VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Deep Learning

Transformer network

Jan Platoš, Radek Svoboda

March 24, 2024

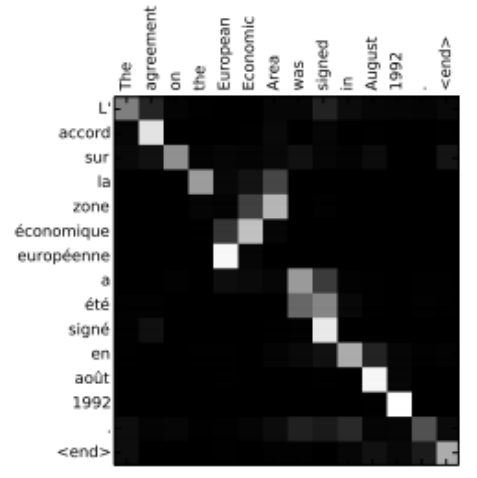
Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB - Technical University of Ostrava

Transformer network

Transformer network

- Recurrent neural network works in synchronized serial way.
- This prevent efficient computation in parallel.
- The attention-mechanism looks at an input sequence and decides at each step which other parts of the sequence are important.
- The attention mechanism helps the decoder to focus on the important part of the input sequence.

Transformer network - Attention



Transformer network - Attention

- The attention allows you to look at the totality of a sentence to make connections between any particular word and its relevant context.
- For each input that the LSTM (Encoder) reads, the attention-mechanism takes into account several other inputs at the same time and decides which ones are important by attributing different weights to those inputs.
- The Decoder will then take as input the encoded sentence and the weights provided by the attention-mechanism.

Transformer network - Attention

- Transformer is a novel architecture that utilize Attention to process Sequence2Sequence tasks.
- It consist on Encoder and Decoder but they are not based on the recurrent connections.
- Encoder and Decoder are composed of modules that can be stacked on top of each other multiple times (Nx).

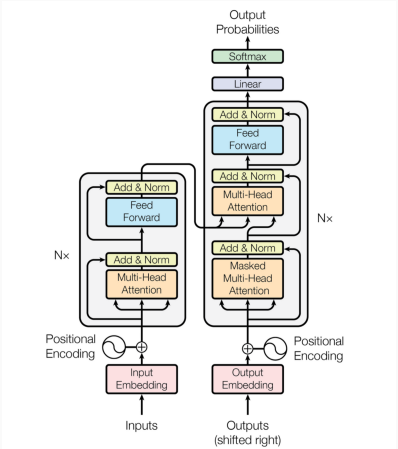


Figure 1: The Transformer - model architecture.

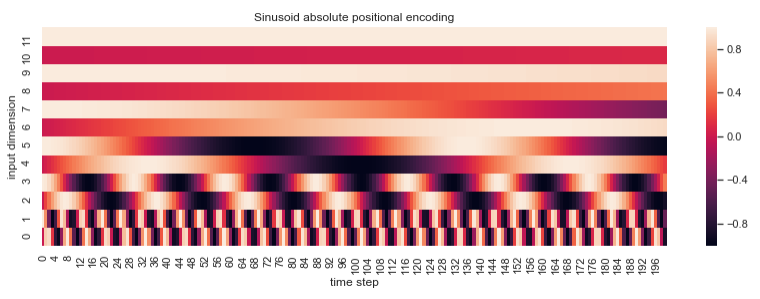
Transformer network - Positional encoding

- The models uses an word embedding for processing.
- Extremely important part is called a **positional encoding**.
- Positional encoding is used to give the order context to the non-recurrent architecture of multi-head attention.
- Positional encoding adds a tensor of the same shape as the input sentence to the encoder and decoder input to maintain the ordering of the words.

Transformer network - Positional encoding

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{model}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right)$$



Transformer network - Multi-head Attention

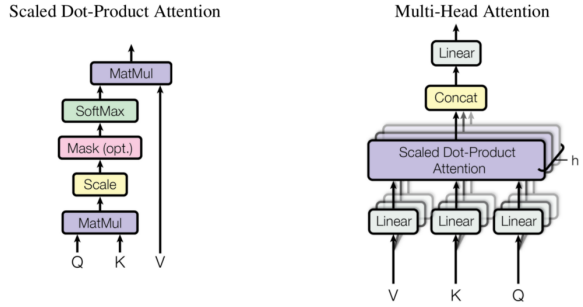


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Transformer network - Multi-head Attention

$$Attention(Q, K, V) = softmax\left(\frac{QK_T}{\sqrt{d_k}}\right)V$$

- Q is a matrix that contains the query (vector representation of one word in the sequence)
- K are all the keys (vector representations of all the words in the sequence).
- V are the values, which are again the vector representations of all the words in the sequence.

Transformer network - Multi-head Attention

- The multi-head attention parallelize the attention mechanism and enrich it with multiple linear projections.
- These linear projection/representations are done by multiplying *Q*, *K* and *V* by weight matrices *W* that are learned during the training.
- Those matrices Q, K and V are different for each position of the attention modules in the structure depending on whether they are in the encoder, decoder or in-between encoder and decoder.
- The reason is that we want to attend on either the whole encoder input sequence or a part of the decoder input sequence.
- The multi-head attention module that connects the encoder and decoder will make sure that the encoder input-sequence is taken into account together with the decoder input-sequence up to a given position.

Transformer network - Training

- It is a Seq2Seq model therefore we need a pair of sentences for training.
- The training procedure uses a principle called *Teacher Enforcing*.
- The output sequence is shifted by single symbol (word or a character).
- We do not want our model to learn how to copy our decoder input during training.
- We want to learn that given the encoder sequence and a particular decoder sequence, which has been already seen by the model, we predict the next word/character.

Transformer network - Training

- If we don't shift the decoder sequence, the model learns to simply 'copy' the decoder input, since the target word/character for position i would be the word/character i in the decoder input.
- By shifting the decoder input by one position, our model needs to predict the target word/character for position i having only seen the word/characters $1, \ldots, i-1$ in the decoder sequence.
- This prevents our model from learning the copy/paste task.
- We fill the first position of the decoder input with a start-of-sentence token, since that place would otherwise be empty because of the right-shift.
- Similarly, we append an end-of-sentence token to the decoder input sequence to mark the end of that sequence and it is also appended to the target output sentence.

Transformer network - Training

- In addition to the right-shifting, the Transformer applies a *mask* to the input in the first multi-head attention module to avoid seeing potential 'future' sequence elements.
- This is specific to the Transformer architecture because we do not have RNNs where we can input our sequence sequentially. Here, we input everything together and if there were no mask, the multi-head attention would consider the whole decoder input sequence at each position.
- The target sequence we want for our loss calculations is simply the decoder input without shifting it and with an end-of-sequence token at the end.

Transformer network - Inference

- Input the full encoder sequence and as decoder input, we take an empty sequence with only a start-of-sentence token on the first position.
- This will output a sequence where we will only take the first element.
- That element will be filled into second position of our decoder input sequence, which now has a start-of-sentence token and a first word/character in it.
- Input both the encoder sequence and the new decoder sequence into the model. Take the second element of the output and put it into the decoder input sequence.
- Repeat this until you predict an end-of-sentence token, which marks the end of the translation.

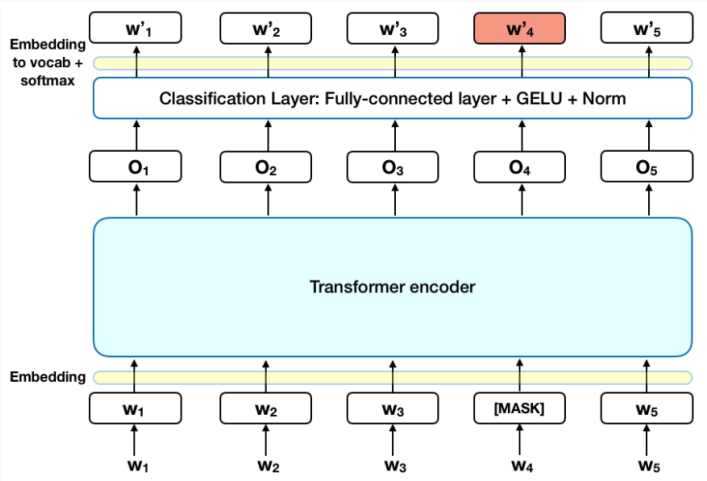
Transformer network - BERT

- BERT - Bidirectional Encoder Representations from Transformers.
- Uses a Transformer-based network with pre-trained deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers.
- BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text.
- Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary.

Transformer network - BERT

- As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once.
- Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).
- The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network.
- The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index.
- When training language models, there is a challenge of defining a prediction goal.

Transformer network - BERT



Transformer network - BERT - Masked LM

- Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token.
- The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.
- In technical terms, the prediction of the output words requires:
  1. Adding a classification layer on top of the encoder output.
  2. Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
  3. Calculating the probability of each word in the vocabulary with softmax.
- The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words.

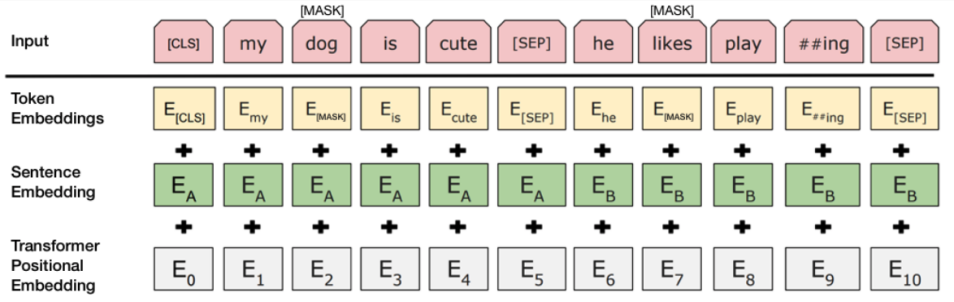
Transformer network - BERT - Next Sentence Prediction (NSP)

- In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document.
- During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence.
- The assumption is that the random sentence will be disconnected from the first sentence.

Transformer network - BERT - Next Sentence Prediction (NSP)

- To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:
  1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
  2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
  3. A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.

Transformer network - BERT - Next Sentence Prediction (NSP)



Transformer network - BERT - Next Sentence Prediction (NSP)

- To predict if the second sentence is indeed connected to the first, the following steps are performed:
  1. The entire input sequence goes through the Transformer model.
  2. The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
  3. Calculating the probability of IsNextSequence with softmax.
- When training the BERT model, Masked LM and Next Sentence Prediction are trained together, with the goal of minimizing the combined loss function of the two strategies.

Transformer network - BERT - How to use BERT (Fine-tuning)

- BERT can be used for a wide variety of language tasks, while only adding a small layer to the core model:
  1. Classification tasks such as sentiment analysis are done similarly to Next Sentence classification, by adding a classification layer on top of the Transformer output for the [CLS] token.
  2. In Question Answering tasks (e.g. SQuAD v1.1), the software receives a question regarding a text sequence and is required to mark the answer in the sequence. Using BERT, a Q&A model can be trained by learning two extra vectors that mark the beginning and the end of the answer.
  3. In Named Entity Recognition (NER), the software receives a text sequence and is required to mark the various types of entities (Person, Organization, Date, etc) that appear in the text. Using BERT, a NER model can be trained by feeding the output vector of each token into a classification layer that predicts the NER label.

Transformer network - References i

1. What is a Transformer? by Maxime, https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04
2. A Beginner's Guide to Attention Mechanisms and Memory Networks https://wiki.pathmind.com/attention-mechanism-memory-network
3. What is Teacher Forcing for Recurrent Neural Networks? by Jason Brownlee https://machinelearningmastery.com/teacher-forcing-for-recurrent-neural-networks/
4. Positional Encoding: Everything You Need to Know by Darjan Salaj https://www.inovex.de/de/blog/positional-encoding-everything-you-need-to-know

Transformer network - References ii

5. Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. ”Effective approaches to attention-based neural machine translation.” arXiv preprint arXiv:1508.04025 (2015).
6. Devlin, Jacob, et al. ”Bert: Pre-training of deep bidirectional transformers for language understanding.” arXiv preprint arXiv:1810.04805 (2018).
7. BERT Explained: State of the art language model for NLP by Rani Horev `https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a`
8. Understanding BERT Transformer: Attention isn’t all you need by Damien Sileo `https://medium.com/synapse-dev/understanding-bert-transformer-attention-isnt-all-you-need-58`

Questions?