

Глубокое обучение и вообще

Соловей Влад и Шигапова Фирюза

15 декабря 2021 г.

Посиделка 11: Transformer

Agenda

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

Быстренькая история взятая из старых лекций

задача seq2seq

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

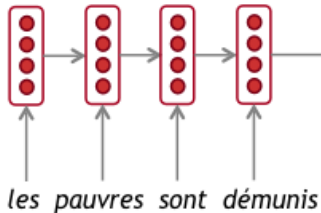
Я буду оставлять некоторые ключевые слова того, чтобы вы могли сами залезть поглубже, если такое погружение потребуется.

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

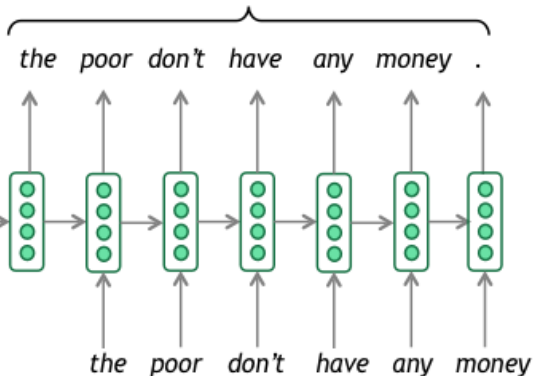
Модели в машинном переводе сравнивают по BLEU score - если в кратце, то эта метрика сравнения полученного машинной перевода и человеческого, насколько мы вообще бьемся. Из проблем данной метрики - если машина перевела правильно, но альтернативно, то BLEU будет низкий....

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.

Greedy decoding

При декодировании может быть следующие проблемы - мы декодируем какое-то конкретное слово. Но что делать если это слово некорректное?

Greedy decoding/beam search

Самый простой подход - селектировать несколько наиболее вероятных слов, а не одно. Мы получим множество предложений, а потом по какой-то эвристике выбирать лучшее из них. Подход хорош всем, кроме скорости. Ему есть альтернатива - называется beam-search.

Какие проблемы мы видим в таком подходе (спойлер, из коробки он не полетел)?

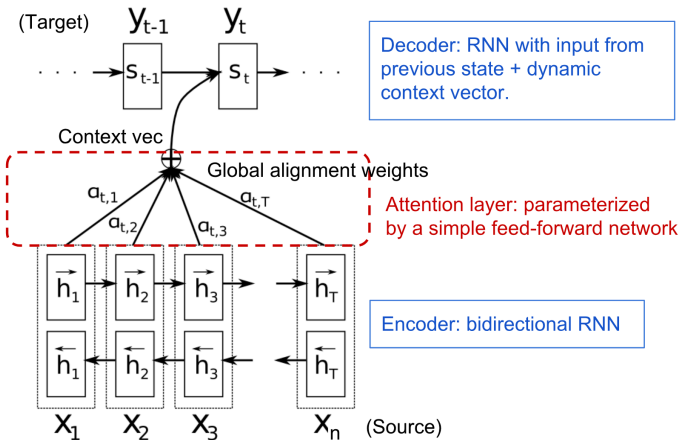
Attention!

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.

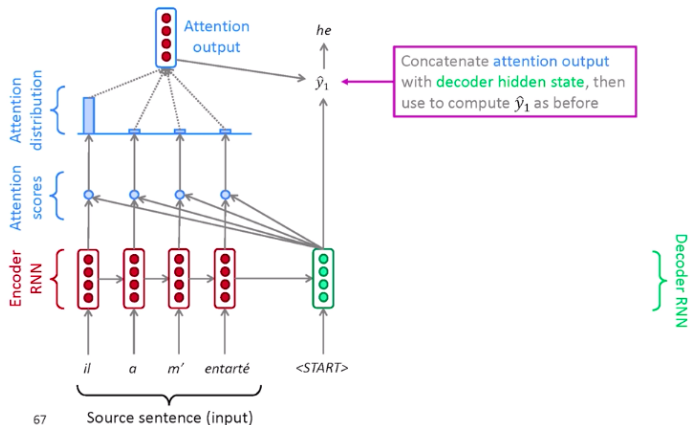
Attention



Additive Attention

Attention

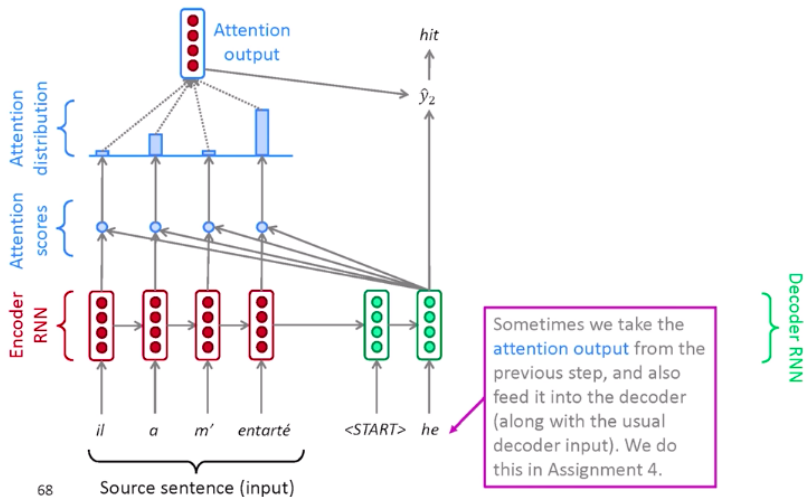
Sequence-to-sequence with attention



ссылочка на оригинал

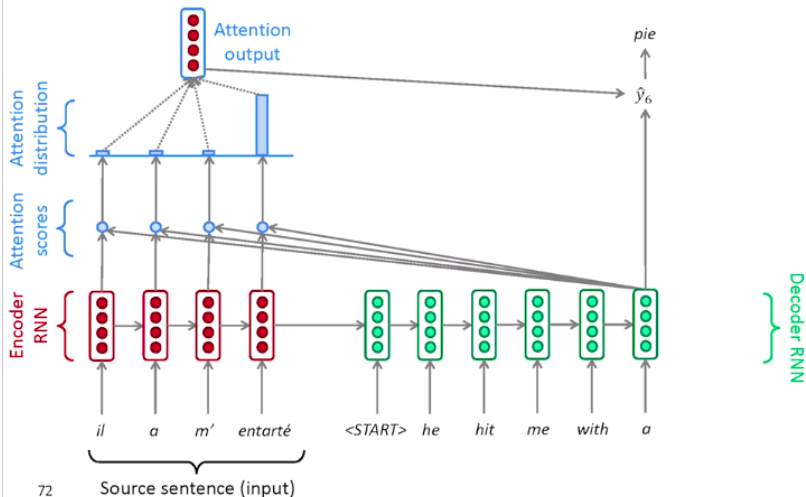
Attention

Sequence-to-sequence with attention



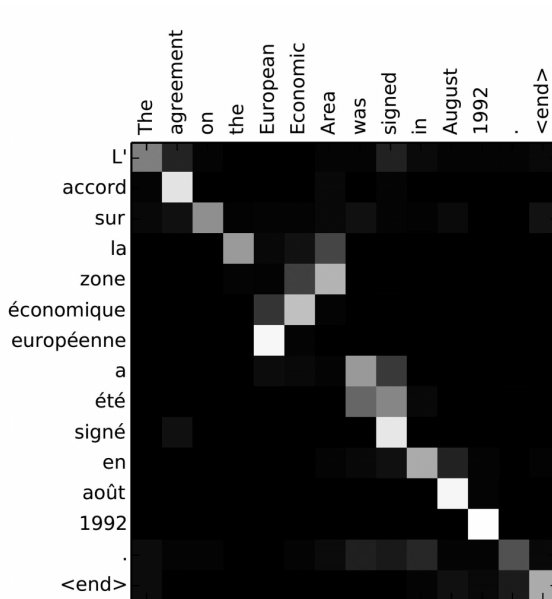
Attention

Sequence-to-sequence with attention



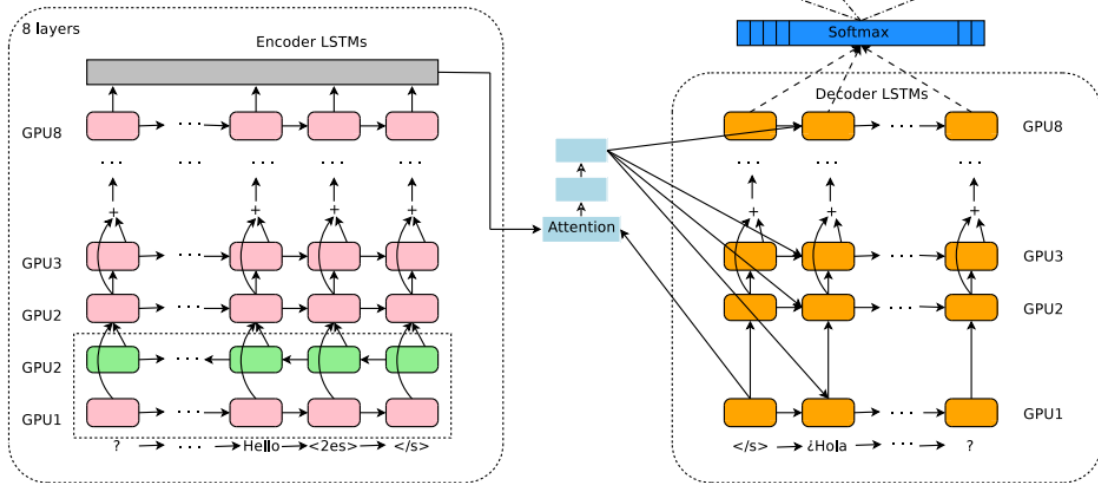
72

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 градиенты, накапливаем инфу в весах сетки.



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

Выводы:

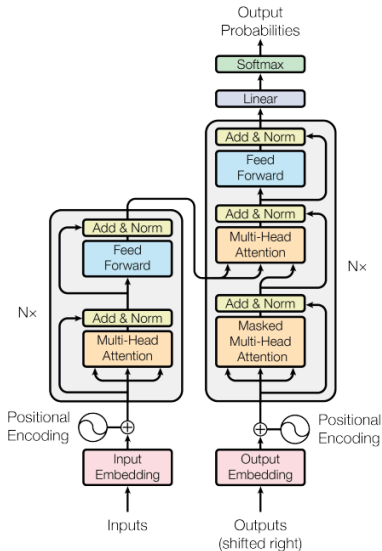
1. 8 слоев LSTM (8 Карл!)
2. в attention 2 слоя dense.
3. Собираем слова из морфем - пытаемся победить out-of-vocabulary.
4. Модель стала иногда сексистом и фашистом - требуются слишком большие дата сет, чтобы учить эту большую прелесть.

Attention is all you need!

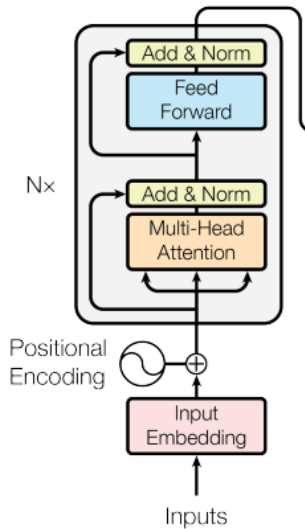
attention is all you need

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

attention is all you need

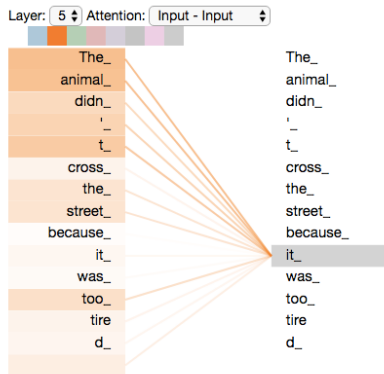


Encoder



Что мы хотим?

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



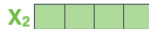
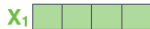
Абстракции!

Input

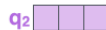
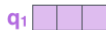
Thinking

Machines

Embedding

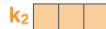


Queries



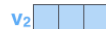
W^Q

Keys



W^K

Values



W^V

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

Mar 2

Input

Embedding

Queries

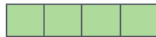
Keys

Values

Score

Thinking

x_1



q_1



k_1



v_1



$$q_1 \cdot k_1 = 112$$

Machines

x_2



q_2



k_2



v_2



$$q_1 \cdot k_2 = 96$$

Mar 3

Input

Embedding

Queries

Keys

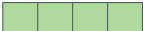
Values

Score

Divide by 8 ($\sqrt{d_k}$)

Softmax

Thinking

x_1 

q_1 

k_1 

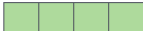
v_1 

$$q_1 \cdot k_1 = 112$$

14

0.88

Machines

x_2 

q_2 

k_2 

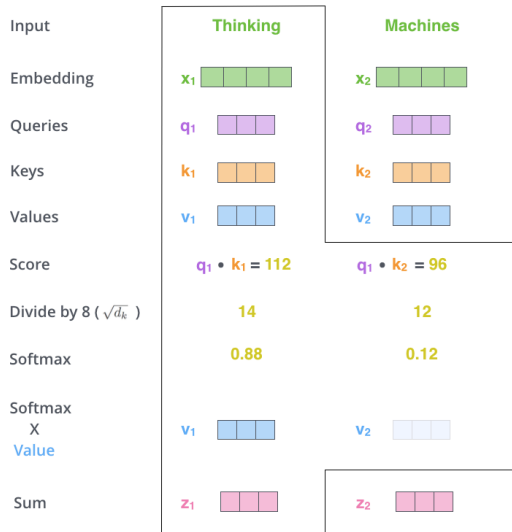
v_2 

$$q_2 \cdot k_2 = 96$$

12

0.12

Mar 4



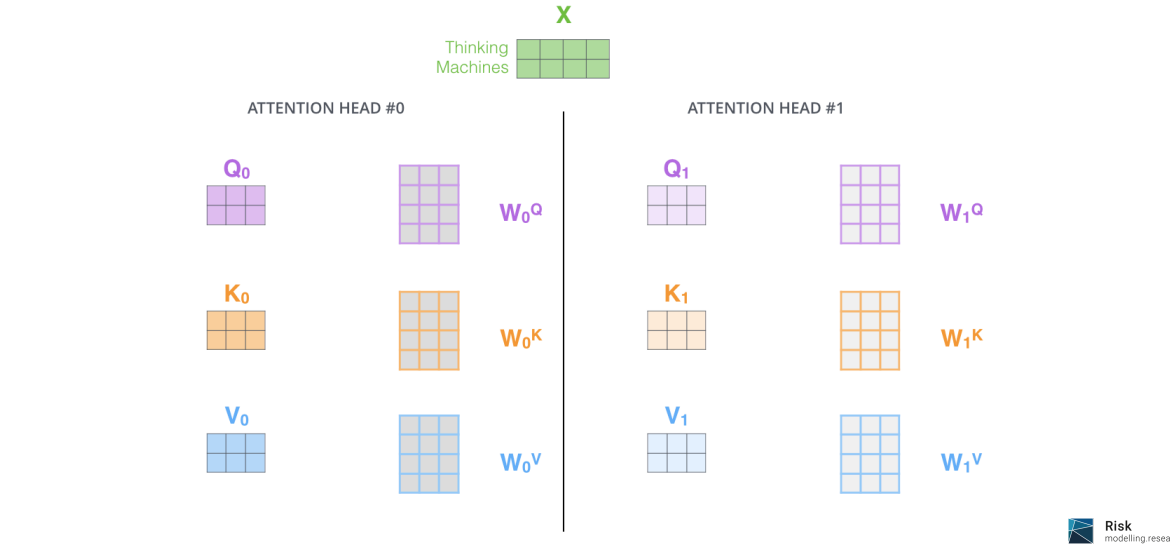
Шар 5

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{W}^{\text{Q}} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \text{Q} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{W}^{\text{K}} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \text{K} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

$$\begin{matrix} \text{X} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} \times \begin{matrix} \text{W}^{\text{V}} \\ \begin{array}{|c|c|c|c|} \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline & & & \\ \hline \end{array} \end{matrix} = \begin{matrix} \text{V} \\ \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline \end{array} \end{matrix}$$

multi head attention



Соединяем!

1) Concatenate all the attention heads

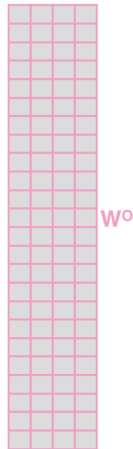


3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



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

\times



Итого

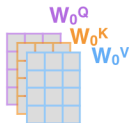
1) This is our input sentence*

Thinking
Machines

2) We embed each word*



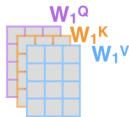
3) Split into 8 heads.
We multiply X or R with weight matrices



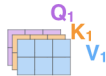
W_0^K
 W_0^V



K_0
 V_0



W_1^K
 W_1^V



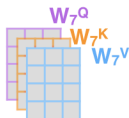
K_1
 V_1



...

...

...



W_7^K
 W_7^V



K_7
 V_7



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

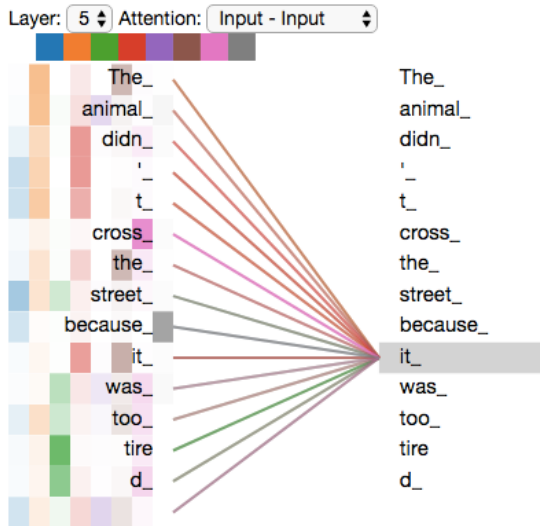


W^O



Z





Итого

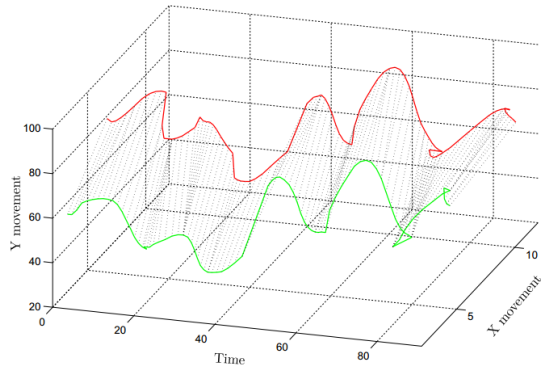
Выходом всего этого дела будут вектора key и value, которые позволят декодеру смотреть на нужные нам кусочки. И бежим смотреть гифки декодера!

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

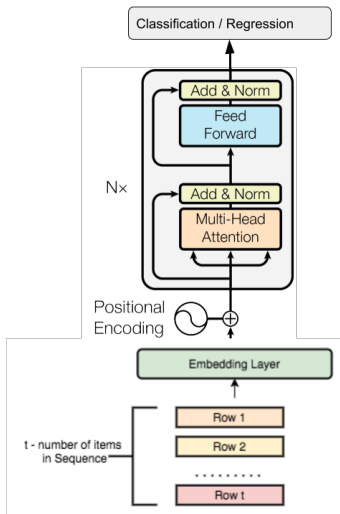
Итого

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

Multivariate Time Series



Multivariate Time Series



<https://arxiv.org/pdf/2010.02803.pdf>

Multivariate Time Series - Positional Encoding

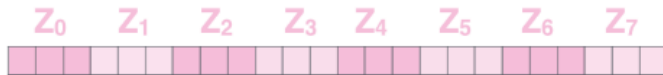
Positional Encoding позволяет учитывать позицию элемента в последовательности

1. Посчитать \cos - \sin и включить в качестве еще одной характеристики в вектор характеристик: $\text{concat}([u, \cos, \sin])$
2. Создать обучаемый вектор, равный размеру входного вектора характеристик, и сложить его с вектором: $u + W_{pos}$
3. Создать обучаемый скаляр и включить в качестве еще одной характеристики в вектор характеристик: $\text{concat}([u, \text{scalar}_{pos}])$

Multivariate Time Series - что можно после Encoder

MLP

Вытягиваем последовательность в один длинный вектор



Multivariate Time Series - что можно после Encoder RNN

Отдаем на вход RNN сети

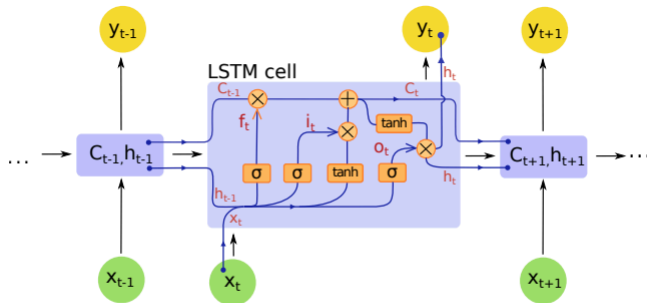
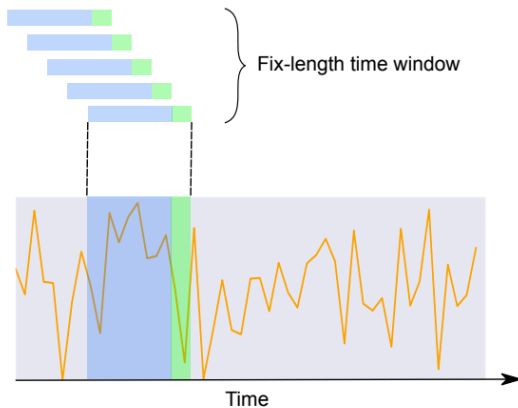


Figure 3. Long Short-Term Memory network and LSTM unit.

Multivariate Time Series - что можно после Encoder CNN

Отдаем на вход CNN сети



Tabular Data

	A	B	C	D	E	F	G	H	I	J	K	L
1	User	Card	Year	Month	Day	Time	Amount	Use Chip	Merchant Name	Merchant City	MCC	Is Fraud?
2	0	0	2019	4	21	9:47	\$6.81	Chip	Chevron	Brandon	5541	No
3	0	0	2019	4	21	10:38	\$10.07	Chip	Anwar Grocery	Brandon	5411	No
4	0	0	2019	4	22	3:53	\$42.61	Chip	Kelly Auto Repair	Brandon	7538	No
5	0	0	2019	4	22	7:28	\$47.66	Chip	Barnes & Noble	Brandon	5942	No
6	0	0	2019	4	22	10:30	\$9.73	Chip	Applebees	Brandon	5812	No
7	0	0	2019	4	23	15:02	\$121.47	Chip	Green Wholesale	Brandon	5300	No
8	0	0	2019	4	23	23:20	\$71.66	Online	Frontier Communications	ONLINE	4814	No
9	0	0	2019	4	24	10:13	\$11.05	Chip	Applebees	Brandon	5812	No
10	0	0	2019	4	24	10:17	\$11.05	Chip	Applebees	Brandon	5812	No

Tabular Data

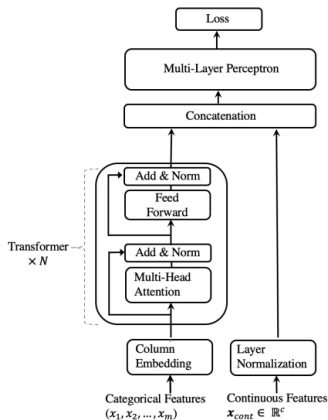


Figure 1: The architecture of TabTransformer.

<https://arxiv.org/pdf/2012.06678.pdf>

Tabular Data

- Для каждой категориальной переменной ставим в соответствие Embedding
- Все категориальные признаки представляем в виде последовательности
- Созданная последовательность идет на вход Transformer

$$E(\mathbf{x}_{cat}) = e_1(x_1), \dots, e_m(x_m)$$