \odot \frac{\vec{v}_i^\mu - \bar{v}_i^\mu \mathbf{1}}{\sqrt{(\sigma_i^\mu)^2 + \epsilon}} + \boldsymbol{\beta}$
Self-Attention (SA)
Token mixer
$A_{ij}^\mu = \mathrm{softmax}_j\left(\frac{(W_Q \vec{v}_i^\mu)^\top (W_K \vec{v}_j^\mu)}{\sqrt{d_k}}\right)$
$\vec{v}_i^\mu \leftarrow \vec{v}_i^\mu + W_O \sum_j A_{ij}^\mu W_V \vec{v}_j^\mu$
Layer Normalization (LN) 2
$\vec{v}_i^\mu \leftarrow \boldsymbol{\gamma} \odot \frac{\vec{v}_i^\mu - \bar{v}_i^\mu \mathbf{1}}{\sqrt{(\sigma_i^\mu)^2 + \epsilon}} + \boldsymbol{\beta}$
Multi-Layer Perceptron
$\vec{v}_i^\mu \leftarrow \vec{v}_i^\mu + \mathrm{MLP}_\theta(\vec{v}_i^\mu)$
De-embedder produces logits
$s_i^\mu = \mathrm{D}_\theta(\vec{v}_i^\mu)$
$P(x_1^\mu) = \mathrm{softmax}(s_1^\mu) = (0.1, 0.02, \ldots, 0.07))$
$P(x_2^\mu) = \mathrm{softmax}(s_2^\mu) = (0.0, 0.4, \ldots, 0.1))$
$P(x_3^\mu) = \mathrm{softmax}(s_3^\mu) = (0.01, 0.2, \ldots, 0.0)$
$P(x_T^\mu) = \mathrm{softmax}(s_T^\mu) = (0.0, 0.9, \ldots, 0.05)$

Some relevant references

- **Basic Transformer**: Vaswani, A. et al. (2017). Attention Is All You Need

- **First GPT study**: Radford, A. et al. (2018). Improving Language Understanding by Generative Pre-Training. OpenAI technical report.

- **GPT-2 and neural scaling laws**: Radford, A. et al. (2019). Language Models are Unsupervised Multitask Learners. OpenAI technical report.

- **GPT-3 and in-context learning**: Brown, T. B. et al. (2020). Language Models are Few-Shot Learners

- **In-context learning**: Wei, J. et al. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

- **Pre-layer normalization**: Xiong, R. et al. (2020). On Layer Normalization in the Transformer Architecture

- **RMSNorm**: Zhang, B. and Sennrich, R. (2019). Root Mean Square Layer Normalization

- **Transfer of μP initialization**: Yang, G. et al. (2022). Tensor Programs V: Tuning Large Neural Networks via Zero-Shot Hyperparameter Transfer