Neural language models: training text representations you've always been waiting for

Halyna Oliinyk @ 1touch.io

1touch.io

What is the motivation?

- continuous space embeddings help to alleviate the curse of dimensionality;
- better probability distributions over sequences of words;
- ability to capture syntax, morphology, semantics, dependencies between sentences, etc.


LET US BEGIN.

ULMFiT

Transfer learning

- inductive transfer learning: source task and target task are different, source domains and target domains may be different or the same;

- transductive transfer learning: target tasks are the same, source and target domains are different;

- unsupervised transfer learning: similar to inductive transfer learning, but designed specifically for unsupervised models.

AWD-LSTM [1]

- mathematical formulation of a standard LSTM is:

$$i_t = \sigma(W^i x_t + U^i h_{t-1})$$
$$f_t = \sigma(W^f x_t + U^f h_{t-1})$$
$$o_t = \sigma(W^o x_t + U^o h_{t-1})$$
$$\tilde{c}_t = \tanh(W^c x_t + U^c h_{t-1})$$
$$c_t = i_t \odot \tilde{c}_t + f_t \odot + \tilde{c}_{t-1}$$
$$h_t = o_t \odot \tanh(c_t)$$

- stochastic gradient descent is defined as:

$$w_{k+1} = w_k - \gamma_k \hat{\nabla} f(w_k),$$

AWD-LSTM [2]

Algorithm 1 Non-monotonically Triggered ASGD (NT-ASGD)

Inputs: Initial point w_0, learning rate γ, logging interval L, non-monotone interval n.

Initialize $k \leftarrow 0, t \leftarrow 0, T \leftarrow 0, \texttt{logs} \leftarrow \texttt{[]}$
while stopping criterion not met **do**
  Compute stochastic gradient $\hat{\nabla} f(w_k)$ and take SGD step (1).
  if $\texttt{mod}(k, L) = 0$ and $T = 0$ **then**
    Compute validation perplexity v.
    if $t > n$ and $v > \min_{l \in \{t-n, \cdots, t\}} \texttt{logs[l]}$ **then**
      Set $T \leftarrow k$
    end if
    Append v to $\texttt{logs}$
    $t \leftarrow t + 1$
  end if
end while
return $\frac{\sum_{i=T}^{k} w_i}{(k-T+1)}$

AWD-LSTM [3]

- variable length backpropagation sequences;
- DropConnect;
- variational dropout;
- embedding dropout;
- weight tying;
- independent embedding size and hidden size;
- activation regularization and temporal activation regularization.

Discriminative fine-tuning

- SGD learning rule with discriminative fine-tuning:

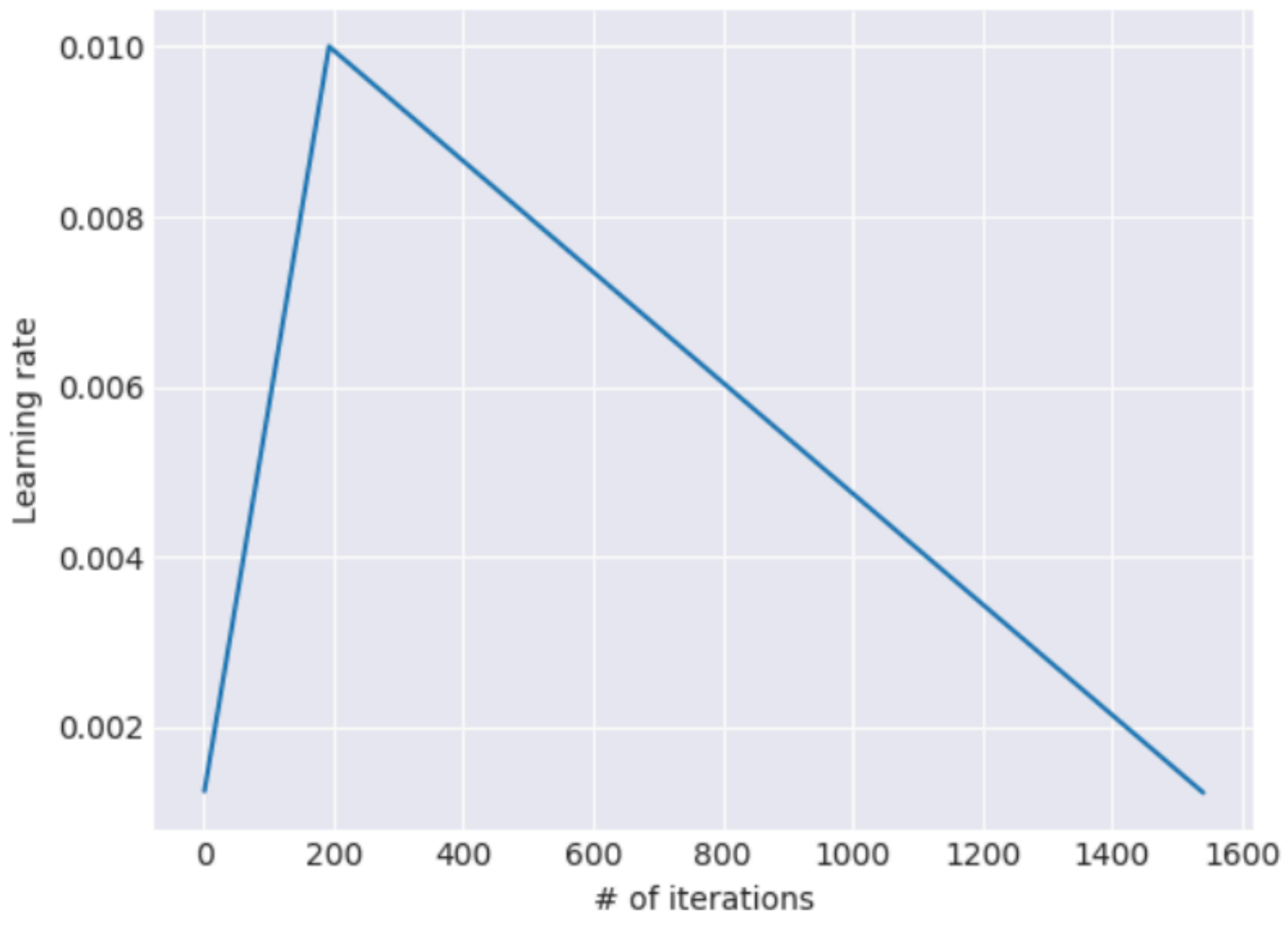
$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta)$$

Slanted triangular rates

$$cut = \lfloor T \cdot cut\_frac \rfloor$$

$$p = \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t - cut}{cut \cdot (1/cut\_frac - 1)}, & \text{otherwise} \end{cases}$$

$$\eta_t = \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio}$$



Batch normalization

- gradient descent step is: $$\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m} \sum_{i=1}^{m} \frac{\partial F_2(\mathrm{x}_i, \Theta_2)}{\partial \Theta_2}$$
- each dimension is normalized as: $$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$
- scaling and shifting normalized value: $$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}.$$
- batch normalizing transform: $$\mathrm{BN}_{\gamma,\beta} : x_{1 \ldots m} \rightarrow y_{1 \ldots m}$$

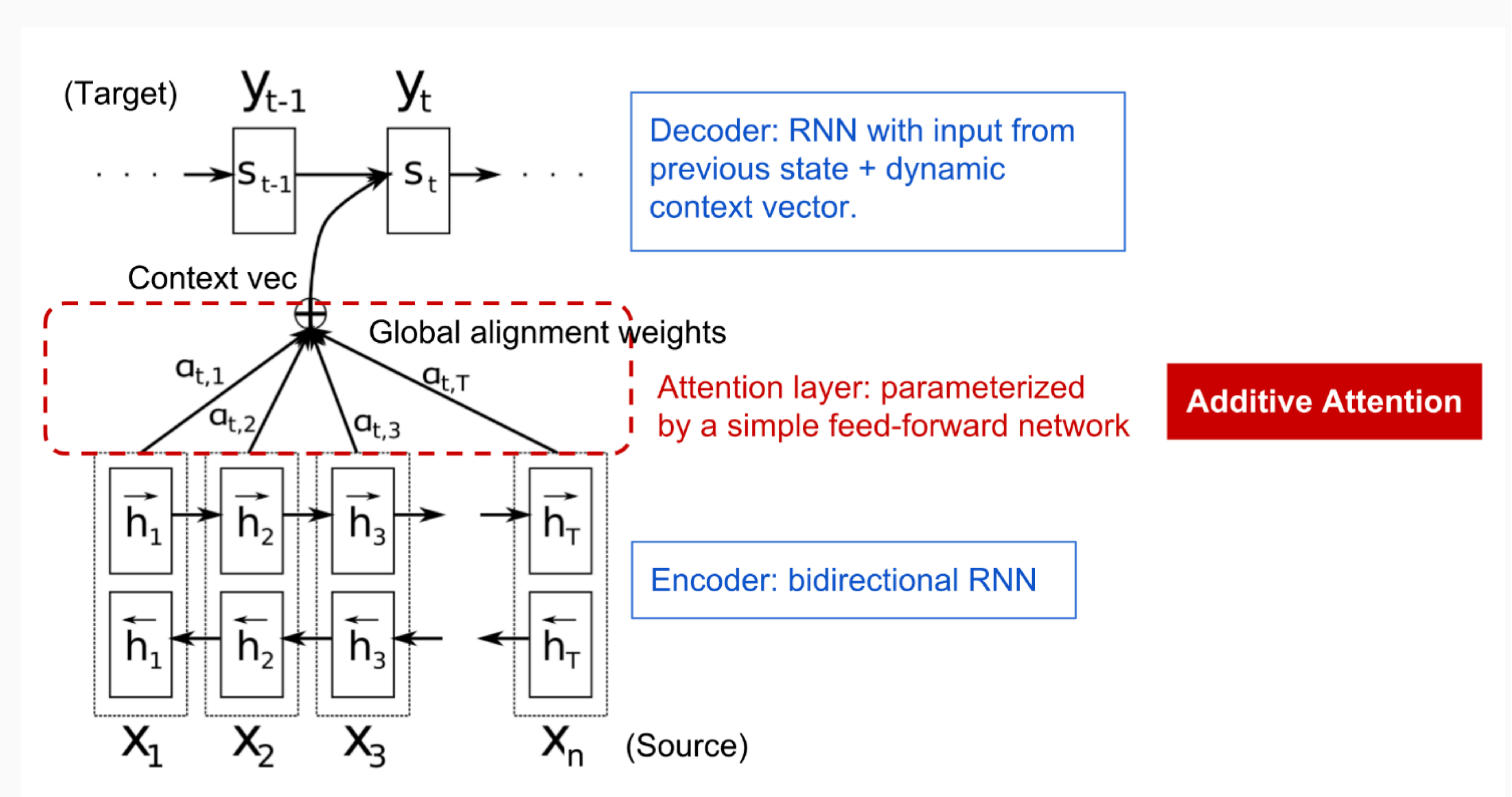
And something more…

- concat pooling: $\mathbf{h}_c = [\mathbf{h}_T, \texttt{maxpool}(\mathbf{H}), \texttt{meanpool}(\mathbf{H})]$
- gradual unfreezing;
- BPTT for text classification;
- bidirectional language model.

ULMFiT = general domain pre-training + target task LM fine-tuning + target task classifier fine-tuning

BERT

General attention mechanism



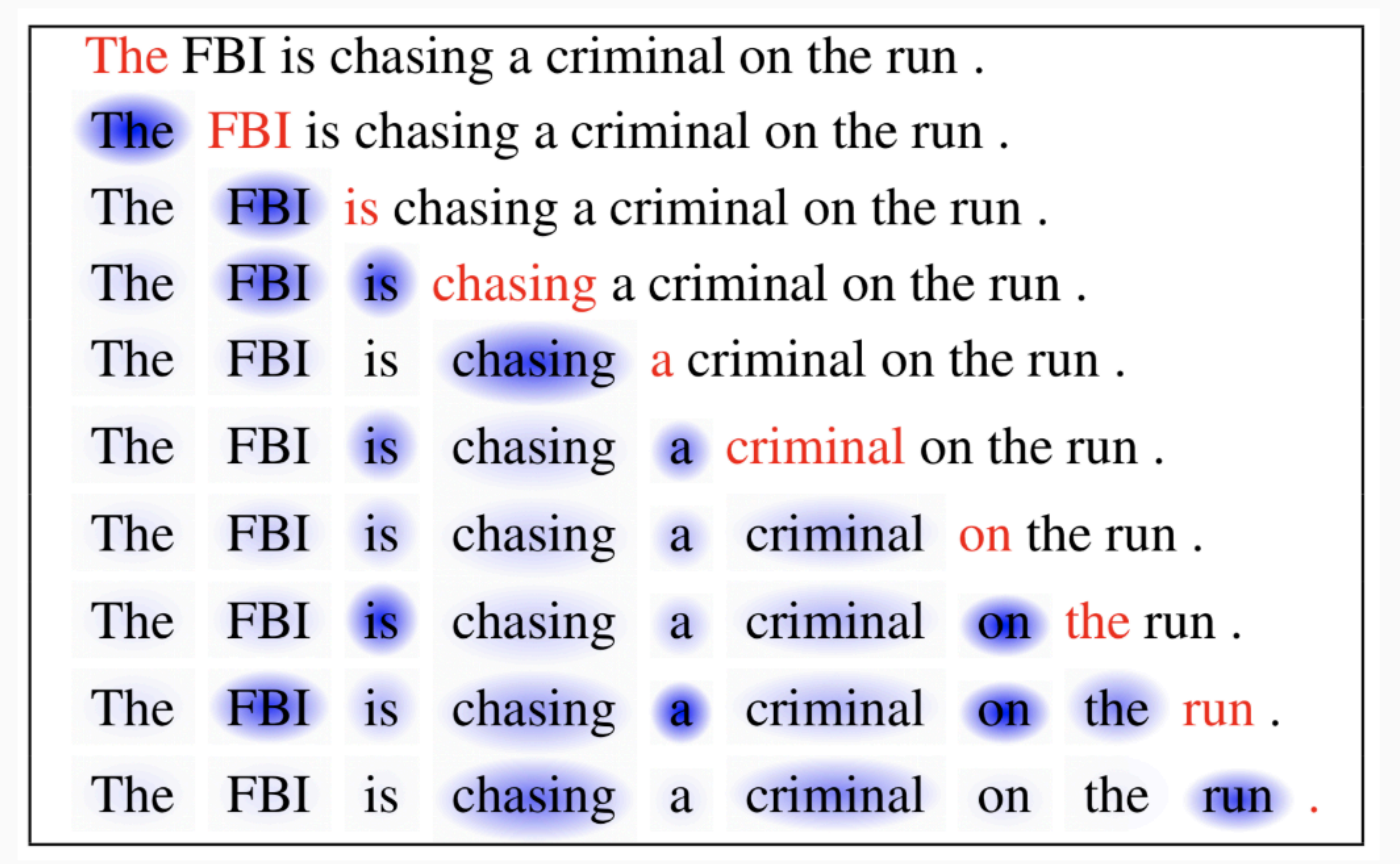
A family of attention mechanisms [1]

Name	Alignment score function
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.

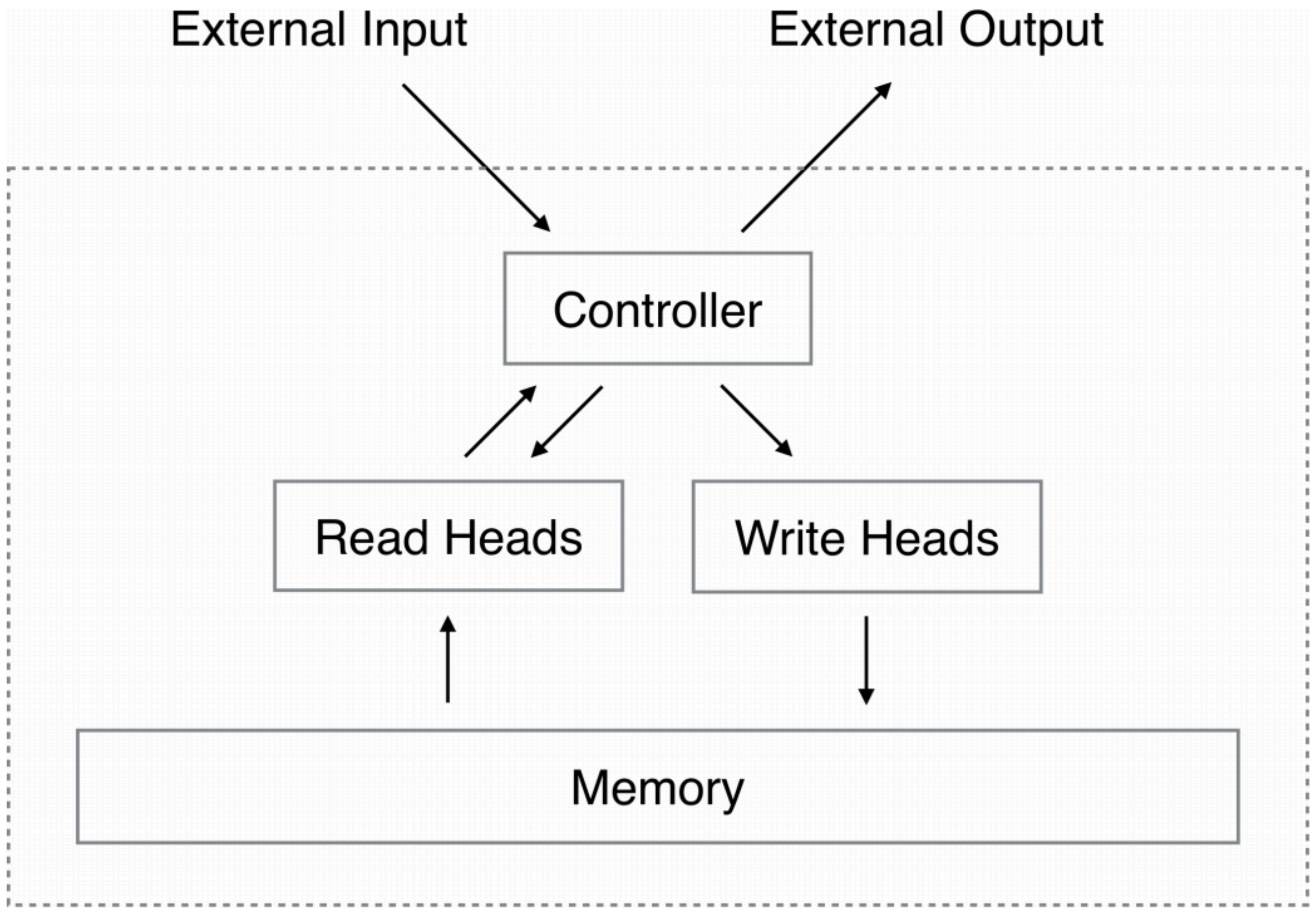
A family of attention mechanisms [2]

Name	Definition
Self-Attention(&)	Relating different positions of the same input sequence. Theoretically the self-attention can adopt any score functions above, but just replace the target sequence with the same input sequence.
Global/Soft	Attending to the entire input state space.
Local/Hard	Attending to the part of input state space; i.e. a patch of the input image.

Self-attention



Neural Turing machine architecture

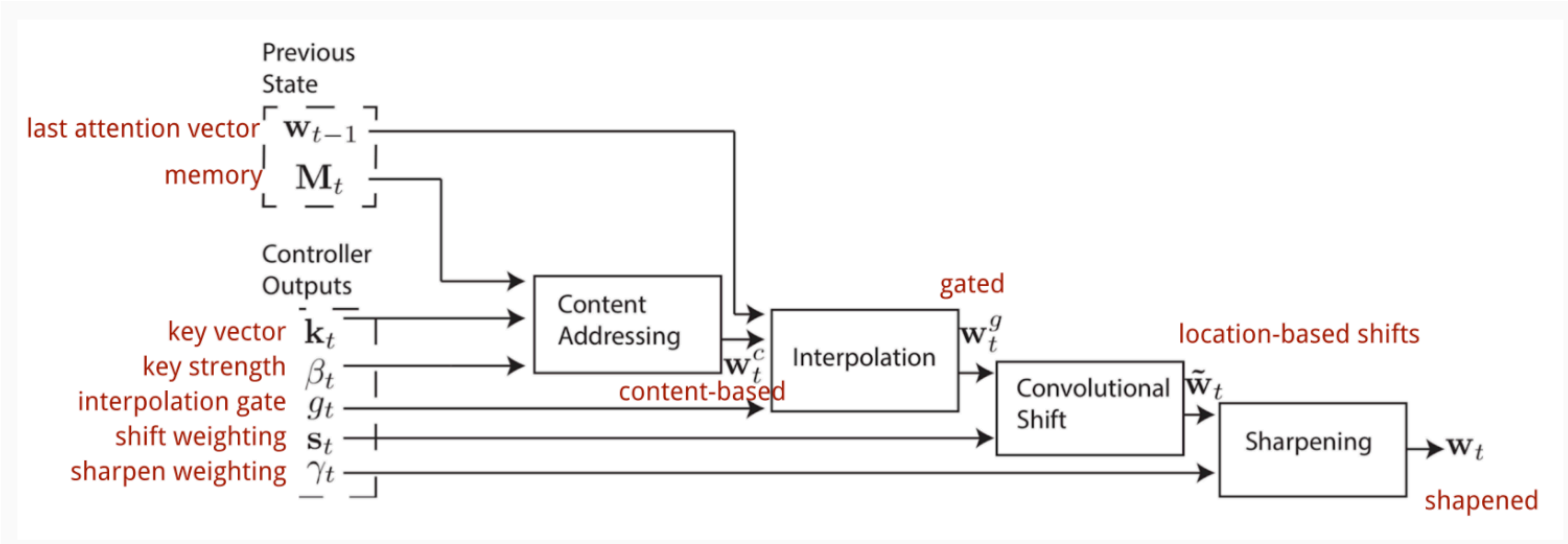


Neural Turing machine main components

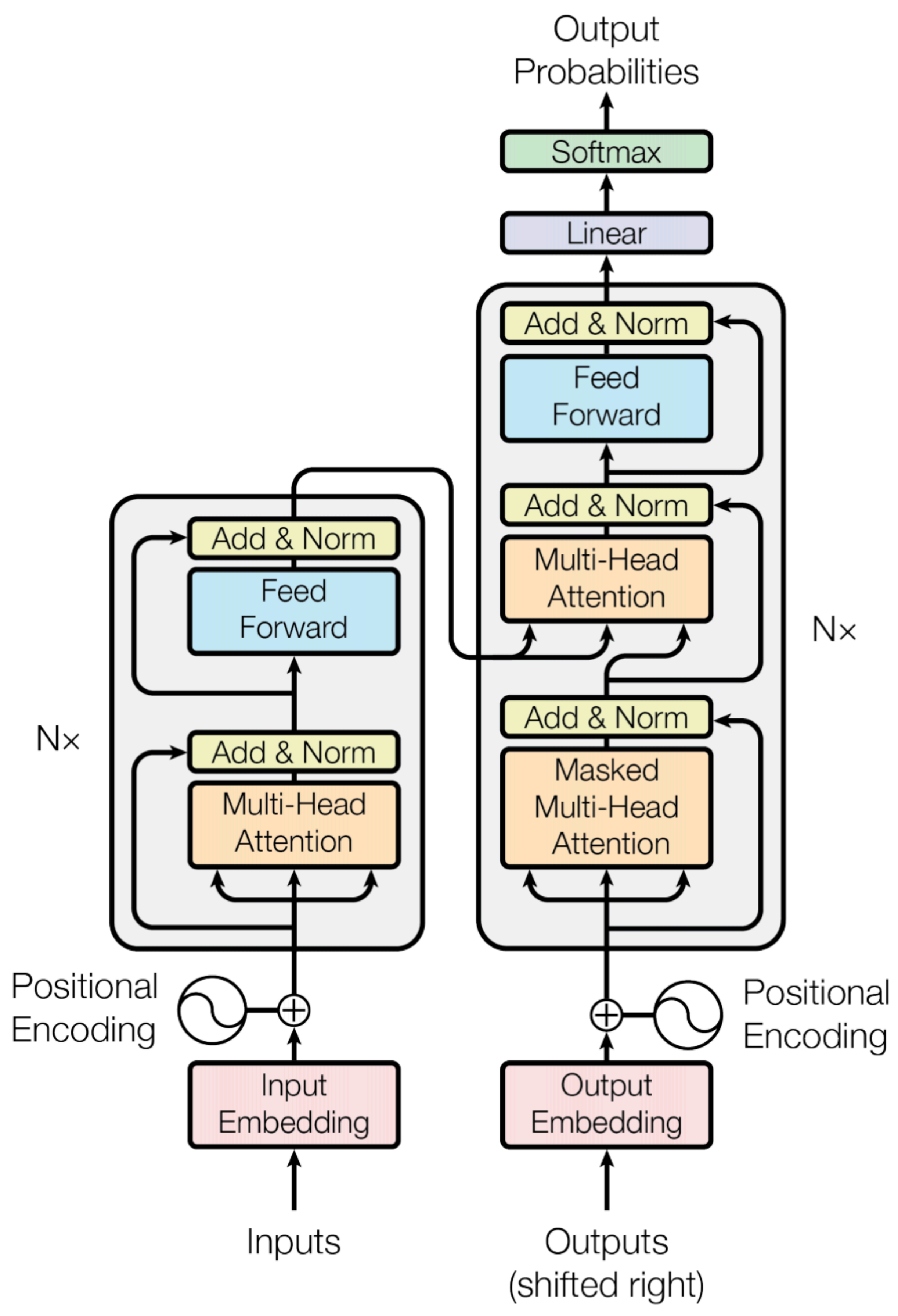
- reading: $\sum_i w_t(i) = 1, \qquad 0 \le w_t(i) \le 1, \forall i. \qquad \mathbf{r}_t \longleftarrow \sum_i w_t(i)\mathbf{M}_t(i),$
- writing: $\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right], \qquad \mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t.$
- focusing by content: $w_t^c(i) \longleftarrow \frac{\exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(i)\big]\Big)}{\sum_j \exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(j)\big]\Big)}. \qquad K\big[\mathbf{u}, \mathbf{v}\big] = \frac{\mathbf{u}\cdot\mathbf{v}}{||\mathbf{u}||\cdot||\mathbf{v}||}.$
- focusing by location: $\mathbf{w}_t^g \longleftarrow g_t \mathbf{w}_t^c + (1 - g_t)\mathbf{w}_{t-1}.$

$$\tilde{w}_t(i) \longleftarrow \sum_{j=0}^{N-1} w_t^g(j)\, s_t(i-j) \qquad w_t(i) \longleftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$

Addressing mechanism in neural Turing machines

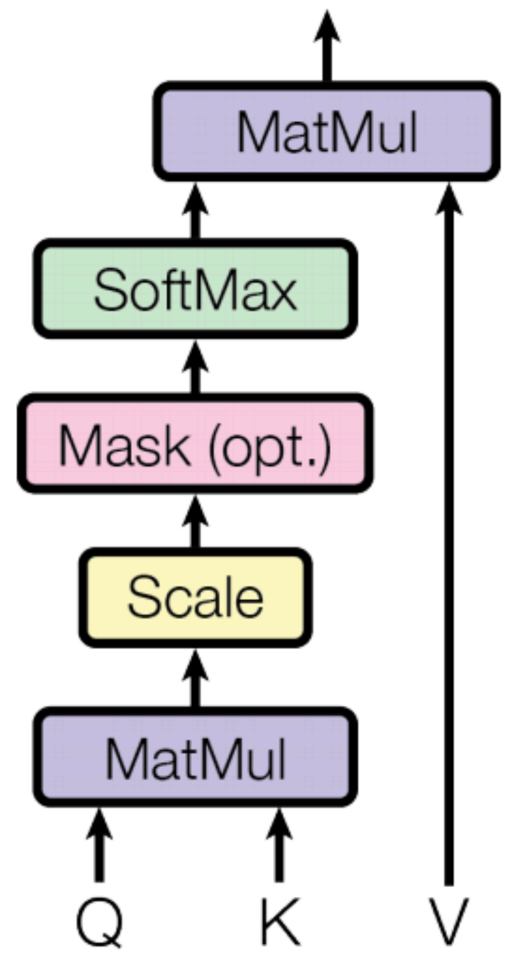


Multi-layer bidirectional transformer encoder [1]

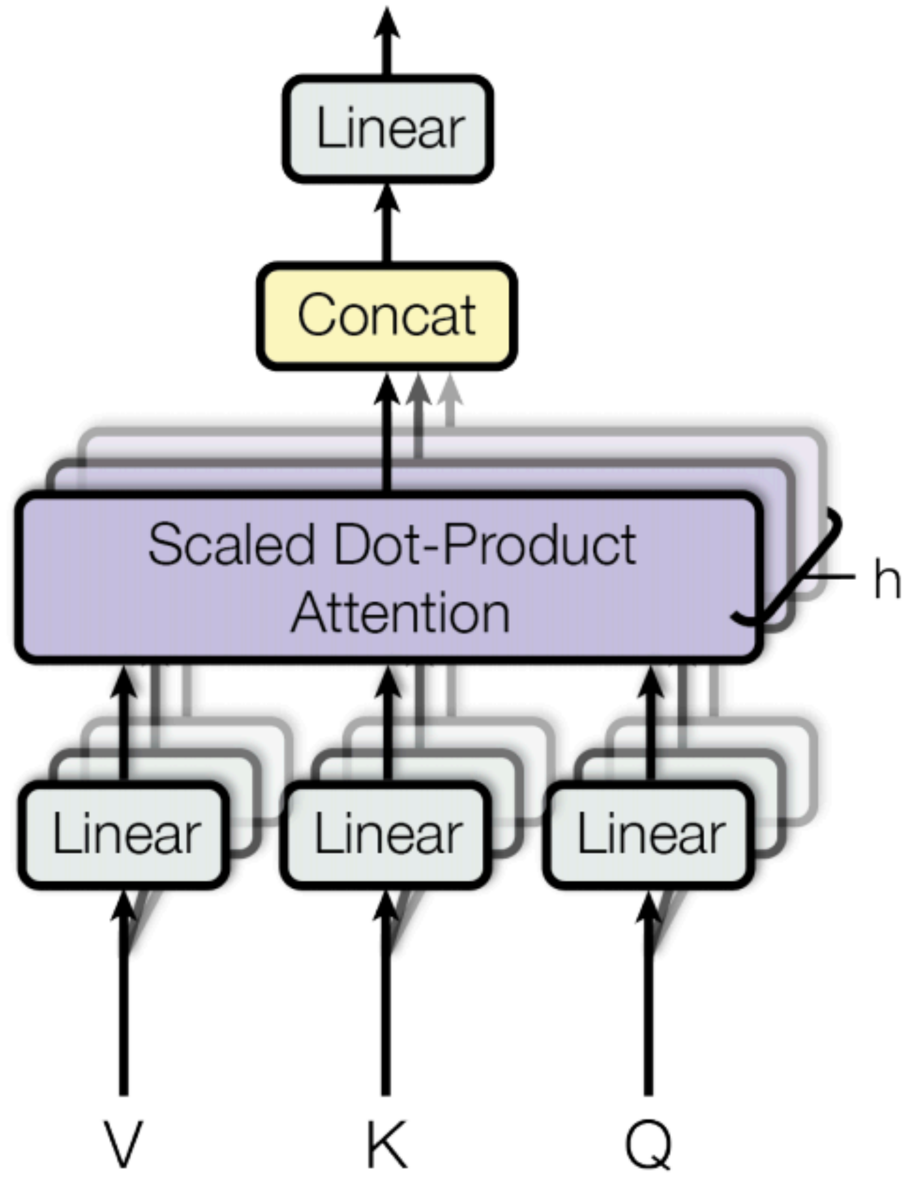


Multi-layer bidirectional transformer encoder [2]

Scaled Dot-Product Attention



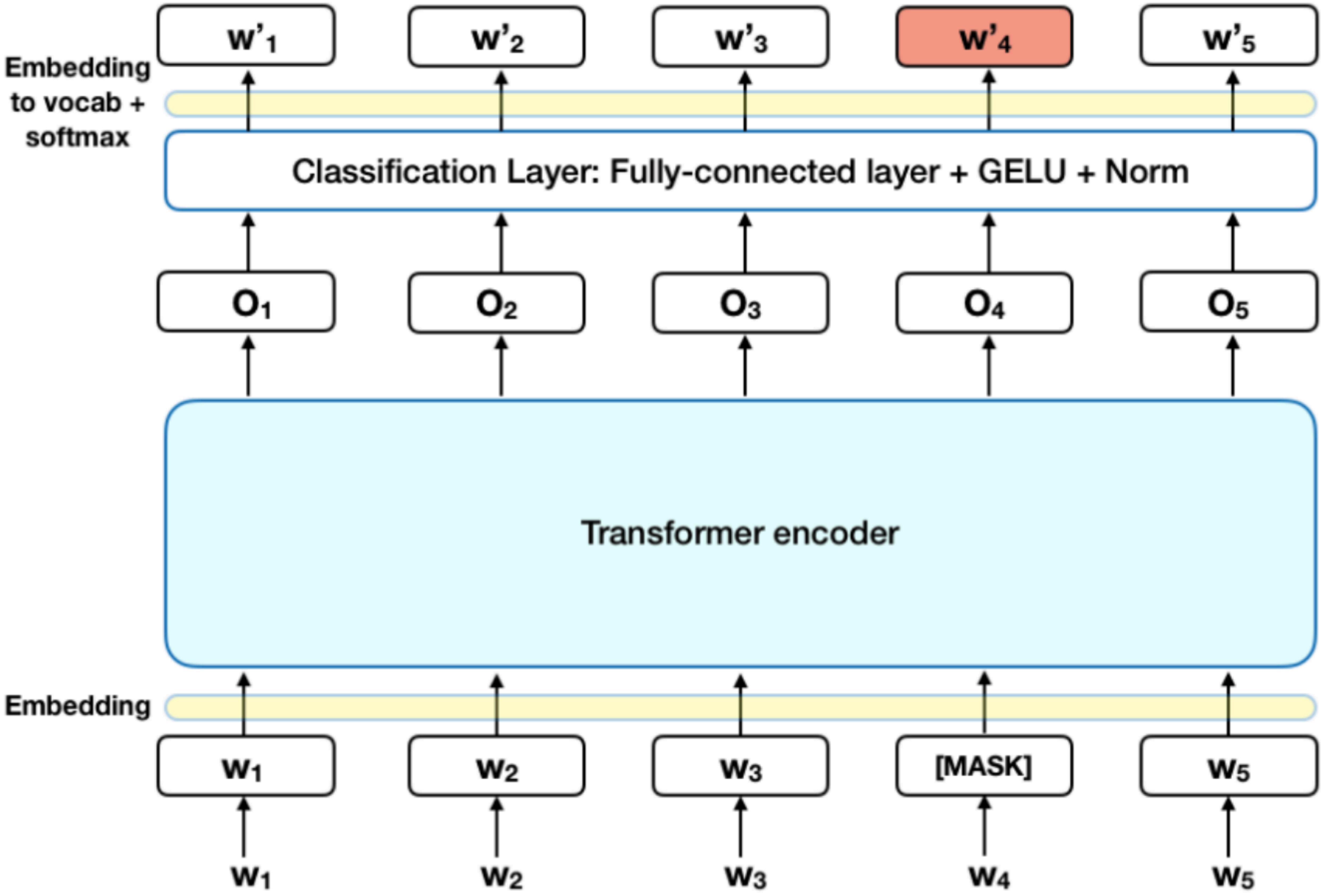
Multi-Head Attention



Attention mechanism of BERT

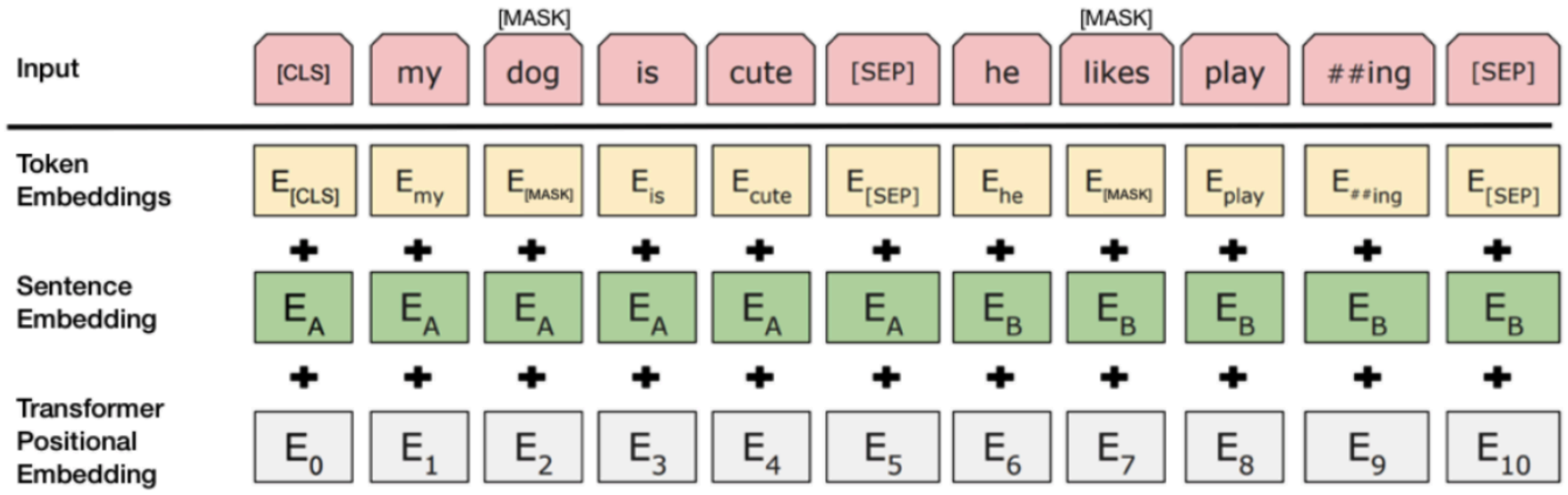
- scaled dot-product attention: $$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$
- multi-head attention: $$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$ $$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Masked LM



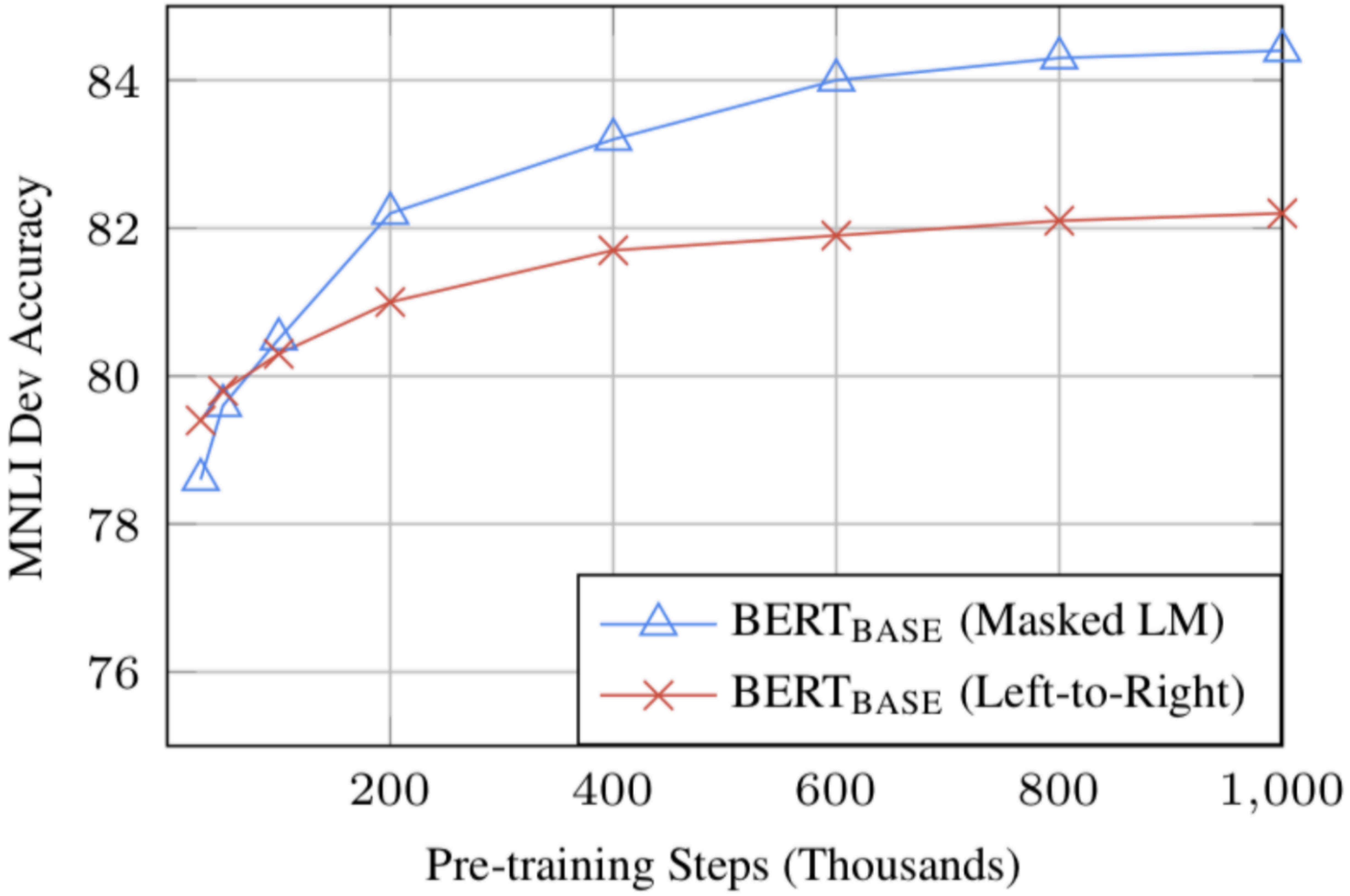
w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

Next sentence prediction

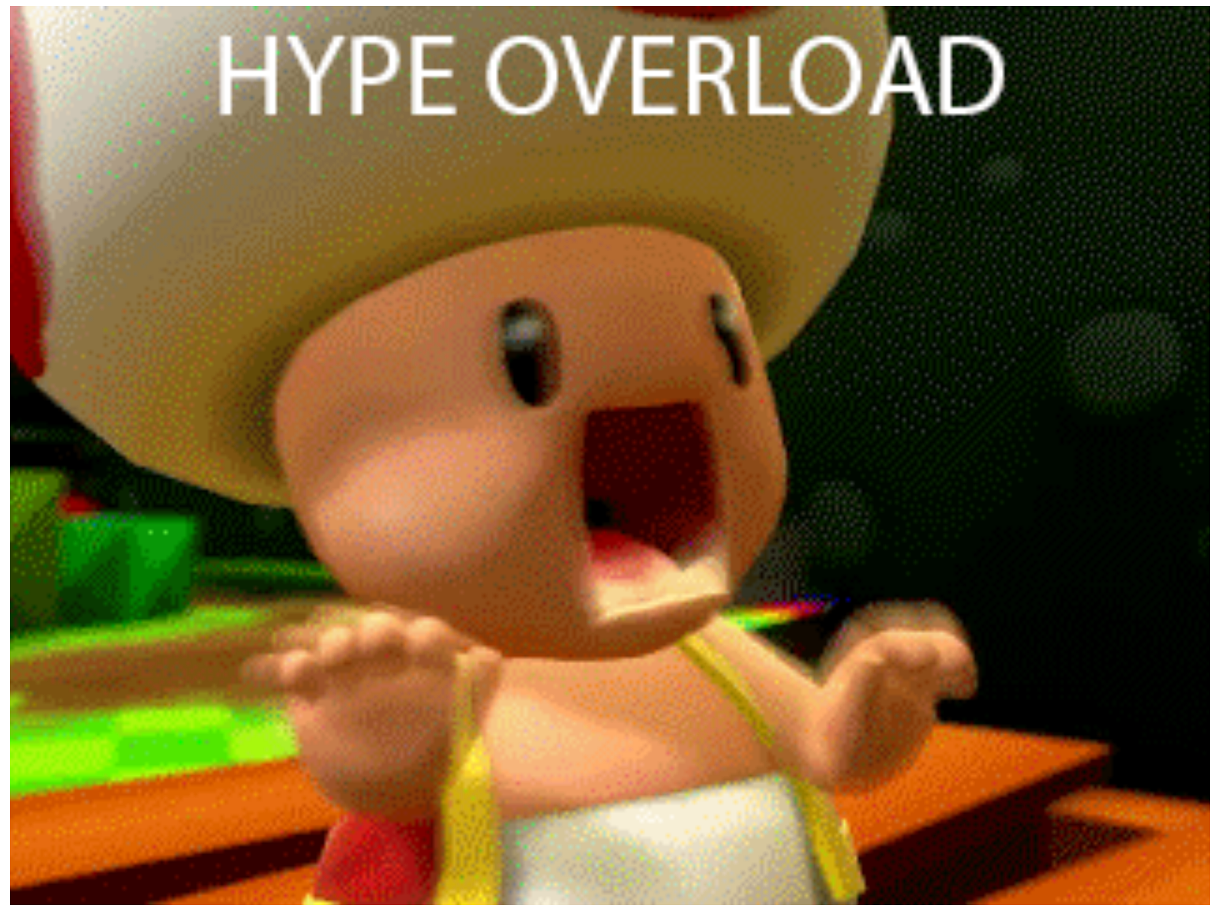


Source: BERT [Devlin et al., 2018], with modifications

BERT pre-training



BERT = pre-training with multi-layer bidirectional transformer encoder + fine-tuning on the target task


HYPE OVERLOAD

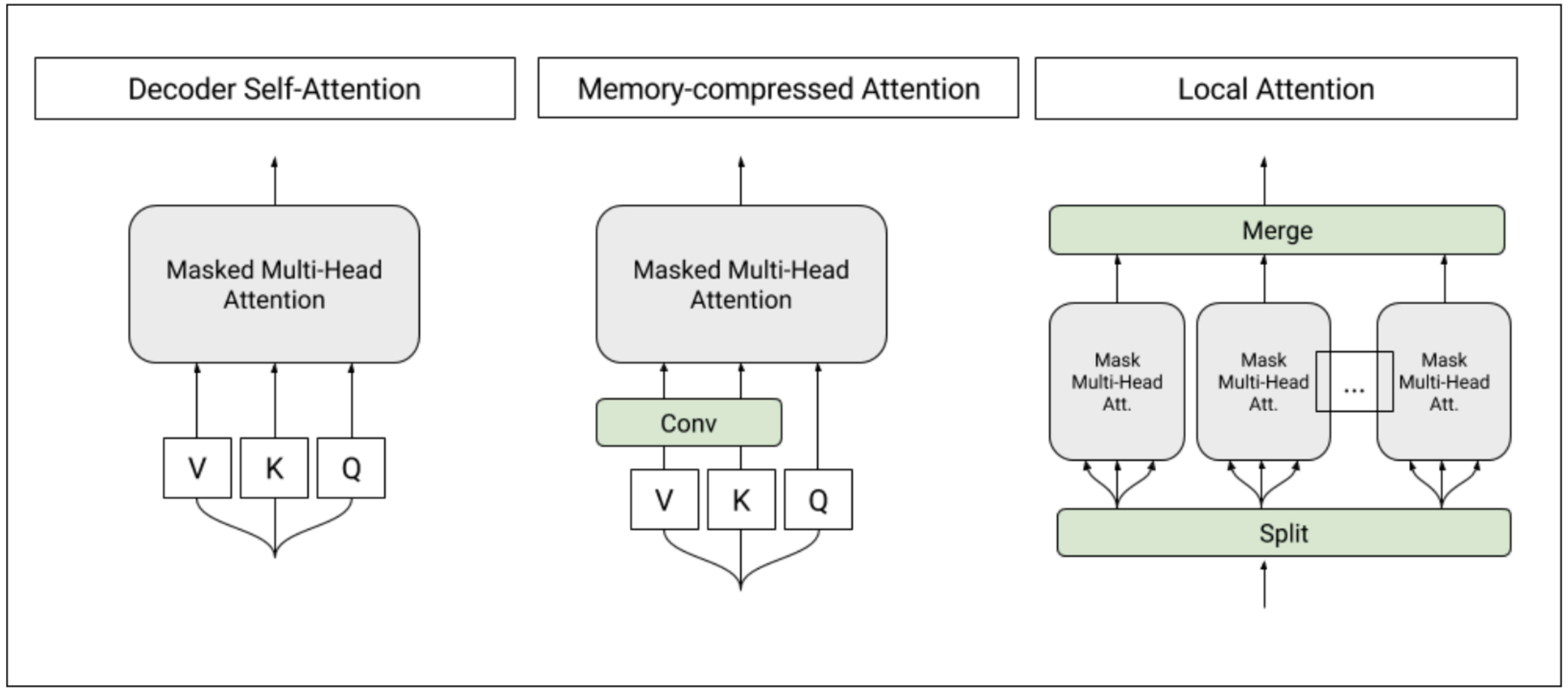
GPT

Learning high-capacity language model [1]

- maximize language modeling objective:

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

- transformer-decoder with memory-compressed attention:



Learning high-capacity language model[2]

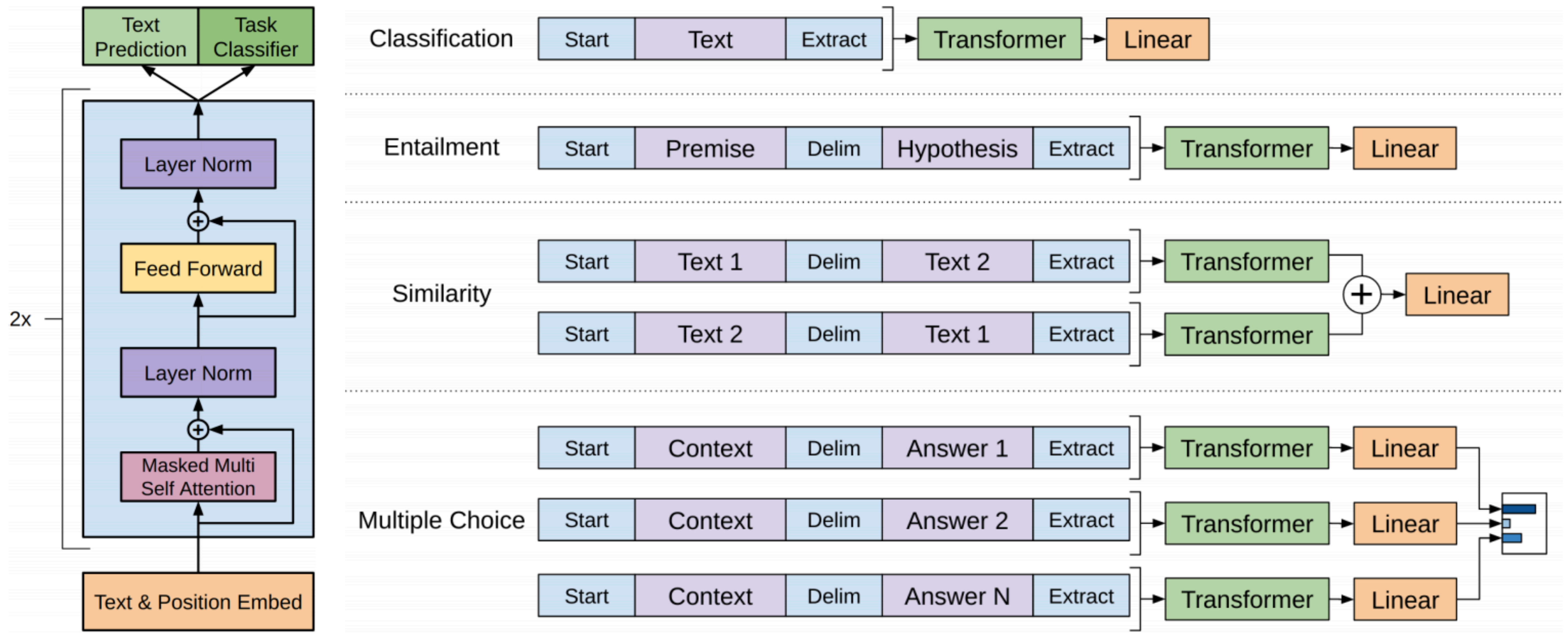
- output distribution over target tokens:

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

Supervised fine-tuning

- linear output layer to predict y: $P(y|x^1, \ldots, x^m) = \texttt{softmax}(h_l^m W_y).$
- objective to maximize: $L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y|x^1, \ldots, x^m).$

Transformer architecture and learning objectives



GPT = learning high-capacity language model + fine-tuning for a target task

ELMo

Bidirectional language models

- forward LM: $$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_1, t_2, \ldots, t_{k-1}).$$
- backward LM: $$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k \mid t_{k+1}, t_{k+2}, \ldots, t_N).$$
- biLM: $$\sum_{k=1}^{N} \Big(\log p(t_k \mid t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k \mid t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \Big).$$

Combining intermediate layer representations

- set of biLM representations:

$$\begin{aligned} R_k &= \{\mathbf{x}_k^{LM}, \overrightarrow{\mathbf{h}}_{k,j}^{LM}, \overleftarrow{\mathbf{h}}_{k,j}^{LM} \mid j = 1, \ldots, L\} \\ &= \{\mathbf{h}_{k,j}^{LM} \mid j = 0, \ldots, L\}, \end{aligned}$$

- task specific weighting of all biLM layers:

$$\mathbf{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} \mathbf{h}_{k,j}^{LM}.$$

ELMo = weighted
bidirectional language model
+ task-specific training

Thank you for attention!