Глубокое обучение и вообще

Бекезин Никита

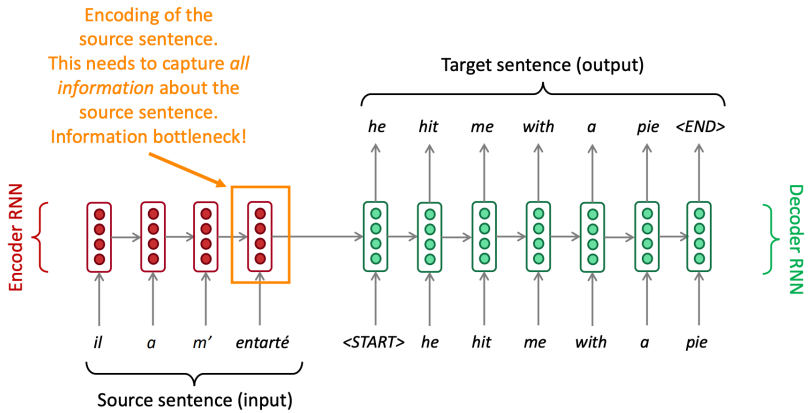
8 декабря 2021 г.

Лекция 14: Быстрое введение в SOTA 2

Agenda

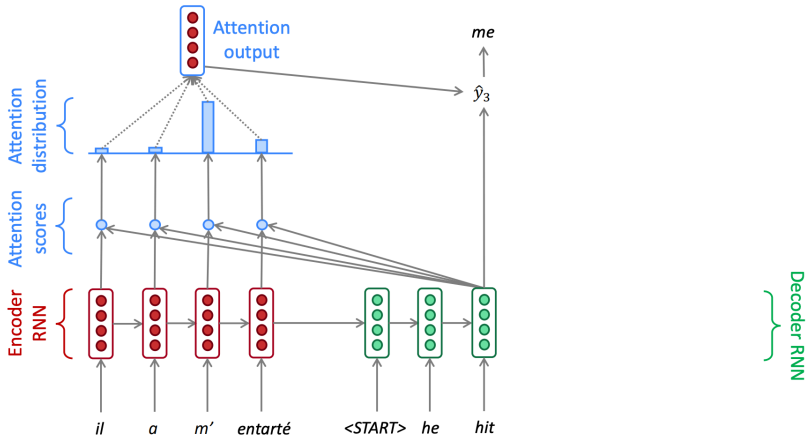
- Seq2seq recap
- Attention recap
- Transformer
- Self-Attention
- BERT

Seq2seq recap



Stanford lectures

Attention recap



Stanford lectures

Attention recap

There are several ways you can compute $\boldsymbol{e} \in \mathbb{R}^N$ from $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N \in \mathbb{R}^{d_1}$ and $\boldsymbol{s} \in \mathbb{R}^{d_2}$:

- Basic dot-product attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{h}_i \in \mathbb{R}$
  - Note: this assumes $d_1 = d_2$
  - This is the version we saw earlier
- Multiplicative attention: $\boldsymbol{e}_i = \boldsymbol{s}^T \boldsymbol{W} \boldsymbol{h}_i \in \mathbb{R}$
  - Where $\boldsymbol{W} \in \mathbb{R}^{d_2 \times d_1}$ is a weight matrix
- Additive attention: $\boldsymbol{e}_i = \boldsymbol{v}^T \tanh(\boldsymbol{W}_1 \boldsymbol{h}_i + \boldsymbol{W}_2 \boldsymbol{s}) \in \mathbb{R}$
  - Where $\boldsymbol{W}_1 \in \mathbb{R}^{d_3 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_3 \times d_2}$ are weight matrices and $\boldsymbol{v} \in \mathbb{R}^{d_3}$ is a weight vector.
  - d_3 (the attention dimensionality) is a hyperparameter

Stanford lectures

Attention recap

- “Free” word alignment
- Better results on long sequences

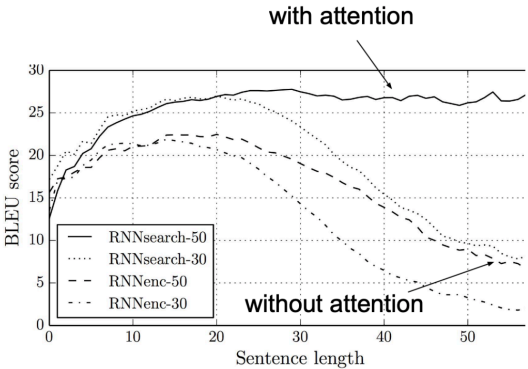
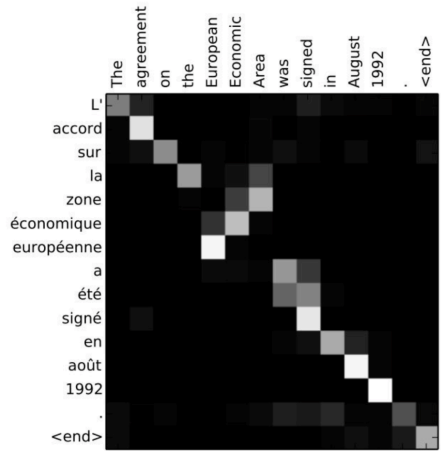


Image source: Neural Machine Translation by Jointly Learning to Align and Translate

Attention advantages

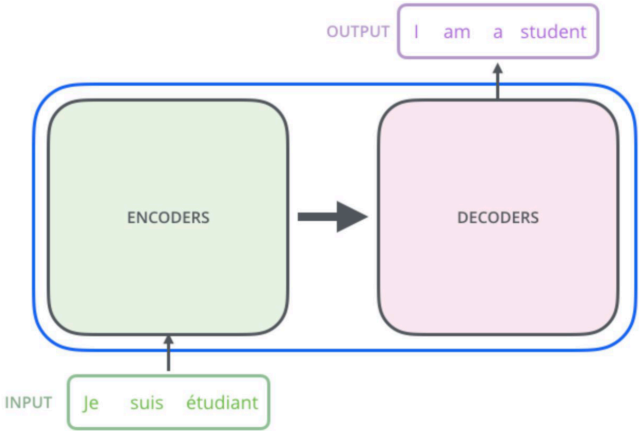


Attention is all you need!

attention is all you need

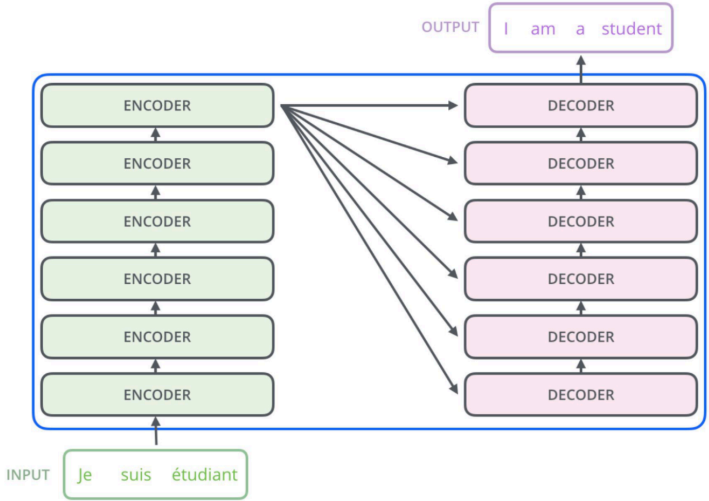
Развитие идеи внимания. Статья вышла в 2017 году и стала мамой всех текущих SOTA моделей. А зачем нам вообще что-то, кроме внимания? Давайте напихаем в энкодер и декодер как можно больше внимания и будем такой штукой его учить.

Transformer



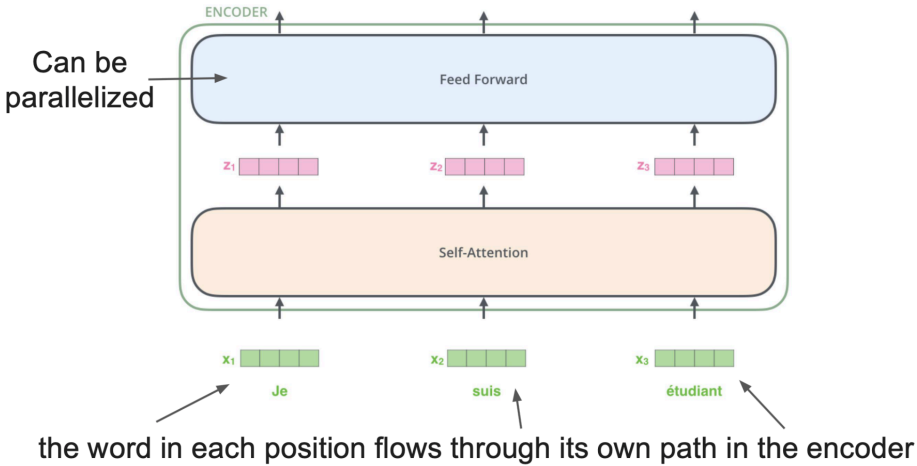
Source

Transformer



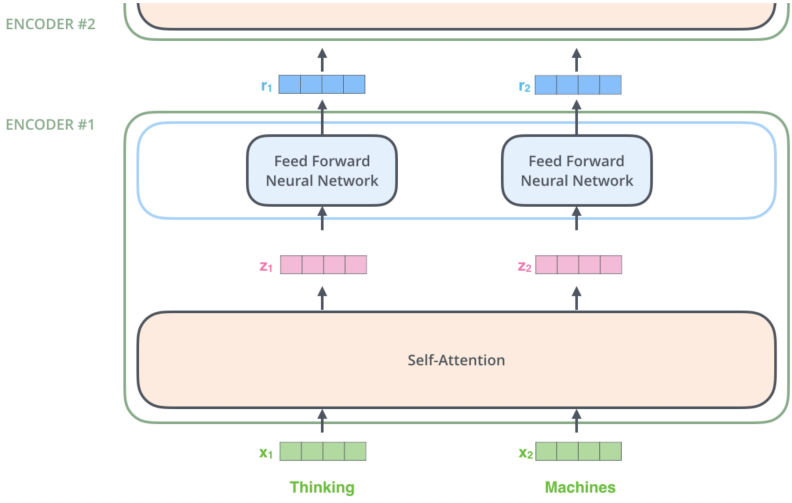
Source

Transformer



Source

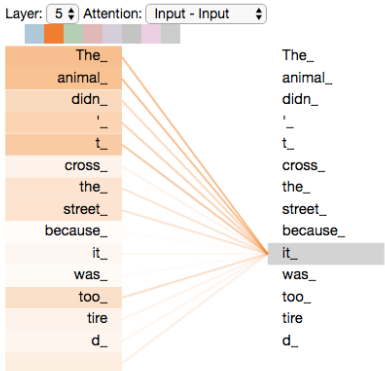
Transformer



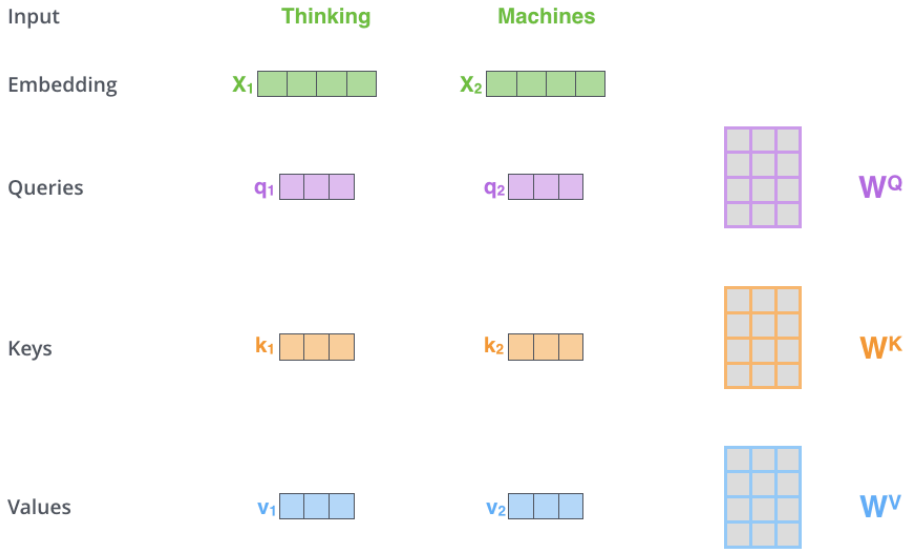
Source

Self-attention или механизм внимания

Есть предложение: ”The animal didn't cross the street because it was too tired”



Шаг 1. Абстракции

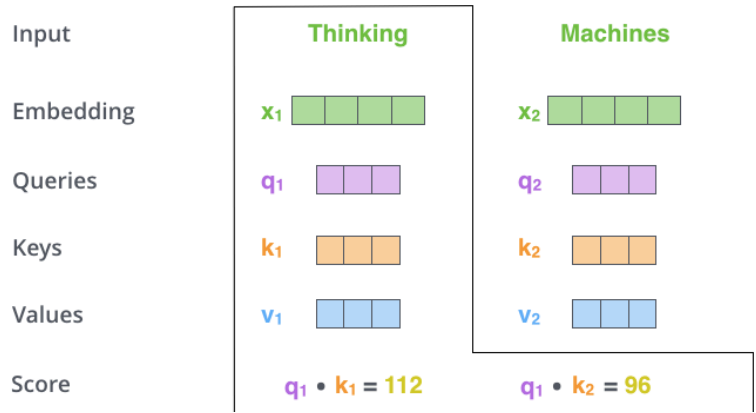


Self-attention

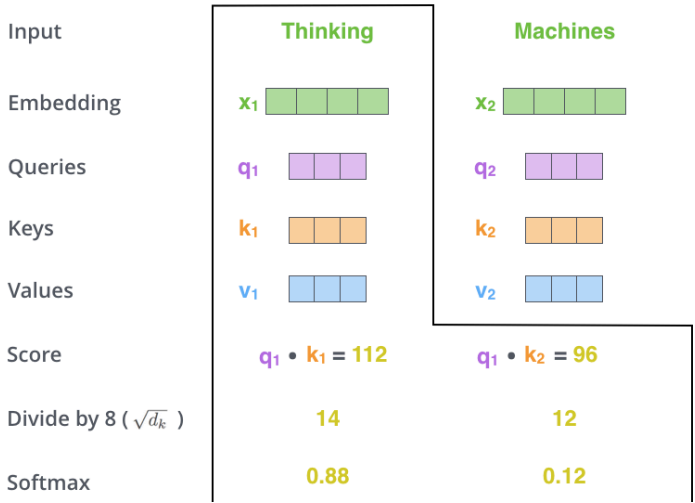
А теперь тоже самое, но словами:

1. Query,key - ищем связи между словами. Ходим по всем со всеми смотрим насколько они связаны. Query - мое текущее слово, key - мое слово с которым я сравниваю себя.
2. Value - то, что мы знаем об этом слове

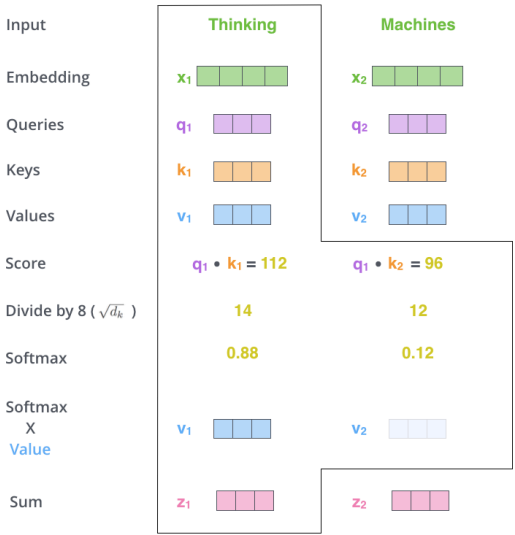
Шаг 2. Расчет attention score



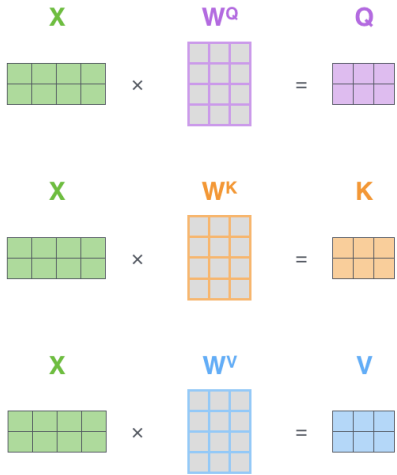
Шаг 3. Шкалирование и Softmax



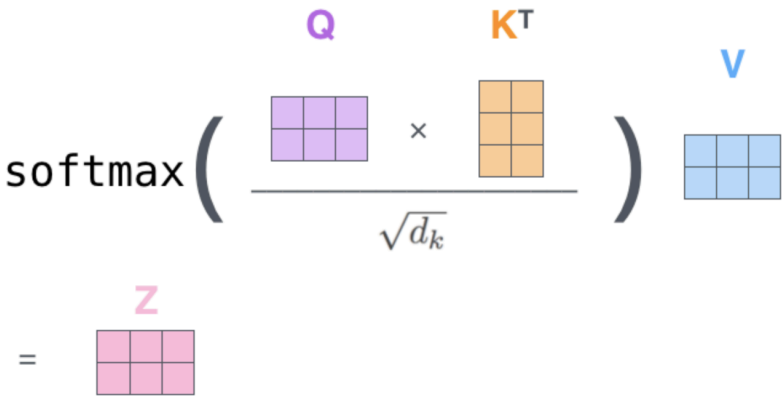
Шаг 4. Сумма взвешенных векторов Value



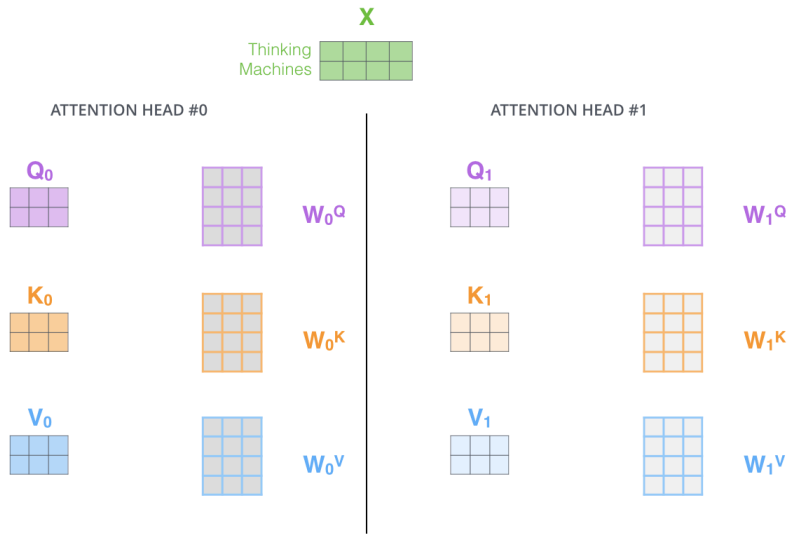
Матричные перемножения



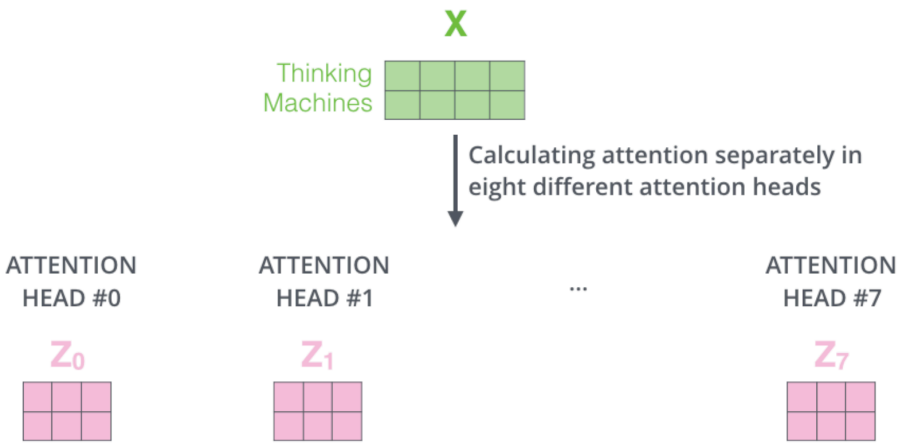
All together



Multi-Head Attention



Multi-Head Attention



Соединяем!

1) Concatenate all the attention heads

2) Multiply with a weight matrix W^O that was trained jointly with the model

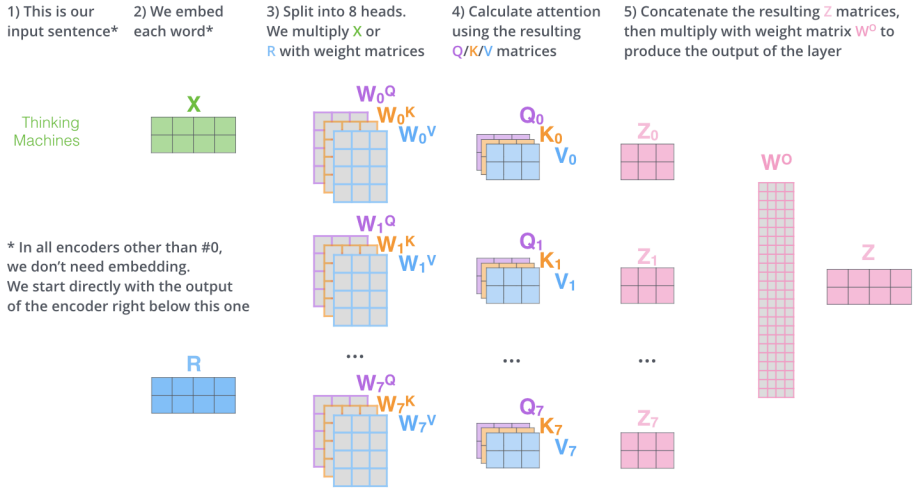
X

3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN

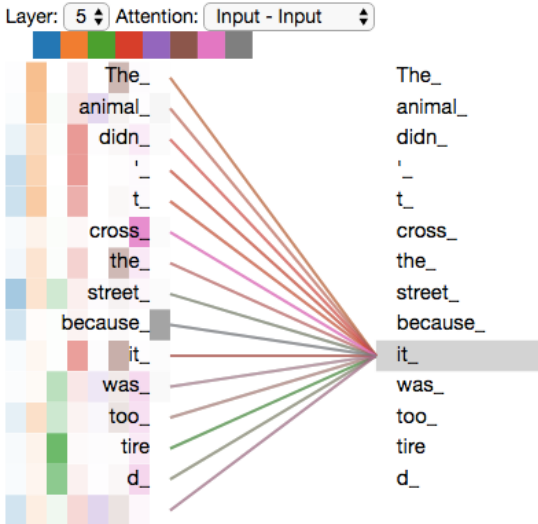
Z

=

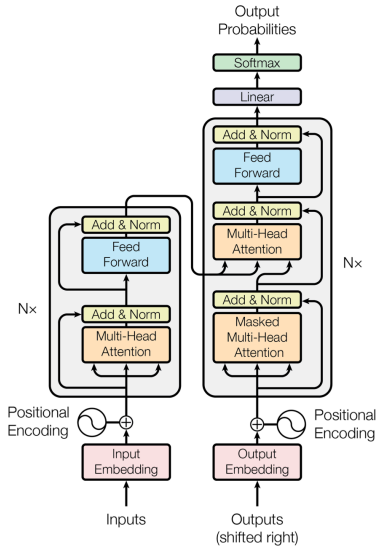
Итого


1) This is our input sentence*
2) We embed each word*
3) Split into 8 heads. We multiply X or R with weight matrices
4) Calculate attention using the resulting Q/K/V matrices
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer
Thinking Machines
X
R
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one
W0Q W0K W0V
W1Q W1K W1V
W7Q W7K W7V
Q0 K0 V0
Q1 K1 V1
Q7 K7 V7
Z0
Z1
Z7
WO
Z

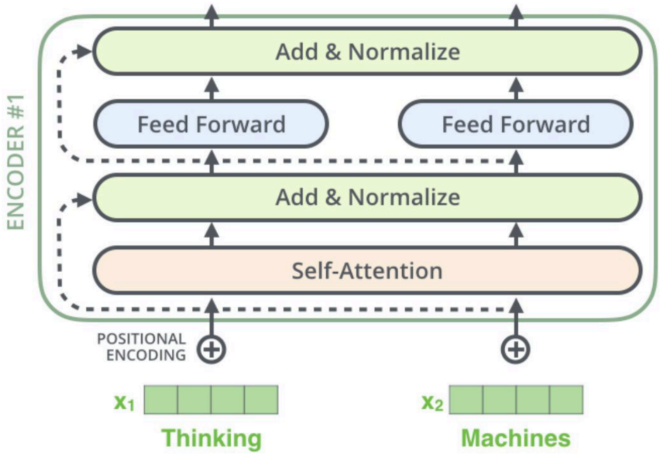
Итого


Layer: 5
Attention: Input - Input
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_
The_
animal_
didn_
'_
t_
cross_
the_
street_
because_
it_
was_
too_
tire
d_

Transformer full architecture



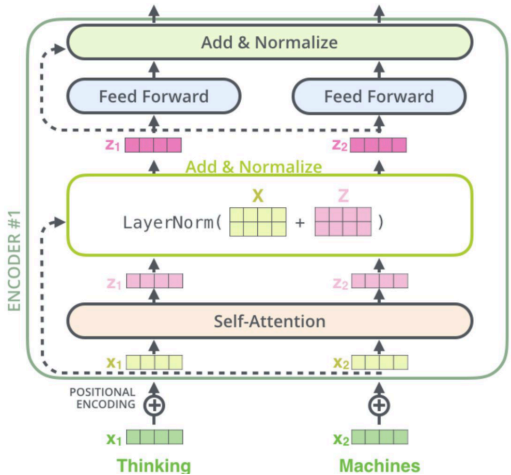
Layer normalization



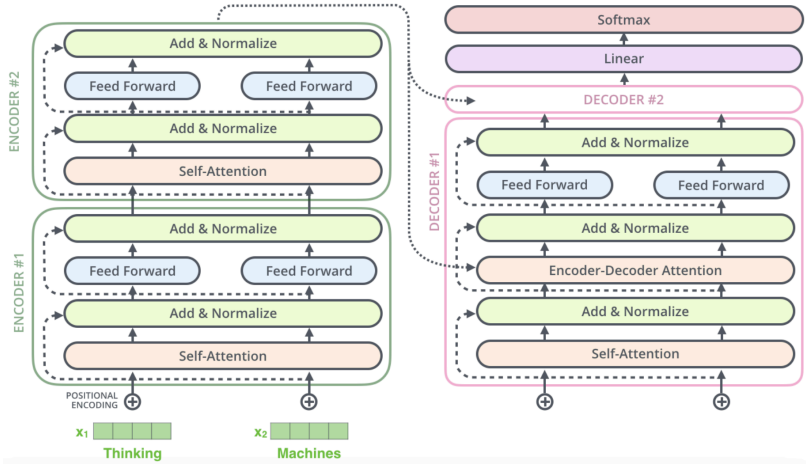
Layer normalization

Like BatchNorm

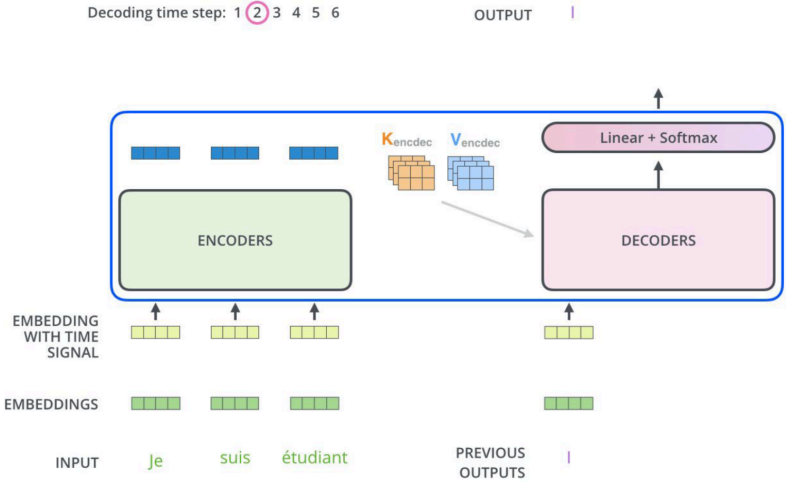
but normalize along all features representing latent vector



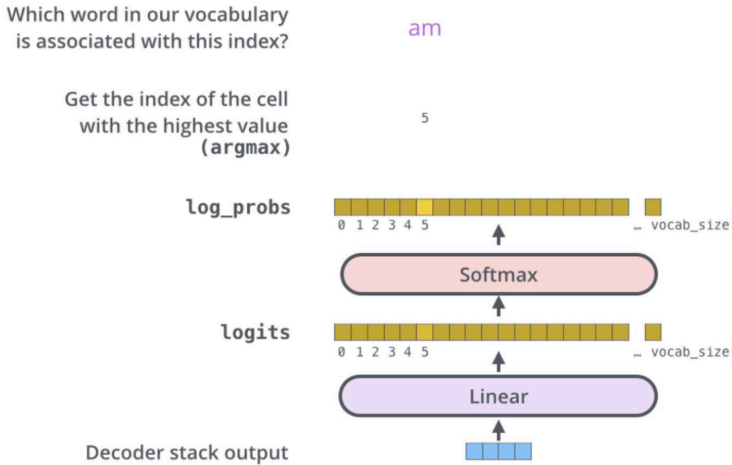
Transformer detailed



Decoder


Decoding time step: 1 2 3 4 5 6
OUTPUT
I
Linear + Softmax
K encdec
V encdec
ENCODERS
DECODERS
EMBEDDING WITH TIME SIGNAL
EMBEDDINGS
INPUT
Je
suis
étudiant
PREVIOUS OUTPUTS
I

Final steps


Which word in our vocabulary is associated with this index?
am
Get the index of the cell with the highest value (argmax)
5
log_probs
0 1 2 3 4 5
… vocab_size
Softmax
logits
0 1 2 3 4 5
… vocab_size
Linear
Decoder stack output

Итого

1. У нас нет никаких слоев, кроме dense
2. Учится очень классно, находит множество взаимосвязей
3. position encoding позволяет учитывать позицию в тексте

И понеслась!!! (развитие дальше - инженерные хаки и закидывание железом)

BERT

Крутой обзорчик с техническими деталями - живет в лекциях МФТИ.
Многие моменты этой лекции заимствованы из него.

Обзор

BERT

Шел 2018 год и гугл сказал - наши комьютеры самые мощные, а данные самые большие!
BERT - Bidirectional Encoder Representations from Transformers
Почему круто - придумали как предобучать без учителя (да, вот оно, вот он наш космос). И потом переиспользовать веса!

BERT

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step

Model:

Dataset:

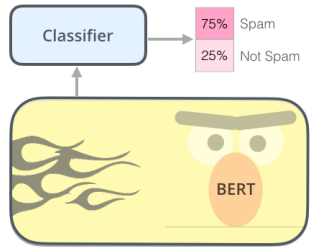
Objective:

Predict the masked word (language modeling)

2 - Supervised training on a specific task with a labeled dataset.

Supervised Learning Step

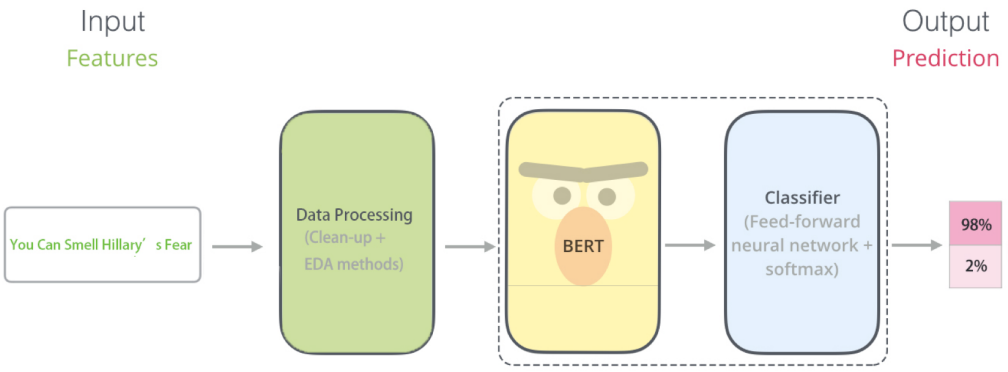
Model: (pre-trained in step #1)



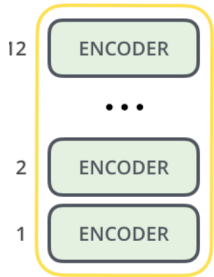
Dataset:

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

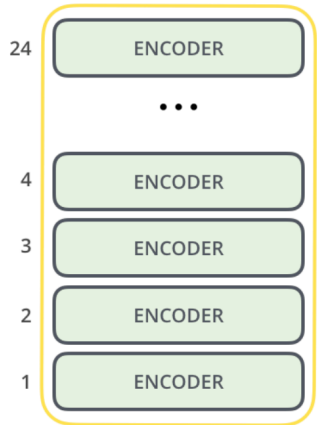
BERT


Input
Features
Output
Prediction
You Can Smell Hillary’ s Fear
Data Processing
(Clean-up +
EDA methods)
BERT
Classifier
(Feed-forward
neural network +
softmax)
98%
2%

BERT: Base, Large



$BERT_{BASE}$



$BERT_{LARGE}$

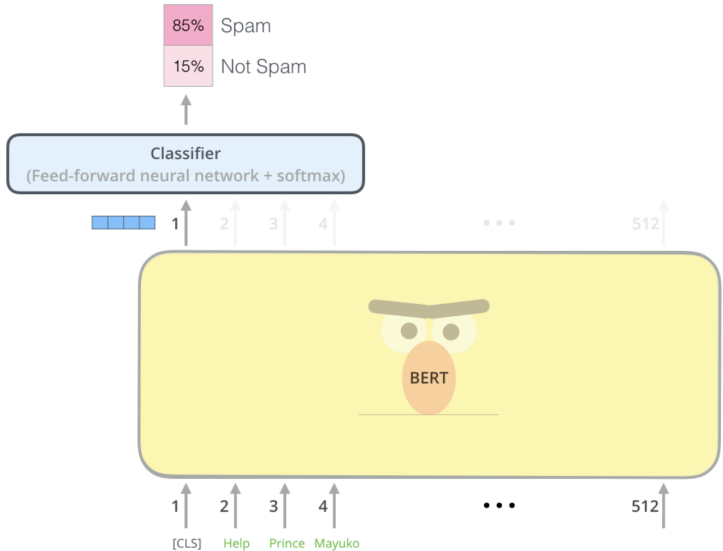
Лежащие внутри идеи и почему он популярный

1. Предобучаем по двум задачам - берем корпус текстов и маскируем часть предложений, заставляем учить и предсказывать маску.
2. И вторая идея - предсказываем следующее слово в предложении
3. Он из коробки знает язык, ему 1-2 эпохи надо подсказать, что с этим знанием делать
4. В готовых либах лежат много готовых под задачи бертов - классификация, вопросно - ответные системы и тому подобное.
5. Опять же - подаем слова кусочками, чтобы как-то решать проблему oov.

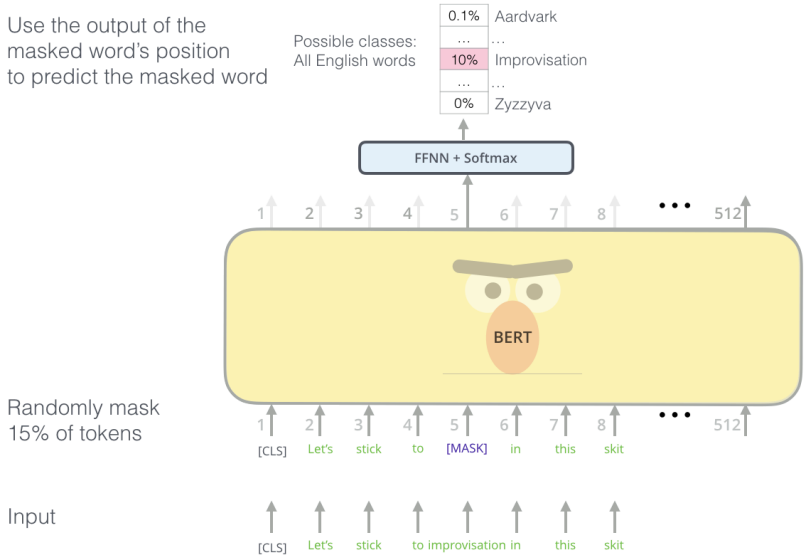
Все мы всех победили - нам не нужно размечать данные для обучения, мы счастливы!

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

Pretraining. Task 1: Masked LM



Pretraining. Task 2: Next Sentence Prediction



Где использовать?

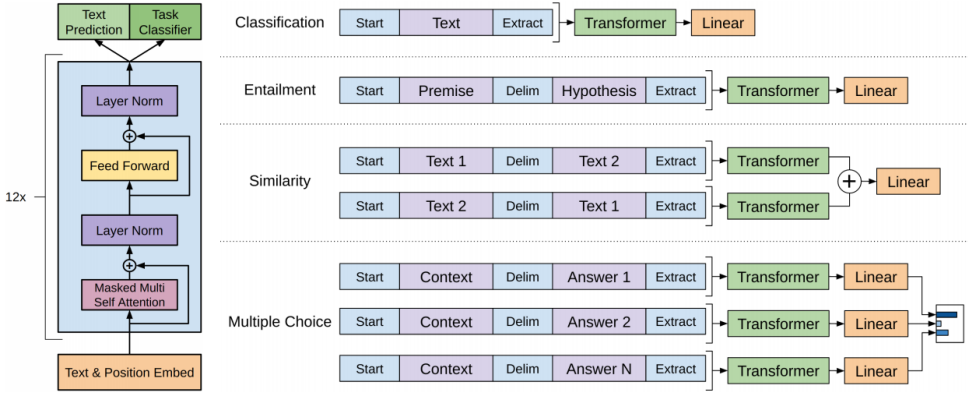
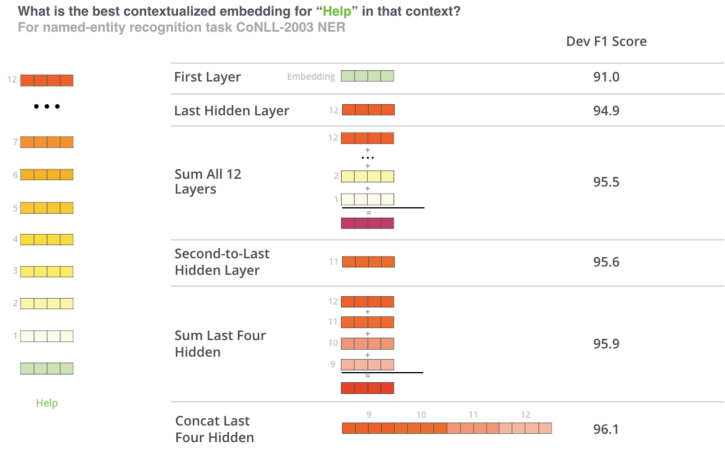


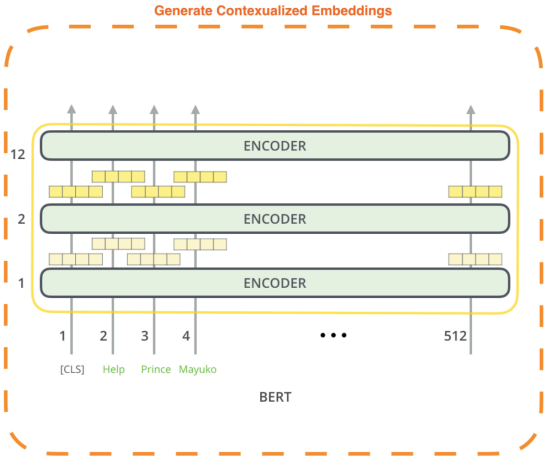
Figure 1: **(left)** Transformer architecture and training objectives used in this work. **(right)** Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

BERT for feature extraction

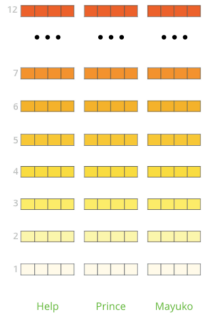
The fine-tuning approach isn't the only way to use BERT. Just like ELMo, you can use the pre-trained BERT to create contextualized word embeddings.



BERT for feature extraction



The output of each encoder layer along each token's path can be used as a feature representing that token.



But which one should we use?

Немного статистики

Model Type	Vocab Size	Hidden Dim	# Params	Model Size (MB)	FLOPS ratio
$\text{BERT}_{\text{DISTILLED}}$	4928	48	1,775,910	6.8	1.3%
		96	5,665,926	22	1.32%
		192	19,169,094	73	4.49%
$\text{BERT}_{\text{BASE}}$	30522	768	110,106,428	420	100%