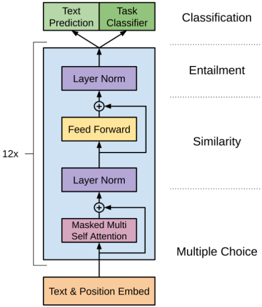
Generative Pre-Trained Transformers

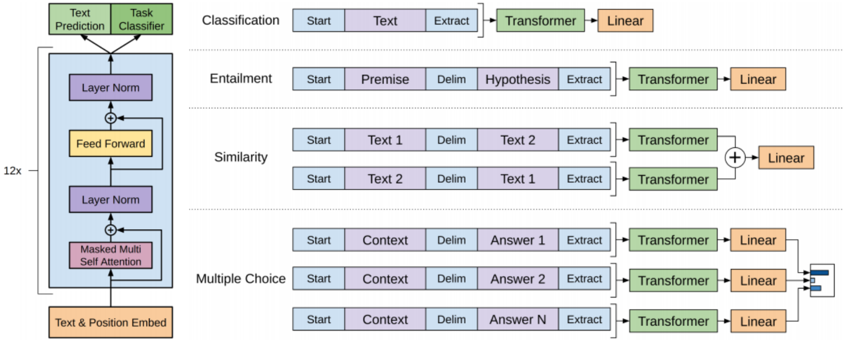
GPT-1 (2018)



Learning goals

- use of the transformer decoder
- input modifications (and how this is useful)

GPT-1



▸ Source: Radford et al., 2018

ARCHITECTURAL DETAILS

- Transformer *decoder* as backbone of the architecture
  - 12-layer-decoder with masked self-attention mechanism
  - Hidden dimension $H = 768$, $A = 12$ Attention heads
  - BPE vocabulary w/ 40k merges
  - Learned positional embeddings (as opposed to fixed, sinusoidal ones in the original Transformer)
- With $U = (w_{t-k}, \ldots, w_{t-1})$

$$\begin{aligned}
\vec{h}_0 &= \vec{U}\vec{W}_e + \vec{W}_p \\
\vec{h}_l &= \mathit{Trafo}(\vec{h}_{l-1}) \forall l \in [1, n] \\
P(w_t) &= \mathit{softmax}(\vec{h}_n \vec{W}_e^\top)
\end{aligned}$$

PRE-TRAINING GPT

- Standard LM objective

$$L_1(\{w_1, \ldots, w_n\}) = \sum_i \log(P(w_t | w_{t-k}, \ldots, w_{t-1}; \Theta))$$

  where $\{w_1, \ldots, w_n\}$ is an *unlabeled* sequence of tokens
- *Resource:* BooksCorpus
  - > 7k unpublished books from various genres
  - contains long texts and thus allows learning long range dependencies

FINE-TUNING GPT

- Linear output layer with softmax activation on top
- Auxiliary language modeling objective during fine-tuning
  $\rightarrow$ Improves generalization
  $\rightarrow$ Accelerates convergence
- Task-specific input transformations
  - *Entailment:*
    Concatenation of premise (p) & hypothesis (h): $[p; \$; h]$
  - *Similarity:* Use both orderings and concatenate resulting representations: $[s_1; \$; s_2]$ and $[s_2; \$; s_1]$
  - *Q&A and Commensense Reasoning:*
    Concatenate context (z), question (q) and each possible answer (a_k): $[z; q; \$, a_k]$
- Fine-tuning is rather quick, 3 epochs were sufficient

FINE-TUNING GPT

- Additional objective:

$$L_2(\{w_1, \ldots, w_n\}) = \sum_{x,y} \log(P(y|w_1, \ldots, w_n))$$

where

- $P(y|w_1, \ldots, w_n) = \mathit{softmax}(h_l^m W_y)$ and
- $\{w_1, \ldots, w_n\}$ is a *labeled* sequence of tokens

- Combining both objectives:

$$L_3(\{w_1, \ldots, w_n\}) = L_2(\{w_1, \ldots, w_n\}) + \lambda \cdot L_1(\{w_1, \ldots, w_n\})$$

SOTA RESULTS

Performance on different benchmarks:

Table 2: Experimental results on natural language inference tasks, comparing our model with current state-of-the-art methods. 5x indicates an ensemble of 5 models. All datasets use accuracy as the evaluation metric.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

Table 3: Results on question answering and commonsense reasoning, comparing our model with current state-of-the-art methods.. 9x means an ensemble of 9 models.

Method	Story Cloze	RACE-m	RACE-h	RACE
val-LS-skip [55]	76.5	-	-	-
Hidden Coherence Model [7]	77.6	-	-	-
Dynamic Fusion Net [67] (9x)	-	55.6	49.4	51.2
BiAttention MRU [59] (9x)	-	60.2	50.3	53.3
Finetuned Transformer LM (ours)	**86.5**	**62.9**	**57.4**	**59.0**

▸ Source: Radford et al., 2018

GPT PARAMETER COUNT

- **We know**:

$$n_{layers} = 12; \quad d_{model} = 768; \quad V = 40000; \quad M = 512;$$

- **Also**:

$$N_{Decoder} = 12 \cdot d_{model}^2 \quad \textit{and} \quad N_{Embedding} = \overbrace{V \times d_{model}}^{\text{token embedding}} + \underbrace{M \times d_{model}}_{\text{pos. embedding}}$$

$$\begin{aligned} \Rightarrow \quad N_{total} &= n_{layers} \cdot N_{Decoder} + N_{Embedding} \\ &= 12 \cdot 12 \cdot 768^2 + 40000 \times 768 + 512 \times 768 \\ &= 116{,}047{,}872 \approx 117M \end{aligned}$$

Note that $N_{Decoder} = 12 \cdot d_{model}^2$ and not $16 \cdot d_{model}^2$ because the Decoder here doesn't do cross attention!