

Глубокое обучение и вообще

Соловей Влад и Андросов Дмитрий

13 сентября 2022 г.

Посиделка Внимательная: Быстрое введение в SOTA

Agenda

- Быстрая история
- seq2seq
- Attention
- Self-attention
- BERT
- ELMO
- Сломанный мозг.....

Быстремькая история взятая из старых лекций

задача seq2seq

спойлер

После этой лекции могут возникнуть огромное количество вопросов - но в современных архитектурах слишком много инженерных хаков, которые лучше осознавать постепенно сами.

Я буду оставлять некоторые ключевые слова того, чтобы вы могли сами залезть поглубже, если такое погружение потребуется.

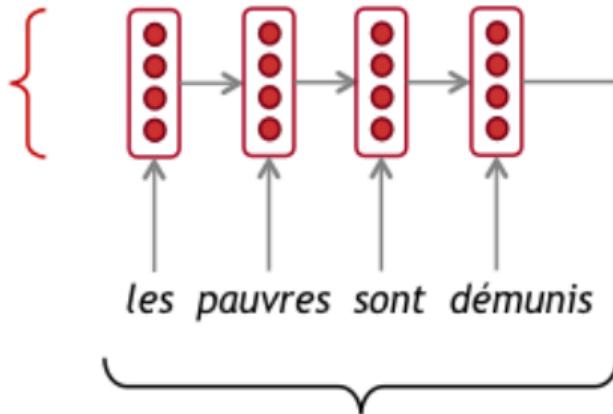
Задача seq2seq - задача, когда мы хотим предсказать по одной последовательности другую Самая стандартная подобная задача - машинный перевод. Нейронные сети ворвались в эту сферу человеческого прогресса в 2014 году

Метрика

Модели в машинном переводе сравнивают по BLEU score - если в кратце, то эта метрика сравнения полученного машиной перевода и человеческого, насколько мы вообще бьемся. Из проблем данной метрики - если машина перевела правильно, но альтернативно, то BLEU будет низкий....

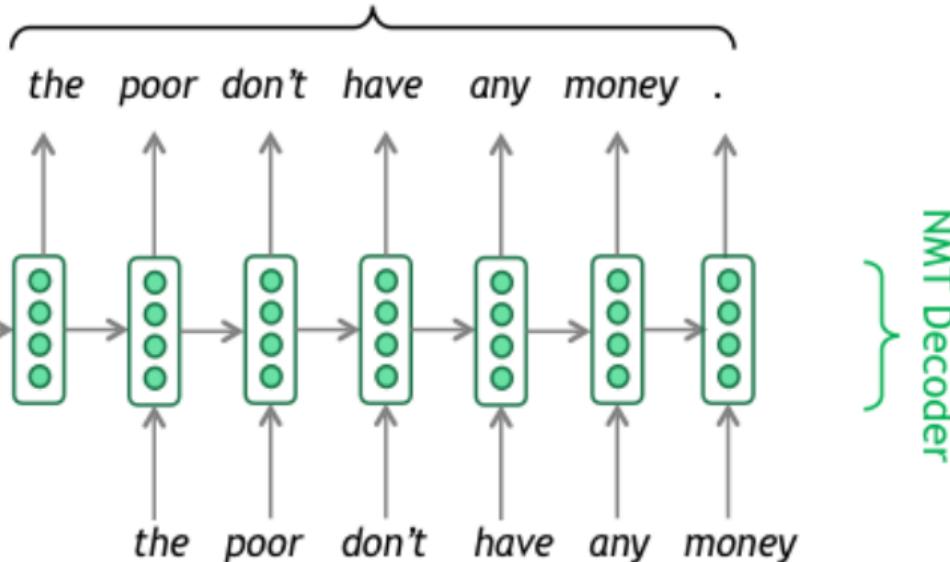
NMT Encoder

We feed in each word from left to right, one at a time. By the end, the NMT system has encoded information about the whole sentence in a numerical format.



French sentence (input)

English translation (output)



The previous outputted word gets added as part of the input into the network next, giving the network some view of the sentence already produced and some context of the words preceding it.

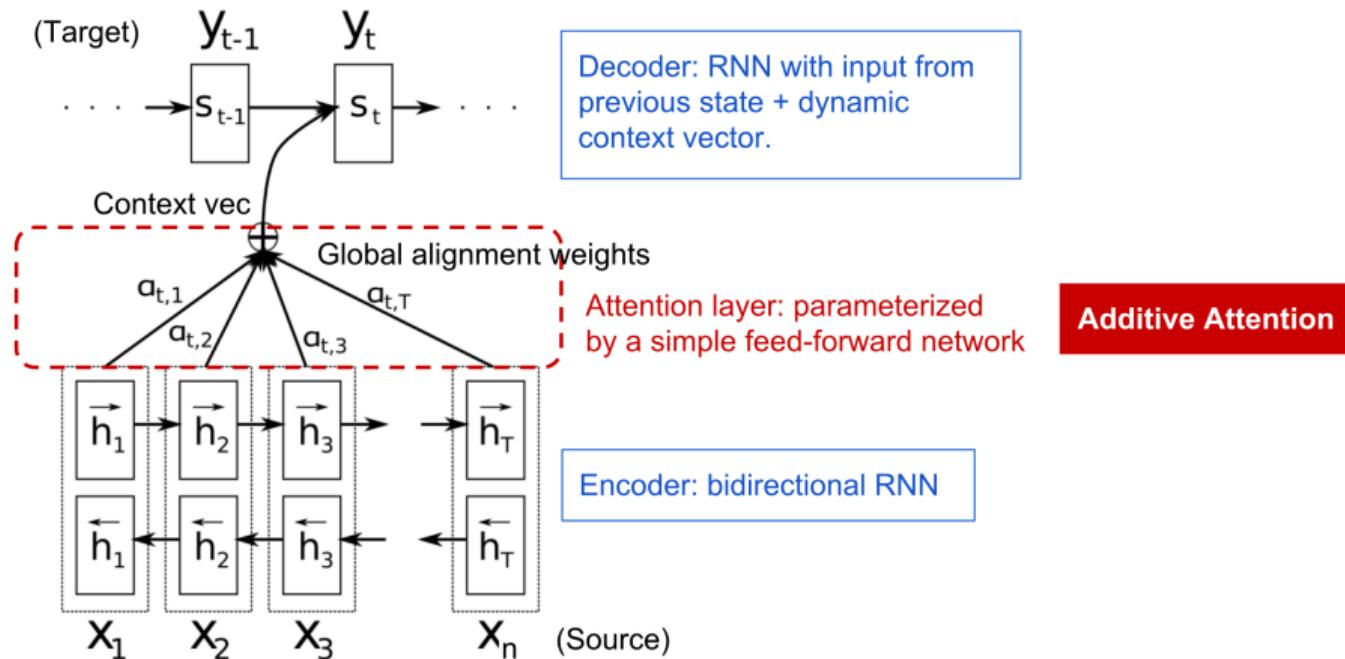
Attention!

Attention

А вот бы использовать не один вектор, а все. Информация то течет и кодируется во всех векторах.....

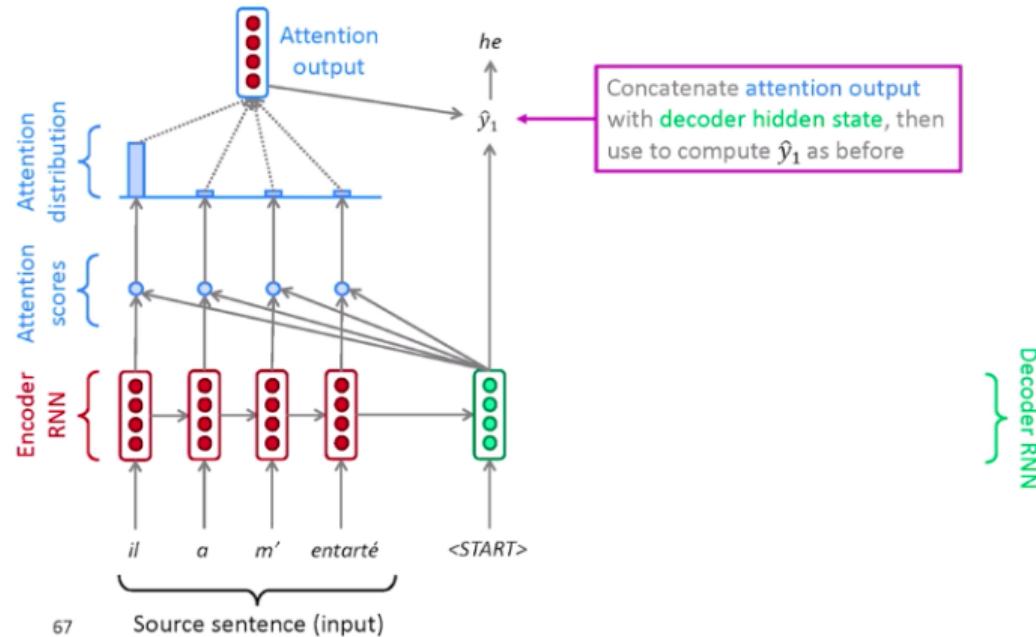
И да - это классная и разумная идея. Нам на встречу приходит концепция внимания.

Attention



Attention

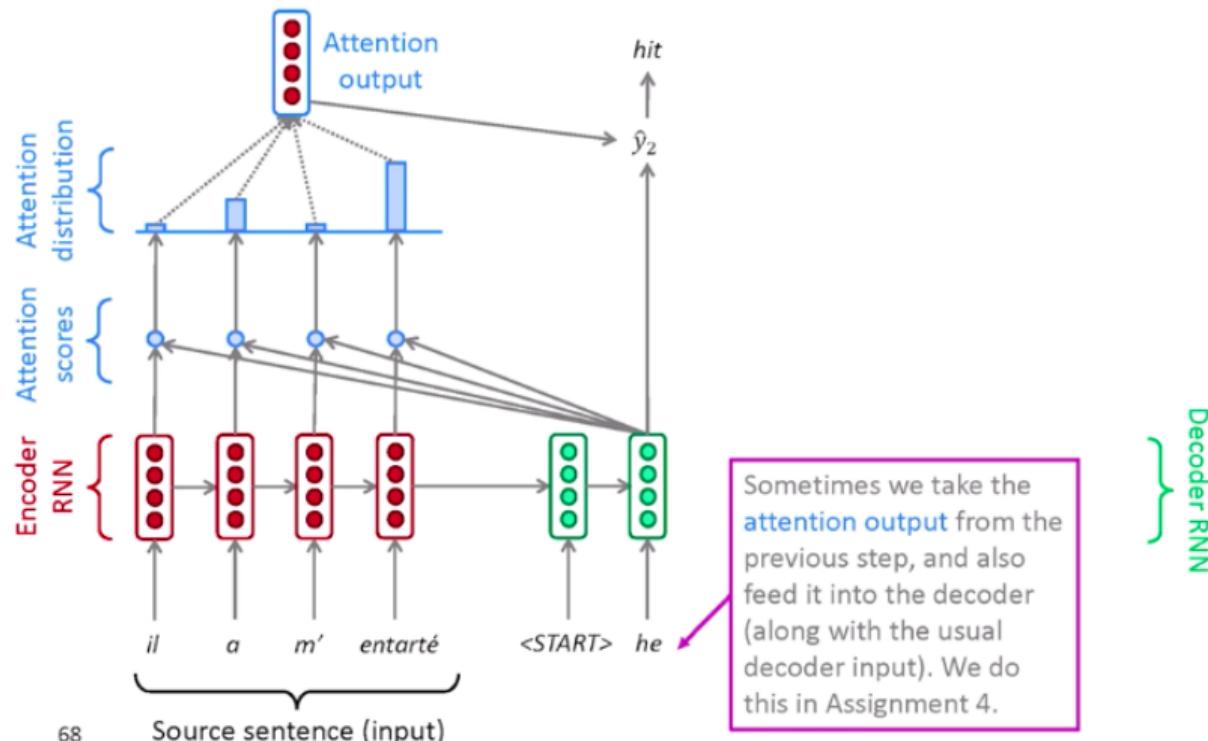
Sequence-to-sequence with attention



ссылочка на оригинал

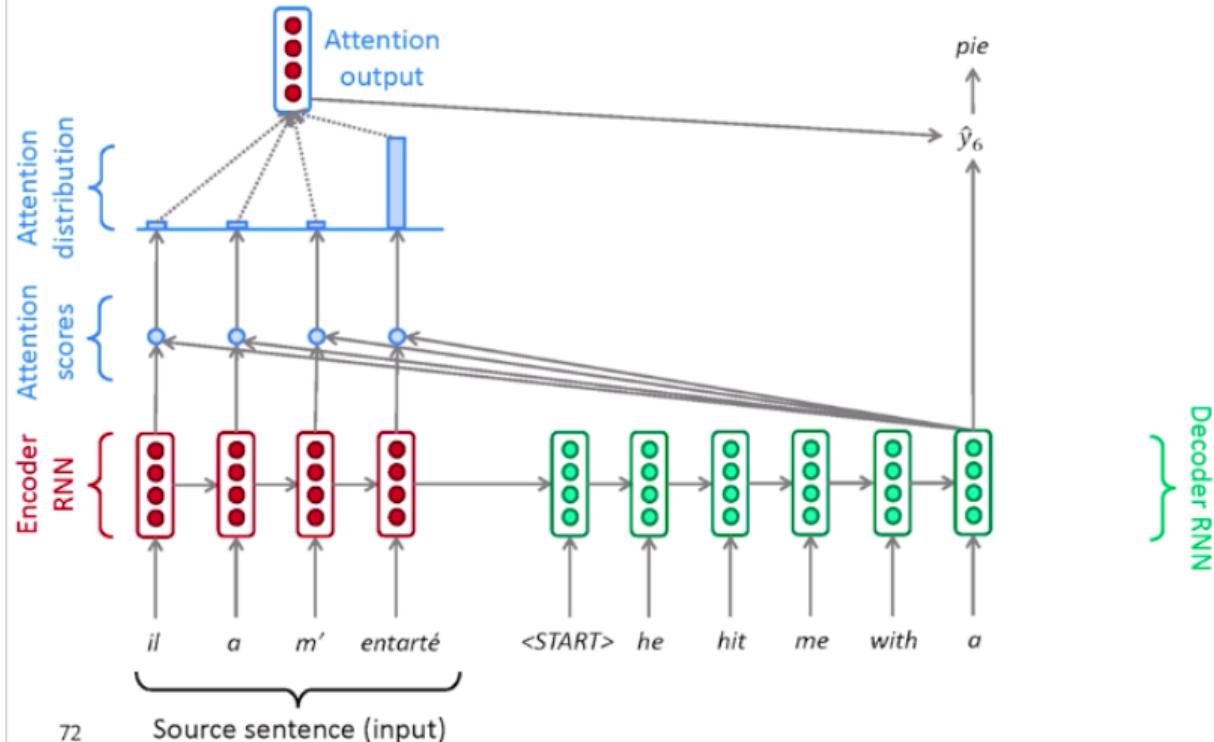
Attention

Sequence-to-sequence with attention

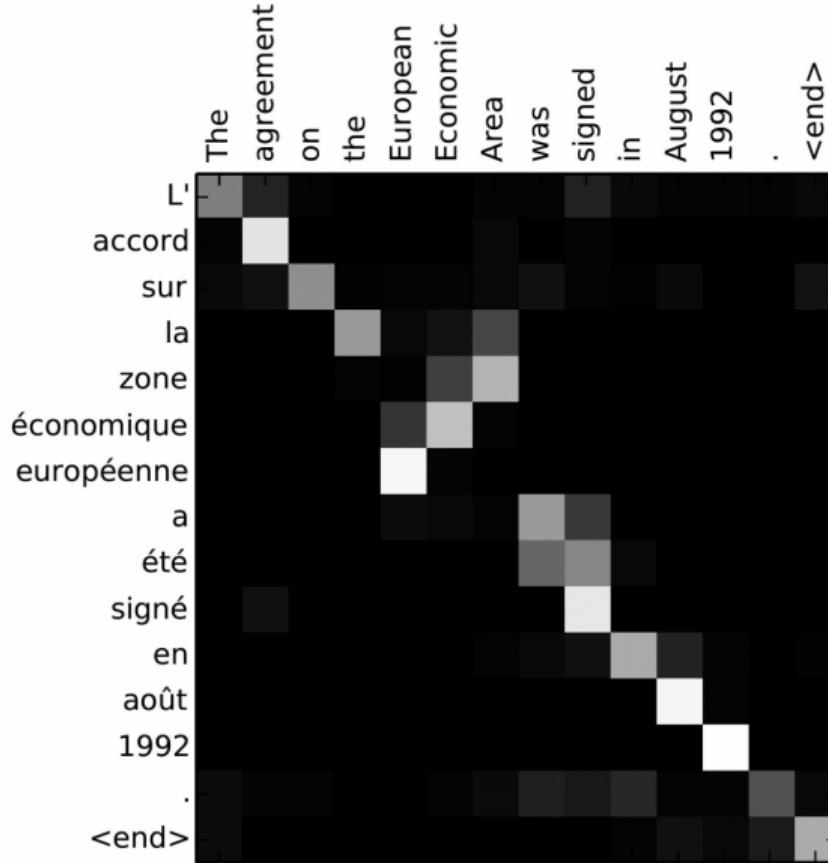


Attention

Sequence-to-sequence with attention



Attention



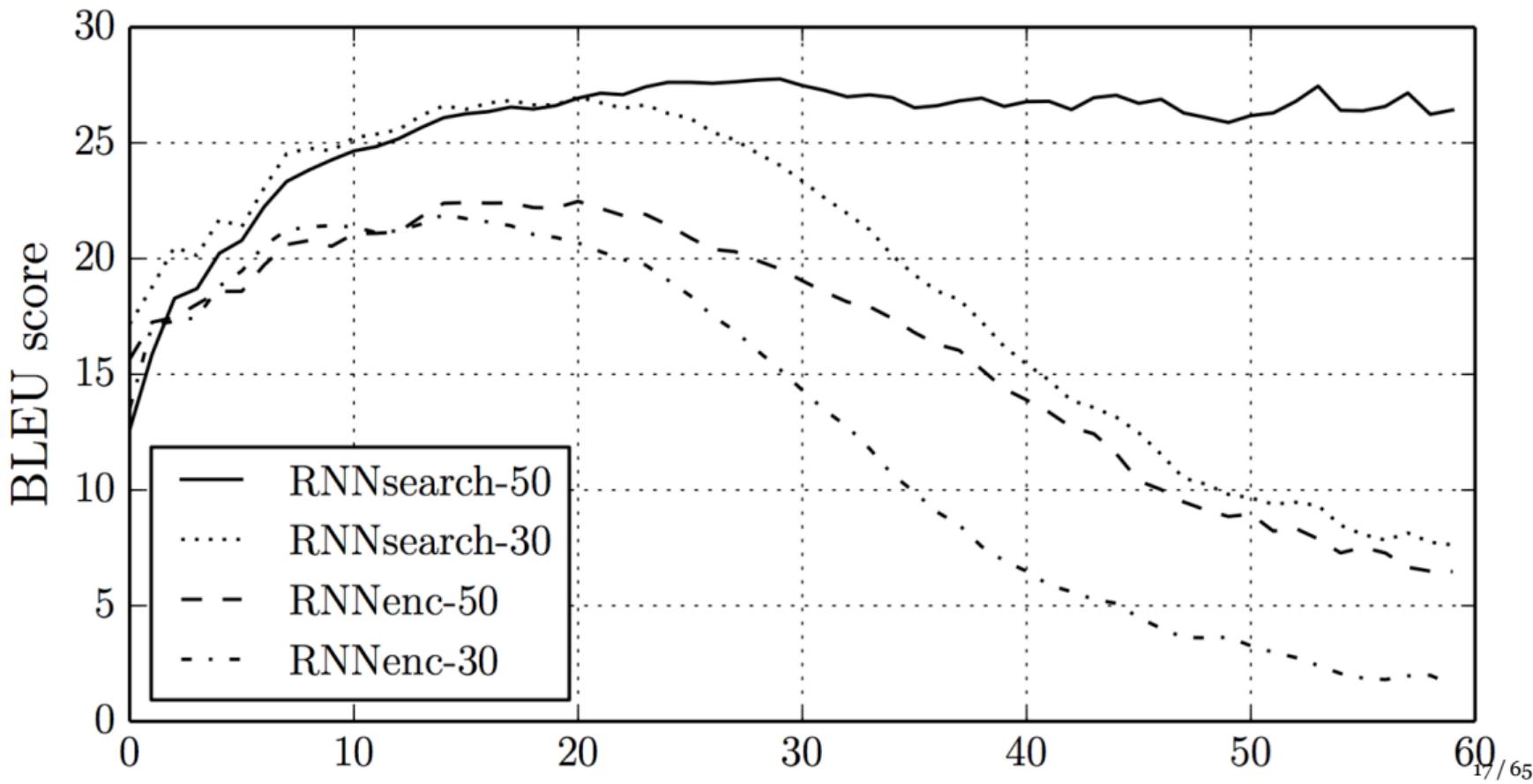
Attention

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

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

Источник: <https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture08-nmt.pdf>

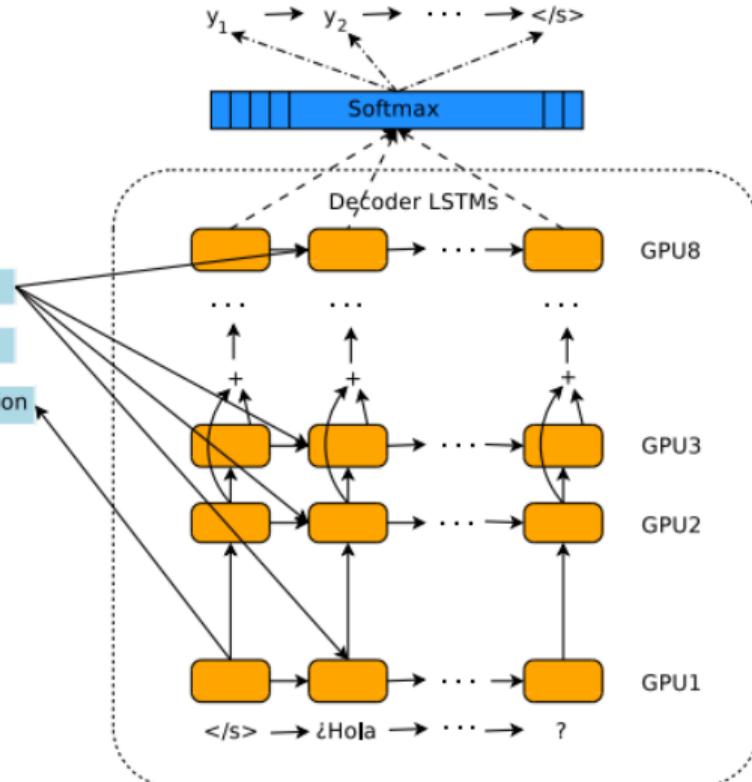
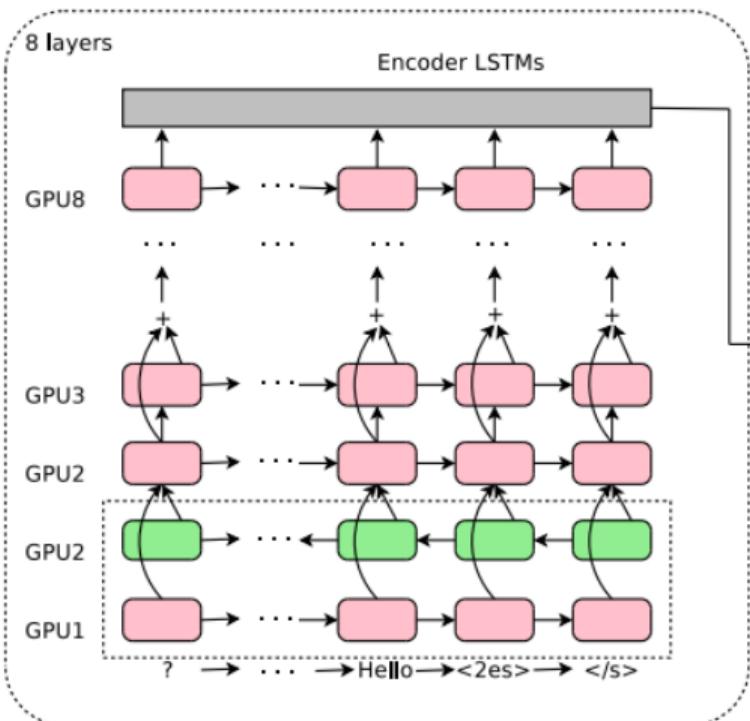
Attention



Attention

Идейно - внимание просто выбирает то из эмбедингов, которое действительно нужно для декодирования. Это просто матричное произведение(а можно взвешивать и без весов) и softmax. У нас все остается дифференцируемым - берем градиентны, накапливаем инфу в весах сетки.

google



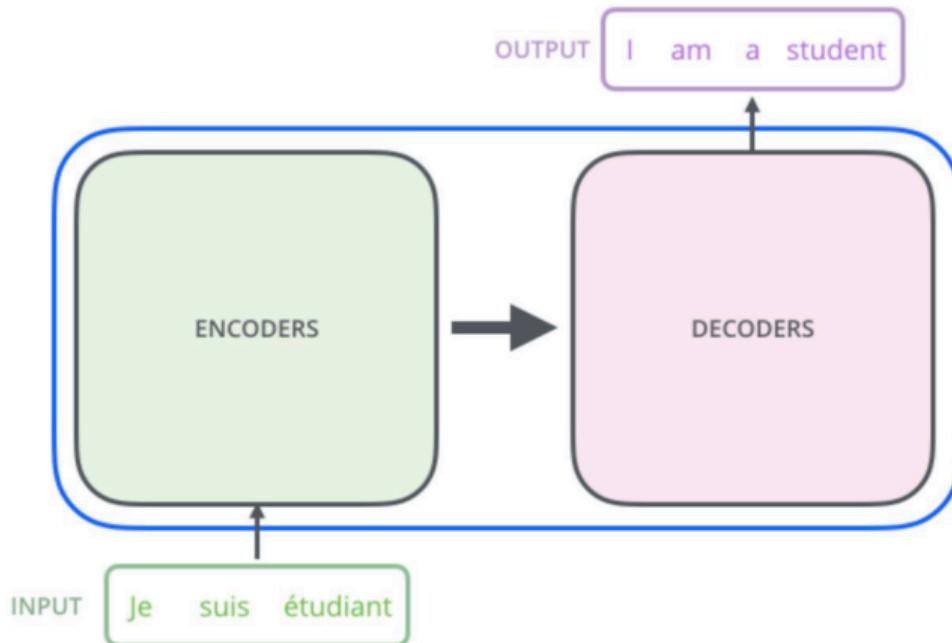
В целом глобальное решение было найдено, осталось закидать проблему железом.

Выводы:

1. 8 слоев LSTM (8 Карл!)
2. в attention 2 слоя dense.
3. Собираем слова из морфем - пытаемся победить out-of-vocabulary.
4. Модель стала иногда сексистом и фашистом - требуются слишком большие данные сети, чтобы учить эту большую прелесть.

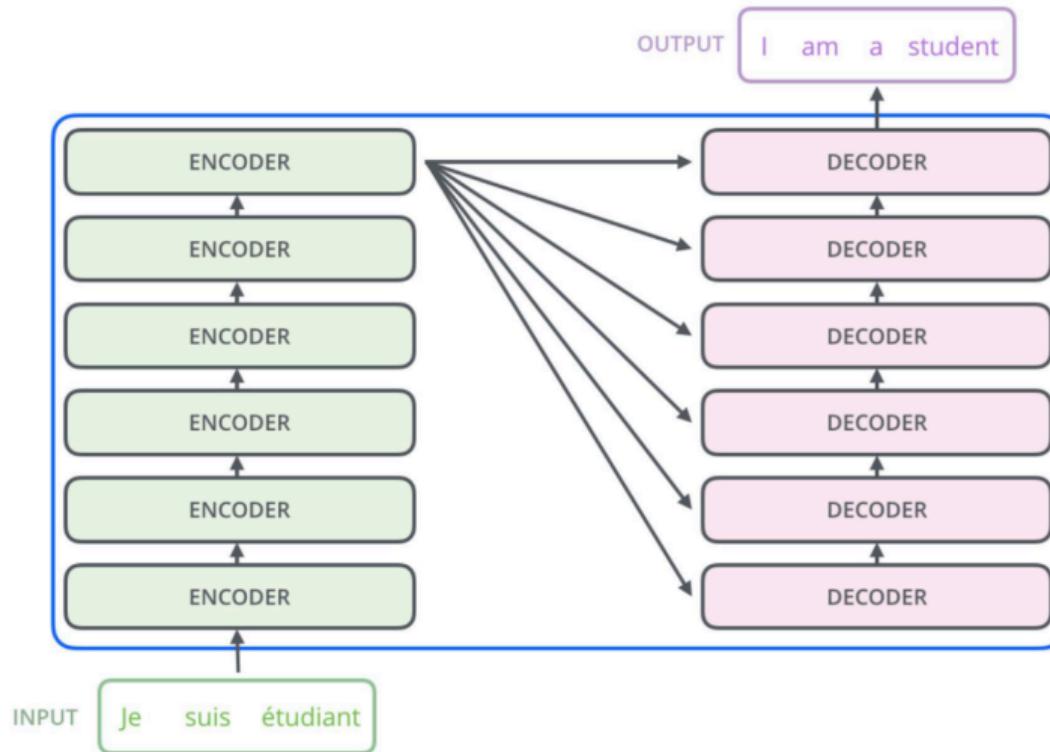
Attention is all you need!

Transformer



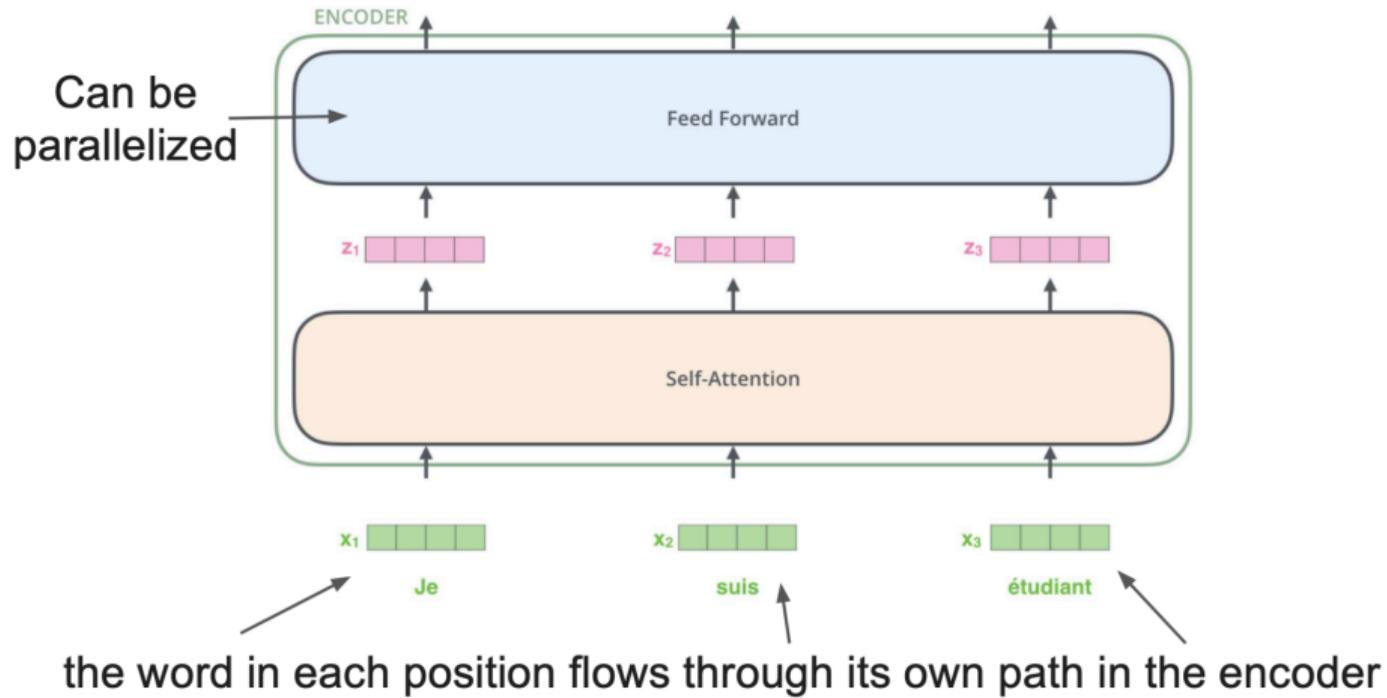
Источник: <https://jalammar.github.io/illustrated-transformer/>

Transformer



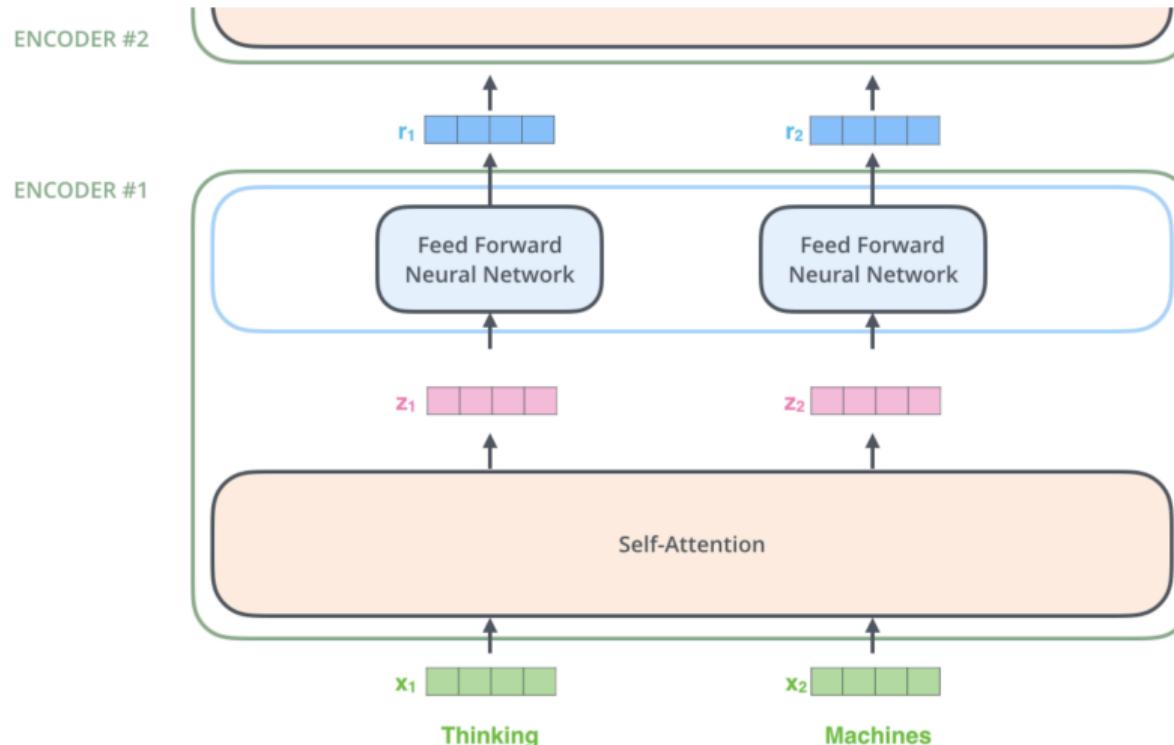
Источник: <https://jalammar.github.io/illustrated-transformer/>

Transformer



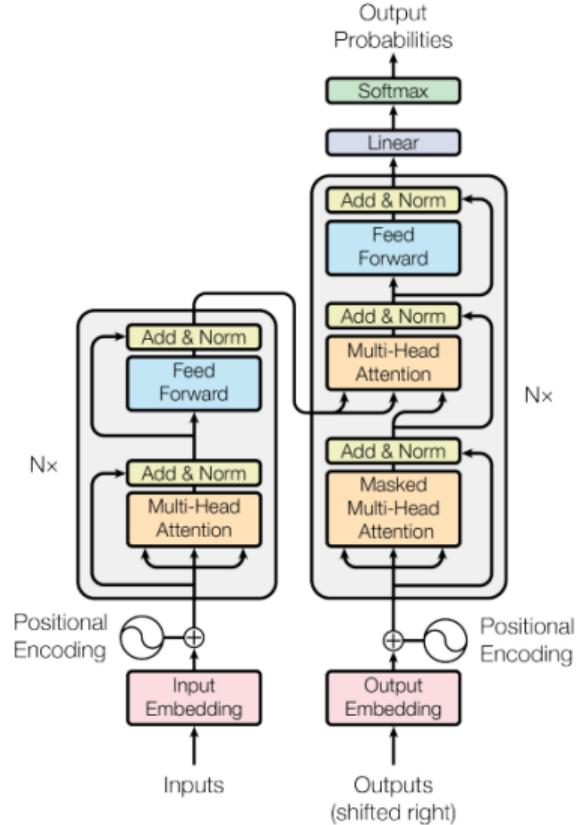
Источник: <https://jalammar.github.io/illustrated-transformer/>

Transformer

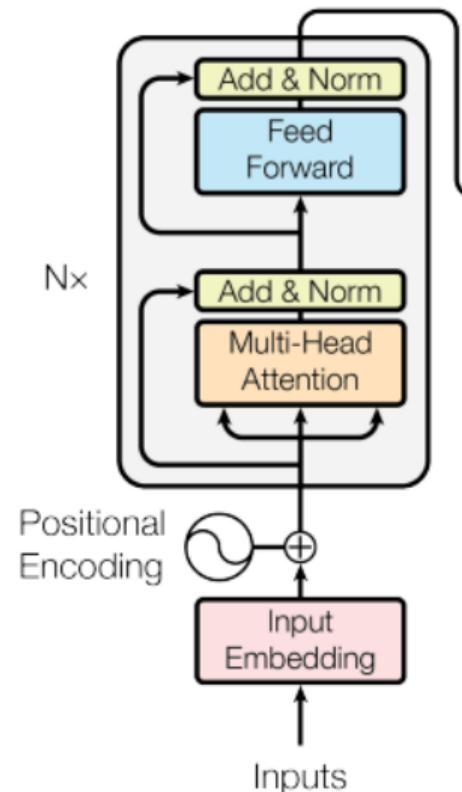


Источник: <https://jalammar.github.io/illustrated-transformer/>

attention is all you need

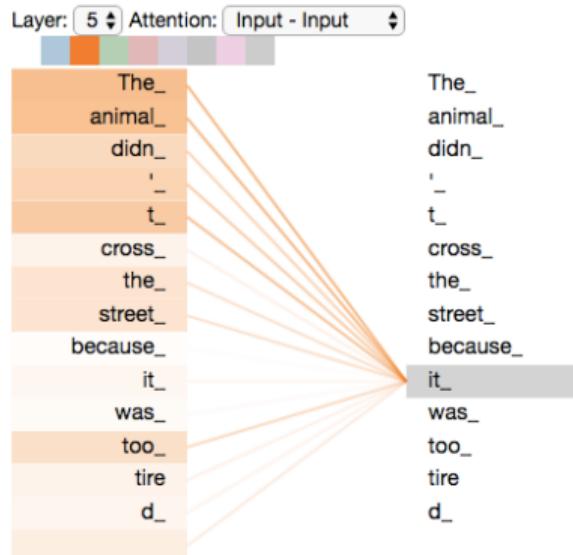


Encoder

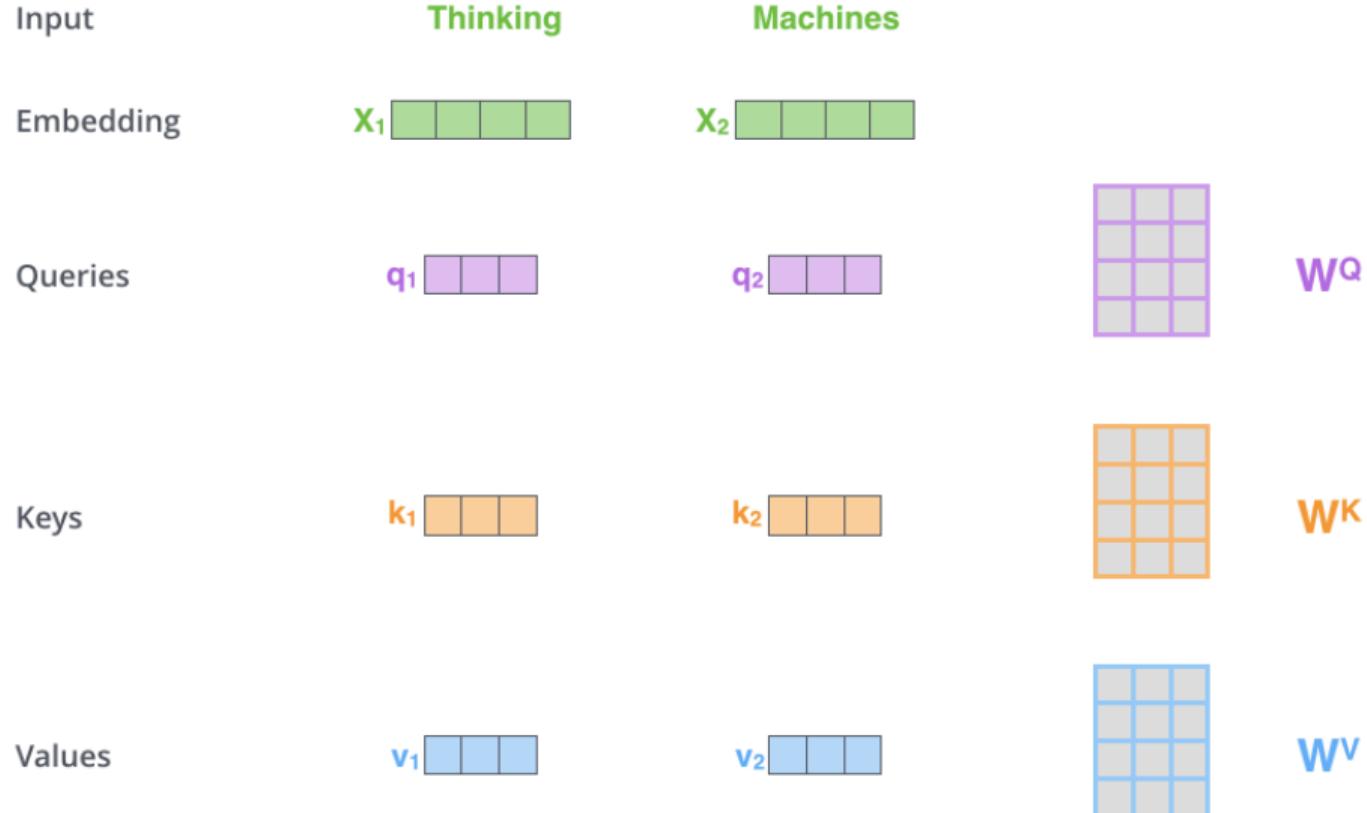


ЧТО МЫ ХОТИМ?

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



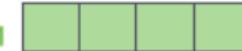
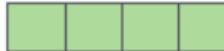
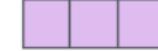
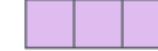
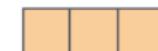
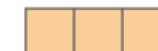
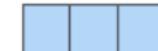
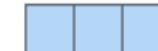
Абстракции!



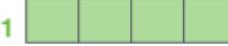
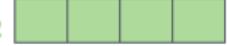
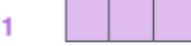
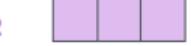
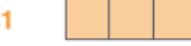
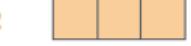
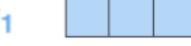
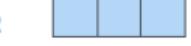
А теперь тоже самое, но словами:

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

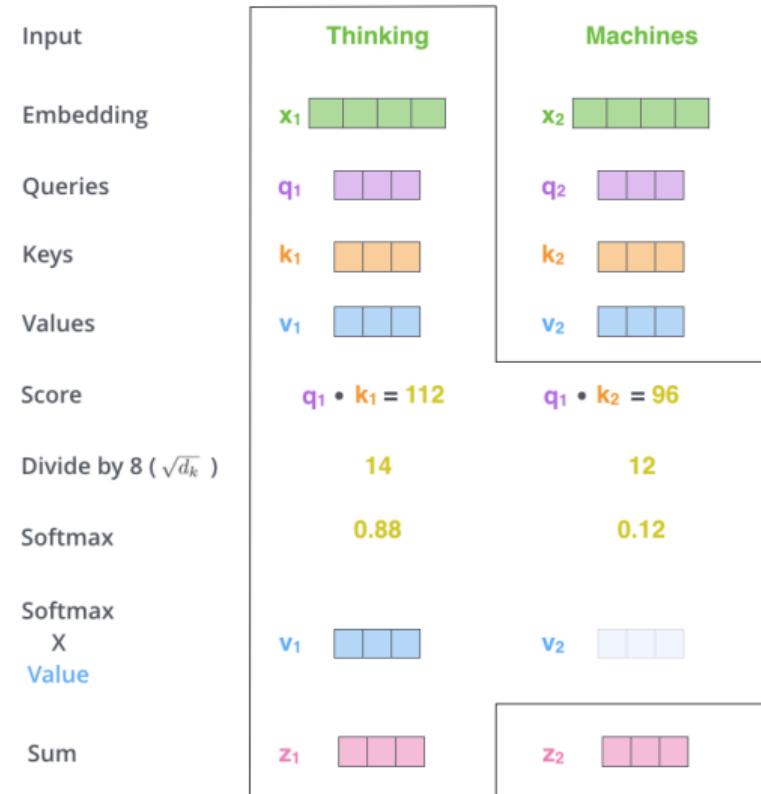
Шаг 2

Input	Thinking		Machines	
Embedding	x_1		x_2	
Queries	q_1		q_2	
Keys	k_1		k_2	
Values	v_1		v_2	
Score	$q_1 \cdot k_1 = 112$		$q_1 \cdot k_2 = 96$	

Шаг 3

Input	Thinking		Machines	
Embedding	x_1		x_2	
Queries	q_1		q_2	
Keys	k_1		k_2	
Values	v_1		v_2	
Score	$q_1 \cdot k_1 = 112$		$q_1 \cdot k_2 = 96$	
Divide by 8 ($\sqrt{d_k}$)	14		12	
Softmax	0.88		0.12	

Шаг 4



Шаг 5

$$X \times W^Q = Q$$

A diagram illustrating matrix multiplication. On the left, a green 3x3 matrix labeled 'X' is multiplied by a purple 3x3 matrix labeled 'W^Q'. The result is a purple 3x3 matrix labeled 'Q'. The matrices are represented as 3x3 grids of colored squares.

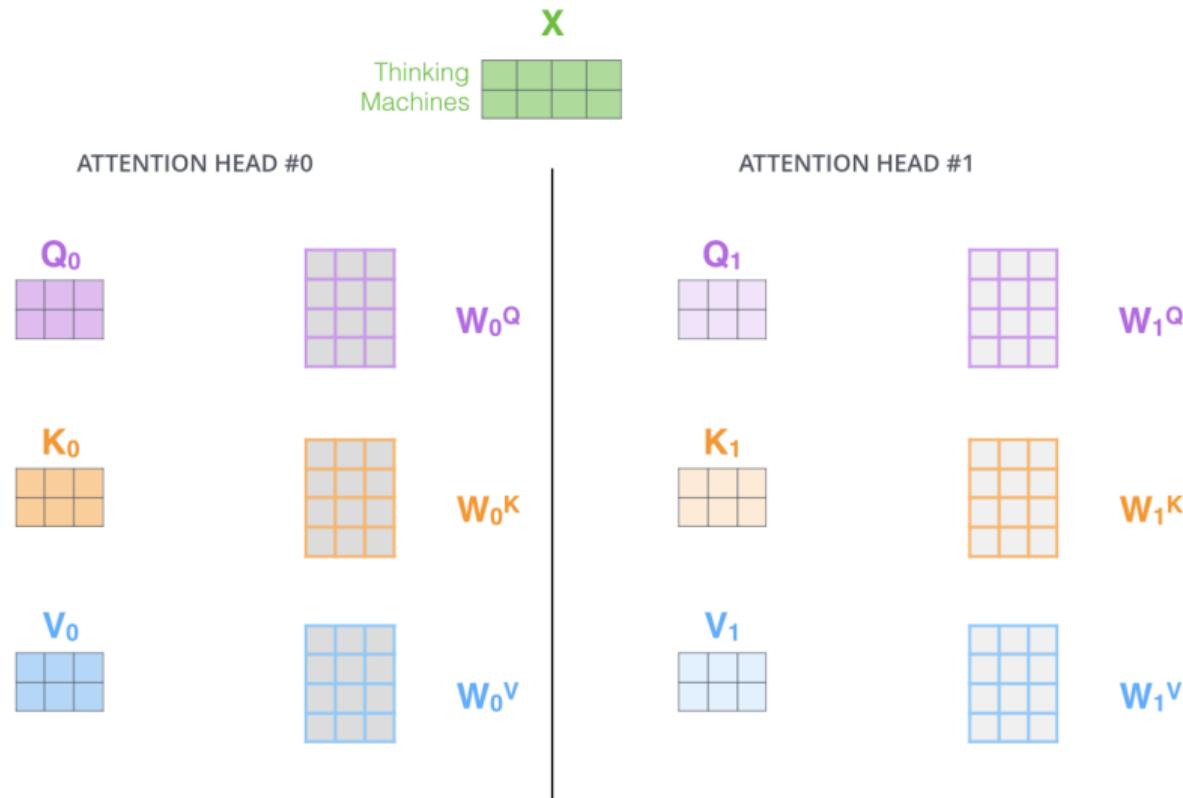
$$X \times W^K = K$$

A diagram illustrating matrix multiplication. On the left, a green 3x3 matrix labeled 'X' is multiplied by an orange 3x3 matrix labeled 'W^K'. The result is an orange 3x3 matrix labeled 'K'. The matrices are represented as 3x3 grids of colored squares.

$$X \times W^V = V$$

A diagram illustrating matrix multiplication. On the left, a green 3x3 matrix labeled 'X' is multiplied by a blue 3x3 matrix labeled 'W^V'. The result is a blue 3x3 matrix labeled 'V'. The matrices are represented as 3x3 grids of colored squares.

multi head attention



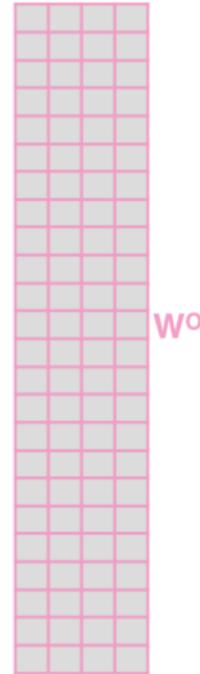
Соединяем!

1) Concatenate all the attention heads



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

\times



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

$$= \begin{matrix} Z \\ \hline \end{matrix}$$

The diagram shows an equals sign followed by a horizontal line. Above the line is the letter Z in red. Below the line is a 3x3 grid of pink squares, representing the concatenated matrix Z .

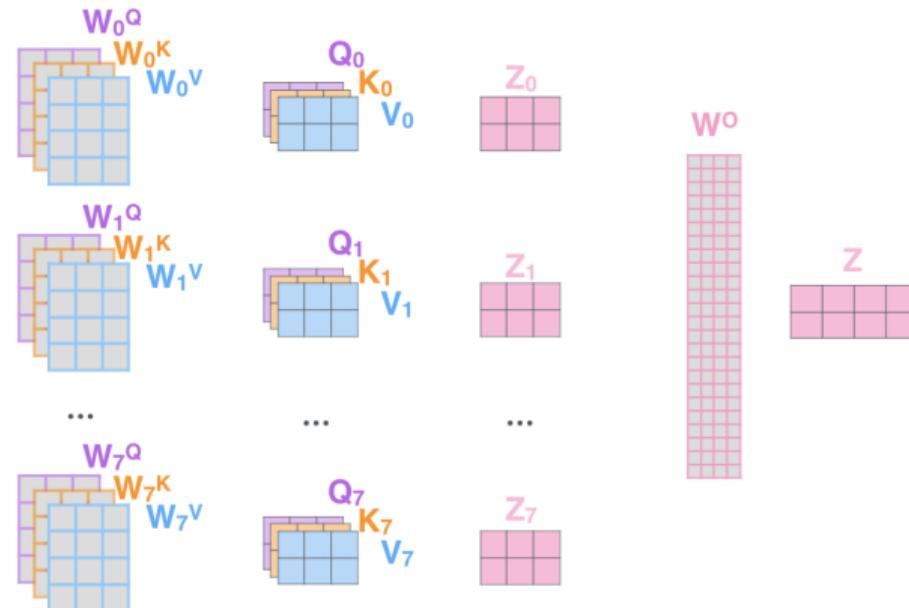
Смотрим раз

1) This is our input sentence*
each word*



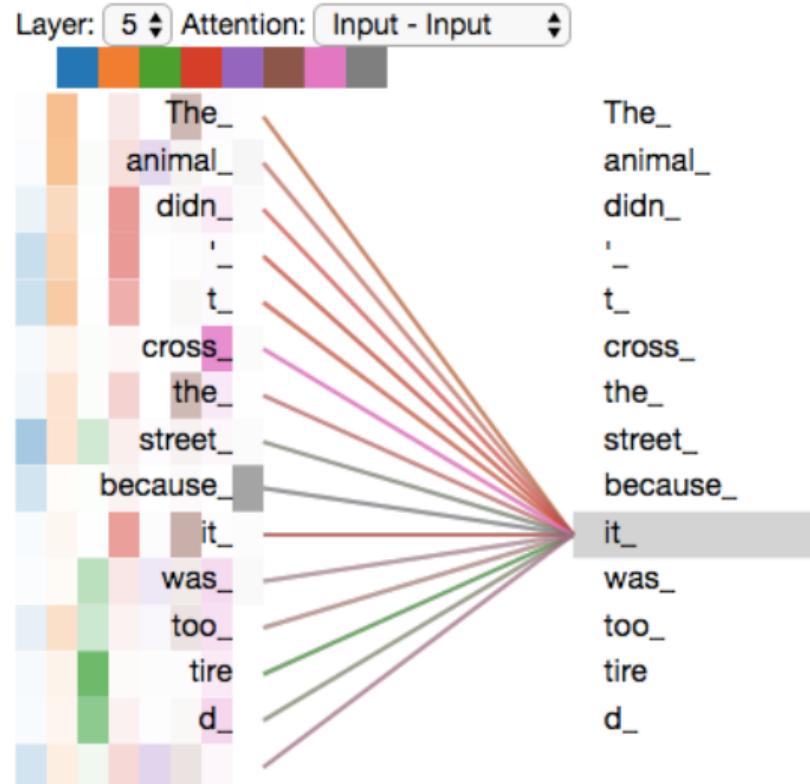
2) We embed each word*

3) Split into 8 heads.
We multiply X or R with weight matrices

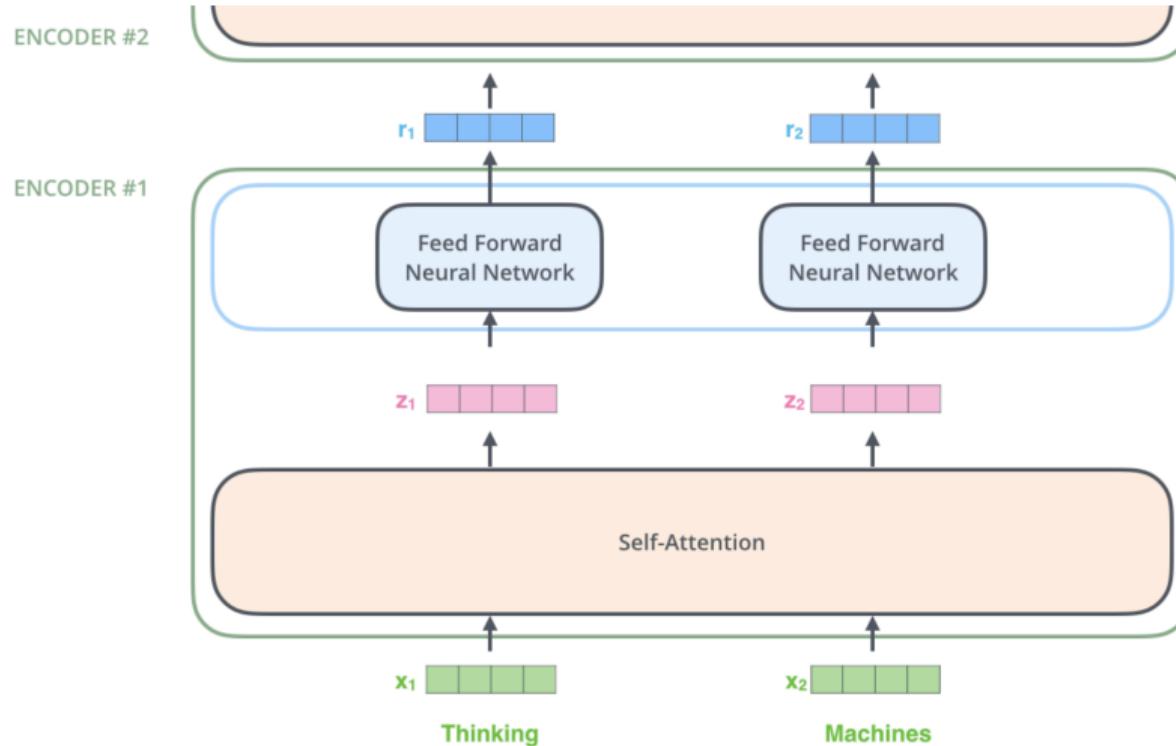


* In all encoders other than #0, we don't need embedding.
We start directly with the output of the encoder right below this one

Смотрим два

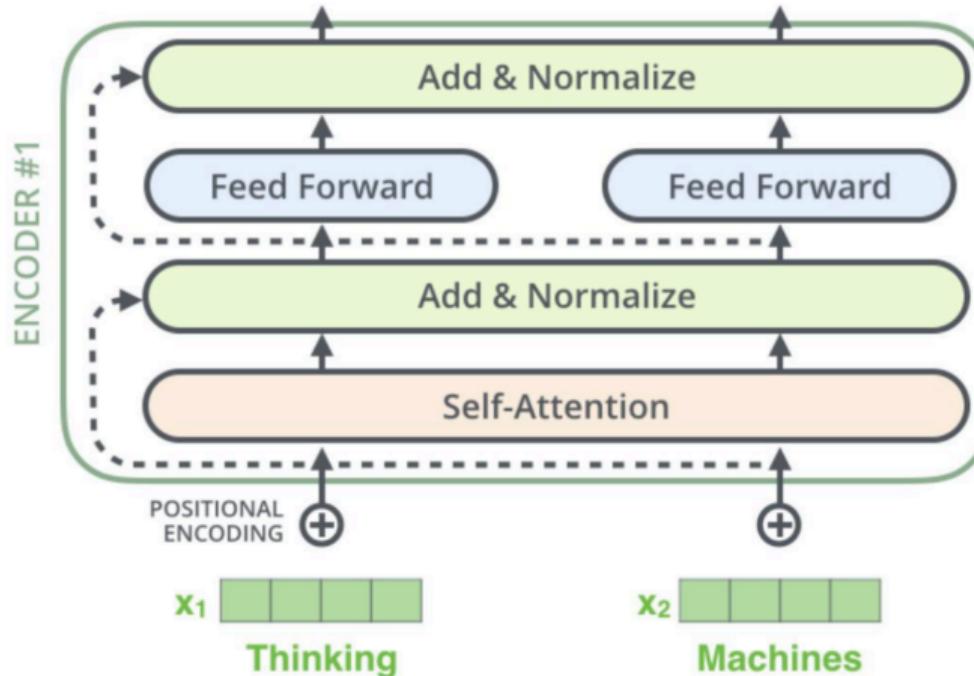


До сих пор архитектура Transformer выглядела так



Источник: <https://jalammar.github.io/illustrated-transformer/>

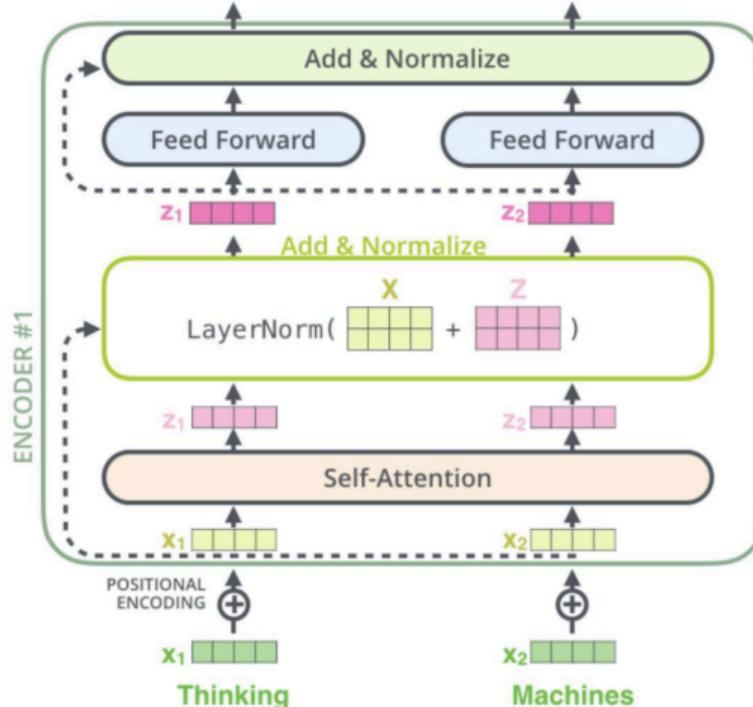
Углубимся в детали: Layer normalization



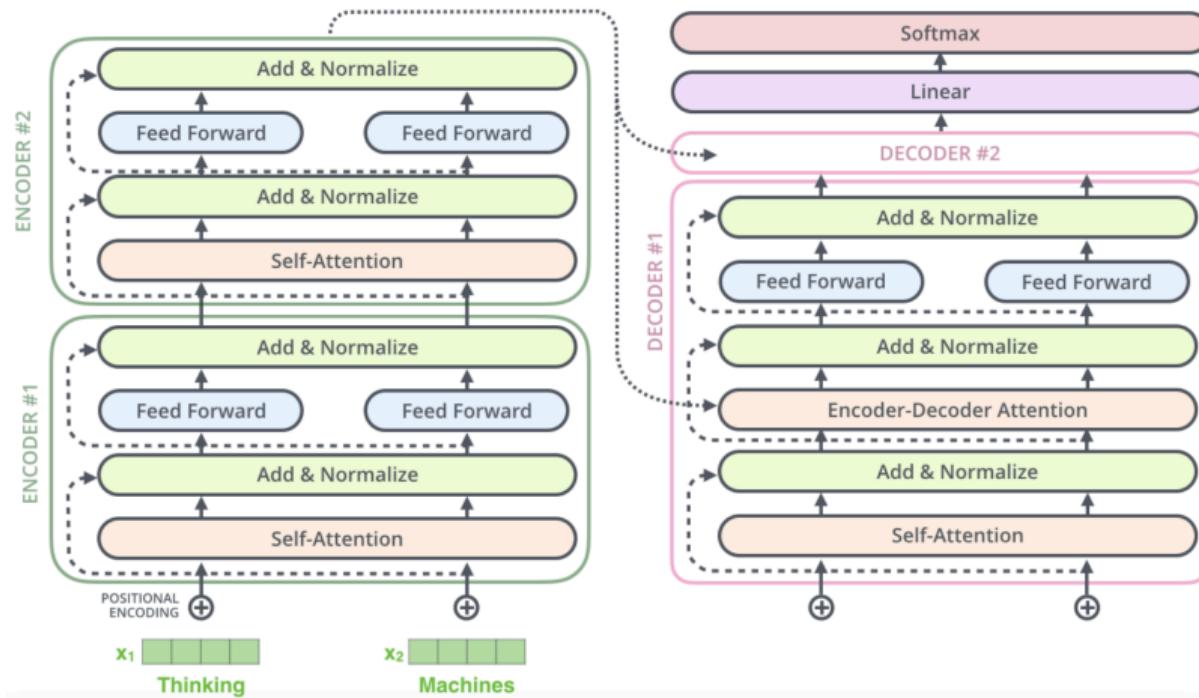
Layer normalization

Like BatchNorm

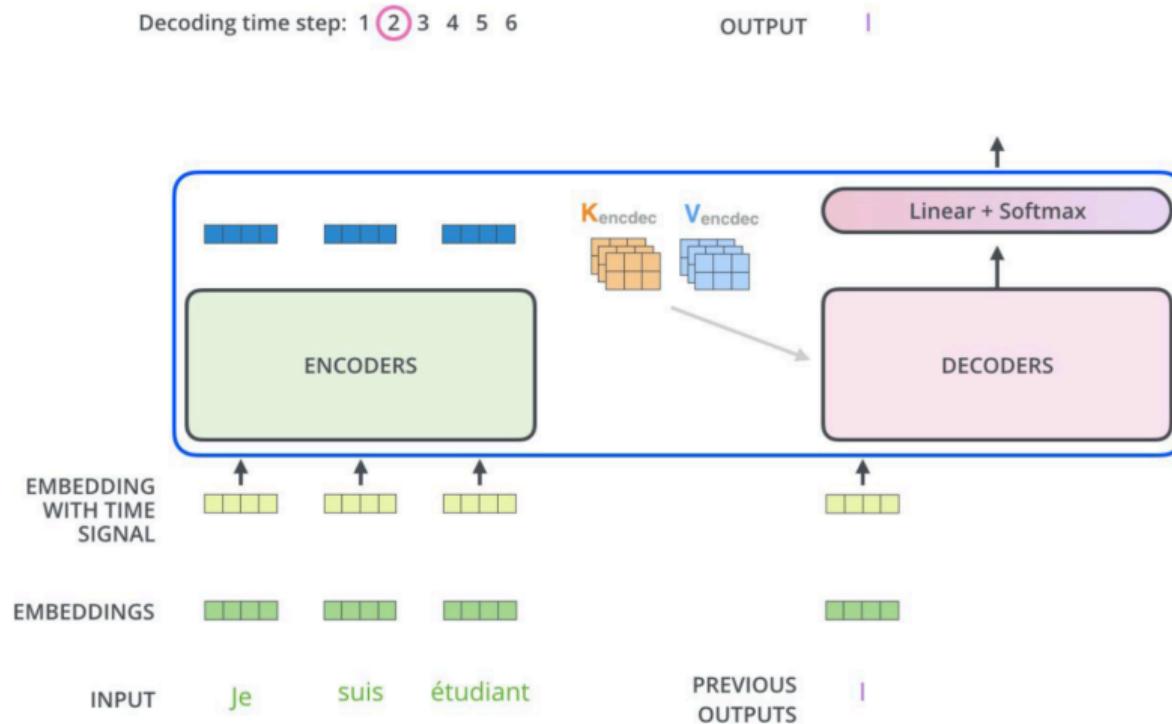
but normalize along
all features
representing latent
vector



Transformer detailed



Encoder-Decoder Attention



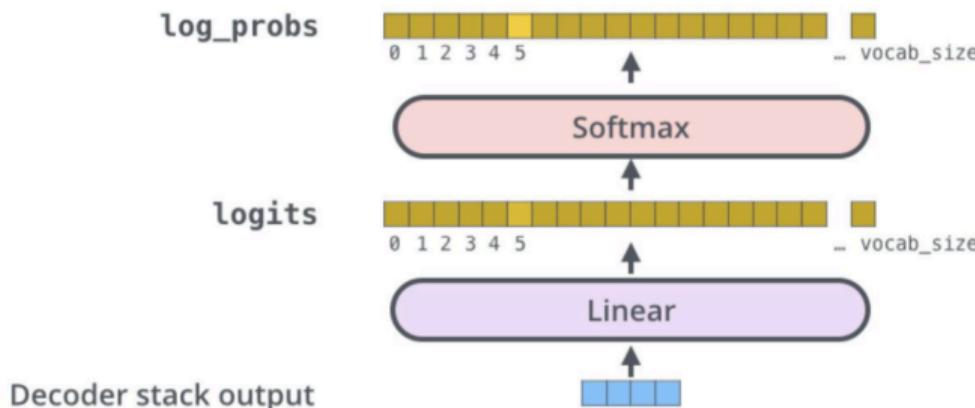
Final steps - Linear layer

Which word in our vocabulary
is associated with this index?

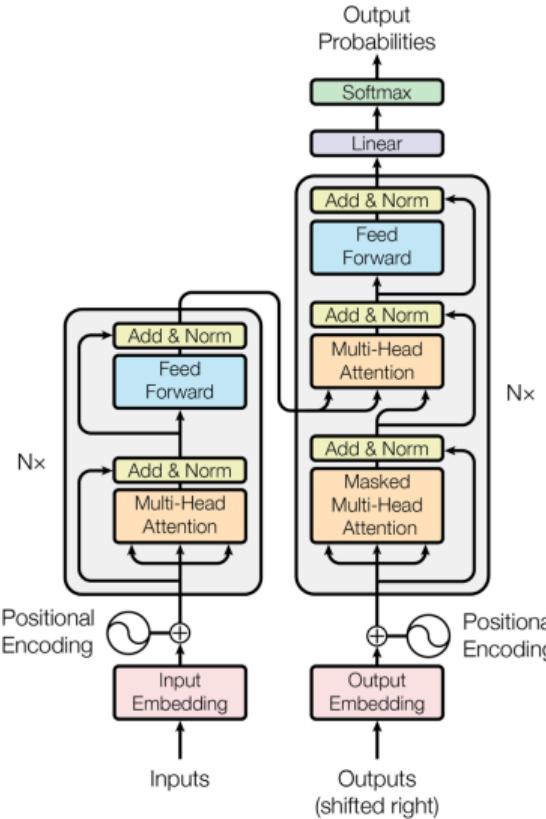
am

Get the index of the cell
with the highest value
(argmax)

5



Transformer full architecture once again



Итого

Выходом всего этого дела будут вектора key и value, которые позволяют декодеру смотреть на нужные нам кусочки. И бежим смотреть гифки декодера!

Объяснение взято отсюда **английский оригинал** и отсюда **лекции мфти**

Итого

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



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

SOTA (ну или история соты)

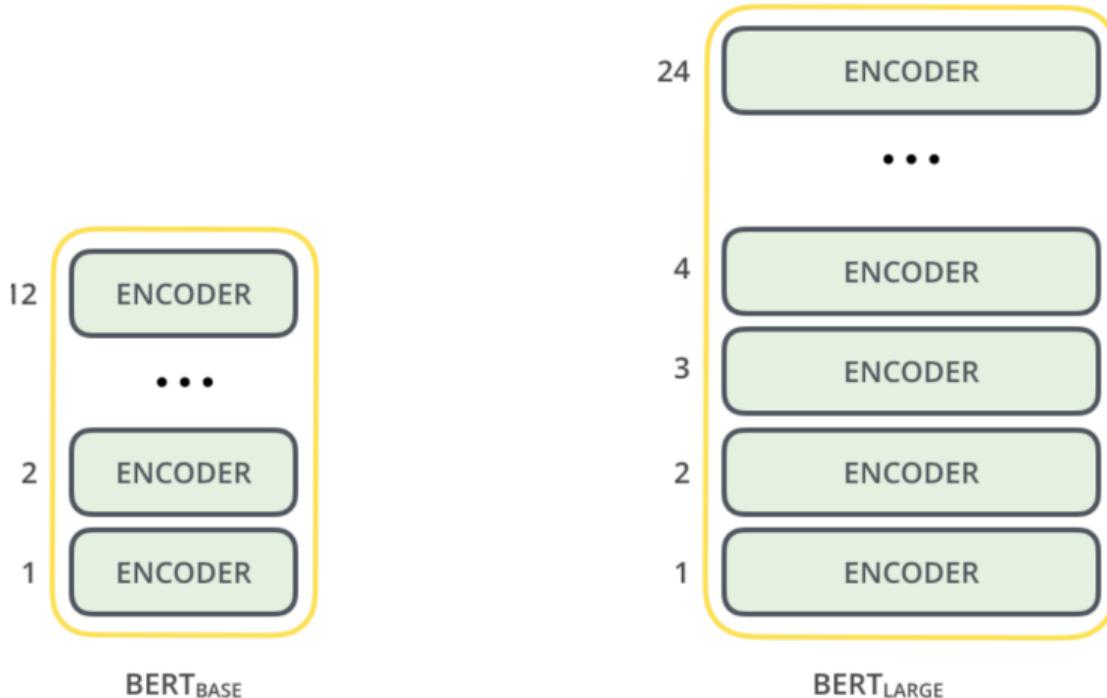
Крутой обзорчик с техническими деталями - живут в тех же лекциях физтеха. При желании можно вкурить. И да, в целом курс достаточно крутой - и крут он тем, что считается, что слушатель не лаптем щи хлебает, а считает градиент на лету, но пока не придумал зачем.

Обзор

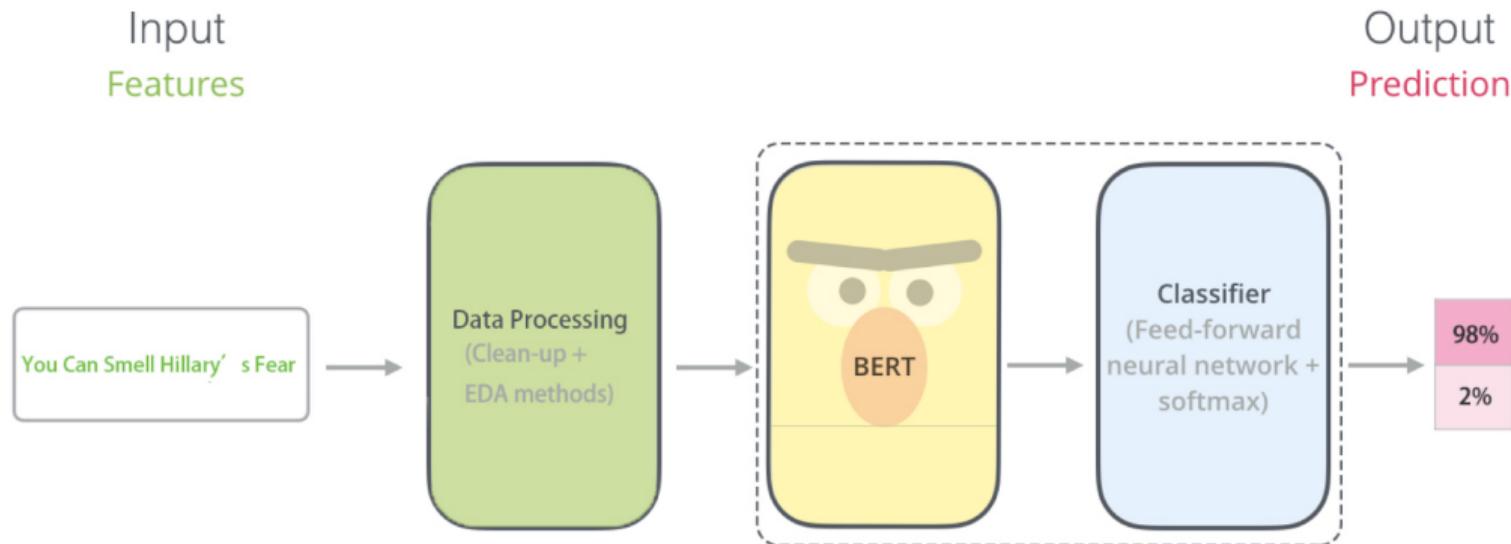
BERT

- Шел 2018 год и Google сказал - наши компьютеры самые мощные, а данные самые большие!
- BERT - Bidirectional Encoder Representations from Transformers
- В чем прелесть - придумали как предобучать без учителя и потом переиспользовать веса!

Архитектура BERT: Base, Large версии



BERT, простейшее применение



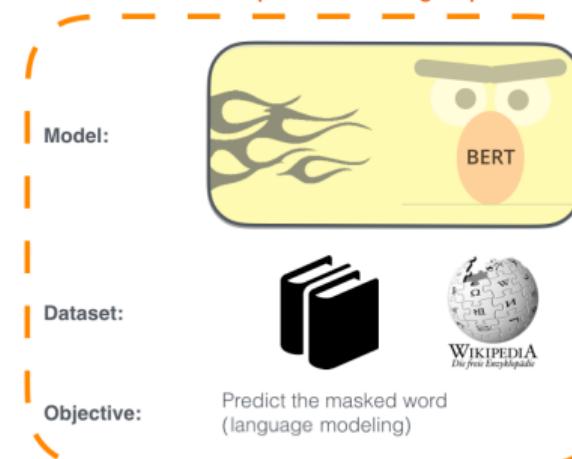
2 фазы BERT

BERT можно логически разделить на 2 фазы: Pre-training и Fine-tuning

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

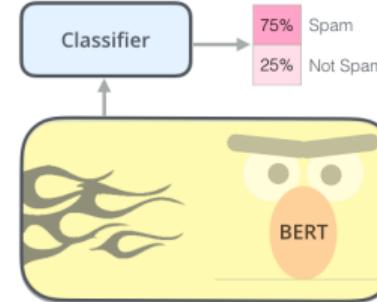
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

Semi-supervised Learning Step



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

Supervised Learning Step



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

Источник: <https://jalammar.github.io/illustrated-bert/>

Лежащие внутри идеи и почему он популярный

1. Pre-training по двум задачам:

- Masked LM: берем корпус текстов и маскируем часть предложений, тренируем предсказывать маску.
- Next sentence prediction: тренируем определять является ли следующее предложение истинно следующим для данного предложения

2. BERT из коробки знает язык, ему 1-2 эпохи надо подсказать, что с этим знанием делать

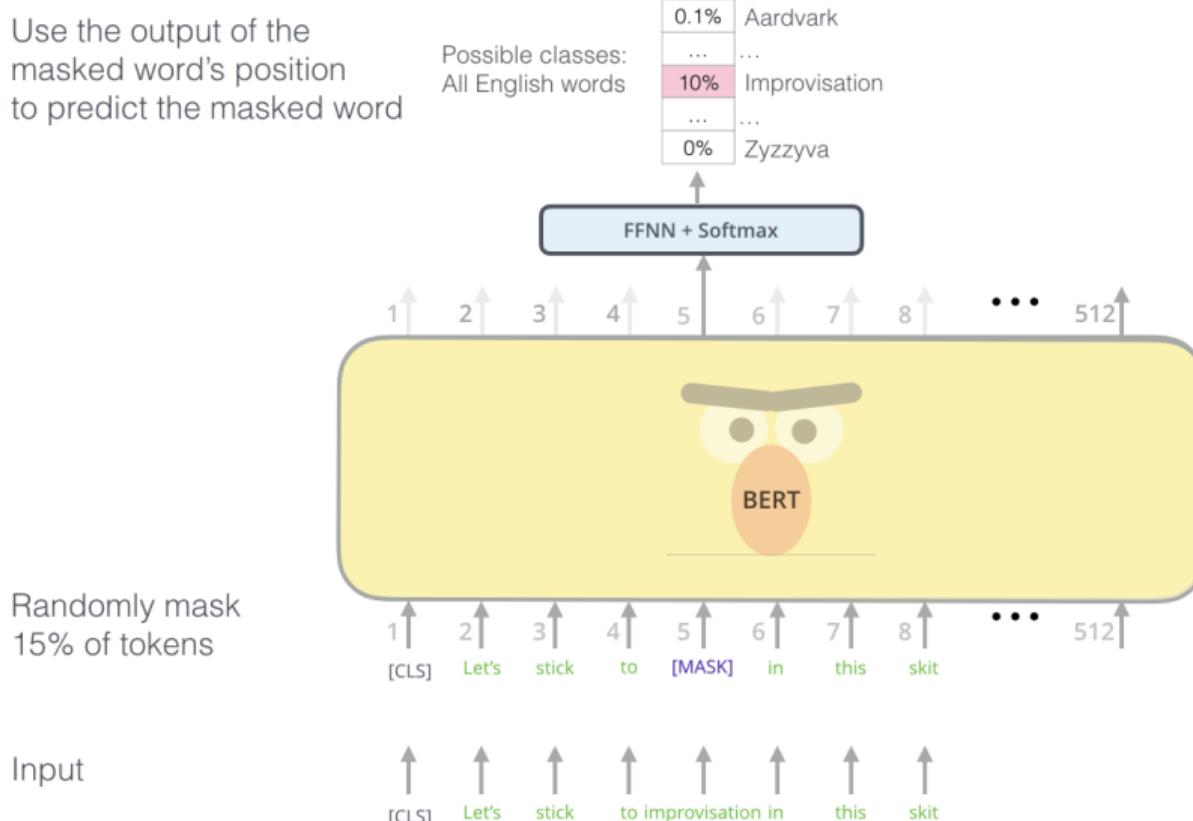
3. В готовых библиотеках лежат много предобученных под разные задачи Бертов - классификация, вопросно - ответные системы и тому подобное.

4. Опять же подаем слова кусочками (WordPiece), чтобы как-то решать проблему OOV.

Победа - нет необходимости размечать данные для обучения!

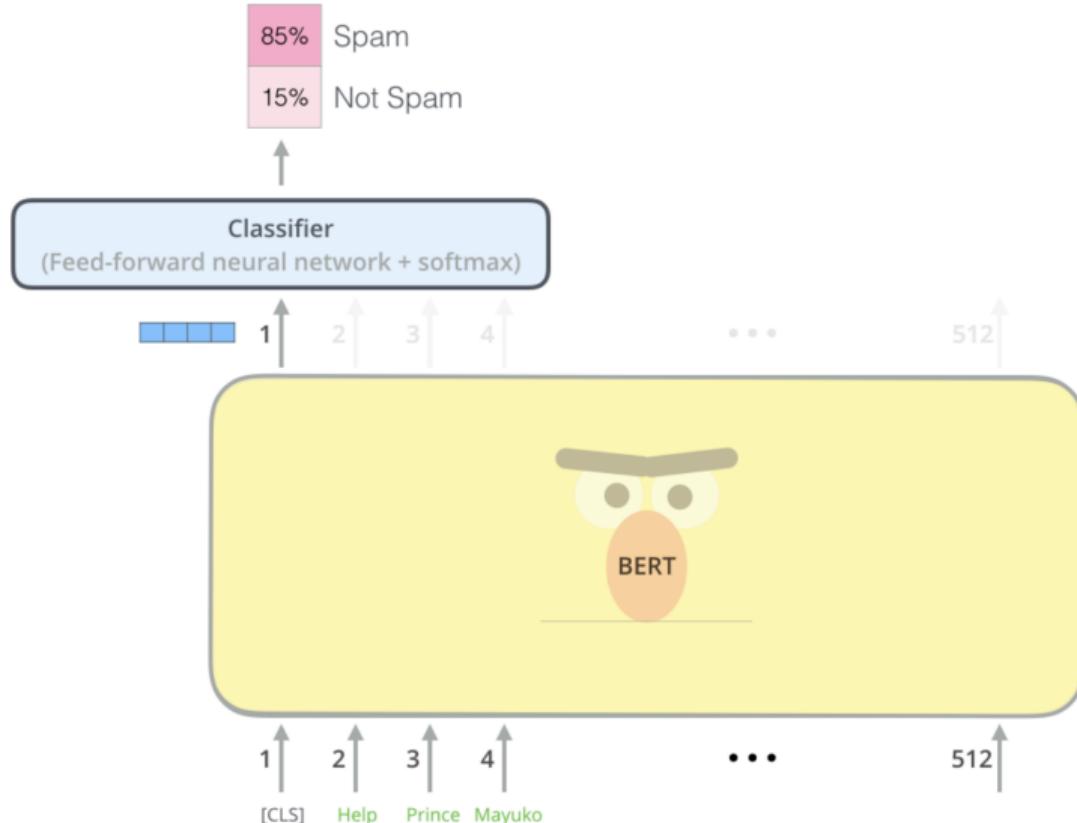
Masked LM Pretraining

Use the output of the masked word's position to predict the masked word



Randomly mask 15% of tokens

Fine-tuning



В каких задачах еще используется?

Spoiler: Картинка не про BERT. Взята из статьи про Open AI Transformer

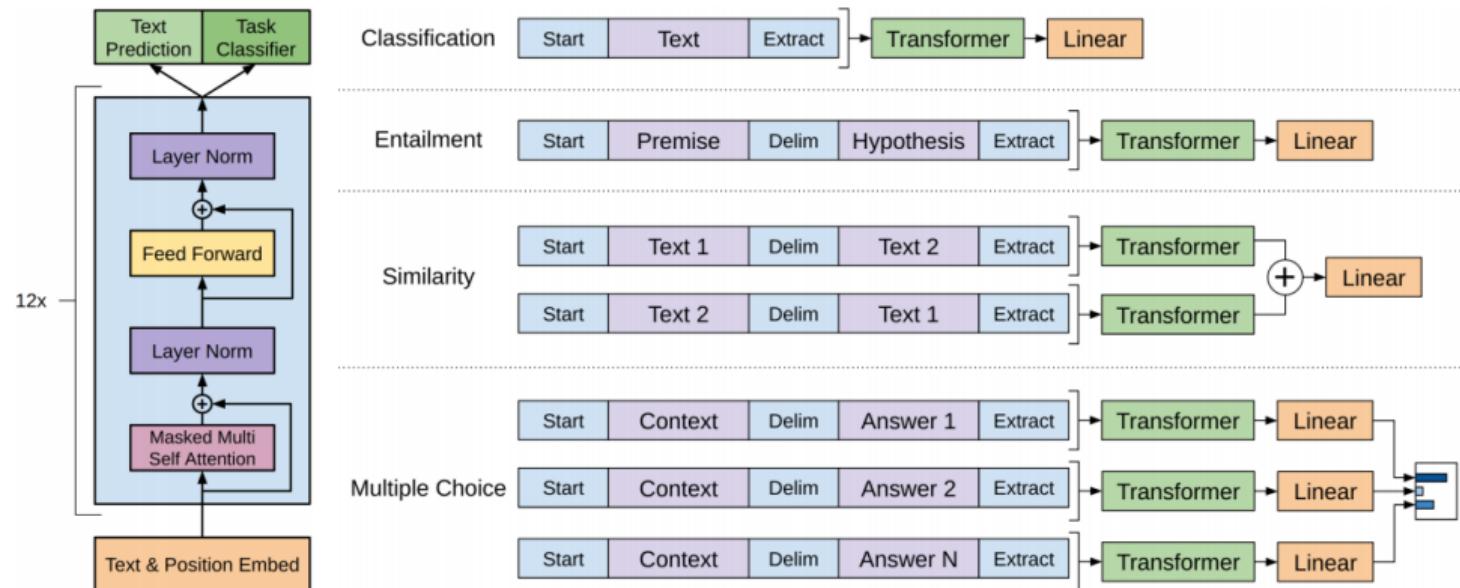
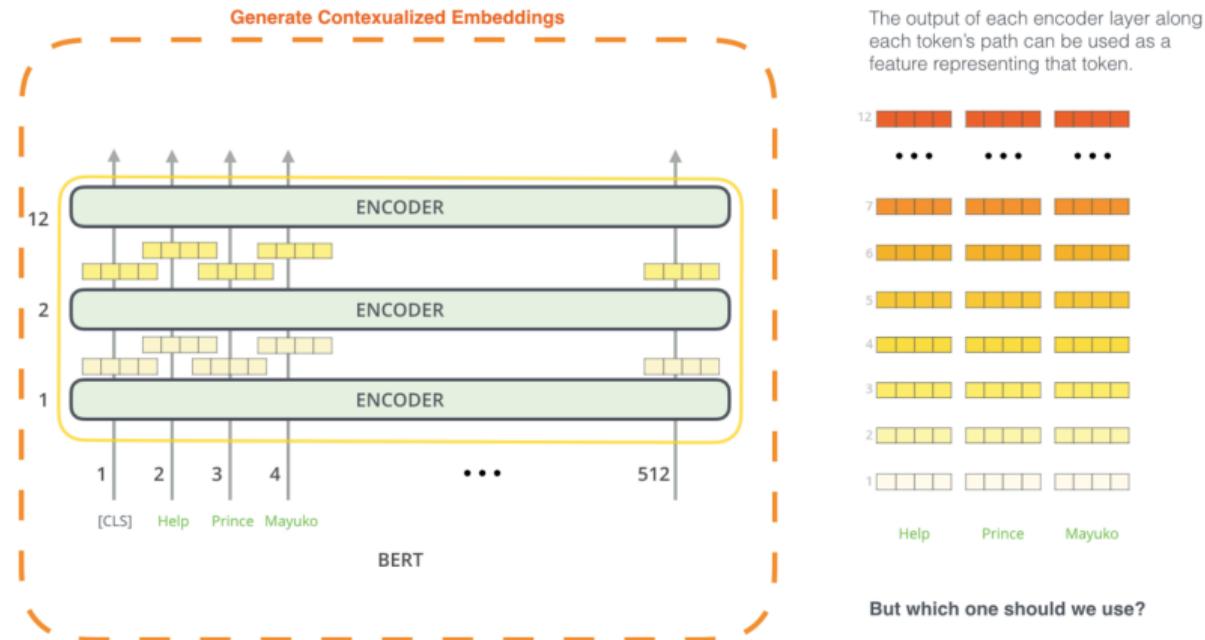


Figure 1: (**left**) Transformer architecture and training objectives used in this work. (**right**) Input transformations for fine-tuning on different tasks. We convert all structured inputs into token sequences to be processed by our pre-trained model, followed by a linear+softmax layer.

BERT for feature extraction

И это еще не все!

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

What is the best contextualized embedding for “Help” in that context?

For named-entity recognition task CoNLL-2003 NER

		Dev F1 Score
12	Embedding	91.0
• • •		
7		
6		
5		
4		
3		
2		
1		
Help		
	First Layer	
	Last Hidden Layer	94.9
	Sum All 12 Layers	95.5
	Second-to-Last Hidden Layer	95.6
	Sum Last Four Hidden	95.9
	Concat Last Four Hidden	96.1

The diagram illustrates the architecture of BERT for feature extraction. It shows the sequence of tokens from 12 down to Help, each with a corresponding vector embedding. The embeddings are represented by colored boxes: orange for tokens 12 through 5, yellow for tokens 4 through 2, white for token 1, and green for the word "Help". The "Embedding" row shows a single green vector. The "First Layer" row shows a vector composed of three green segments. The "Last Hidden Layer" row shows a vector composed of four orange segments. The "Sum All 12 Layers" row shows a vector resulting from summing the embeddings of all 12 tokens. The "Second-to-Last Hidden Layer" row shows a vector resulting from summing the embeddings of tokens 11 through 9. The "Sum Last Four Hidden" row shows a vector resulting from summing the embeddings of tokens 9 through 6. The "Concat Last Four Hidden" row shows a vector resulting from concatenating the embeddings of tokens 9 through 12.

Немного статистики

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

ELMO

Серия вопросов в зал

Как работают разные эмбединги?

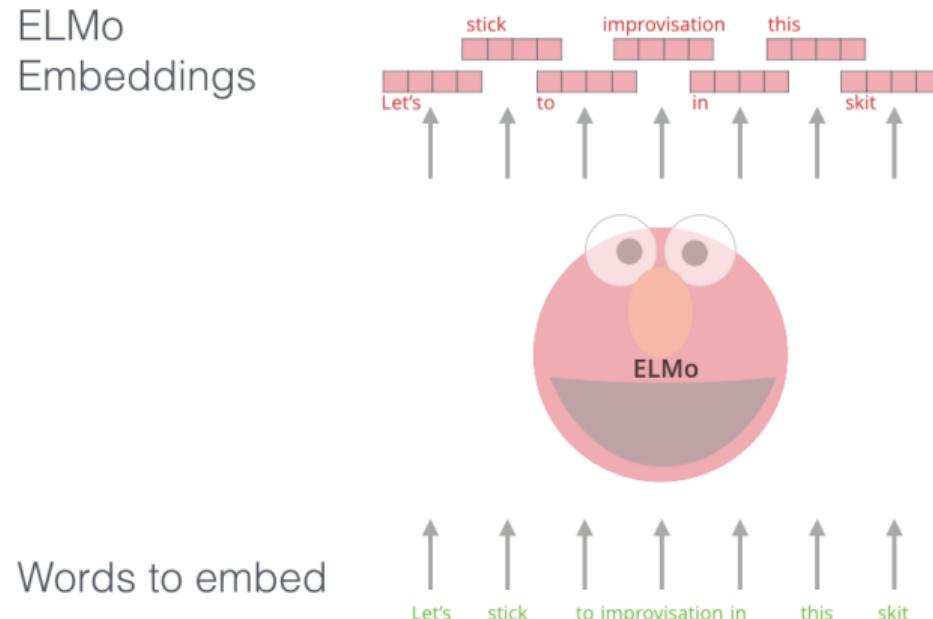
В чем, по вашему мнению, их главная проблема?

Картиночка про решение проблемы



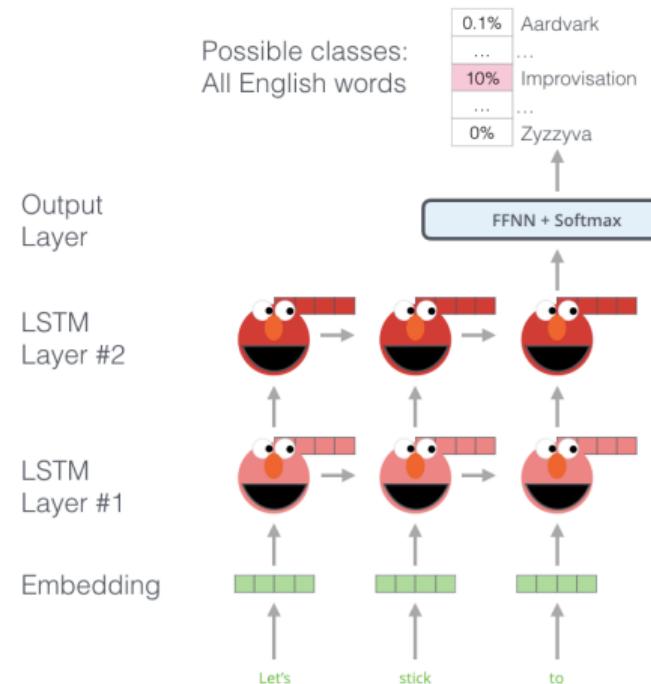
ELMO

Захватываем контекст предложения через biderictional LSTM. Таким образом мы захватываем и контекст предложения (да, надо очень очень много данных, не обучайте это дома)



ELMO

Учится понимать язык ELMO следующим образом - оно берет большой дата сет и пытается предсказать следующее слово в предложении.



ELMO

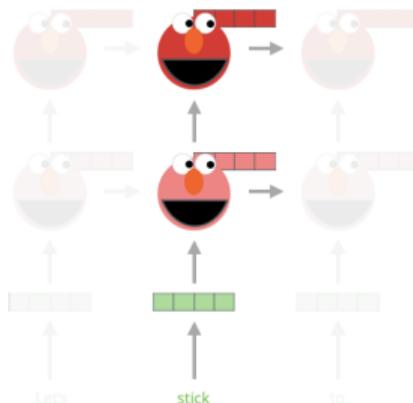
Применяем

Embedding of “stick” in “Let’s stick to” - Step #2

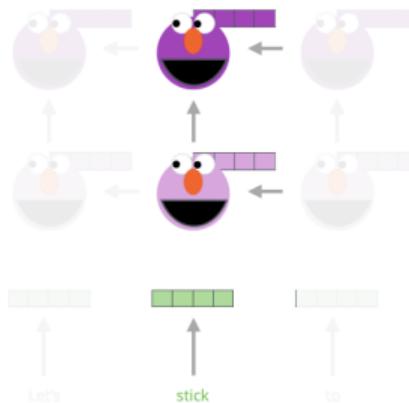
1- Concatenate hidden layers



Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context

ELMO

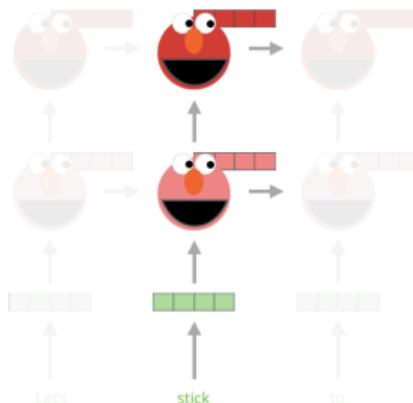
Применяем

Embedding of “stick” in “Let’s stick to” - Step #2

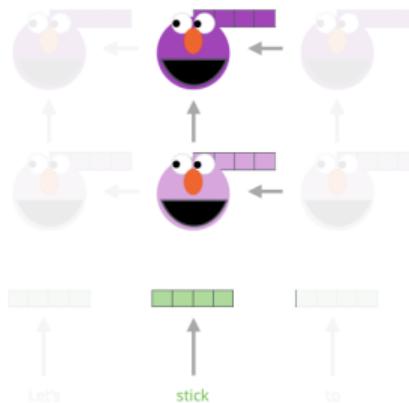
1- Concatenate hidden layers



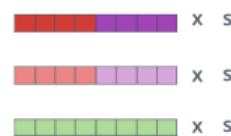
Forward Language Model



Backward Language Model



2- Multiply each vector by a weight based on the task



3- Sum the (now weighted) vectors



ELMo embedding of “stick” for this task in this context

Итого

Почему это все стало круто и популярно?

1. Идея с вниманием стала ключевой - таким образом мы можем тянуть информацию через всю последовательность
2. Внимание можно параллелизовать, LSTM намного сложнее
3. Придумали как сделать так, чтобы не размечать данные
4. Купили много GPU
5. Посадили кучу инженеров, которые заставили это все учиться!

Ну и маленькая мысль в конце - за курс мы разобрали кубики нейронных сетей и посмотрели какой лютый треш можно из этих кубиков делать. Современный моделист в нейронных сетях - скорее инженер, нежели аналитик