Attention!

22th November, 2022

Skoltech

Alexey Zaytsev, Assistant professor

Models of Sequential Data

3 generations of Machine translation models

Skoltech

Machine translation: a sequence to sequence problem

Translate a sentence from one language to another

source language

target language

$\boldsymbol{x}$ → $\boldsymbol{y}$

A la guerre comme a la guerre

На войне как на войне

First generation: Early machine translation, 50s

Cold war child: Russian to English IBM 701 Translator

Doctor Dostert predicted that *"five, perhaps three years hence, interlingual meaning conversion by electronic process in important functional areas of several languages may well be an accomplished fact. "* (1954)

Rule-based approach, uses a dictionary to map Russian words to English

Skoltech

https://www.ibm.com/ibm/history/exhibits/701/701_translator.html
https://youtu.be/8ZtdVUB007A

Second generation: Statistical machine translation, 90s-2010s

Learn probabilistic model from data

$$p(\boldsymbol{y}|\boldsymbol{x})$$

To translate: for an input English sequence $\boldsymbol{x}$ find the most probable Russian sentence $\boldsymbol{y}$

$$p(\boldsymbol{y}|\boldsymbol{x}) \to \max_{\boldsymbol{y}}$$

Bayesian perspective:

$$p(\boldsymbol{y}|\boldsymbol{x}) \sim p(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y})$$

Skoltech

Translation model: learn from parallel corpus

Language model: learn from monolingual corpus

Learning translation model

Learn translation model from data

$$p(\boldsymbol{x}|\boldsymbol{y})$$

- Large amount of parallel data
- Alignment

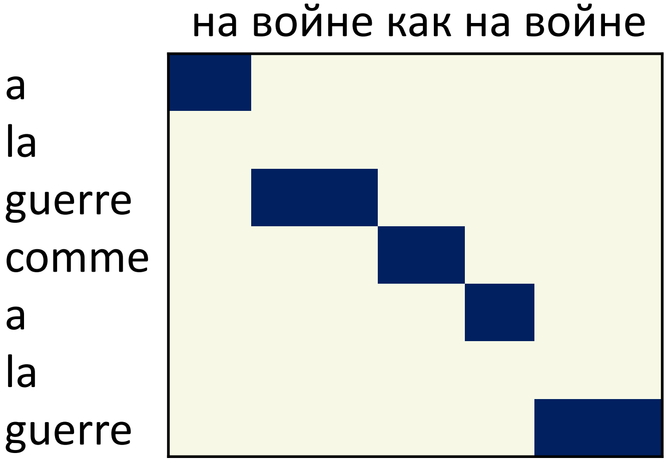
$$p(\boldsymbol{x}, \boldsymbol{a}|\boldsymbol{y})$$

correspondence between words in different languages

Alignment types:

- one-to-one
- spurious words
- one-to-many
- Many-to-one

Many problems on the way



Decoding

The optimization problem is hard

$$p(\boldsymbol{y}|\boldsymbol{x}) \rightarrow \max_{\boldsymbol{y}}$$

- Full search is not possible
- Heuristic search algorithm to search for the best translation: look through a tree of possible options

Skoltech

The best SMT systems are very complex

- Language itself is very complex
- Many details we don’t even mention
- Separately designed subcomponents
- Tricky feature engineering
- Extra information
- The language changes – we need to maintain the system

All difficulties we saw about pre-Neural approaches to sequence processing multiplied x100!

Third generation: Neural Machine Translation!

Neural machine translation

Our goal is to do machine translation with one Neural network

It works with two RNNs

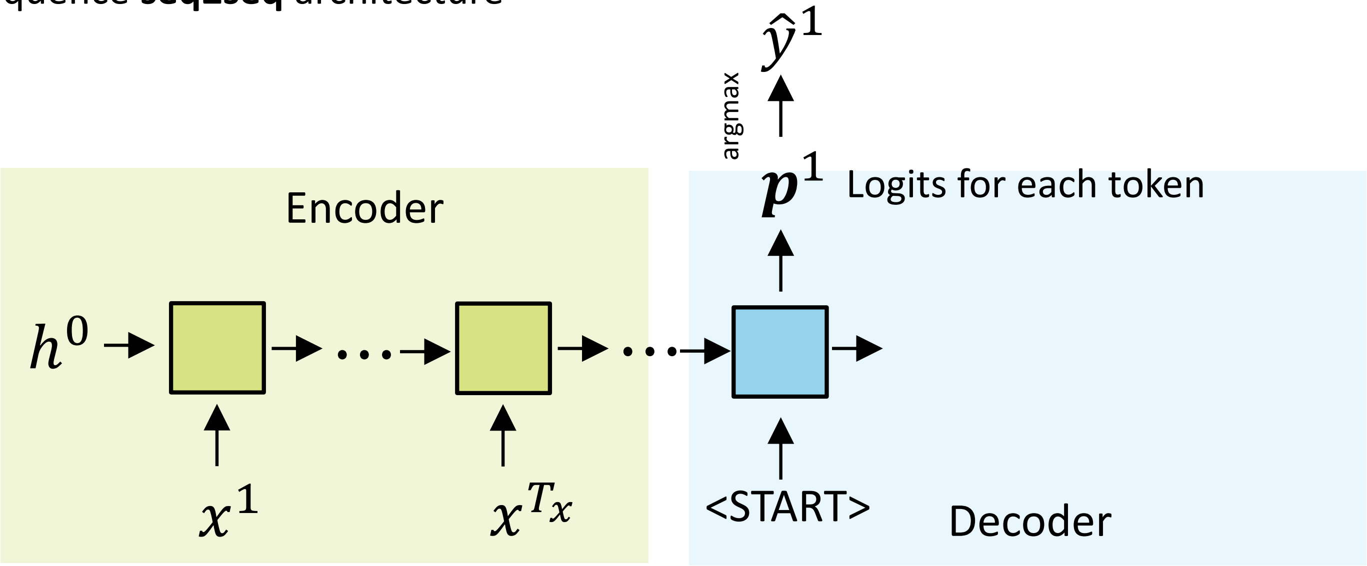
Sequence to sequence **seq2seq** architecture

Neural machine translation

Our goal is to do machine translation with one Neural network

It works with two RNNs

Sequence to sequence **seq2seq** architecture

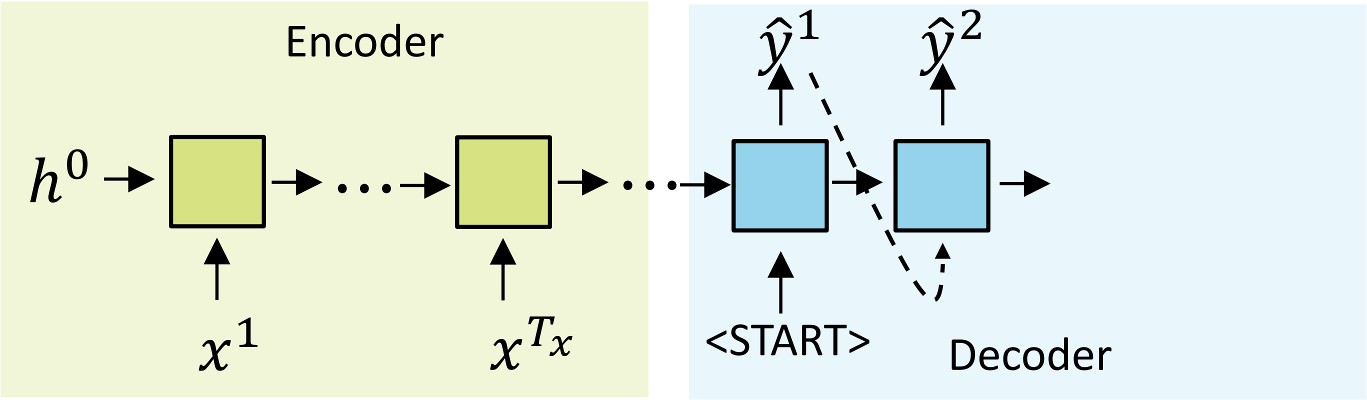


Neural machine translation

Our goal is to do machine translation with one Neural network

It works with two RNNs

Sequence to sequence **seq2seq** architecture



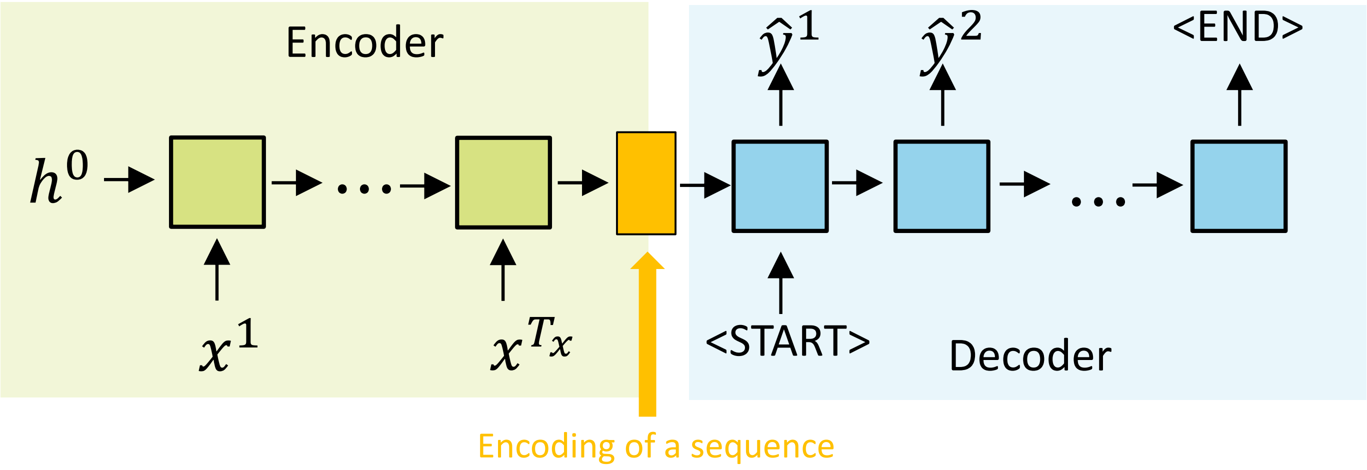
Skoltech

Neural machine translation

Our goal is to do machine translation with one Neural network

It works with two RNNs

Sequence to sequence **seq2seq** architecture

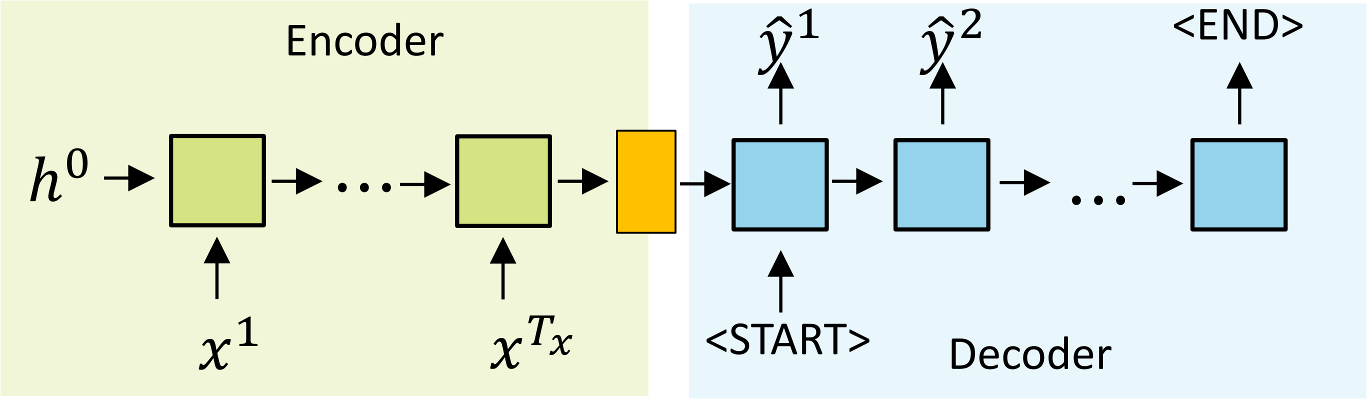


Neural machine translation

Our goal is to do machine translation with one Neural network

It works with two RNNs

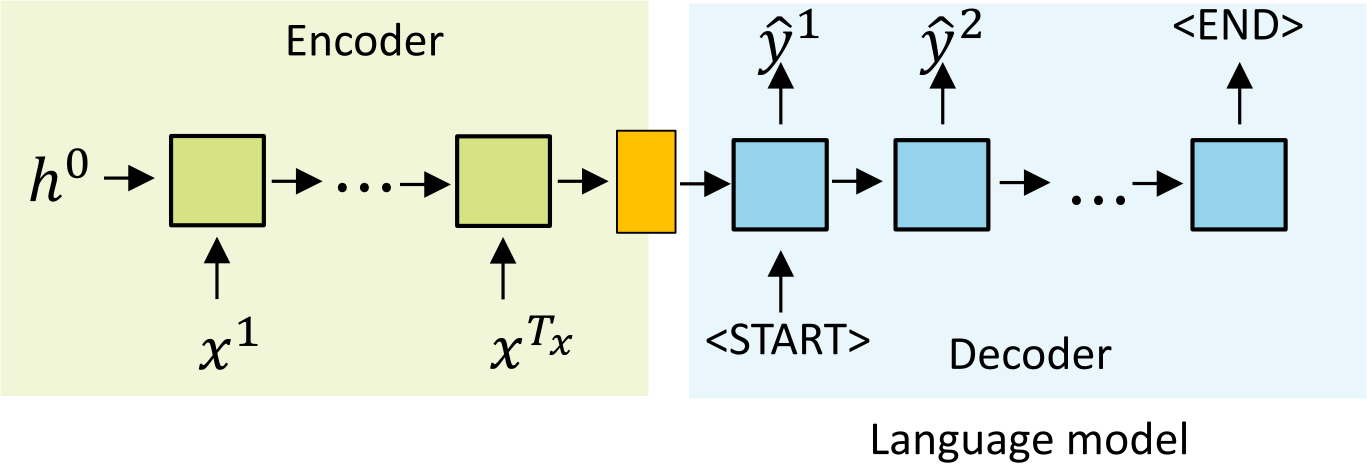
Sequence to sequence **seq2seq** architecture



Language model

New world of seq2seq models

Summarization: long text – text summary
Dialogue: one phrase – another phrase
Parsing: input – output parse as a sequence
Code generation: task description – python code



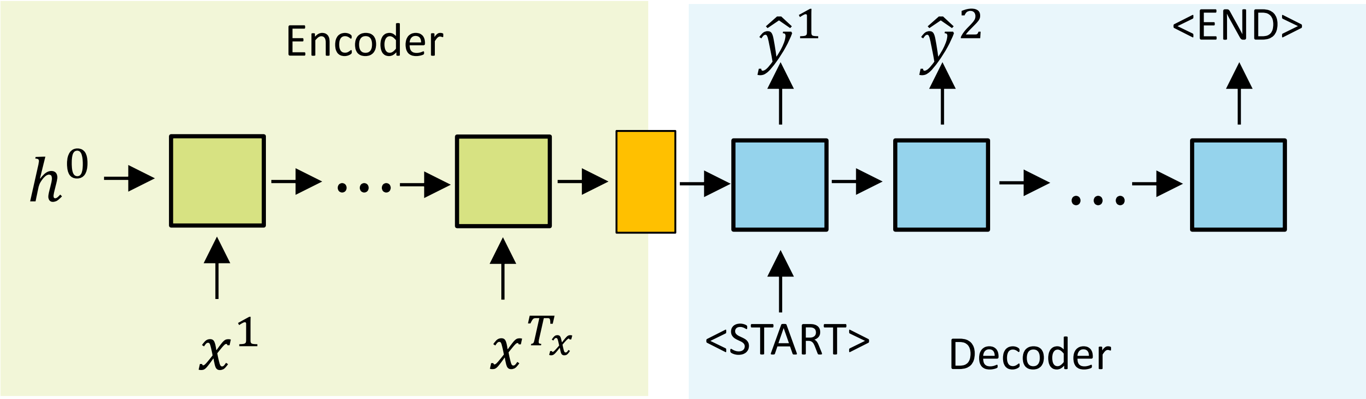
NMT directly models the conditional language model

Learn probabilistic model from data

$$p(\boldsymbol{y}|\boldsymbol{x}) = p(y_T|y_1, y_2, \dots, y_{T-1}, \boldsymbol{x})\, p(y_{T-1}|y_1, y_2, \dots, y_{T-2}, \boldsymbol{x}) \dots p(y_1|\boldsymbol{x})$$

Each term is an RNN block

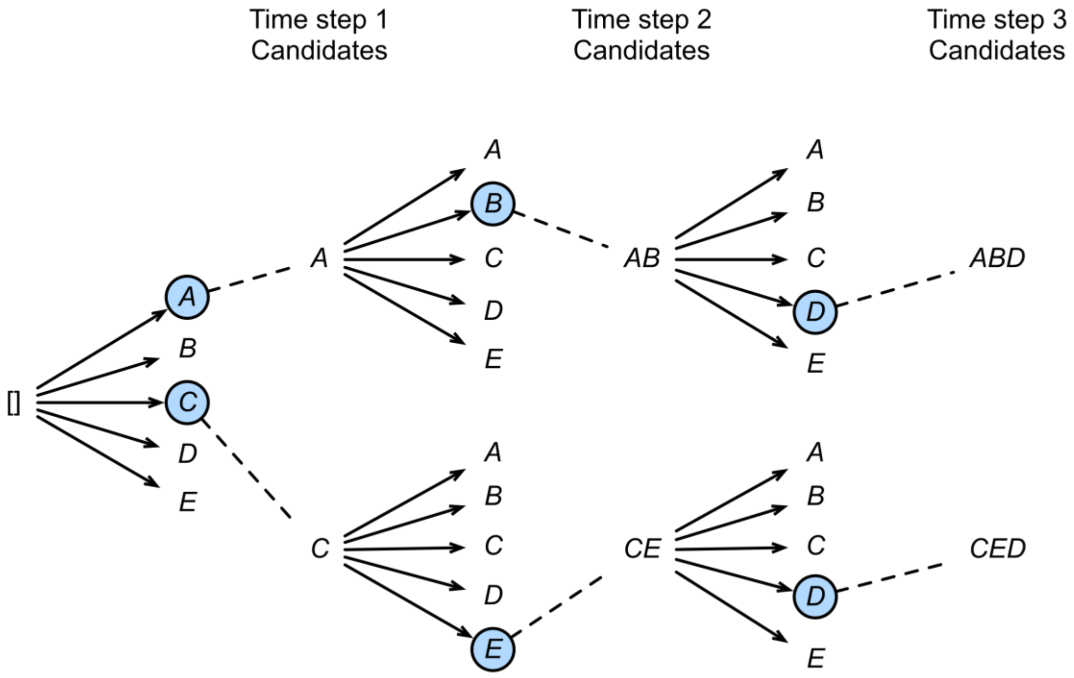
Loss function: compare logits to true words, now we have a logloss if we have a parallel corpus



Beam search decoding

Problem:
We generate tokens one by one from the very beginning
Wrong decision at some step leads to wrong translation as a whole

Solution:
"Beam search": Keep a population of solutions and select the most probable sequences at each step



Beam search decoding: practical implementation

Stopping criteria:

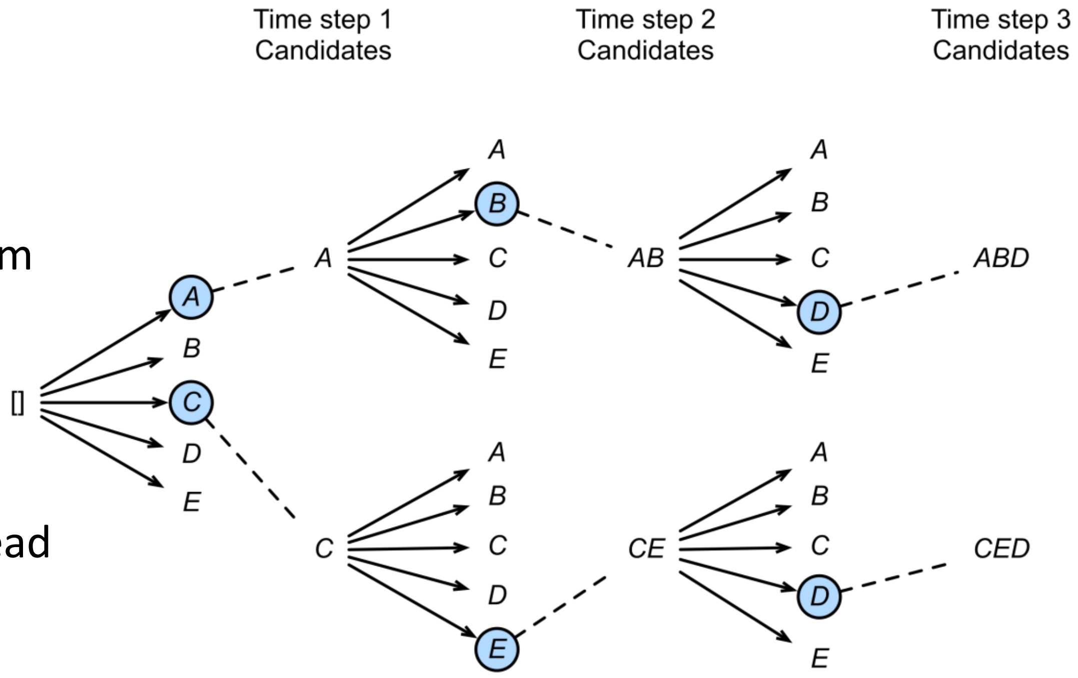
- Generated <END> token
- Reached maximum sequence length

We have $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_m\}$ after beam search

Selection criterion:

Compare normalized $\tilde{p}(\boldsymbol{y}_i|\boldsymbol{x})$ instead of $p(\boldsymbol{y}_i|\boldsymbol{x})$

$$\tilde{p}(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{|\boldsymbol{y}|} p(\boldsymbol{y}|\boldsymbol{x})$$



Advantages of NMT

- Better performance compared to SLT (Statistical Learning Translation)
- Single end2end neural network
- Much less human engineering effort
    - No feature engineering
    - Same method for all language pairs

Evaluations of Machine Translation: BLEU

"the closer a machine translation is to a professional human translation, the better it is“

Precision: share of words from $\hat{\boldsymbol{y}}$ that appear in $\boldsymbol{y}$

True $\boldsymbol{y}$	the	cat	is	on	the	mat
Candidate $\hat{\boldsymbol{y}}$	the	the	the	the	the	the

Unigram precision is 6/6 = 1

Limit with number of words from references: there are 2 “the” in $\boldsymbol{y}$

BLEU modified unigram precision is 2/6 = 0.33

Skoltech

Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. J. 2002. *BLEU: a method for automatic evaluation of machine translation*. ACL-2002: 40th Annual meeting of the Association for Computational Linguistics

BLEU evaluation examples

True $\boldsymbol{y}$	the	cat	is	on	the	mat
Candidate $\hat{\boldsymbol{y}}$	the	the	cat	cat	cat	the

Unigram precision is 6/6 = 1
Bigram precision is 1/5 = 0.2
BLEU unigram score 2/6
BLEU bigram score is 1/5

Good BLEU scores is about 0.7
Higher scores mean overfitting or other problems

Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. J. 2002. *BLEU: a method for automatic evaluation of machine translation*. ACL-2002: 40th Annual meeting of the Association for Computational Linguistics

General advices on BLEU

In practice we use **n-gram modified precision** with n = 4 that "best correlates to human judgement".

Good BLEU scores is about 0.7

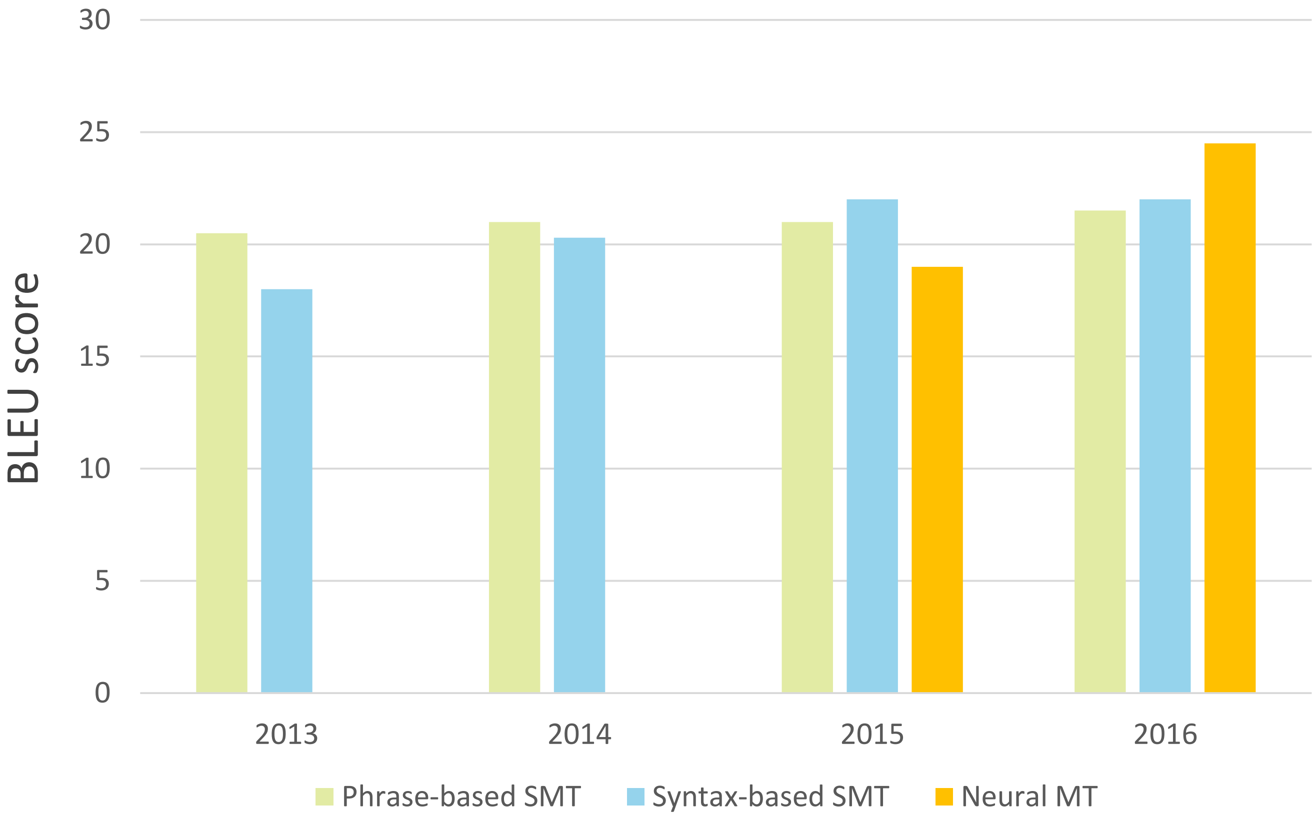
Higher scores mean overfitting or other problems

Recall is also important

Skoltech

Papineni, K.; Roukos, S.; Ward, T.; Zhu, W. J. 2002. *BLEU: a method for automatic evaluation of machine translation*. ACL-2002: 40th Annual meeting of the Association for Computational Linguistics

Evaluations of Machine Translation

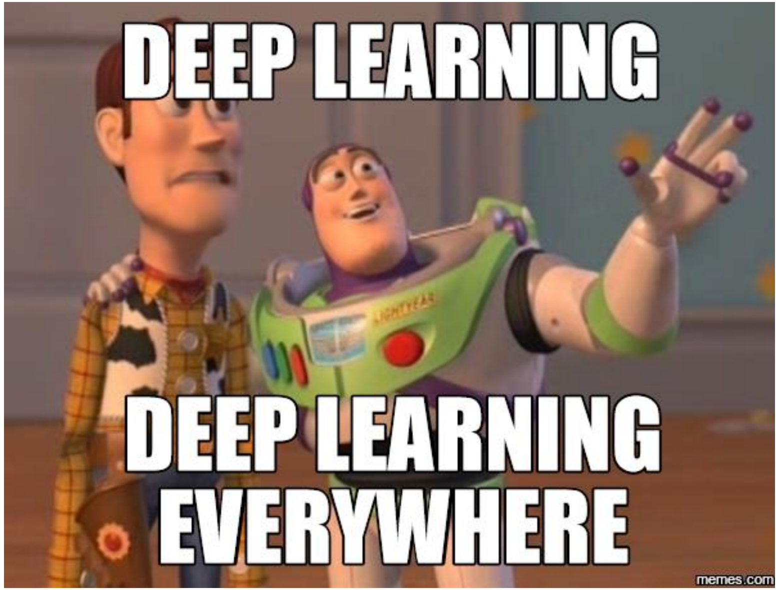


Skoltech

source: Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 8 – Translation, Seq2Seq, Attention

NMT: success story for deep learning

2014: first seq2seq paper published
2016: Google translate switches from SMT to NMT



SMT: hundreds of engineers *for many years*
NMT: handful of engineers *in a few months*

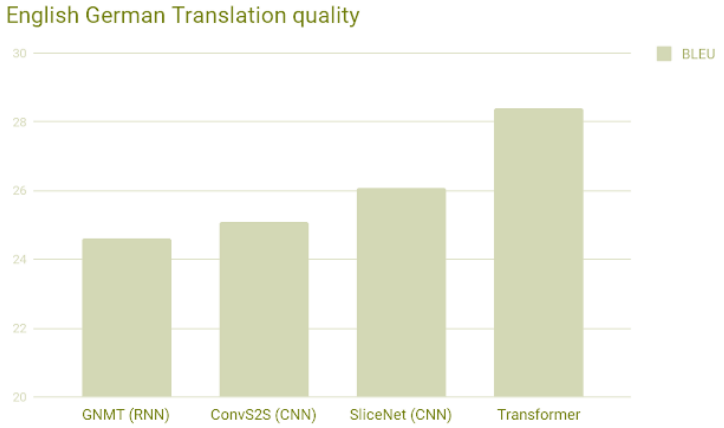
Skoltech

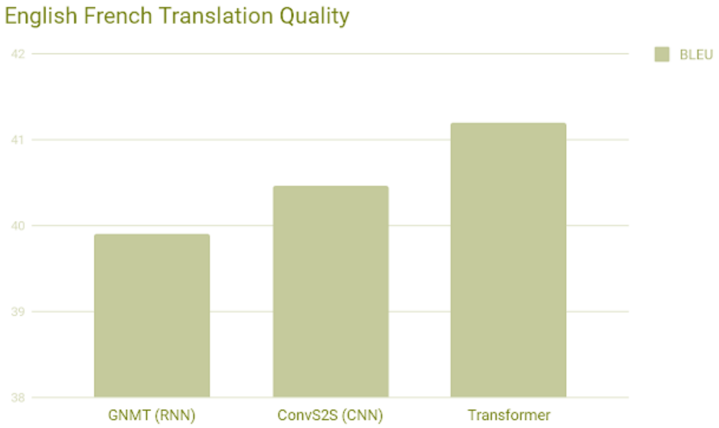
NMT problems

- Out-of-vocabulary words
- Domain mismatch: training and test data
- Context over longer texts
- Low-resource language pairs

New state of the art: **attention** is all we need

attention

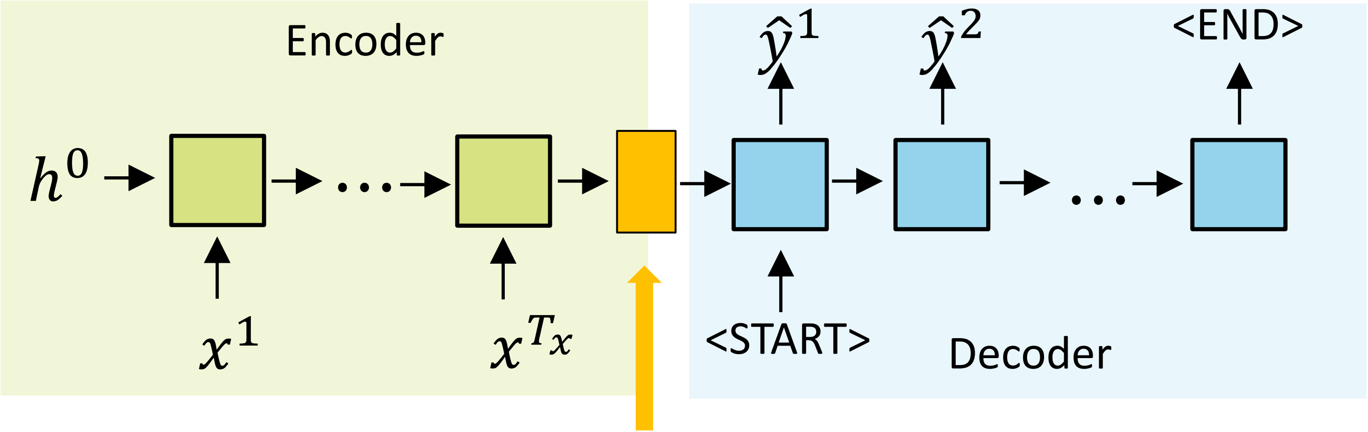




Skoltech

https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Bottleneck in seq2seq models



All information about the sequence is in this vector

Skoltech

Attention

- Solution to the bottleneck problem
- Direction connection between parts of input and output sequence

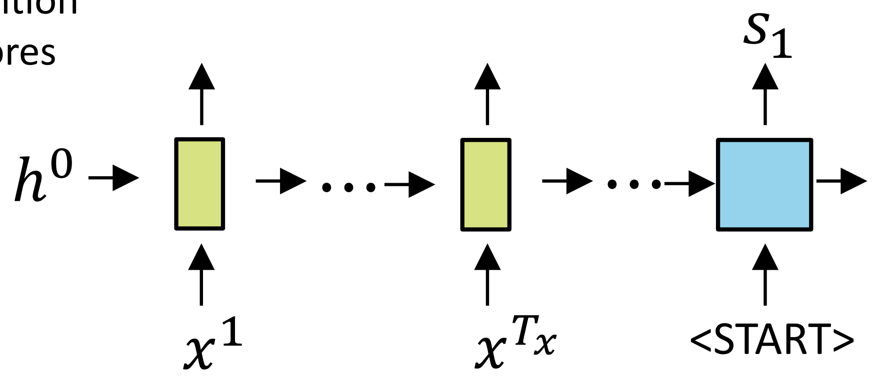
Skoltech

Sequence 2 sequence with attention

Attention
output

Attention
distribution

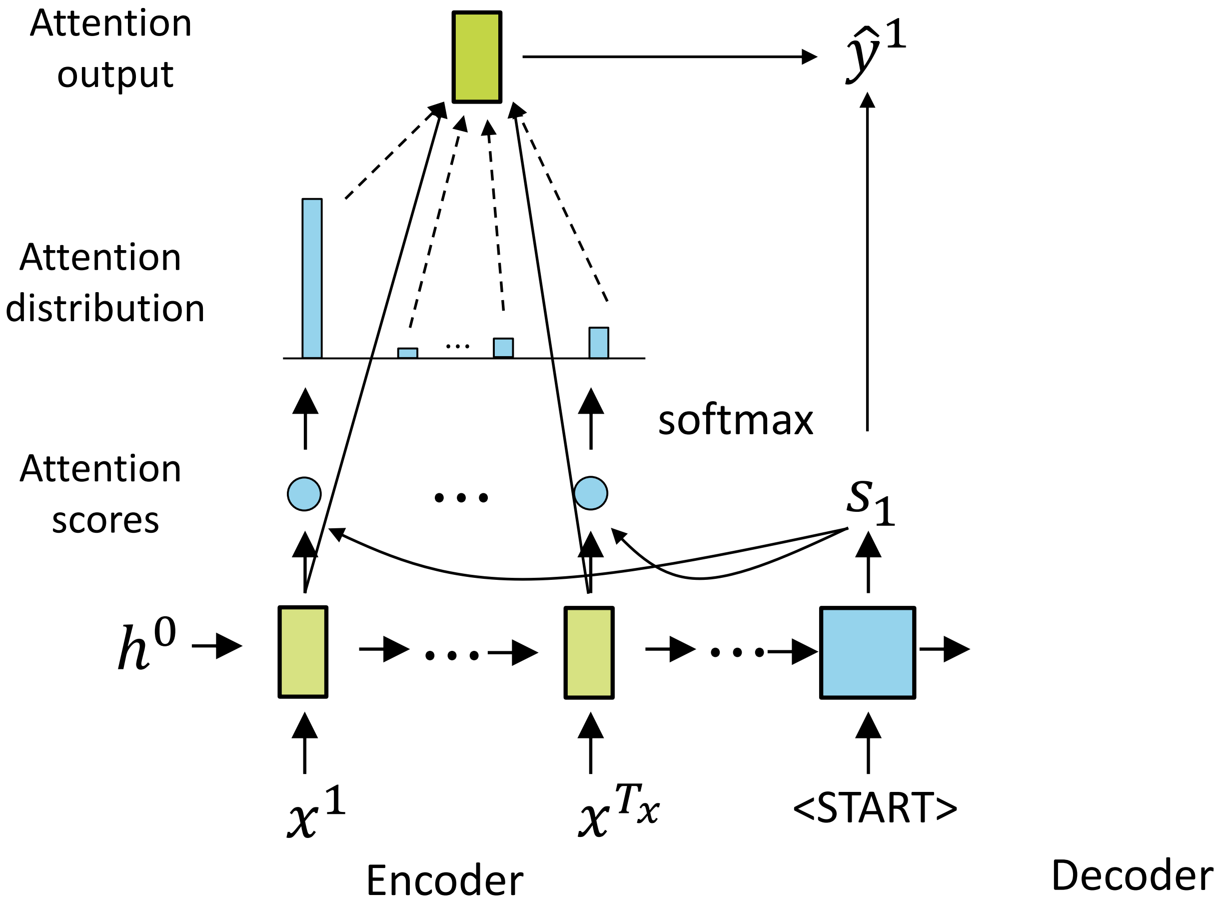
Attention
scores



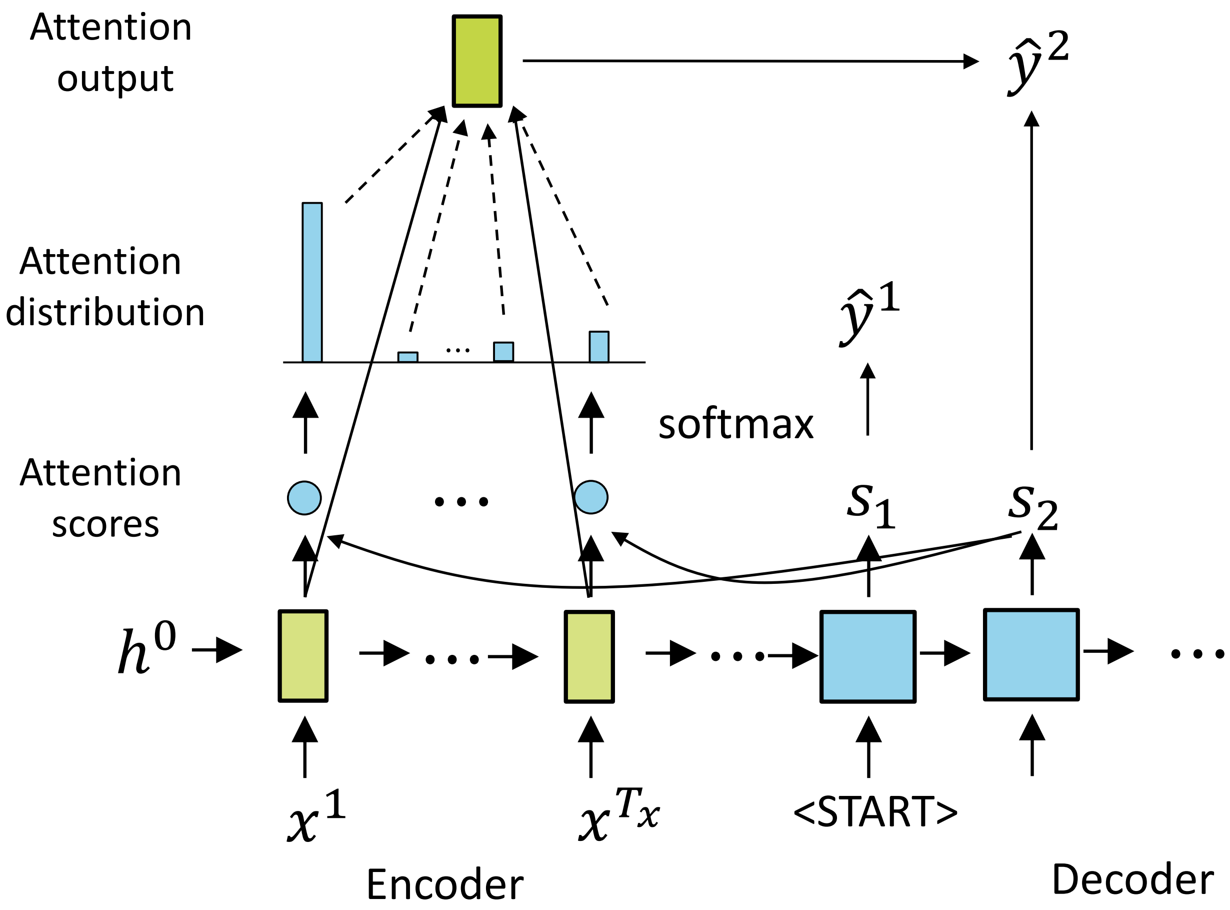
Encoder

Decoder

Sequence 2 sequence with attention



Sequence 2 sequence with attention



Skoltech

Attention: formulas

- First RNN produces encoder hidden states $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{T_x} \in \mathbb{R}^h$
- Decoder hidden state $\boldsymbol{s}_t \in \mathbb{R}^h$ at time step t
- Attention scores for step t:
  $\boldsymbol{e}^{\boldsymbol{t}} = [\boldsymbol{s}_t^{\boldsymbol{T}} \boldsymbol{h}_1, \ldots, \boldsymbol{s}_t^{\boldsymbol{T}} \boldsymbol{h}_{T_x}] \in \mathbb{R}^{T_x}$
- Softmax to get attention distribution: all values are positive, sum of all values is 1:
  $\boldsymbol{\alpha}^{\boldsymbol{t}} = \mathrm{softmax}(\boldsymbol{e}^{\boldsymbol{t}}) \in \mathbb{R}^{T_x}$
- Attention output $\boldsymbol{a}_t$ is a weighted sum of hidden states:

$$\boldsymbol{a}_t = \sum_{i=1}^{T_x} \alpha_i^t \boldsymbol{h}_i \in \mathbb{R}^h$$

- We concatenate the attention output $\boldsymbol{a}_t$ with the decoder hidden state $\boldsymbol{s}_t$ and proceed to the non-attention part of our seq2seq model

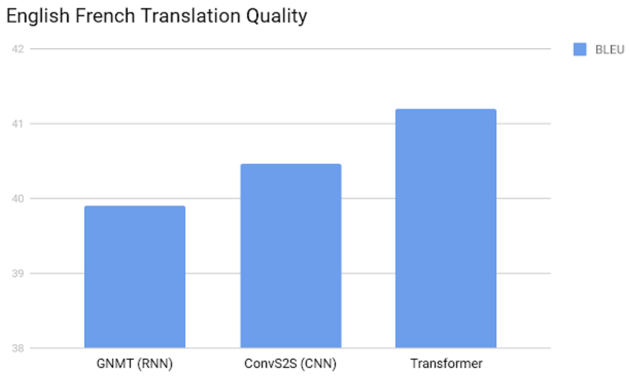
$$[\boldsymbol{a}_t, \boldsymbol{s}_t] \in \mathbb{R}^{2h}$$

Skoltech

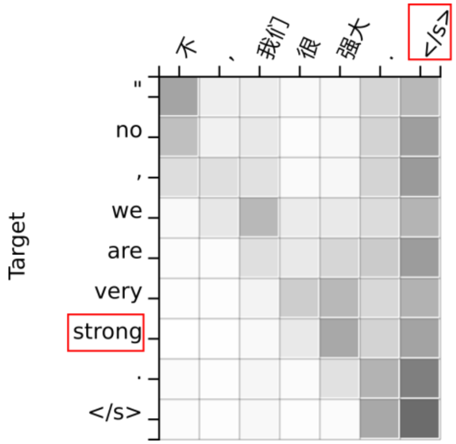
Attention is just great

- Significantly improves performance of NMT
- Solves the bottleneck problem
  - All encoder tokens are connected to all decoder tokens
- No more vanishing gradients
  - All to All connection
- Provides some interpretability
  - see alignment figure

- Similar to RNN seq2seq, but greater!



BLEU scores (higher is better) of single models on the standard WMT newstest2014 English to French translation benchmark.



Attention is a general deep learning idea

We can use attention in many architectures and many tasks

- Other NLP problems
- Sequential data processing
- Graph Neural Networks

Key value interpretation:

s_i - query to a database	Hidden state of the decoder
k_i - keys in the database	Hidden state of the encoder
h_i - values in the database	Hidden state of the encoder

- Calculate correspondence $e(s_i, k_i)$
- Calculate weights on the base of correspondence values
- Extract information as weighted sum of values $\sum_{i=1}^{n} \alpha_i \boldsymbol{h}_i$

Skoltech

Transformers

Skoltech

Key value interpretation

$\boldsymbol{q}_i$ - query to a database — Hidden state of the *decoder*

$\boldsymbol{k}_j$ - keys in the database — Hidden state of the *encoder*

$\boldsymbol{v}_j$ - values in the database — Hidden state of the *encoder*

We **calculate** attention scores

$$\alpha_j = e(\boldsymbol{q}_i, \boldsymbol{k}_j) = \boldsymbol{s}_i^T \boldsymbol{k}_j$$

$$\boldsymbol{\alpha} = \mathrm{softmax}(\boldsymbol{\alpha})$$

Then we extract the information as weighted sum of values

$$\mathbf{a}_{\mathrm{i}} = \sum_{j=1}^{T_x} \alpha_j \boldsymbol{v}_j$$

Skoltech

Matrix key value interpretation

$\boldsymbol{q}_i$ - query to a database — Hidden state of the *decoder*

$\boldsymbol{k}_j$ - keys in the database — Hidden state of the *encoder*

$\boldsymbol{v}_j$ - values in the database — Hidden state of the *encoder*

We calculate correspondences

$$A(q, K, V) = \sum_i \frac{\exp(\boldsymbol{q}_i^T \boldsymbol{k}_j)}{\sum_l \exp(\boldsymbol{q}_i^T \boldsymbol{k}_l)} \boldsymbol{v}_j$$

$$A(Q, K, V) \ = \mathrm{softmax}\left(\mathrm{QK^T}\right)\mathrm{V}$$

Skoltech

Scaled attention values

For large dimension of the space of keys d_k:

- Large variances dot products $\boldsymbol{q}_i^T \boldsymbol{k}_j$
- Softmax only pays attention to some keys
- Gradients are small, hard to learn

Old formula:

$$A(Q, K, V) = \mathrm{softmax}\left(\mathrm{QK^T}\right)\mathrm{V}$$

New scaled formula:

$$A(Q, K, V) = \mathrm{softmax}\left(\mathrm{QK^T}/\sqrt{\mathrm{d_k}}\right)\mathrm{V}$$

Skoltech

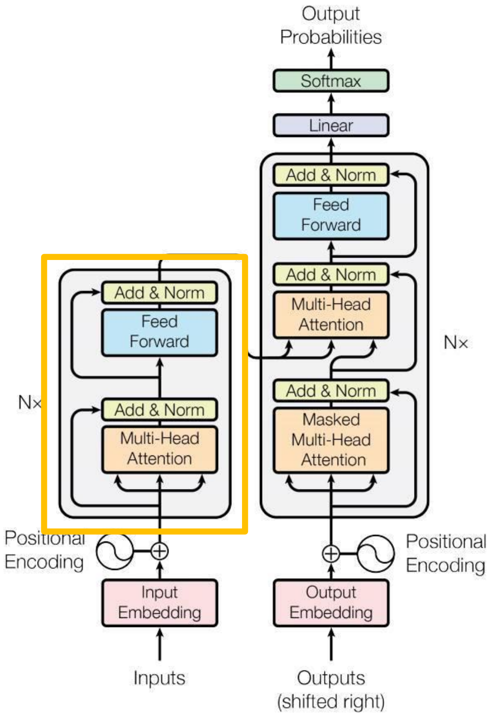
Transformer is based on the same idea

Now we completely drop RNN part

Also we repeat *self-attention* many times

Further we'll consider separate parts:

- Multi-head attention
- Feed Forward



Skoltech

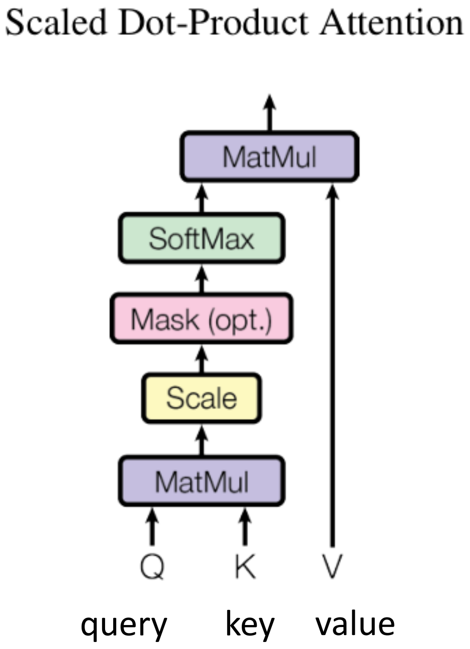
https://pytorch.org/tutorials/beginner/transformer_tutorial.html

Attention / Self-attention block

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

d_k is the dimension of query and key,
we scale to take control of large values of dot-product in high dimensions

A possible option is to replace scaled dot-product used here with additive attention: a single-hidden layer neural network.



Skoltech

“Attention is all you need” paper

Self-attention block

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

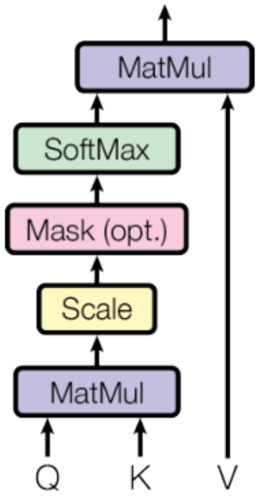
We produce queries, keys, and values using initial word embeddings

$$\mathrm{Q} = \mathrm{XW^Q}, \dim\left(\mathrm{W^Q}\right) = \mathrm{d_x} \times \mathrm{d}_q,$$

$$\mathrm{K} = \mathrm{XW}^K, \dim\left(\mathrm{W^Q}\right) = \mathrm{d_x} \times \mathrm{d}_k,$$

$$\mathrm{V} = \mathrm{XW}^V, \dim\left(\mathrm{W^Q}\right) = \mathrm{d_x} \times \mathrm{d}_v,$$

Scaled Dot-Product Attention



query key value

Skoltech

“Attention is all you need” paper

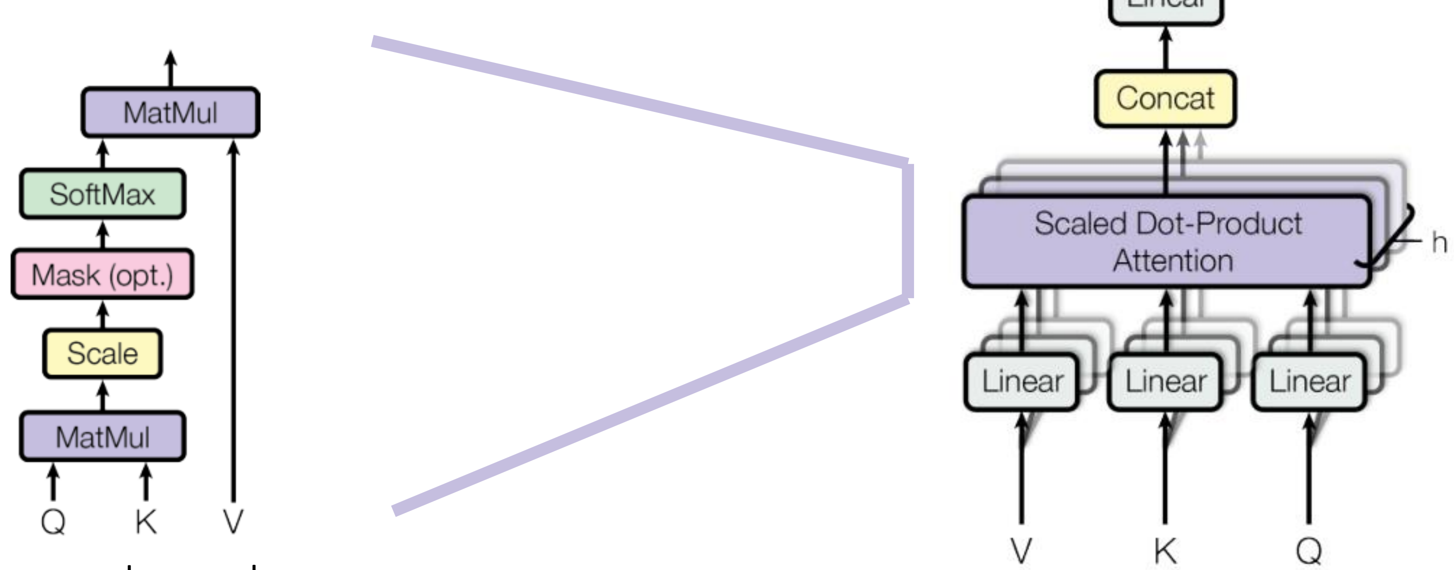
Multi-Head attention

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

h heads in total

Scaled Dot-Product Attention

Multi-Head Attention

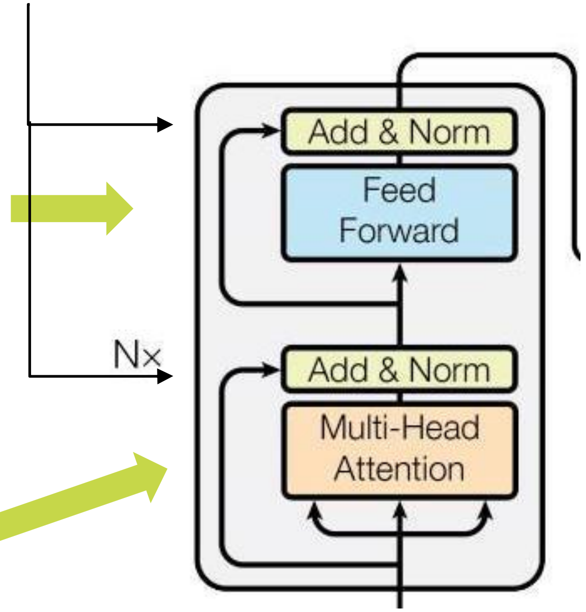


query key value

“Attention is all you need” paper

Full block

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$



Two linear transformation with ReLU activation in between

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Multi-head attention

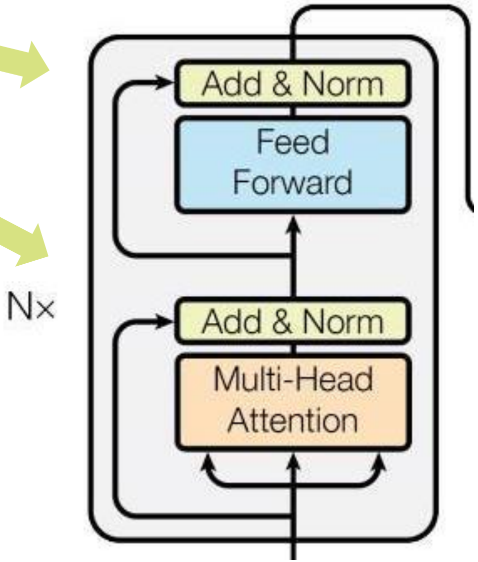
$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

There are 6 consecutive *Full blocks* in the paper transformer architecture

“Attention is all you need” paper

Full block: normalization and residual connection

$$\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$$



Why? Speeds up training!

- Similar to Batch Normalization
- But can be used with batch size 1
- Can be used with RNNs and Transformers

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

Skoltech

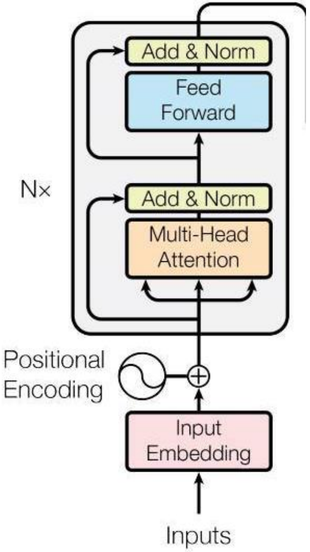
Layer Normalization, https://arxiv.org/abs/1607.06450

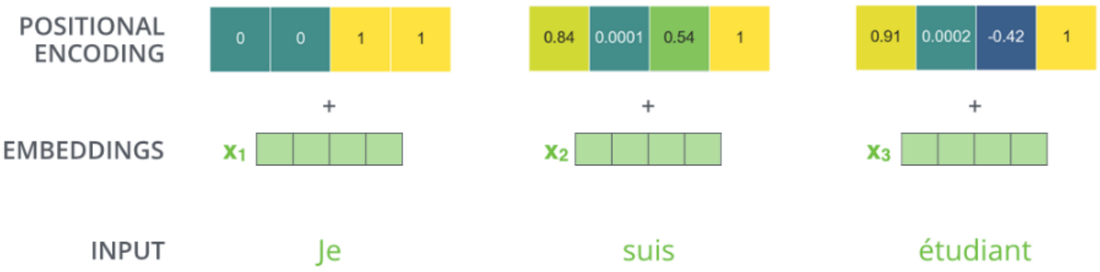
Position encoding

In addition to usual embeddings of inputs we use position encoding to capture position

They are not one-hot vectors, as we want to handle various-length sequences

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





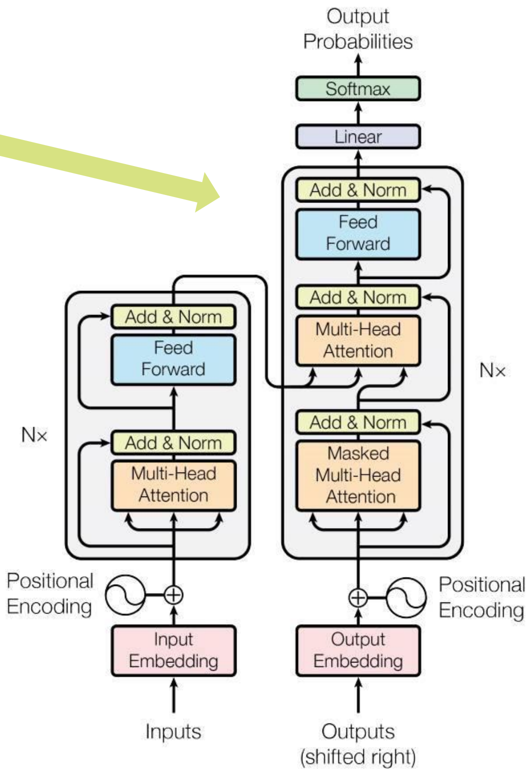
Real example of positional encoding with a toy embedding size of 4

“Attention is all you need” paper http://jalammar.github.io/illustrated-transformer/

Decoder is similar with another N=6 blocks

Additional sublayer to take into account attentions from encoder

We generate one token and proceed to the next token generation

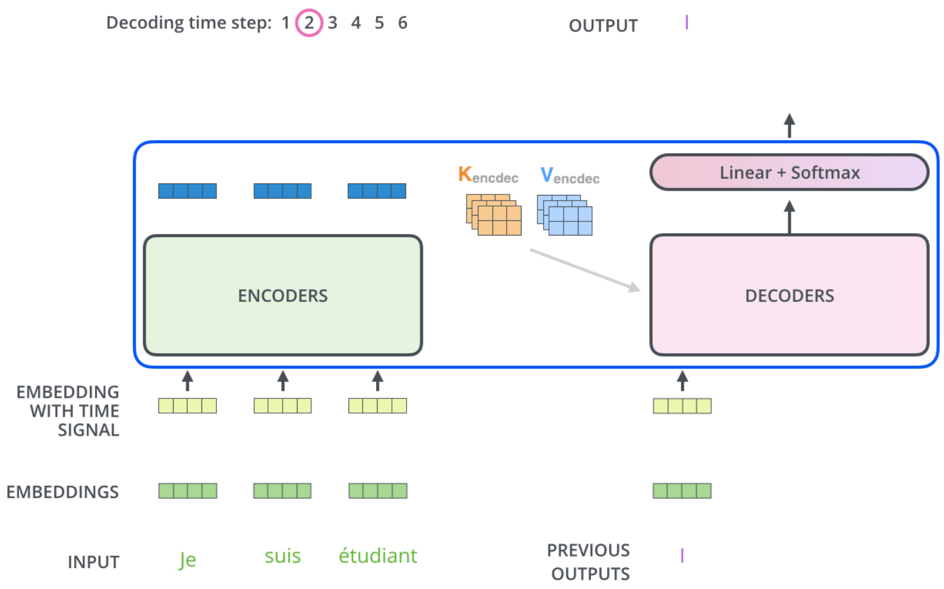


Skoltech

“Attention is all you need” paper

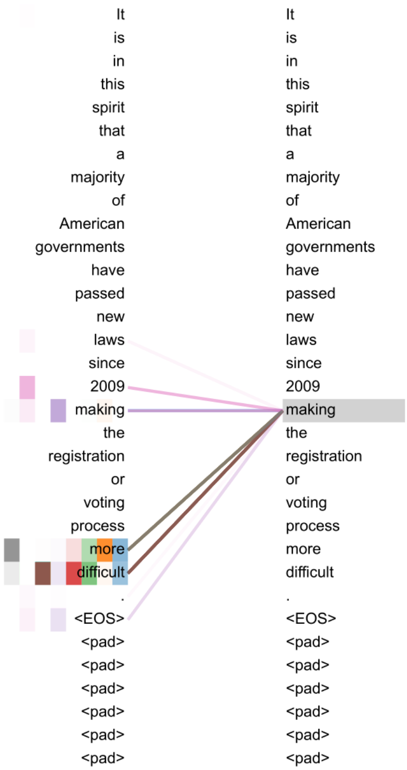
Decoder inference

We generate one token and proceed to the next token generation



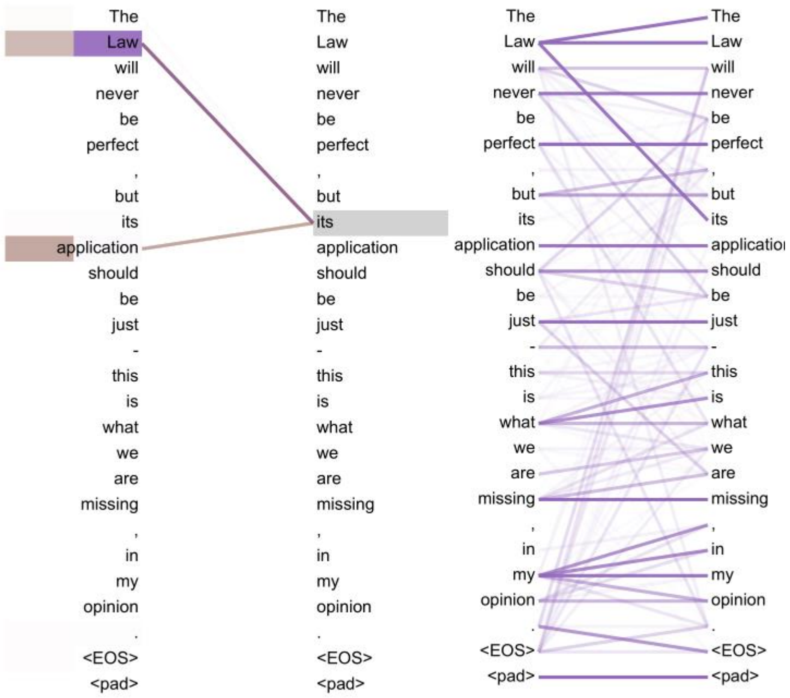
http://jalammar.github.io/illustrated-transformer/

Attention visualizations



“Attention is all you need” paper

Attention visualizations

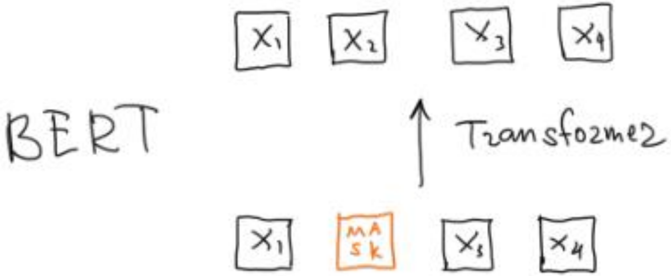


“Attention is all you need” paper

Transformer training

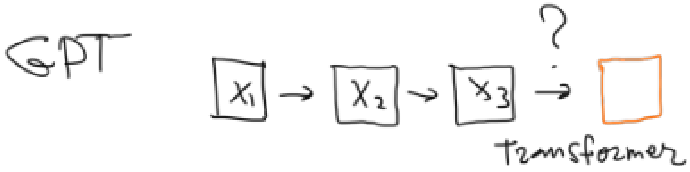
- Masked language model:

Replace random tokens with masks, try to reconstruct them using a Neural network



- Next token prediction

Predict next token



We don’t need labeled examples, we just create them

Efficiency of transformers

Brown, T. B., Mann, B., Ryder, N. et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}	Batch Size	Learning Rate
GPT-3 Small	125M	12	768	12	64	0.5M	6.0×10^{-4}
GPT-3 Medium	350M	24	1024	16	64	0.5M	3.0×10^{-4}
GPT-3 Large	760M	24	1536	16	96	0.5M	2.5×10^{-4}
GPT-3 XL	1.3B	24	2048	24	128	1M	2.0×10^{-4}
GPT-3 2.7B	2.7B	32	2560	32	80	1M	1.6×10^{-4}
GPT-3 6.7B	6.7B	32	4096	32	128	2M	1.2×10^{-4}
GPT-3 13B	13.0B	40	5140	40	128	2M	1.0×10^{-4}
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128	3.2M	0.6×10^{-4}

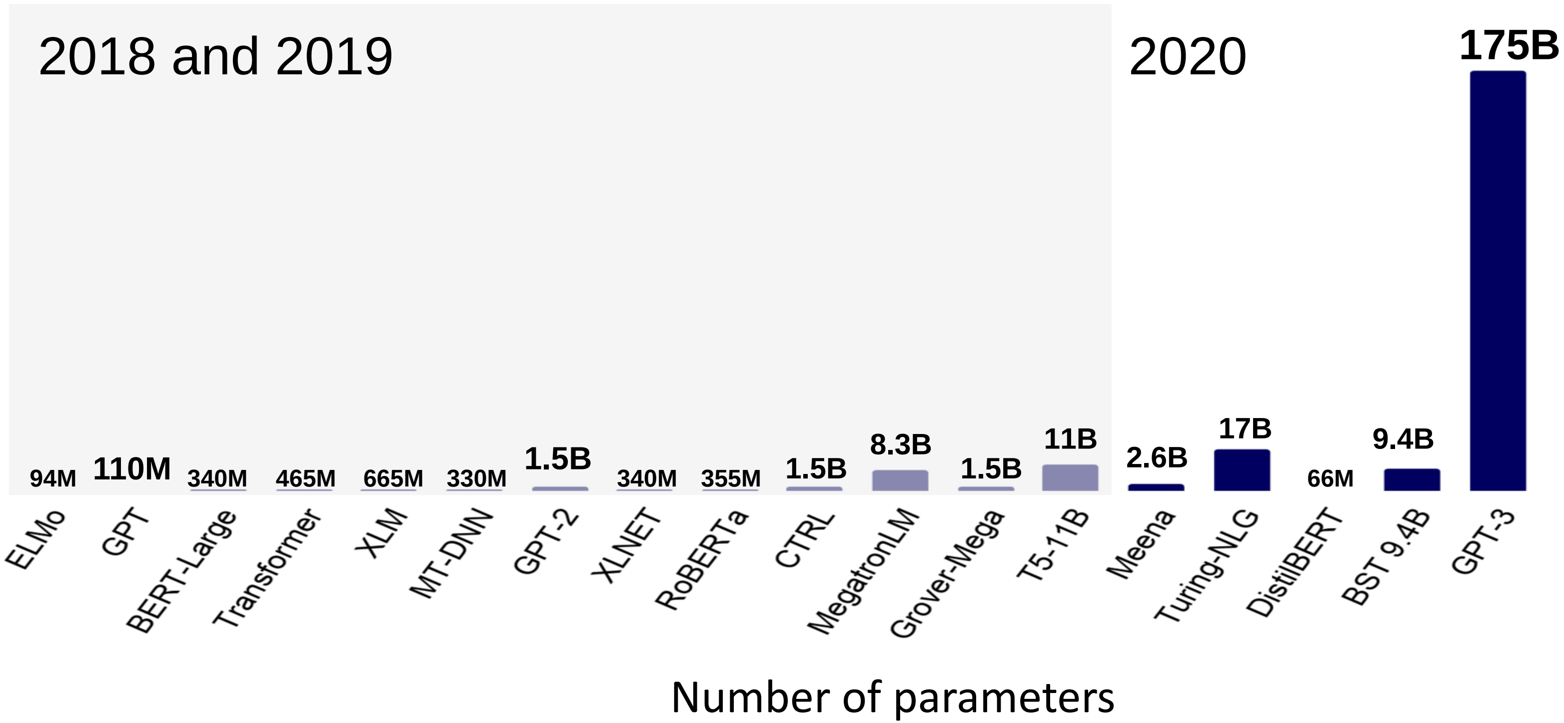
Table 2.1: Sizes, architectures, and learning hyper-parameters (batch size in tokens and learning rate) of the models which we trained. All models were trained for a total of 300 billion tokens.

	Mean accuracy	95% Confidence Interval (low, hi)	t compared to control (p-value)	"I don't know" assignments
Control (deliberately bad model)	86%	83%–90%	-	3.6 %
GPT-3 Small	76%	72%–80%	3.9 (2e-4)	4.9%
GPT-3 Medium	61%	58%–65%	10.3 (7e-21)	6.0%
GPT-3 Large	68%	64%–72%	7.3 (3e-11)	8.7%
GPT-3 XL	62%	59%–65%	10.7 (1e-19)	7.5%
GPT-3 2.7B	62%	58%–65%	10.4 (5e-19)	7.1%
GPT-3 6.7B	60%	56%–63%	11.2 (3e-21)	6.2%
GPT-3 13B	55%	52%–58%	15.3 (1e-32)	7.1%
GPT-3 175B	52%	49%–54%	16.9 (1e-34)	7.8%

Table 3.11: Human accuracy in identifying whether short (~200 word) news articles are model generated. We find that human accuracy (measured by the ratio of correct assignments to non-neutral assignments) ranges from 86% on the control model to 52% on GPT-3 175B. This table compares mean accuracy between five different models, and shows the results of a two-sample T-Test for the difference in mean accuracy between each model and the control model (an unconditional GPT-3 Small model with increased output randomness).

Skoltech

Transformer models are monstrous



Skoltech

Training cost estimation: 10-50 MLN US$ for GPT-3

Conclusions

- Attention + positional encoding are all you need
- The attention idea is pretty general

Skoltech

Sources

- Stanford CS224N: NLP with Deep Learning | Winter 2019 | Lecture 8 – Translation, Seq2Seq, Attention

- https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

Skoltech