Lecture 10. Attention Models and Transformers

Lecture 10. Attention Models and Transformers

Alex Avdyushenko

Kazakh-British Technical University

November 12, 2022

Good afternoon, dear students. Today we are going to talk about attention and transformers — probably the most breakthrough architecture of the last maybe five years of deep learning. This is a very interesting approach, and again it is quite similar to human information processing.

- List the disadvantages of convolutional neural networks
- Write the formula for the simplest (vanilla) recurrent network module
- What kind of filters (gates) does LSTM have?

Ok, great, and as always we start with traditional five-minutes questions on the previous lecture. Please, write answers or send photos with them directly to me in private messages here in Teams or may be in Telegram, but choose only one option please :)

Disadvantages of vanilla Recurrent Neural Network
Recall

We use one hidden vector

$$h_t = f_W(h_{t-1}, x_t)$$

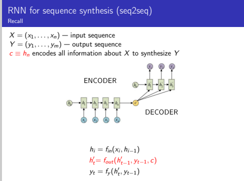
As a function f_W we set a linear transformation with a non-linear component-wise "sigmoid":

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$
$$y_t = W_{hy} h_t$$

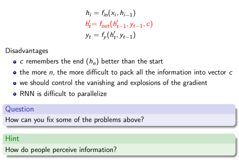
Disadvantages

1. input and output sequence lengths must match
2. "reads" the input only from left to right, does not look ahead
3. therefore it is not suitable for machine translation, question answering tasks and others

Now, let's remember vanilla recurrent neural network and it's disadvantages.



Ok, next step. How can we solve this problems? For example, we can use seq2seq architecture.



Now, as is often the case we have new, more difficult problems with seq2seq architecture.

2022-11-12

└─Let's count the number of passes

Basketball!

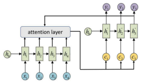
Now let's have some fun and count the number of passes in the video.

RNN with attention mechanism

$a(h, h')$ is the similarity function of input h and output h' (for example, dot product or $\exp(h^T h')$ and others)

α_{ti} — importance of input i for output t (attention score), $\sum_i \alpha_{ti} = 1$

c_t — input context vector for output t



So, how can we use this mechanism for improving our network architecture? We use function a, it is the similarity function of two vectors. And so on...

The Main Areas for Attention Models

Converting one sequence to another, i.e. seq2seq

- Machine translation
- Question answering
- Text summarization
- Annotation of images, videos (multimedia description)
- Speech recognition
- Speech synthesis

Sequence processing

- Classification of text documents
- Document Sentiment Analysis

Attention models allow us to solve a lot of problems from different areas.

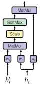
Vector Similarity Functions

$a(h, h') = h^T h'$ is the scalar (inner) product
$a(h, h') = exp(h^T h')$ — norm becomes SoftMax
$a(h, h') = h^T W h'$ — with the learning parameter matrix W
$a(h, h') = w^T \tanh(Uh + Vh')$ is additive attention with w, U, V

Linear vector transformations query, key, value:

$a(h_i, h'_{t-1}) = (W_k h_i)^T (W_q h'_{t-1}) / \sqrt{d}$
$\alpha_{ti} = \text{SoftMax}_i a(h_i, h'_{t-1})$
$c_t = \sum_i \alpha_{ti} W_v h_i$

${W_q}_{d \times dim(h')}, {W_k}_{d \times dim(h)}, {W_v}_{d \times dim(h)}$ — linear neuron weight matrices, possible simplification $W_k \equiv W_v$



Dichao Hu. An introductory survey on attention mechanisms in NLP problems. 2018

Let's discuss in detail the math inside attention. The first one is Similarity Functions of two vectors. Of course in the simplest case it could be the scalar (inner) product. Also you could use the trainable parameters and our favorite linear transformation layer with non-linear sigmoid function and weights vector. But the most common procedure is to have query, key, value parameters matrices.

Attention formula

q is the query vector for which we want to calculate the context
$K = (k_1, \dots, k_n)$ — key vectors compared with the query
$V = (v_i, \dots, v_n)$ — value vectors forming the context
$a(k_i, q)$ — score of relevance (similarity) of key k_i to query q
c is the desired context vector relevant to the query

Attention Model

This is a 3-layer network that computes a convex combination of v_i values relevant to the query q:

$$c = Attn(q, K, V) = \sum_i v_i \mathrm{SoftMax}_i a(k_i, q)$$

$c_t = Attn(W_q h'_{t-1}, W_k H, W_v H)$ is the example from the previous slide, where $H = (h_1, \dots, h_n)$ are input vectors, h'_{t-1} is output
Self-attention:
$c_i = Attn(W_q h_i, W_k H, W_v H)$ is a special case when $h' \in H$

In general case we could say, that it is three-layer network, and attention mechanism chooses the best way to store information from the input to the one output vector c. And one important case needs to be highlighted here. It is called self-attention.

Multi-Head Attention

Idea: J different attention models are jointly trained to highlight various aspects of the input information (for example, parts of speech, syntax, idioms):
$c^j = \text{Attn}(W_q^j q, W_k^j H, W_v^j H), j = 1, \dots, J$
Variants of aggregating the output vector:
$c = \frac{1}{J} \sum_{j=1}^{J} c^j$ — averaging
$c = [c^1 \dots c^J]$ — concatenation
$c = [c^1 \dots c^J]W$ — to return to the desired dimension
Regularization: to make aspects of attention as different as possible, rows $J \times n$ of matrices $A, \alpha_{ji} = \text{SoftMax}_i a(W_k^j h_i, W_q^j q)$, decorrelated ($\alpha_s^T \alpha_j \to 0$) and sparse ($\alpha_j^T \alpha_j \to 1$):

$$\|AA^T - I\|^2 \to \min_{\{W_k^j, W_q^j\}}$$

Zhouhan Lin, Y.Bengio et al. A structured self-attentive sentence embedding. 2017

Ok, and the last generalization in this lecture — is multi-head attention. Here you do all the same, but having J different attention models :)

Ok, finally, what is it, who knows?

Additions and remarks

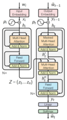
- a lot of such blocks $N = 6$, $h_i \to \Box \to z_i$ are connected in series
- calculations can be easily parallelized in x_i
- it is possible to use pre-trained x_i embeddings
- it is possible to learn embeddings x_i of words $w_i \in V$
- Layer Normalization (LN), $x, \mu, \sigma \in \mathbb{R}^d$

$$LN_s(x; \mu, \sigma) = \mu_s + \sigma_s \frac{x_s - \overline{x}}{\sigma_x}$$

$$\overline{x} = \frac{1}{d}\sum_{s=1}^{d} x_s, \sigma_x^2 = \frac{1}{d}\sum_{s=1}^{d}(x_s - \overline{x})$$

Layer Normalization is quite easy — you train the mean and variance values of components of your vectors

Architecture of Transformer Decoder



Autoregressive sequence synthesis
$y_0 = \langle BOS \rangle$ — start symbol embedding;
for all $t = 1, 2, \ldots$:

1. masking data "from the future": $h_t = y_{t-1} + p_t;\ H_t = (h_1, \ldots, h_t)$
2. multi-head self-attention: $h'_t = LN_{sc} \circ MH_j \circ Attn(W^j_q h_t, W^j_k H_t, W^j_v H_t)$
3. multi-head attention on the Z vectors: $h''_t = LN_{sc} \circ MH_j \circ Attn(\tilde{W}^j_q h'_t, \tilde{W}^j_k Z, \tilde{W}^j_v Z)$
4. two-layer neural network: $y_t = LN_{sc} \circ FFN(h''_t)$
5. linear predictive layer: $p(\tilde{w}|t) = SoftMax_{\tilde{w}}(W_y y_t + b_y)$
6. generation $\tilde{w}_t = \arg\max_{\tilde{w}} p(\tilde{w}|t)$ **until** $\tilde{w}_t \neq \langle EOS \rangle$

Let's move on to the second part of transformer architecture — the decoder.

Temporary page!

LATEX was unable to guess the total number of pages correctly. A
was some unprocessed data that should have been added to the
this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page
away, because LATEX now knows how many pages to expect for t
document.