Lecture 10. Attention Models and Transformers

Alex Avdyushenko

Kazakh-British Technical University

November 12, 2022

Five-minutes block

Five-minutes block

- 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?

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 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

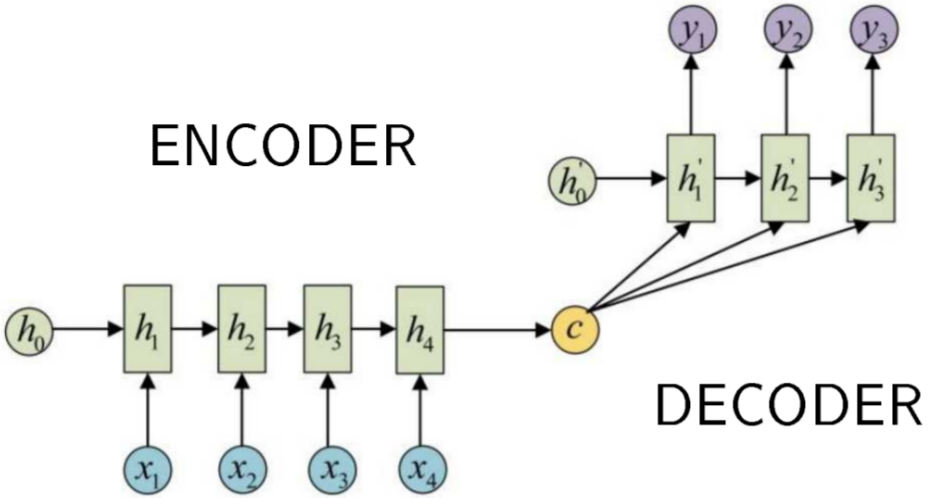
RNN for sequence synthesis (seq2seq)

Recall

$X = (x_1, \ldots, x_n)$ — input sequence

$Y = (y_1, \ldots, y_m)$ — output sequence

$c \equiv h_n$ encodes all information about X to synthesize Y



$$h_i = f_{in}(x_i, h_{i-1})$$

$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$

$$y_t = f_y(h'_t, y_{t-1})$$

$$h_i = f_{in}(x_i, h_{i-1})$$
$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$
$$y_t = f_y(h'_t, y_{t-1})$$

$$h_i = f_{in}(x_i, h_{i-1})$$
$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$
$$y_t = f_y(h'_t, y_{t-1})$$

Disadvantages

- c remembers the end (h_n) better than the start
- the more n, the more difficult to pack all the information into vector c
- we should control the vanishing and explosions of the gradient
- RNN is difficult to parallelize

$$h_i = f_{in}(x_i, h_{i-1})$$
$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$
$$y_t = f_y(h'_t, y_{t-1})$$

Disadvantages

- c remembers the end (h_n) better than the start
- the more n, the more difficult to pack all the information into vector c
- we should control the vanishing and explosions of the gradient
- RNN is difficult to parallelize

Question

How can you fix some of the problems above?

$$h_i = f_{in}(x_i, h_{i-1})$$
$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c)$$
$$y_t = f_y(h'_t, y_{t-1})$$

Disadvantages

- c remembers the end (h_n) better than the start
- the more n, the more difficult to pack all the information into vector c
- we should control the vanishing and explosions of the gradient
- RNN is difficult to parallelize

Question

How can you fix some of the problems above?

Hint

How do people perceive information?

Let's count the number of passes

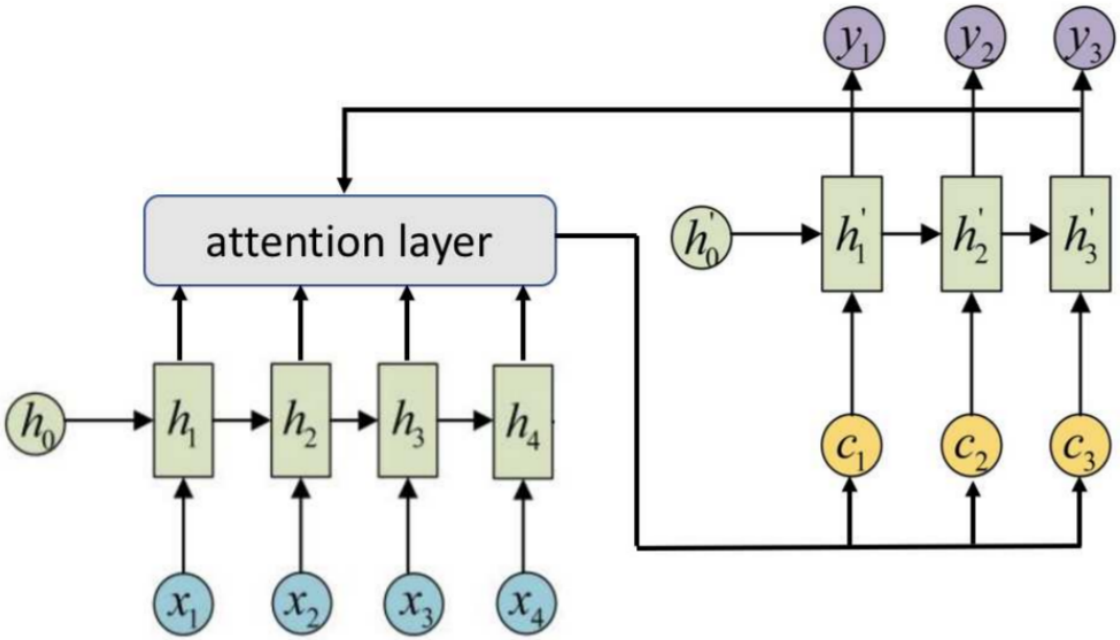
Basketball!

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



$$h_i = f_{in}(x_i, h_{i-1})$$

$$\alpha_{ti} = \text{norm}_i a(h_i, h'_{t-1}), \;\; \text{norm}_i(p) = \frac{p_i}{\sum_j p_j}$$

$$c_t = \sum_i \alpha_{ti} h_i$$

$$h'_t = f_{out}(h'_{t-1}, y_{t-1}, c_t)$$

$$y_t = f_y(h'_t, y_{t-1}, c_t)$$

- you can enter learnable parameters in a and c_t
- it is possible to refuse recurrence both in h_i and in h'_t

---

Bahdanau et al. Neural machine translation by jointly learning to align and translate. 2015

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

The Main Areas for Attention Models

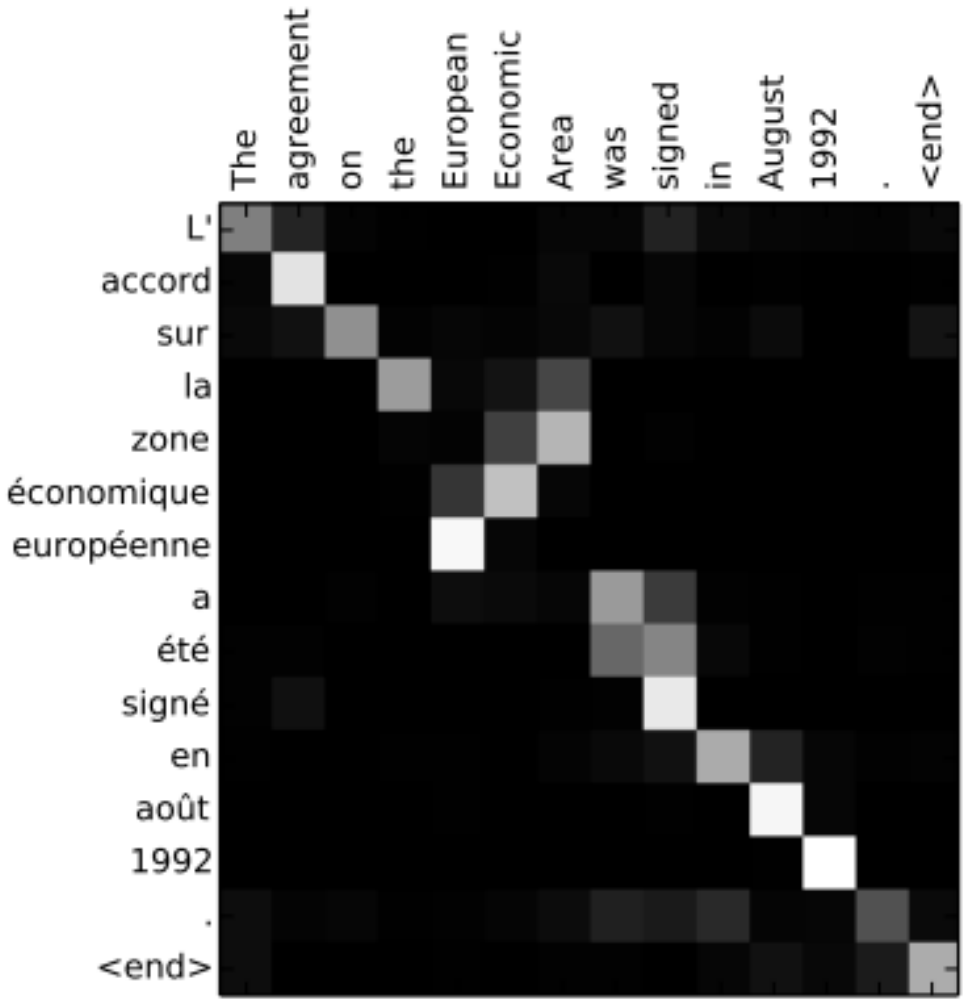
Converting one sequence to another, i.e. seq2seq

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

Sequence processing

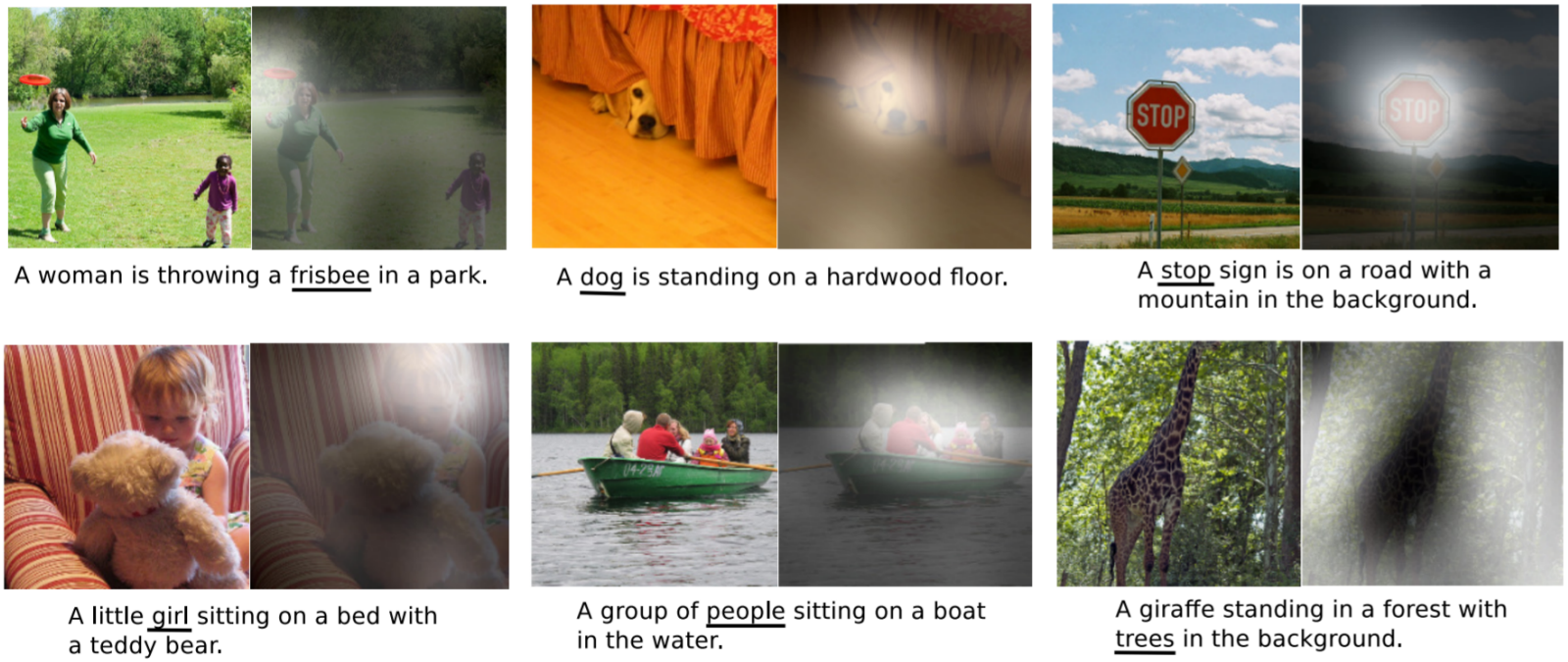
- Classification of text documents
- Document Sentiment Analysis

Attention in machine translation



Attention Model Interpretability: Visualization of α_{ti}

Attention for annotating images



When generating a word for an image description, the visualization shows which areas of the image the model pays attention to.

Kelvin Xu et al. Show, attend and tell: neural image caption generation with visual attention. 2016

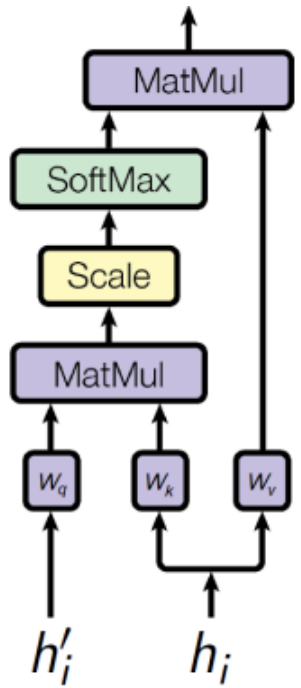
Vector Similarity Functions

$a(h, h') = h^T h'$ 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

Attention formula

q is the query vector for which we want to calculate the context
$K = (k_1, \ldots, k_n)$ — key vectors compared with the query
$V = (v_i, \ldots, v_n)$ — 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 formula

q is the query vector for which we want to calculate the context
$K = (k_1, \ldots, k_n)$ — key vectors compared with the query
$V = (v_i, \ldots, v_n)$ — 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:

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

Attention formula

q is the query vector for which we want to calculate the context
$K = (k_1, \ldots, k_n)$ — key vectors compared with the query
$V = (v_i, \ldots, v_n)$ — 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:

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

$c_t = Attn(W_q h'_{t-1}, W_k H, W_v H)$ is the example from the previous slide, where $H = (h_1, \ldots, h_n)$ are input vectors, h'_{t-1} is output

Attention formula

q is the query vector for which we want to calculate the context
$K = (k_1, \ldots, k_n)$ — key vectors compared with the query
$V = (v_i, \ldots, v_n)$ — 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:

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

$c_t = Attn(W_q h'_{t-1}, W_k H, W_v H)$ is the example from the previous slide, where $H = (h_1, \ldots, h_n)$ 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 $h' \in H$

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):

$c^j = \text{Attn}(W_q^j q, W_k^j H, W_v^j H), j = 1, \ldots, J$

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):

$c^j = \text{Attn}(W_q^j q, W_k^j H, W_v^j H), j = 1, \ldots, J$

Variants of aggregating the output vector:

$c = \frac{1}{J} \sum_{j=1}^{J} c^j$ — averaging

$c = [c^1 \ldots c^J]$ — concatenation

$c = [c^1 \ldots c^J] W$ — to return to the desired dimension

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):

$c^j = \text{Attn}(W_q^j q, W_k^j H, W_v^j H), j = 1, \ldots, J$

Variants of aggregating the output vector:

$c = \frac{1}{J} \sum_{j=1}^{J} c^j$ — averaging

$c = [c^1 \ldots c^J]$ — concatenation

$c = [c^1 \ldots c^J] W$ — 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

Not covered in detail in this lecture

- hierarchical attention (e.g. for classifying documents): words $\in$ sentences $\in$ documents
- Graph Attention Network (GAT): multi-class multi-label classification of graph vertices

---

Z.Yang, A.Smola et al. Hierarchical attention networks for document classification. 2016

Petar Velickovic et al. Graph Attention Networks. ICLR-2018

Transformer for machine translation

Transformer is a neural network architecture based on attention models and fully connected layers, **without RNN**

Transformer for machine translation

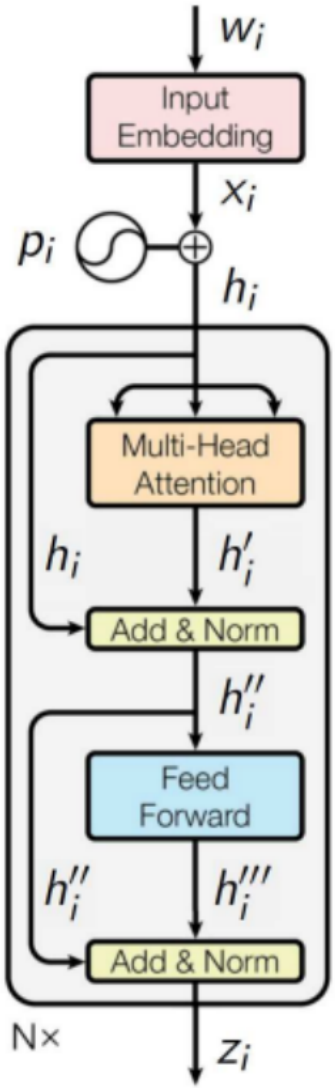
Transformer is a neural network architecture based on attention models and fully connected layers, **without RNN**
Scheme of data transformations in machine translation:

- $S = (w_1, \ldots, w_n)$ — sentence from words in the input language
  $\downarrow$ trainable or pre-trained word vectorization $\downarrow$
- $X = (x_1, \ldots, x_n)$ — embeddings of input sentence words
  $\downarrow$ encoder transformer $\downarrow$
- $Z = (z_1, \ldots, z_n)$ — contextual embeddings
  $\downarrow$ transformer-decoder $\downarrow$
- $Y = (y_1, \ldots, y_m)$ — embeddings of output sentence words
  $\downarrow$ generation of words from the constructed language model $\downarrow$
- $\tilde{S} = (\tilde{w}_1, \ldots, \tilde{w}_m)$ — sentence words in target language

---

Vaswani et al. (Google) Attention is all you need. 2017

Architecture of Transformer Encoder



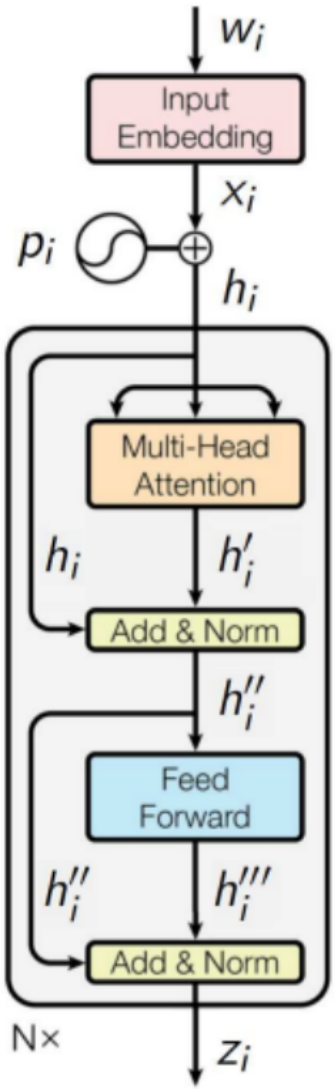
1. position vectors p_i are added:

$$h_i = x_i + p_i, \quad H = (h_1, \ldots, h_n)$$

$d=dim\ \ x_i, p_i, h_i=512$

$dim\ \ H=512\times n$

Architecture of Transformer Encoder



1. position vectors p_i are added:

$$h_i = x_i + p_i, \quad H = (h_1, \ldots, h_n)$$

$d{=}dim\ \ x_i, p_i, h_i{=}512$

$dim\ \ H{=}512 \times n$

2. multi-head self-attention:

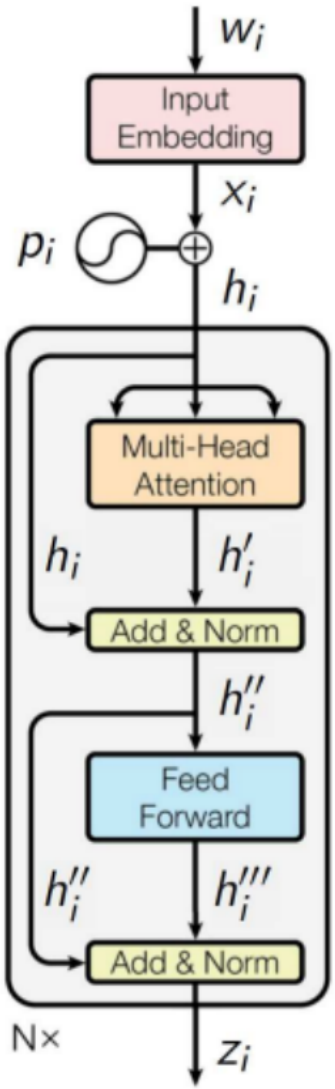
$$h_i^j = Attn(W_q^j h_i, W_k^j H, W_v^j H)$$

$j{=}1,\ldots,J{=}8$

$dim\ \ h_i^j{=}64$

$dim\ \ W_q^j, W_k^j, W_v^j{=}64 \times 512$

Architecture of Transformer Encoder



1. position vectors p_i are added:

$$h_i = x_i + p_i, \;\; H = (h_1, \ldots, h_n)$$

$d{=}dim\;\; x_i, p_i, h_i{=}512$

$dim\;\; H{=}512 \times n$

2. multi-head self-attention:

$$h_i^j = Attn(W_q^j h_i, W_k^j H, W_v^j H)$$

$j{=}1,\ldots,J{=}8$

$dim\;\; h_i^j{=}64$

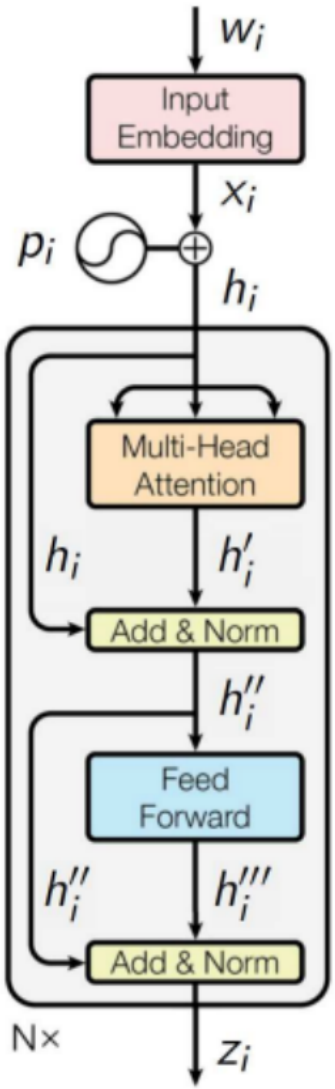
$dim\;\; W_q^j, W_k^j, W_v^j{=}64 \times 512$

3. concatenation:

$$h'_i = MH_j(h_i^j) \equiv [h_i^1 \ldots h_i^J]$$

$dim\;\; h'_i{=}512$

Architecture of Transformer Encoder



1. position vectors p_i are added:

$$h_i = x_i + p_i, \;\; H = (h_1, \ldots, h_n)$$

$d{=}dim\;\; x_i, p_i, h_i{=}512$

$dim\;\; H{=}512 \times n$

2. multi-head self-attention:

$$h_i^j = Attn(W_q^j h_i, W_k^j H, W_v^j H)$$

$j{=}1,\ldots,J{=}8$

$dim\;\; h_i^j{=}64$

$dim\;\; W_q^j, W_k^j, W_v^j{=}64 \times 512$

3. concatenation:

$$h'_i = MH_j(h_i^j) \equiv [h_i^1 \ldots h_i^J]$$

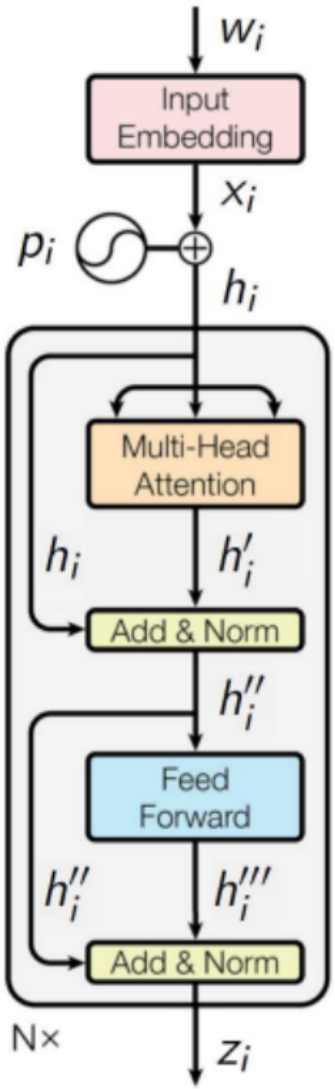
$dim\;\; h'_i{=}512$

4. Skip-connection and layer normalization:

$$h''_i = LN(h'_i + h_i; \mu_1, \sigma_1)$$

$dim\;\; h''_i, \mu_1, \sigma_1{=}512$

Architecture of Transformer Encoder



1. position vectors p_i are added:

$$h_i = x_i + p_i, \;\; H = (h_1, \ldots, h_n)$$

$d{=}dim\;\; x_i, p_i, h_i{=}512$

$dim\;\; H{=}512 \times n$

2. multi-head self-attention:

$$h_i^j = Attn(W_q^j h_i, W_k^j H, W_v^j H)$$

$j{=}1,\ldots,J{=}8$

$dim\;\; h_i^j{=}64$

$dim\;\; W_q^j, W_k^j, W_v^j{=}64 \times 512$

3. concatenation:

$$h'_i = MH_j(h_i^j) \equiv [h_i^1 \ldots h_i^J]$$

$dim\;\; h'_i{=}512$

4. Skip-connection and layer normalization:

$$h''_i = LN(h'_i + h_i; \mu_1, \sigma_1)$$

$dim\;\; h''_i, \mu_1, \sigma_1{=}512$

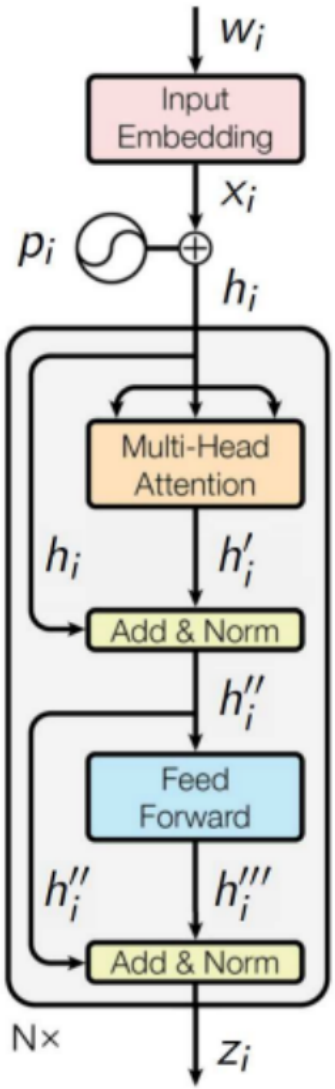
5. Fully connected two-layer network FFN:

$$h'''_i = W_2 ReLU(W_1 h''_i + b_1) + b_2$$

$dim\;\; W_1{=}2048 \times 512$

$dim\;\; W_2{=}512 \times 2048$

Architecture of Transformer Encoder



1. position vectors p_i are added:

$$h_i = x_i + p_i, \;\; H = (h_1, \ldots, h_n)$$

$d{=}dim\;\; x_i, p_i, h_i{=}512$

$dim\;\; H{=}512 \times n$

2. multi-head self-attention:

$$h_i^j = Attn(W_q^j h_i, W_k^j H, W_v^j H)$$

$j{=}1,\ldots,J{=}8$

$dim\;\; h_i^j{=}64$

$dim\;\; W_q^j, W_k^j, W_v^j{=}64 \times 512$

3. concatenation:

$$h'_i = MH_j(h_i^j) \equiv [h_i^1 \ldots h_i^J]$$

$dim\;\; h'_i{=}512$

4. Skip-connection and layer normalization:

$$h''_i = LN(h'_i + h_i; \mu_1, \sigma_1)$$

$dim\;\; h''_i, \mu_1, \sigma_1{=}512$

5. Fully connected two-layer network FFN:

$$h'''_i = W_2 ReLU(W_1 h''_i + b_1) + b_2$$

$dim\;\; W_1{=}2048 \times 512$

$dim\;\; W_2{=}512 \times 2048$

6. Skip-connection and layer normalization:

$$z_i = LN(h'''_i + h''_i; \mu_2, \sigma_2)$$

$dim\;\; z_i, \mu_2, \sigma_2{=}512$

Additions and remarks

- a lot of such blocks $N = 6, h_i \to \square \to z_i$ 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), $x, \mu, \sigma \in \mathbb{R}^d$

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

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

Positional encoding

The positions of the words i are encoded by the vectors p_i so that

- the more $|i - j|$, the more $\|p_i - p_j\|$
- number of positions is not limited

Positional encoding

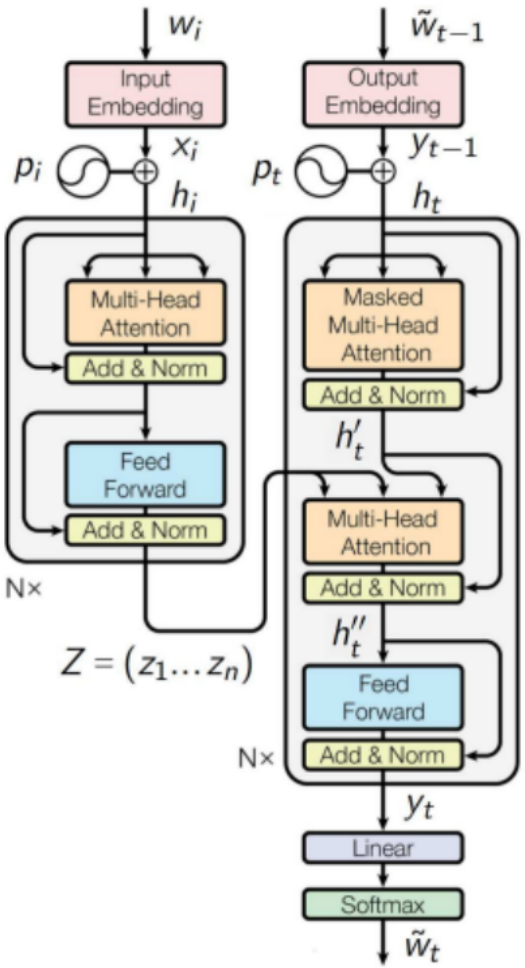
The positions of the words i are encoded by the vectors p_i so that

- the more $|i - j|$, the more $\|p_i - p_j\|$
- number of positions is not limited

For example,
$c_j = \text{Attn}(q_j, K, V) = \sum_i (v_i + w^v_{i \boxminus j}) \text{SoftMax}_i a(k_i + w^k_{i \boxminus j}, q_j)$,
where $i \boxminus j = \max(\min(i - j, \delta), -\delta)$ is the truncated difference, $\delta = 5..16$

Shaw, Uszkoreit, Vaswani. Self-attention with relative position representations. 2018

Architecture of Transformer Decoder



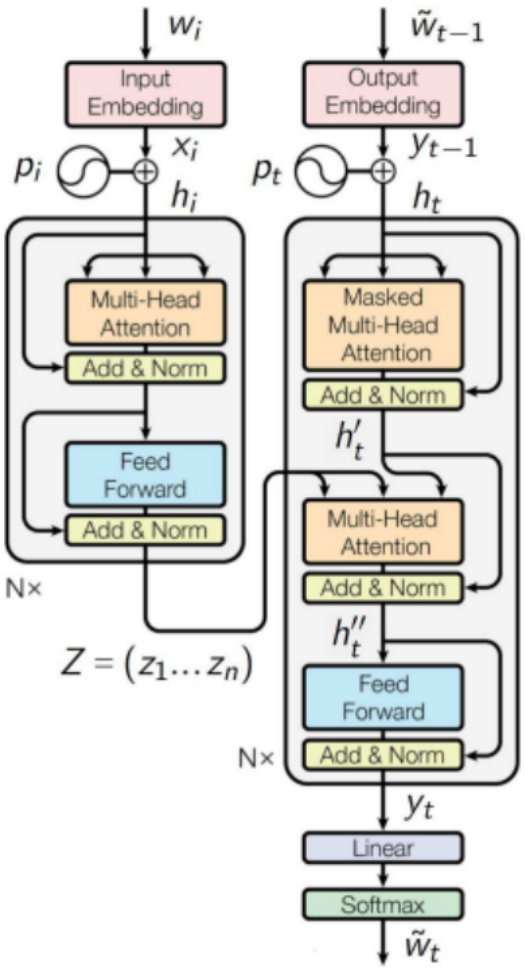
Autoregressive sequence synthesis

$y_0 = \langle BOS \rangle$ — start symbol embedding;

for all $t = 1, 2, \dots$:

1. masking data "from the future":
   $h_t = y_{t-1} + p_t$; $H_t = (h_1, \dots, h_t)$

Architecture of Transformer Decoder



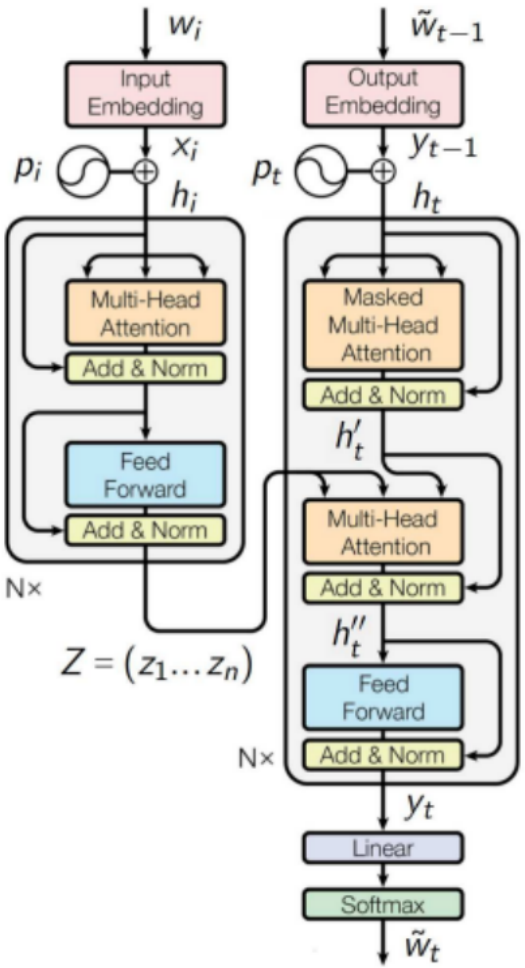
Autoregressive sequence synthesis

$y_0 = \langle BOS \rangle$ — start symbol embedding;

for all $t = 1, 2, \dots$:

1. masking data "from the future":
   $h_t = y_{t-1} + p_t;\ \ H_t = (h_1, \dots, h_t)$
2. multi-head self-attention:
   $h'_t = LN_{sc} \circ MH_j \circ Attn(W_q^j h_t, W_k^j H_t, W_v^j H_t)$

Architecture of Transformer Decoder



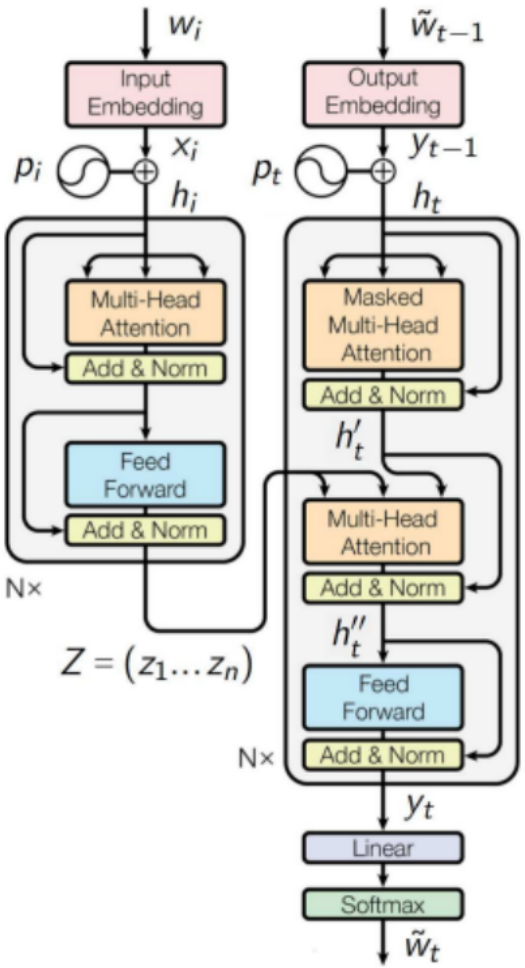
Autoregressive sequence synthesis

$y_0 = \langle BOS \rangle$ — start symbol embedding;

for all $t = 1, 2, \dots$:

1. masking data "from the future":
   $h_t = y_{t-1} + p_t;\ \ H_t = (h_1, \dots, h_t)$
2. multi-head self-attention:
   $h'_t = LN_{sc} \circ MH_j \circ Attn(W_q^j h_t, W_k^j H_t, W_v^j H_t)$
3. multi-head attention on the Z vectors:
   $h''_t = LN_{sc} \circ MH_j \circ Attn(\tilde{W}_q^j h'_t, \tilde{W}_k^j Z, \tilde{W}_v^j Z)$

Architecture of Transformer Decoder



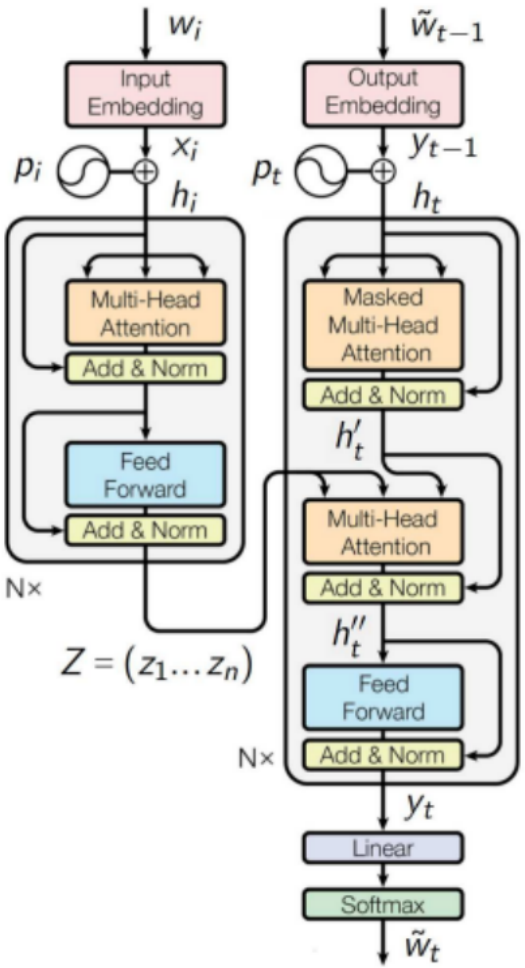
Autoregressive sequence synthesis

$y_0 = \langle BOS \rangle$ — start symbol embedding;

for all $t = 1, 2, \dots$:

1. masking data "from the future":
   $h_t = y_{t-1} + p_t;\ \ H_t = (h_1, \dots, h_t)$
2. multi-head self-attention:
   $h'_t = LN_{sc} \circ MH_j \circ Attn(W_q^j h_t, W_k^j H_t, W_v^j H_t)$
3. multi-head attention on the Z vectors:
   $h''_t = LN_{sc} \circ MH_j \circ Attn(\tilde{W}_q^j h'_t, \tilde{W}_k^j Z, \tilde{W}_v^j Z)$
4. two-layer neural network:
   $y_t = LN_{sc} \circ FFN(h''_t)$

Architecture of Transformer Decoder



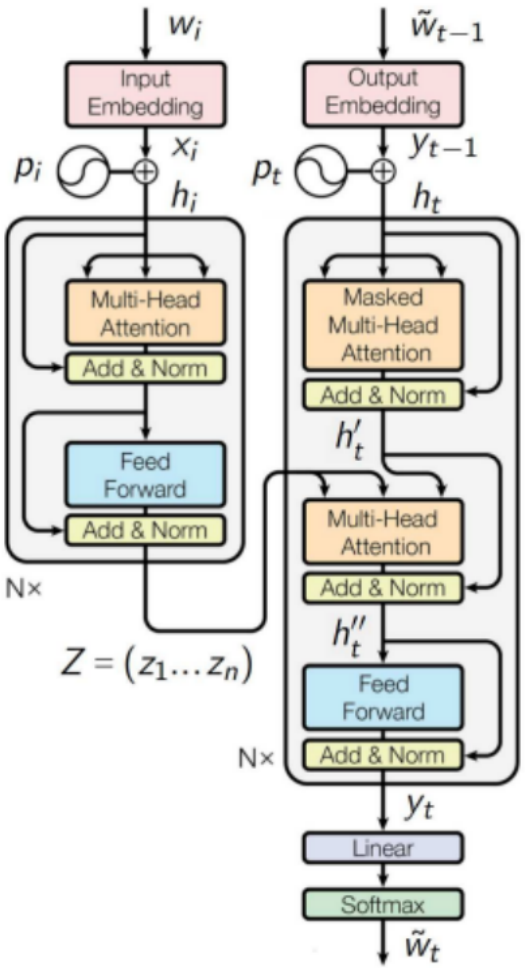
Autoregressive sequence synthesis

$y_0 = \langle BOS \rangle$ — start symbol embedding;

for all $t = 1, 2, \dots$:

1. masking data "from the future":
   $h_t = y_{t-1} + p_t; \;\; H_t = (h_1, \dots, h_t)$
2. multi-head self-attention:
   $h'_t = LN_{sc} \circ MH_j \circ Attn(W_q^j h_t, W_k^j H_t, W_v^j H_t)$
3. multi-head attention on the Z vectors:
   $h''_t = LN_{sc} \circ MH_j \circ Attn(\tilde{W}_q^j h'_t, \tilde{W}_k^j Z, \tilde{W}_v^j Z)$
4. two-layer neural network:
   $y_t = LN_{sc} \circ FFN(h''_t)$
5. linear predictive layer:
   $p(\tilde{w}|t) = SoftMax_{\tilde{w}}(W_y y_t + b_y)$

Architecture of Transformer Decoder



Autoregressive sequence synthesis

$y_0 = \langle BOS \rangle$ — start symbol embedding;

for all $t = 1, 2, \dots$:

1. masking data "from the future":
   $h_t = y_{t-1} + p_t;\ \ H_t = (h_1, \dots, h_t)$
2. multi-head self-attention:
   $h'_t = LN_{sc} \circ MH_j \circ Attn(W_q^j h_t, W_k^j H_t, W_v^j H_t)$
3. multi-head attention on the Z vectors:
   $h''_t = LN_{sc} \circ MH_j \circ Attn(\tilde{W}_q^j h'_t, \tilde{W}_k^j Z, \tilde{W}_v^j Z)$
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**
   $\tilde{w}_t \neq \langle EOS \rangle$

Training and validation criteria for machine translation

Criteria for learning parameters of the neural network W on the training set of sentences S with translation $\tilde{S}$:

$$\sum_{(S,\tilde{S})} \sum_{\tilde{w}_t \in \tilde{S}} \ln p(\tilde{w}_t | t, S, W) \to \max_W$$

Training and validation criteria for machine translation

Criteria for learning parameters of the neural network W on the training set of sentences S with translation $\tilde{S}$:

$$\sum_{(S,\tilde{S})} \sum_{\tilde{w}_t \in \tilde{S}} \ln p(\tilde{w}_t | t, S, W) \to \max_W$$

Criteria for evaluating models (non-differentiable) based on a sample of pairs of sentences "translation S, standard S_0":
BiLingual Evaluation Understudy:

$$\text{BLEU} = \min\left(1, \frac{\sum \text{len}(S)}{\sum \text{len}(S_0)}\right) \times$$

$$\underset{(S_0,S)}{\text{mean}} \left(\prod_{n=1}^{4} \frac{\#n\text{-gram from } S, \text{incoming in } S_0}{\#n\text{-gram in } S} \right)^{\frac{1}{4}}$$

BERT — Bidirectional Encoder Representations from Transformers

The BERT transformer is a decoderless encoder that can be trained to solve a wide range of NLP problems.
Scheme of data transformations in NLP tasks:

- $S = (w_1, \ldots, w_n)$ — sentence from words in the input language
  $\downarrow$ learning embeddings with transformer $\downarrow$
- $X = (x_1, \ldots, x_n)$ — embeddings of input sentence words
  $\downarrow$ encoder transformer $\downarrow$
- $Z = (z_1, \ldots, z_n)$ — contextual embeddings
  $\downarrow$ additional training for a specific task $\downarrow$
- Y - output text / markup / classification, etc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (Google AI Language)
BERT: pre-training of deep bidirectional transformers for language understanding. 2019.

MLM (masked language modeling) criterion for BERT training

The masked language modeling criterion is built automatically from texts (self-supervised learning):

$$\sum_{S} \sum_{i \in M(S)} \ln p(w_i | i, S, W) \to \max_{W},$$

where $M(S)$ is a subset of masked tokens from S,

$$p(w | i, S, W) = \underset{w \in V}{\text{SoftMax}}(W_z z_i(S, W_T) + b_z)$$

is a language model that predicts the i-th sentence token S, $z_i(S, W_T)$ is the contextual embedding of the i-th sentence token S at the output of the transformer with parameters W_T,
W — all transformer and language model parameters

NSP (next sentence prediction) criterion for BERT training

The criterion for predicting the relationship between NSP sentences is built automatically from texts (self-supervised learning):

$$\sum_{(S,S')} \ln p\left(y_{SS'}|S,S',W\right) \to \max_{W},$$

where $y_{SS'} = [S \text{ followed by } S']$ is the binary classification of a pair of sentences,

$$p(y|S,S',W) = \underset{y\in\{0,1\}}{\text{SoftMax}}\left(W_y \tanh(W_s z_0(S,S',W_T) + b_s) + b_y\right)$$

— probabilistic classification model for pairs (S,S'), $z_0(S,S',W_T)$ — contextual embedding of token $\langle\text{CLS}\rangle$ for a sentence pair written as $\langle\text{CLS}\rangle\, S\, \langle\text{SEP}\rangle\, S'\, \langle\text{SEP}\rangle$

A few more notes about transformers

- Fine-tuning: model $f(Z(S, W_T), W_f)$, sample $\{S\}$ and criterion $\mathcal{L}(S, f) \to \max$
- Multi-task learning: for additional training on a set of tasks $\{t\}$, models $f_t(Z(S, W_T), W_f)$, samples $\{S\}_t$ and sum of criteria $\sum_t \lambda_t \sum_S \mathcal{L}_t(S, f_t) \to \max$
- GLUE, SuperGLUE, Russian SuperGLUE — sets of test problems for understanding natural language
- Transformers are usually built not on words, but on tokens received by BPE (Byte-Pair Encoding) or WordPiece

A few more notes about transformers

- Fine-tuning: model $f(Z(S, W_T), W_f)$, sample $\{S\}$ and criterion $\mathcal{L}(S, f) \to \max$
- Multi-task learning: for additional training on a set of tasks $\{t\}$, models $f_t(Z(S, W_T), W_f)$, samples $\{S\}_t$ and sum of criteria $\sum_t \lambda_t \sum_S \mathcal{L}_t(S, f_t) \to \max$
- GLUE, SuperGLUE, Russian SuperGLUE — sets of test problems for understanding natural language
- Transformers are usually built not on words, but on tokens received by BPE (Byte-Pair Encoding) or WordPiece
- First transformer: $N = 6, d = 512, J = 8$, weights 65M
- $\text{BERT}_{\text{BASE}}$, GPT-1: $N = 12, d = 768, J = 12$, weights 110M
- $\text{BERT}_{\text{LARGE}}$, GPT-2: $N = 24, d = 1024, J = 16$, weights 340M-1000M

A few more notes about transformers

- Fine-tuning: model $f(Z(S, W_T), W_f)$, sample $\{S\}$ and criterion $\mathcal{L}(S, f) \to \max$
- Multi-task learning: for additional training on a set of tasks $\{t\}$, models $f_t(Z(S, W_T), W_f)$, samples $\{S\}_t$ and sum of criteria $\sum_t \lambda_t \sum_S \mathcal{L}_t(S, f_t) \to \max$
- GLUE, SuperGLUE, Russian SuperGLUE — sets of test problems for understanding natural language
- Transformers are usually built not on words, but on tokens received by BPE (Byte-Pair Encoding) or WordPiece
- First transformer: $N = 6, d = 512, J = 8$, weights 65M
- $\text{BERT}_{\text{BASE}}$, GPT-1: $N = 12, d = 768, J = 12$, weights 110M
- $\text{BERT}_{\text{LARGE}}$, GPT-2: $N = 24, d = 1024, J = 16$, weights 340M-1000M
- GPT-3: weights 175 billion
- Gopher (DeepMind): 280 billion
- Turing-Megatron (Microsoft + Nvidia): 530 billion

Summary

- Attention models were first built into RNNs or CNNs, but they turned out to be self-sufficient
- Based on them, the Transformer architecture was developed, various variants of which (BERT, GPT-3, XLNet, ELECTRA, etc.) are currently SotA in natural language processing tasks, and not only
- Multi-head self-attention (MHSA) has been proven to be equivalent to a convolutional network [Cordonnier, 2020]

Vaswani et al. Attention is all you need. 2017.
Dichao Hu. An Introductory Survey on Attention Mechanisms in NLP Problems. 2018.
Xipeng Qiu et al. Pre-trained models for natural language processing: A survey. 2020.
Cordonnier et al. On the relationship between self-attention and convolutional layers. 2020

What else can you see?

- GPT-3 on wikipedia: https://en.wikipedia.org/wiki/GPT-3
- Grigory Sapunov about transformers in 2021 (in Russian)
- Nikolai Grigoriev (SE @ Deepmind) about history and modern language models, February 14, 2022
- Seminar of WMC Euler and MCS on formalization of mathematical proofs in Lean and deep learning (in Russian)