STEVENS INSTITUTE OF TECHNOLOGY

Deep Learning for Recommendation (II)

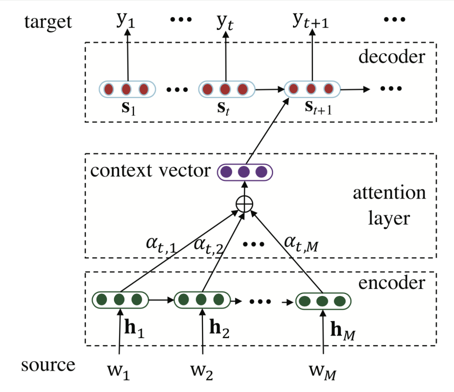
Rensheng Wang,
https://sit.instructure.com/courses/55957

April 28, 2022

Attention-Based Neural Networks

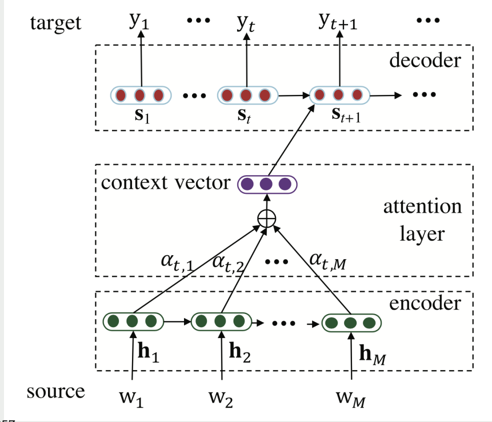
- Attention is a useful tool in deep learning. It is originally proposed to dynamically and selectively collect information from the source sentence in an encoder-decoder model in neural machine translation (NMT) (Bahdanau et al., 2015).
- Attention based Model:

  Figure below shows an encoder-decoder model with the additive attention mechanism.



Attention-Based Neural Networks

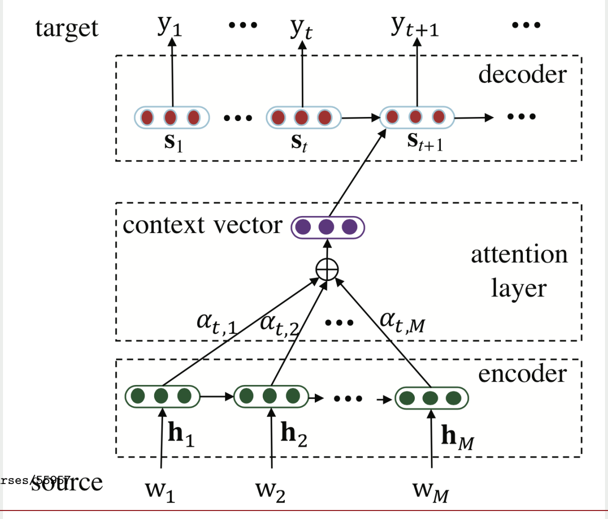
- Input sequence $(w_1, w_2, \cdots, w_M)$ length M; Output sequence $(y_1, y_2, \cdots, y_N)$ length N.
- The encoder (e.g., an RNN) creates a hidden state $\boldsymbol{h}_i$ at each input position $w_i (i = 1, \cdots, M)$. The decoder constructs a hidden state $\boldsymbol{s}_t = f(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, c_t)$ at output position $t(t = 1, \cdots, N)$, where f is the function of the decoder, $\boldsymbol{s}_{t-1}$ and $\boldsymbol{y}_{t-1}$ are the state and output of the previous position, and $\boldsymbol{c}_t$ is the context vector at the position



Attention-Based Neural Networks

- The context vector is defined as the sum of hidden states at all input positions, weighted by attention scores:

$$\boldsymbol{c}_t = \sum_{i=1}^{M} \alpha_{t,i} \boldsymbol{h}_i$$



Context Vector and Attention Score

- The attention score in context vector $\alpha_{t,i}$ is defined as:

$$\alpha_{t,i} = \frac{\exp(g(\boldsymbol{s}_t, \boldsymbol{h}_i))}{\sum_{j=1}^{M} \exp(g(\boldsymbol{s}_t, \boldsymbol{h}_j))}$$

- The function $g(\cdot)$ is determined by the hidden state of the previous output position and the context vector of the **current output position**.

- For example, a feed-forward network with a single hidden layer:

$$g(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^T \tanh\left(\mathbf{W}_a[\boldsymbol{s}_t, \boldsymbol{h}_i]\right)$$

  where $\mathbf{v}_a$ and $\mathbf{W}_a$ are parameters.

- We can see that the context vector ct selectively and dynamically combines the information of the entire input sequence with the attention mechanism.

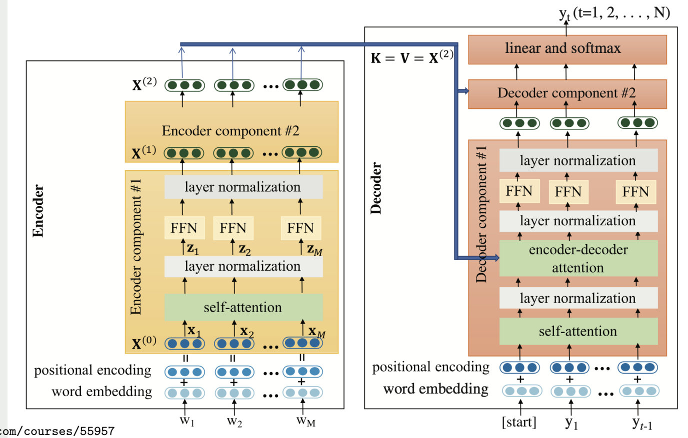
- Compared to the traditional encoder-decoder model in which only a single vector is used, multiple vectors are used to capture the information of the encoder regardless of the distance.

Transformer

- Transformer (Vaswani et al., 2017) is another attention-based neural network under the encoder and decoder framework.
- Different from the aforementioned model which sequentially reads the input sequence (left-to-right or right-to-left), Transformer reads the entire input sequence at once. The characteristic enables it to learn the model by considering both the left and the right context of a word.
- Transformer consists of an encoder for transforming the input sequence of words into a sequence of vectors (internal representation) and a decoder for generating an output sequence of words one by one given the internal representation.

Transformer

- The encoder is a stack of encoder components with an identical structure, and the decoder is also a stack of decoder components with an identical structure, where the encoder and decoder have the same number of components.
- Each encoder component or layer consists of a self-attention sublayer and a feed-forward network sublayer.



Transformer

- It receives a sequence of vectors (packed into a matrix) as input, processes the vectors with the self-attention sublayer, and then passes them through the feed-forward network sublayer.
- Finally, it sends the vectors as output to the next encoder component.
- Specifically, the input is a sequence of words $(w_1, w_2, \cdots, w_M)$ with length M.
- Each word w_i is represented by a vector $\mathbf{x}_i$ as a sum of the word embedding and positional encoding of it.
- The vectors are packed into a matrix $\mathbf{X}(0) = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M]^T$.
- The self-attention sub-layer converts $\mathbf{X}(0)$ into $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_M]^T$ through self-attention

$$\mathbf{Z} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

Self-Attention

- The main idea behind self-attention is that instead of using a fixed embedding for each token, we can use the whole sequence to compute a weighted average of each embedding.
- Another way to formulate this is to say that given a sequence of token embeddings $x_1, \cdots, x_M$, self-attention produces a sequence of new embeddings $x'_1, \cdots, x'_M$, where each x'_i is a linear combination of all the x_j: $x'_i = \sum_{j=1}^{M} a_{ji} xj$ where a_{ji} are called attention weights and are normalized so that $\sum_j a_{ji} = 1$.
- To see why averaging the token embeddings might be a good idea, consider the word "apple". You might think of a fruit, but if you were given more context, like apple devices, then you would realize that "apple" refers to the computer company. Similarly, we can create a representation for "apple" that incorporates this context by combining all the token embeddings in different proportions, perhaps by assigning a larger weight a_{ji} to the token embeddings for "device". Embeddings that are generated in this way are called contextualized embeddings and predate the invention of transformers in language models like ELMo.
- As we explained above, depending on the context, two different representations for "apple" can be generated via self-attention.

Self-Attention

- Self-Attention Parameters:
  - $\mathbf{K}$, $\mathbf{V}$, and $\mathbf{Q}$ are matrices of key vectors, value vectors, and query vectors respectively;
  - d_k is the dimensionality of the key vector;
  - $\mathbf{K}$ is the resulting matrix consisting of M vectors.
- The matrices $\mathbf{K}$, $\mathbf{V}$, and $\mathbf{Q}$ are calculated as

$$\begin{aligned} \mathbf{Q} &= \mathbf{X}^{(0)}\mathbf{W}_Q \\ \mathbf{K} &= \mathbf{X}^{(0)}\mathbf{W}_K \\ \mathbf{V} &= \mathbf{X}^{(0)}\mathbf{W}_V \end{aligned}$$

  where $\mathbf{W}_Q$, $\mathbf{W}_K$ and $\mathbf{W}_V$ are embedding matrices.
- After that, the vectors $\mathbf{z}_i$ in $\mathbf{Z}$ are independently processed by the feed-forward network sub-layer.

Self-Attention

- In each sub-layer, a residual connection is employed, followed by layer-normalization.
- The output of the encoder component is represented as

$$\mathbf{X}(1) = [\mathbf{x}(1), \mathbf{x}(1), \cdots, \mathbf{x}(1)]^T.$$

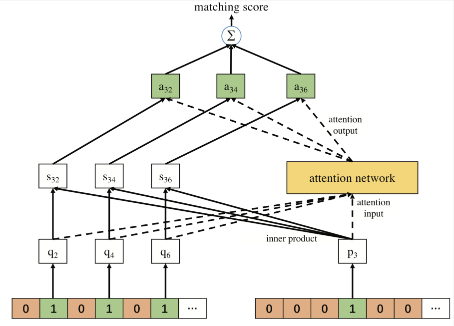
- $\mathbf{X}^{(1)}$ is then fed into the next encoder component.
- The encoder finally outputs the vectors (representations) corresponding to all input words, denoted as $\mathbf{X}^{\text{enc}}$.

Decoder in Transformer

- Each decoder component or layer in the decoder consists of a self-attention sub-layer, an encoder-decoder attention sub-layer, and a feed-forward network sub-layer.
- The sub-layers have the same architecture as that of the encoder component.
- After encoding, the output of the encoder is used to represent the key and value vectors: $\mathbf{K} = \mathbf{V} = \mathbf{X}^{\text{enc}}$, which are then used for "encoder-decoder attention" in each decoder component.
- The decoder sequentially generates words for all output positions $1, 2, \cdots, N$.
- At each position $1 \leq t \leq N$, the bottom decoder component receives the previously outputted words "[start], $y_1, y_2 \cdots, y_{t-1}$", masks the future positions, and outputs internal representations for the next decoder component.
- Finally, the word at position t, denoted as $\mathbf{v}_t$, is selected according to a probabilistic distribution generated by the softmax layer on the top decoder component.
- The process is repeated until a special symbol (e.g., "[end]") is generated, or the maximal length is reached.

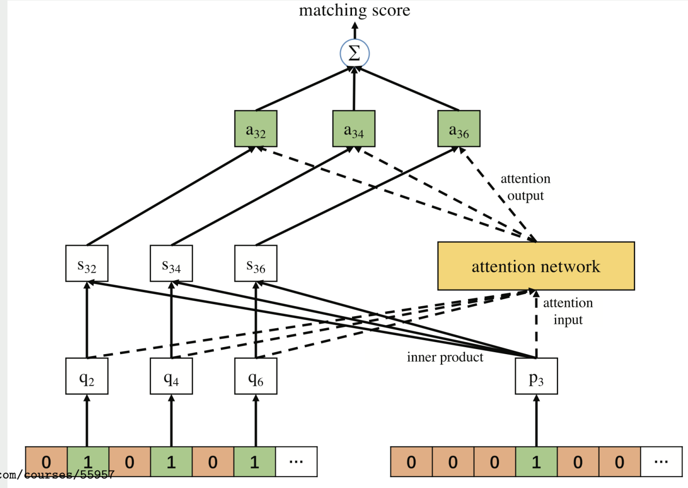
Attention-Based Recommendation

- One observation in the learning of user representation is that histor- ical items may not equally contribute to the modeling of the users preference. For example, a user may choose a trendy item based on its high popularity rather than his/her own interest.
- Although, in principle, an MLP learned from interaction history may be able to capture the complicated relationships, the process is too implicit and there is no guarantee for that.
- To solve the problem, the Neural Attentive Item Similarity (NAIS) model (He et al., 2018a) employs a neural attention network to explicitly learn the weight of each historical item.



Neural Attentive Item Similarity (NAIS) model

- NAIS uses a learnable weight on each interacted item of a user.
- Let $\mathcal{Y}_u$ be the set of interacted items of user u, and each item i is associated with two ID embedding vectors $\mathbf{p}_i$ and $\mathbf{q}_i$ to represent its role as a target item and a historical item, respectively.



Neural Attentive Item Similarity (NAIS) model

- Let $\mathcal{Y}_u$ be the set of interacted items of user u, and each item i is associated with two ID embedding vectors $\mathbf{p}_i$ and $\mathbf{q}_i$ to represent its role as a target item and a historical item, respectively.
- The matching function in NAIS is formulated as

$$f(u,i) = \left(\sum_{j \in \mathcal{Y}_u \backslash \{i\}} a_{ij} \mathbf{q}_j \right)^T \mathbf{p}_i$$

$$a_{ij} = \frac{\exp(g(\mathbf{p}_i, \mathbf{q}_j))}{\left[\sum_{j \in \mathcal{Y}_u \backslash \{i\}} a_{ij} \exp(g(\mathbf{p}_i, \mathbf{q}_j)) \right]^\beta}$$

  where a_{ij} is an attention weight that controls the weight of the historical item j in an estimation of the user us matching score on the target item i.
- The output of $g(\cdot)$ is further processed by a smoothed softmax function where β is in $(0,1)$ to smooth the weighted sum of active users ((the default value of β is 0.5).