Attention mechanisms in deep neural networks

Martin Tutek

February 22, 2018

1 Original attention mechanism and its variants

Essentially, the attention mechanism is a way to find an answer to a query. The query in this problem is a vector of real numbers, while the answer is a linear re-combination of values produced by encoding an input sequence. The attention mechanism is supposed to find the *best* linear re-combination for a given query.

- Query: $q \in \mathbb{R}^{d_q}$
- Keys: $K \in \mathbb{R}^{T \times d_k}$
- Values: $V \in \mathbb{R}^{T \times d_v}$

Where T is the time dimension (variable over different input instances), and $d_{|\cdot|}$ are the dimensions of respective embeddings. Keys and values are mapped one-to-one.

We define the *energy* as a result of a function f mapping from a key and query onto $\mathbb{R}$.

$$e_i = f(q, k_i)$$

The energy values are then normalized with a softmax function to produce a probability distribution over all keys

$$a_i = softmax(e_i)$$

We then use these values as the coefficients of a linear combination over the values

$$att = \sum_i a_i h_i$$

1.1 Usage

Initially (Bahdanau, Cho, and Bengio 2014), the attention mechanism was introduced in a machine translation (sequence to sequence) problem to mitigate the problem of learning long dependencies. After a RNN encoder encoded the input sequence into a sequence of hidden states, we use the current decoder state as the query and the encoder hidden states as both the keys and the values.

Essentially, we find relevant information for the word we are currently translating in the encoded input sequence. For this to work, it is essential that the embedding spaces of the input and output vocabularies are somewhat aligned (similar words in languages should be close together).

Other examples of usage include self-attention and multi-head attention, referenced in a later chapter.

1.2 Variants

1.2.1 MLP attention

(Bahdanau, Cho, and Bengio 2014) f is *parametrized* by a feed-forward neural network (multi-layer perceptron)

$$a_i(q, k) = w_2^T tanh(W_1[q; k_i])$$

- $[q; k]$ are the concatenated query and key
- W_1 is a linear operator
- w_2 is a parameter vector

1.2.2 Bilinear attention

(Luong, Pham, and Manning 2015)

f is parametrized by a matrix $W \in \mathbb{R}^{d_q \times d_k}$ (a bilinear operator)

$$a(q, k) = q^T W k$$

1.2.3 Dot product attention

(Luong, Pham, and Manning 2015)

f is *parameter-free*, however d_q **must** be equal to d_k.

$$a(q, k) = q^T \cdot k$$

2 Extensions

2.1 Self-attention (inter-attention)

(Cheng, Dong, and Lapata 2016) : The LSTMN model *"uses attention to induce relations between tokens"*

Idea: use attention over previous LSTM states (keys, values) with the current LSTM state as the query.

$$a_i^{(t)} = v^T tanh(W_h h_i + W_x x^{(t)} + W_{\hat{h}} \hat{h}^{(t-1)})$$
$$s_i^{(t)} = softmax(a_i^{(t)})$$

Where x^t is the current input, $\hat{h}^{(t-1)}$ the hidden state in the **previous timestep**, v a parameter vector and h_i the previous hidden states (i < t - 1).

Note: only the hidden state h is used in the computation, and not the cell state c!

Then the state vectors (c, h) are updated:

$$\begin{bmatrix} \hat{h}^t \\ \hat{c}^t \end{bmatrix} = \sum_{i=1}^{t-1} s_i^{(t)} \begin{bmatrix} h_i \\ c_i \end{bmatrix} \tag{1}$$

and then replace the un-altered state vectors in further LSTM computations.

Attention fusion: how to use self-attention in a sequence-to-sequence task where a decoder network, along with using self-attention, queries the encoder network (intra-attention).

- **Shallow attention fusion** treats the LSTMN model as a standard LSTM and uses intra-attention on top of it.
- **Deep attention fusion** adds an additional gating mechanism into the LSTM cell update based on intra-attention. Formula in chapter 4. of paper.

2.2 A structured self-attentive sentence embedding

(Lin et al. 2017)

1. Run embedded sentence through BiLSTM
2. Self-attention over the BiLSTM hidden states
3. Use a MLP for a downstream task

Attention:

(1) a MLP attention with a tanh hidden layer as in (1.2.1)
(2) *Matrix attention* – the second weight of the attention MLP is a matrix instead of a vector, resulting in a matrix aggregation instead of a vector

$$A = softmax(W_2 tanh(W_1 H^T))$$

Where the softmax is applied along the second dimension of the input. The MLP has no bias! The matrix A is then multiplied with the matrix of hidden states (of dim $T \times H$), resulting in a sentence embedding matrix $M = AH$

We expect (hope) that the sentence embedding matrix will capture different aspects, however this is not necessary the case since the matrix M can *"suffer from redundancy problems"*. The authors attempt to mitigate this by introducing a regularization penalty term P.

$$P = \left\|(AA^T - I)\right\|_F^2$$

Intermezzo: Frobenius norm:

The Frobenius or Hilbert-Schmidt norm of the matrix A is defined as:

$$\begin{aligned} \|A\|_F &= \sqrt{\sum_i^m \sum_j^n |a_{ij}|^2} \\ &= \sqrt{trace(A^T A)} \\ &= \sqrt{\sum_i^{\min\{m,n\}} \sigma_i^2(A)} \end{aligned} \tag{2}$$

where σ_i is a singular value of A.

Effect of regularization:

The matrix A is row-normalized (each out of r rows should focus on one aspect), and each row represents one module of attention. The matrix AA^T contains the dot products of $\mathbf{a_i}$ and $\mathbf{a_j}$ on the location i, j. Obviously, the matrix AA^T is a

square matrix with diagonal elements equal to one (since $i = j$), therefore the subtraction of the identity matrix.

The remainder of the dot products can be seen as a measure of similarity between two discrete probability distributions over the same discrete space (the input sequence). The dot product of the pdfs produces a number between 0 and 1, 0 meaning the distributions are completely different, and 1 meaning they are equal. Therefore, the larger the overlap between the distributions, the higher the penalty is going to be.

Experiments: Yelp dataset, Age dataset, SNLI (* SNLI model described)

2.3 Attention-over-Attention

(Cui et al. 2016)

- Document, query $\in R^{|\mathbb{D}|\cdot 2h}, R^{|\mathbb{Q}|\cdot 2h}$
- D, Q are sequence lengths of document and query respectively
- Shared embedding spaces for query and document (uses one embedding matrix for their joined vocabulary)
- two BiGRU embed query and document ($h_{doc} \in D \cdot 2h$, $h_{query} \in Q \cdot wh$)
- matrix multiplication over the shared embedding dimension $2d$ produces the `pair-wise matching score`

$$M = h_{doc}^{T} \cdot h_{query} \in R^{DxQ}$$

-

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. "Neural Machine Translation by Jointly Learning to Align and Translate." *arXiv Preprint arXiv:1409.0473*.

Cheng, Jianpeng, Li Dong, and Mirella Lapata. 2016. "Long Short-Term Memory-Networks for Machine Reading." *arXiv Preprint arXiv:1601.06733*.

Cui, Yiming, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. "Attention-over-Attention Neural Networks for Reading Comprehension." *arXiv Preprint arXiv:1607.04423*.

Lin, Zhouhan, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. "A Structured Self-Attentive Sentence

Embedding." *arXiv Preprint arXiv:1703.03130*.

Luong, Minh-Thang, Hieu Pham, and Christopher D Manning. 2015. "Effective Approaches to Attention-Based Neural Machine Translation." *arXiv Preprint arXiv:1508.04025*.