Lesson 18

- Attention
- Transformers

Next argument

Attention

Attention

Attention is the ability to focus on different parts of the input, according to the requirements of the problem being solved.

It is an **essential** component of any intelligent behaviour, with potential application to a wide range of domains.

A differential mechanism

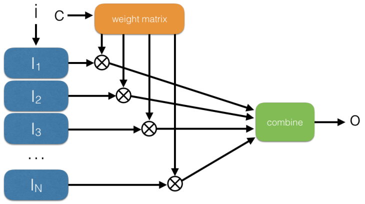
From the point of view of Neural Networks, we would expect the attention mechanism to be **differentiable**, so that we can learn where to focus by standard **backpropagation techniques**.

The current approach (not necessarily the best one) is to **focus everywhere**, just to **different extents**.

Attention as gating maps

Attention mechanisms can be implemented as gating functions.

The gating maps are dynamically generated by some neural net, allowing to focus on different part on the input at (e.g.) different times.



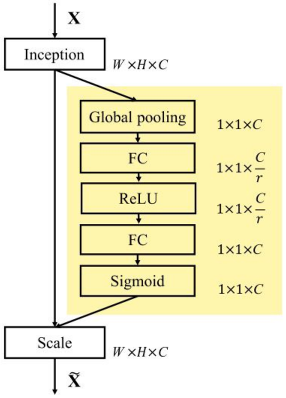
Picture from The fall of RNN / LSTM by E.Colurciello

The forget map, input map and output map in LSTMs are examples of attention mechanisms.

Another example: squeeze and excitation

SE layers are a building component of Squeeze and Excitation Networks

SE layers implement a form of self attention, allowing to focus on particular channels in a dinamical way, according to the input under consideration.



SE-Inception Module

A modular multi purpose layer

The most typical attention layer is based on the **key-value** paradigm, implementing a sort of associative memory.

We access this memory with **queries** to be matched with keys.

The resulting **scores** generate a boolean map that is used to weight values.

Attention: the key-value approach

For each key k_i compute the **scores** a_i as

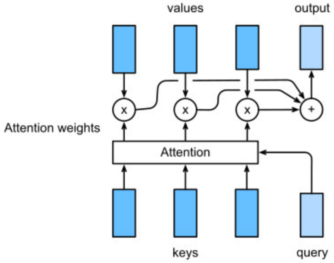
$$a_i = \alpha(q, k_i)$$

obtain **attention weights** via softmax:

$$\vec{b} = softmax(\vec{a})$$

retrun a weighted sum of the values:

$$o = \sum_{i=1}^{n} b_i v_i$$



In many applications, values are also used as keys (self-attention).

Typical score functions

Different score functions lead to different attention layers.

Two commonly used approaches are:

- Dot product.

  $$\alpha(q,k) = q \cdot k/\sqrt(d)$$

  The query and the key must have the same dimension d
- MLP: α is computed by a neural network (usually composed by a single layer):

  $$\alpha(k,q) = tanh(W_k\vec{k} + W_q\vec{q})$$

See Attention layer for the Keras implementation of this layer.

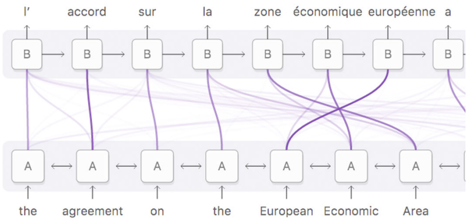
An application to translation

Neural Machine Translation by jointly learning to align and to translate, D.Bahdanau, K.Cho, Y.Bengio (2015)

alignment	identify which parts of the input sequence are relevant to each word in the output
translation	is the process of using the relevant information to select the appropriate output.

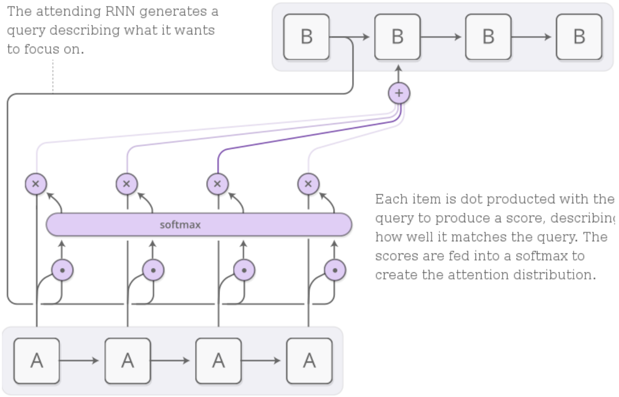
Attention to alignement

Alignement is a form of attention!



Picture from Attention and Augmented Recurrent Neural Networks, by C.Olah and S.Carter.

Producing attention maps


The attending RNN generates a query describing what it wants to focus on.
B
B
B
B
+
×
×
×
×
softmax
Each item is dot producted with the query to produce a score, describing how well it matches the query. The scores are fed into a softmax to create the attention distribution.
A
A
A
A

Picture from Attention and Augmented Recurrent Neural Networks.

Producing attention maps

"The decoder decides parts of the source sentence to pay attention to. By letting the decoder have an attention mechanism, we relieve the encoder from the burden of having to encode all information in the source sentence into a fixed- length vector. With this new approach the information can be spread throughout the sequence of annotations, which can be selectively retrieved by the decoder accordingly".

Attention and Augmented Recurrent Neural Networks.

Next Argument

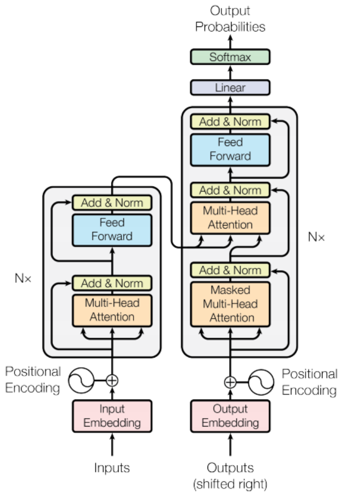
Transformers

Transformers

Transformers have been introduced in Attention is All You Need, one of the most influential works of recent years.

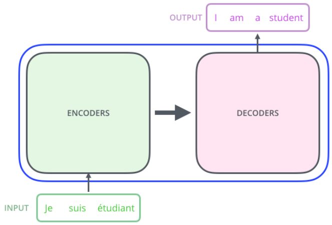
Transformers have rapidly become the model of choice for NLP.

Applications like Bert and GPT, (with all relative families) are based on Transformers.

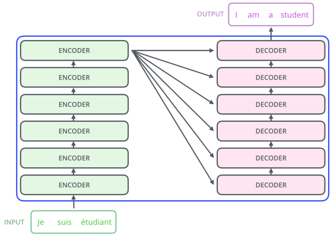


Encoder and decoder

A transformer has a traditional encoder-decoder structure, with connections between them.



The encoding component is a stack of **encoders**. Similarly, the decoding component is a stack of **decoders**.

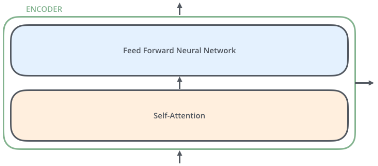


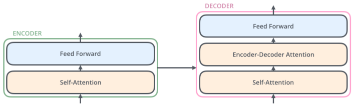
Pictures from The annotated transformer

Encoder and decoder modules

The encoder is organized as a self-attention layer (query, key and value are shared), followed by feedfoward component (a couple of dense layers).

The decoder is similar, with an additional attention layer that helps the decoder to focus on relevant parts of the input sentence.

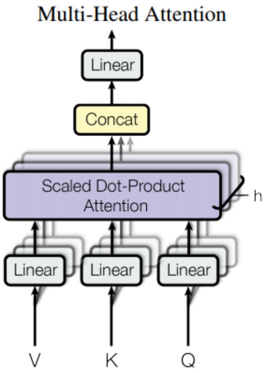




Multi head attention

Using multiple heads for attention expands the model's ability to focus on different positions, for different purposes.

As a result, multiple "representation subspaces" are created, focusing on potentially different aspects of the input sequence.



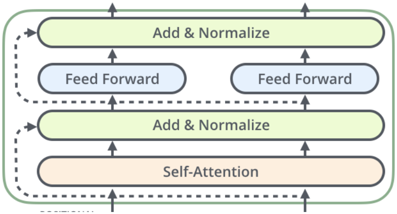
Masking the future

Typically, in a Transformer module, we can apply a boolean mask to the input, to hide part of its content.

This is frequently used in the decoder to prevent it to attend at future positions during generation.

Residual connections

Each sub-layer (self-attention, ffnn) in each encoder has a residual connection around it, and it is followed by a layer-normalization step.



Positional encoding

Positional encoding is added to word embeddings to give the model some information about the **relative position** of the words in the sentence.

The positional information is a vector of the same dimensions d_{model}, of the word embedding.

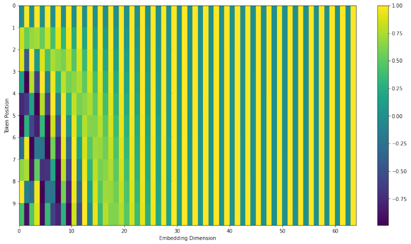
The authors use sine and cosine functions of different frequencies:

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

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

Sinusoids at work

According to the previous encoding, each dimension i of the PE vector corresponds to a sinusoid, where the wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$.



"We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a **linear function** *of PE_{pos}."*

The annotated transformer

Additional links

- Attention is all you need
- The annotated transfomer
- Tensorflow transformer tutorial
- D2L lesson on transformers
- The illustrated Transfomer
- The positional encoding