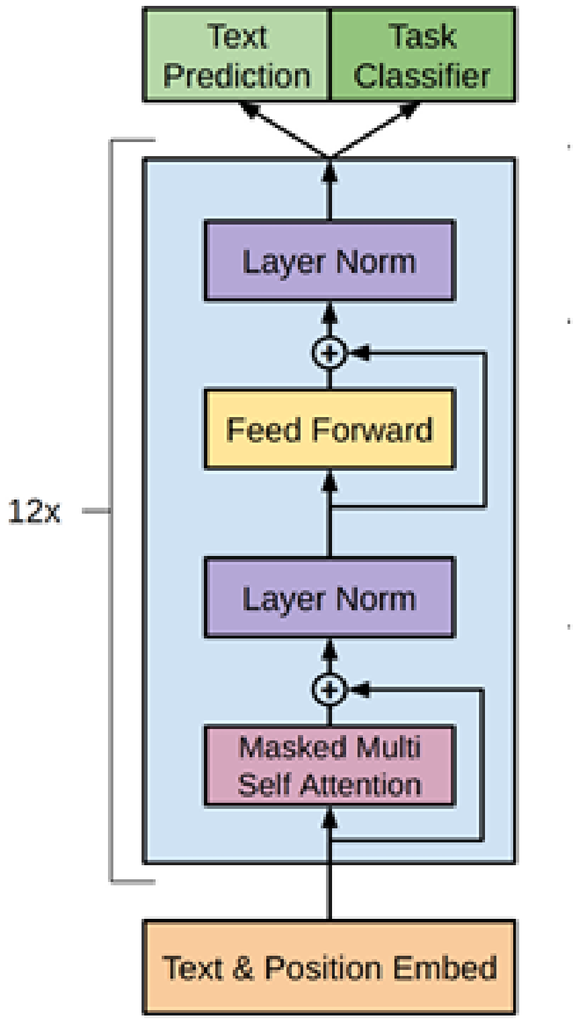


GPT- 1

Generative Pre-Training

DngBack

Key Idea

- Bottleneck: Lack of labled data
- 2 step training process:
    - Generative Pre-training (on unlabeled data)
    - Discriminative fine-tuning (on labeled data)
- Pre-training on large BookCorpus dataset (7000 books)
- Based on decoder architecture from original Transformer

Architecture

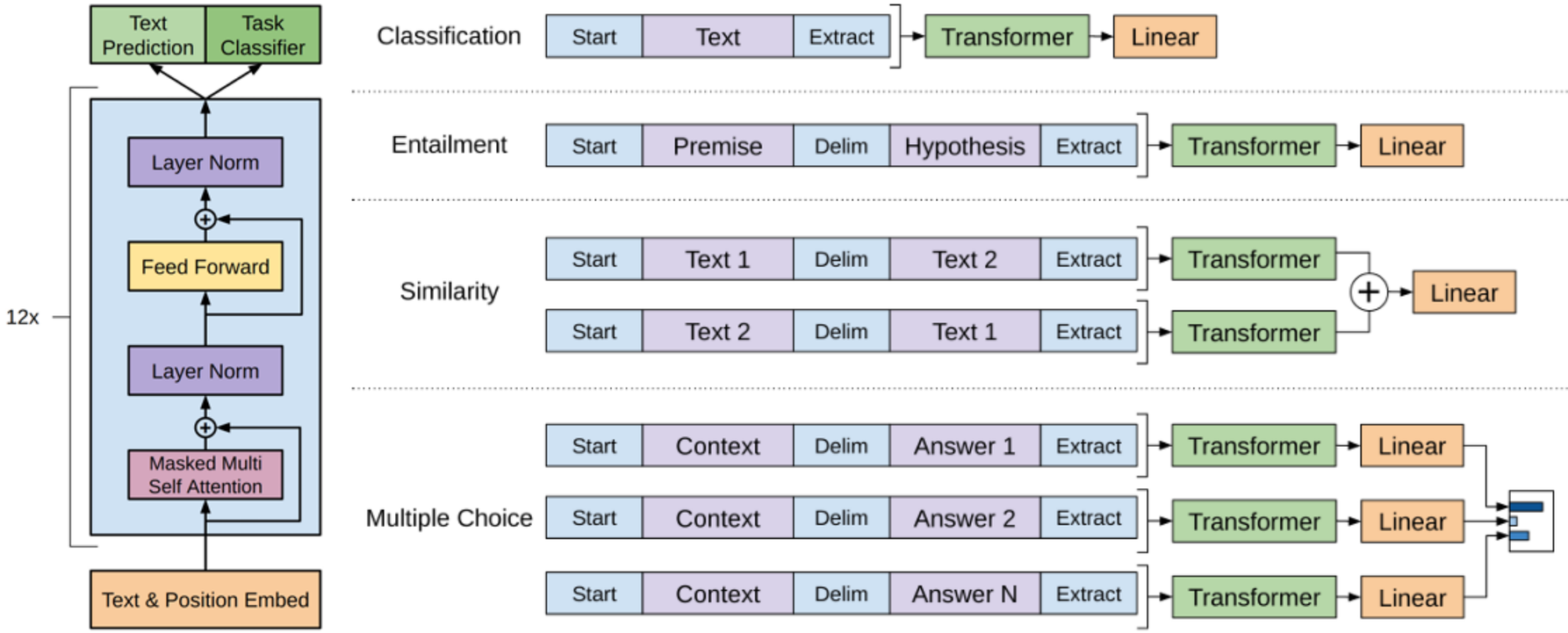


Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

Unsupervised pre-training

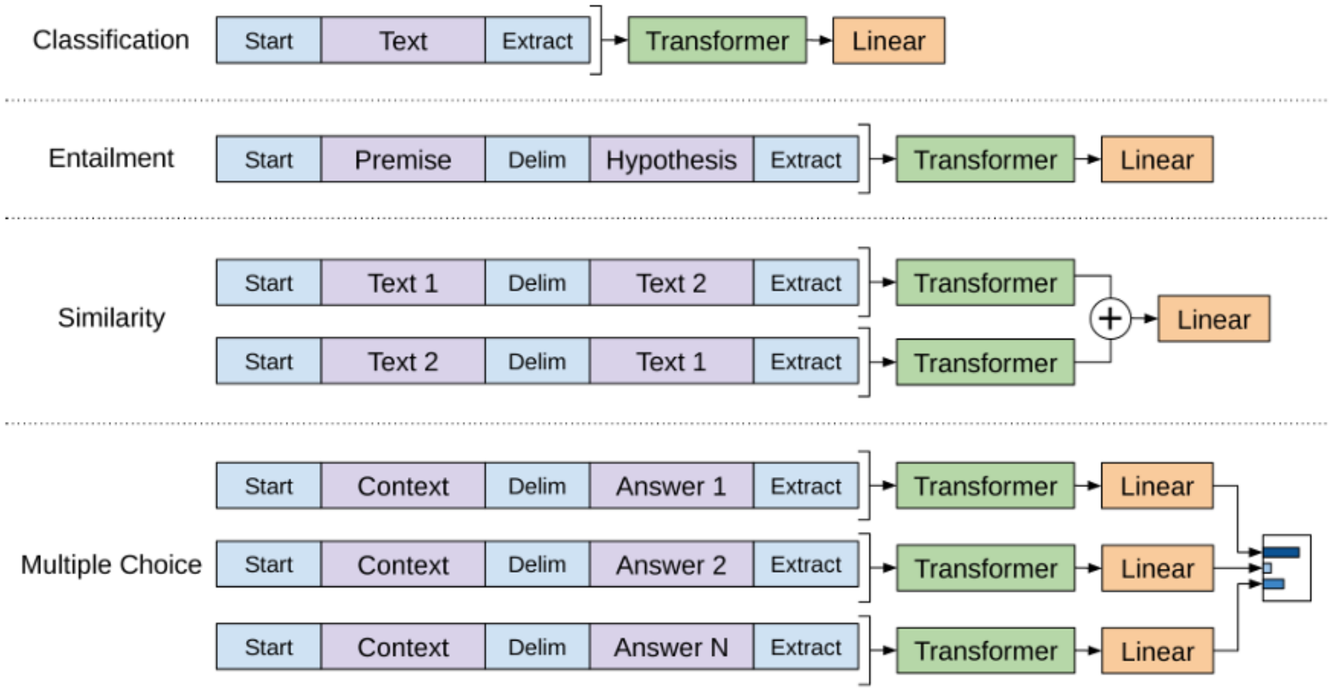
$$L_1(\mathcal{U}) = \sum \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta)$$

In our experiments, we use a multi-layer *Transformer decoder* [34] for the language model, which is a variant of the transformer [62]. This model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feedforward layers to produce an output distribution over target tokens:

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \texttt{transformer\_block}(h_{l-1}) \forall i \in [1, n] \\ P(u) &= \texttt{softmax}(h_n W_e^T) \end{aligned} \tag{2}$$

where $U = (u_{-k}, \ldots, u_{-1})$ is the context vector of tokens, n is the number of layers, W_e is the token embedding matrix, and W_p is the position embedding matrix.

Supervised Fine-Tuning



$$P(y|x^1,\ldots,x^m) = \texttt{softmax}(h_l^m W_y).$$

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1,\ldots,x^m).$$

$$L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C})$$

Reference

- https://www.youtube.com/watch?v=3IweGfgytgY
- https://www.youtube.com/watch?v=LOCzBgSV4tQ

Thanks for watching!