

Лекция 8

Трансформеры и механизм внимания

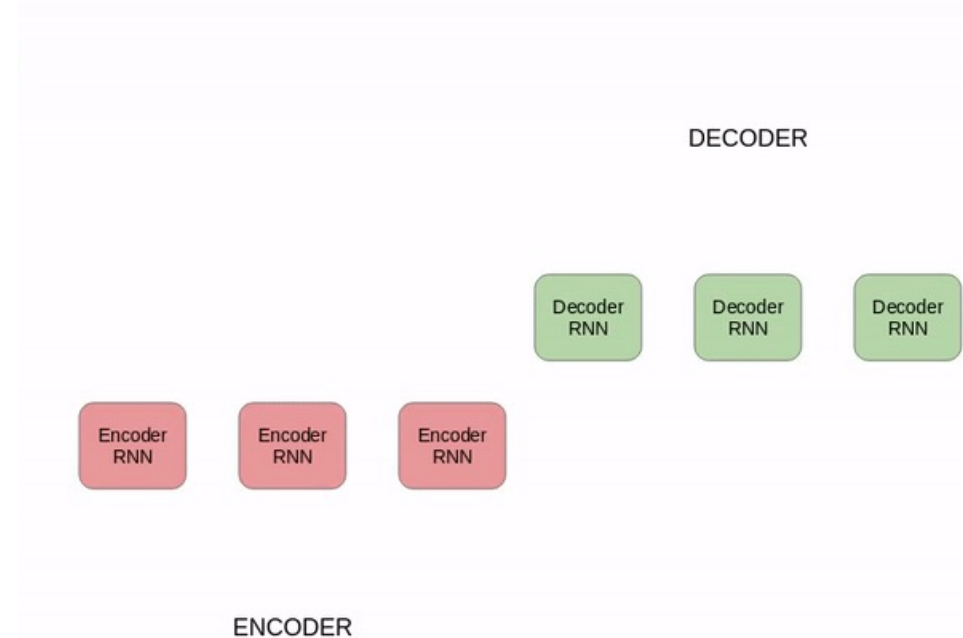
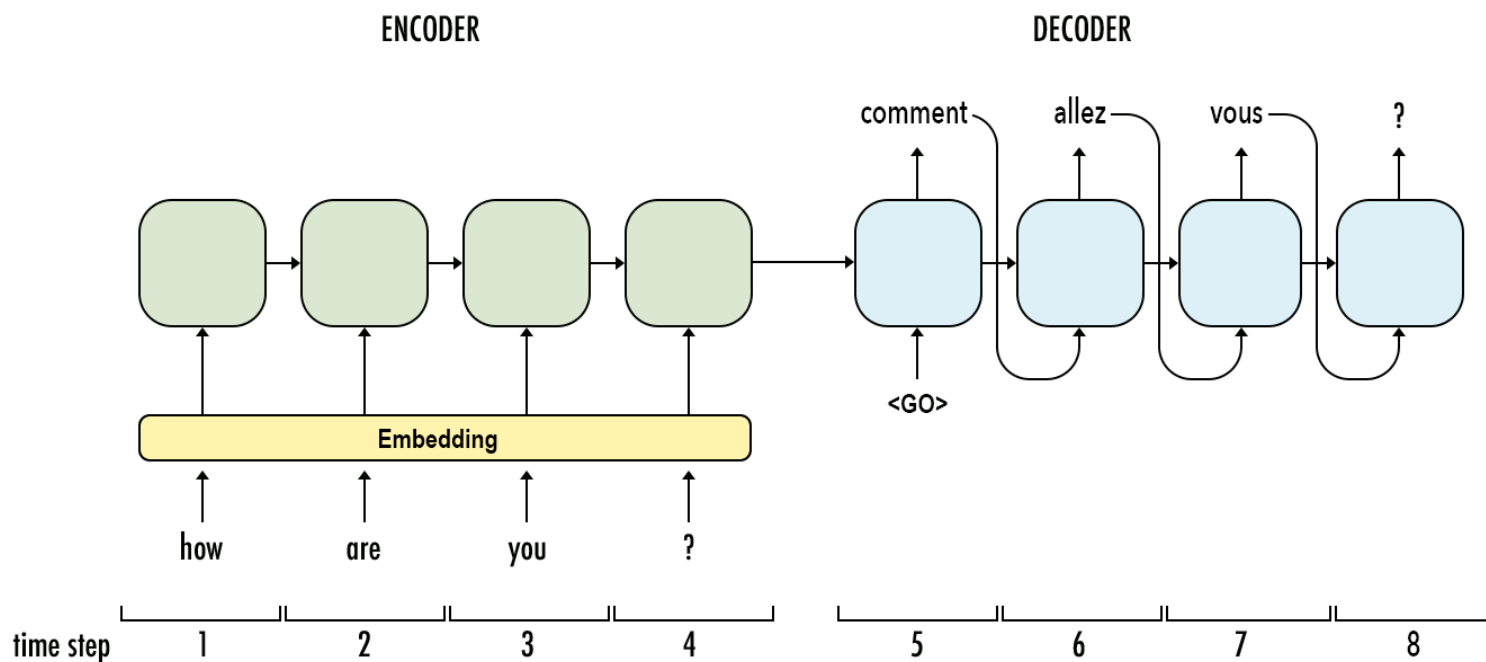
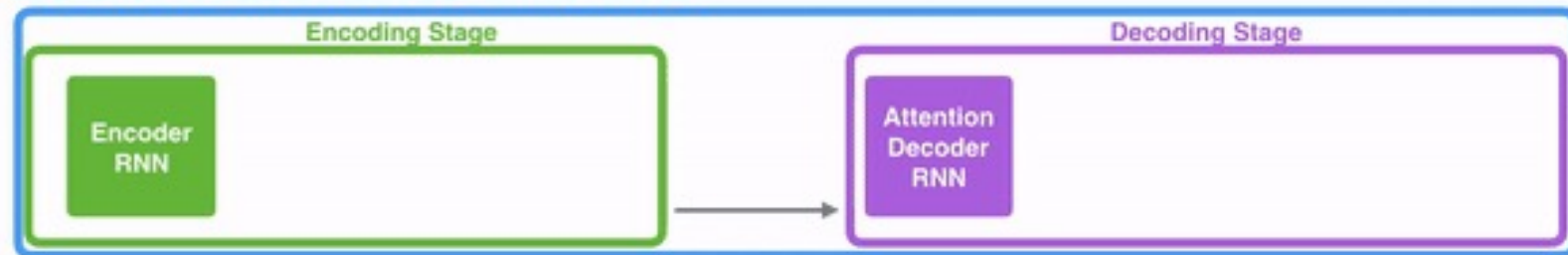
Разработка нейросетевых систем

Канев Антон Игоревич

Seq2seq

- Для машинного перевода
- Состоит из двух частей, например две LSTM
- Можно добавить внешнее внимание

Neural Machine Translation SEQUENCE TO SEQUENCE MODEL WITH ATTENTION



Последовательность -> Последовательность

Машинный перевод

Пример:

ITA: Il gatto si e' seduto sul tappetino.



EN: The cat sat on the mat.

Подход:

seq2seq: имеем один RNN для кодирования исходного предложения, а другой RNN - для предсказания целевого предложения.

В обычной seq2seq для сети decoder мы передаем последнее состояние encoder, но оно не меняется (уже конечное после всех слов) при генерации новых слов с помощью decoder.

Поэтому применим механизм внешнего внимания. Целевой RNN учится (мягко) выравнивать предложение с помощью механизма внимания (обращать внимание на разные слова в зависимости от текста, который уже сгенерировали).

Последовательность -> Последовательность

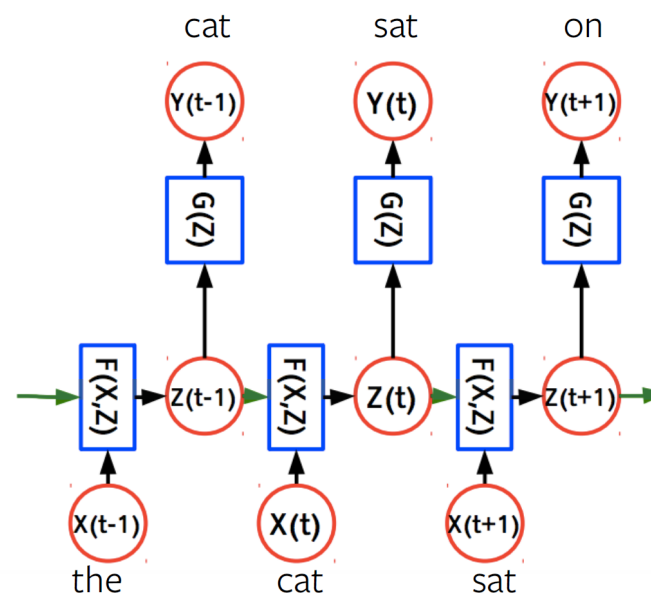
Машинный перевод

Пример:

EN: The cat sat on the mat.

Подход:

Необходимо получить последовательность перевода, генерируя слова одно за другим. Но остается вопрос – с чего начать перевод?



Y. LeCun's diagram

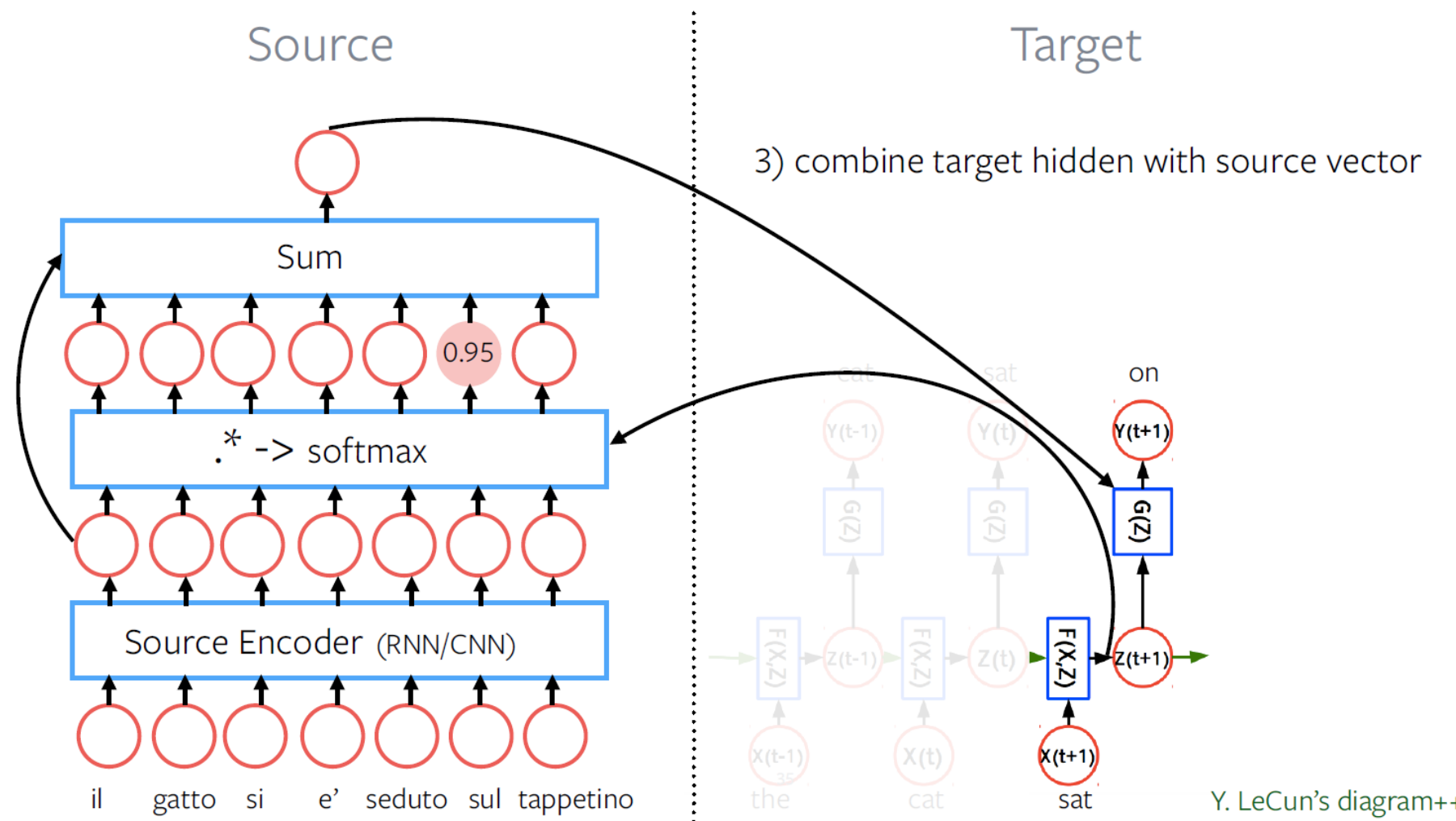
Последовательность -> Последовательность

Внимание RNN

На какое слово из
исходного текста
нужно обратить
внимание?

3 этапа:

- Считаем по всем словам значения в энкодере
- Используем декодер чтобы посчитать внимание
- В генерирующей LSTM используем внимание от всех слов в Encoder

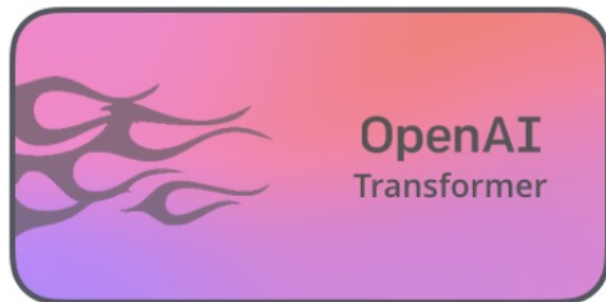


Attention is all you need

- Механизм внимания получил широкое распространение с появлением трансформеров, представленных впервые Google в 2017 году
- Google BERT, Open AI GPT стали применяться повсеместно для решения разных практических задач
- Transformer (encoder+decoder), BERT (только encoder), GPT (только decoder)



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



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

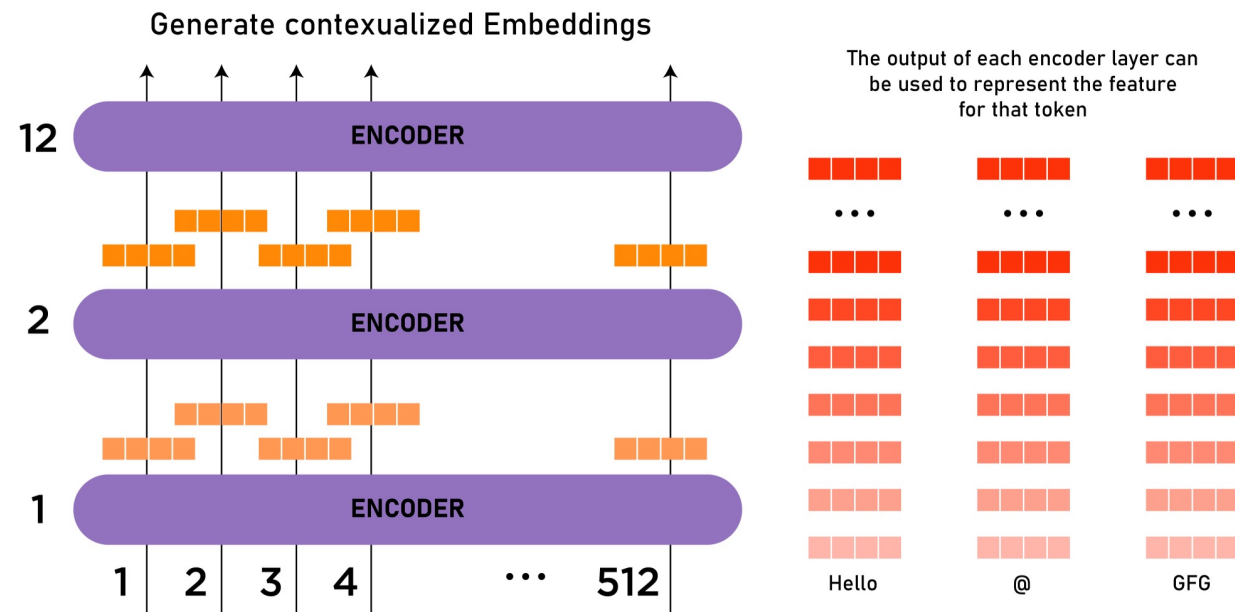
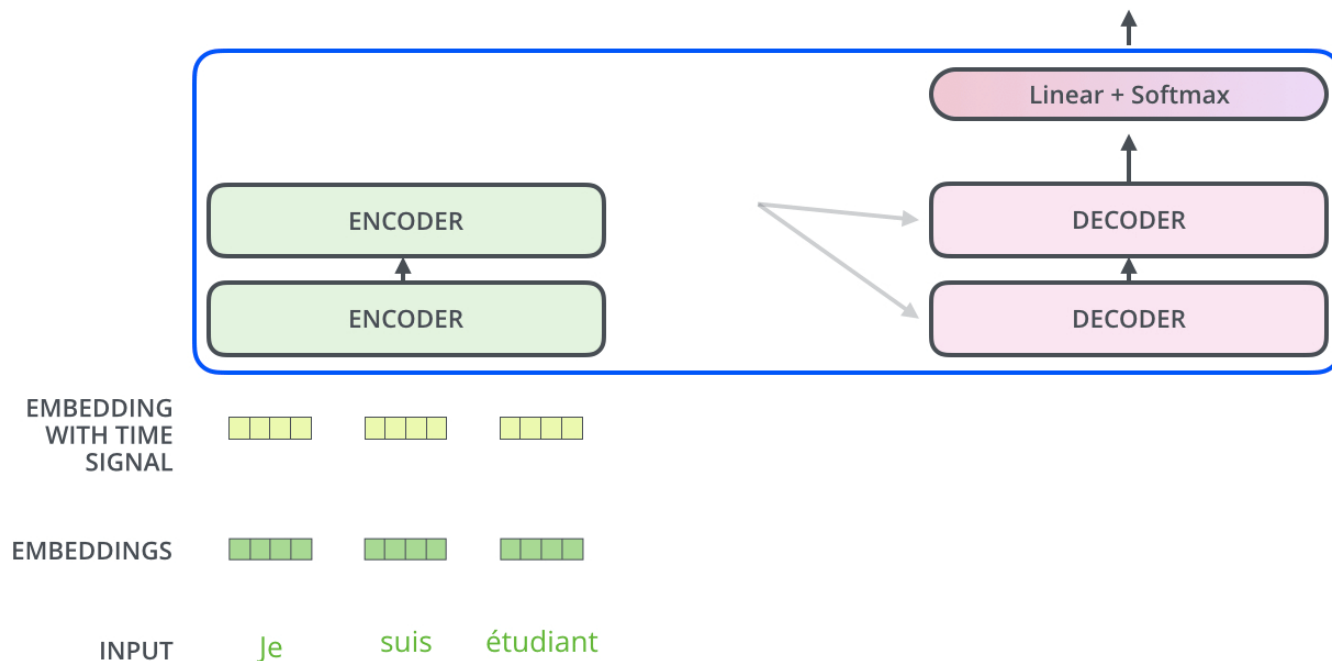
<https://arxiv.org/abs/1706.03762>

Трансформер

- Трансформер с механизмом внимания
- BERT, GPT

Decoding time step: 1 2 3 4 5 6

OUTPUT



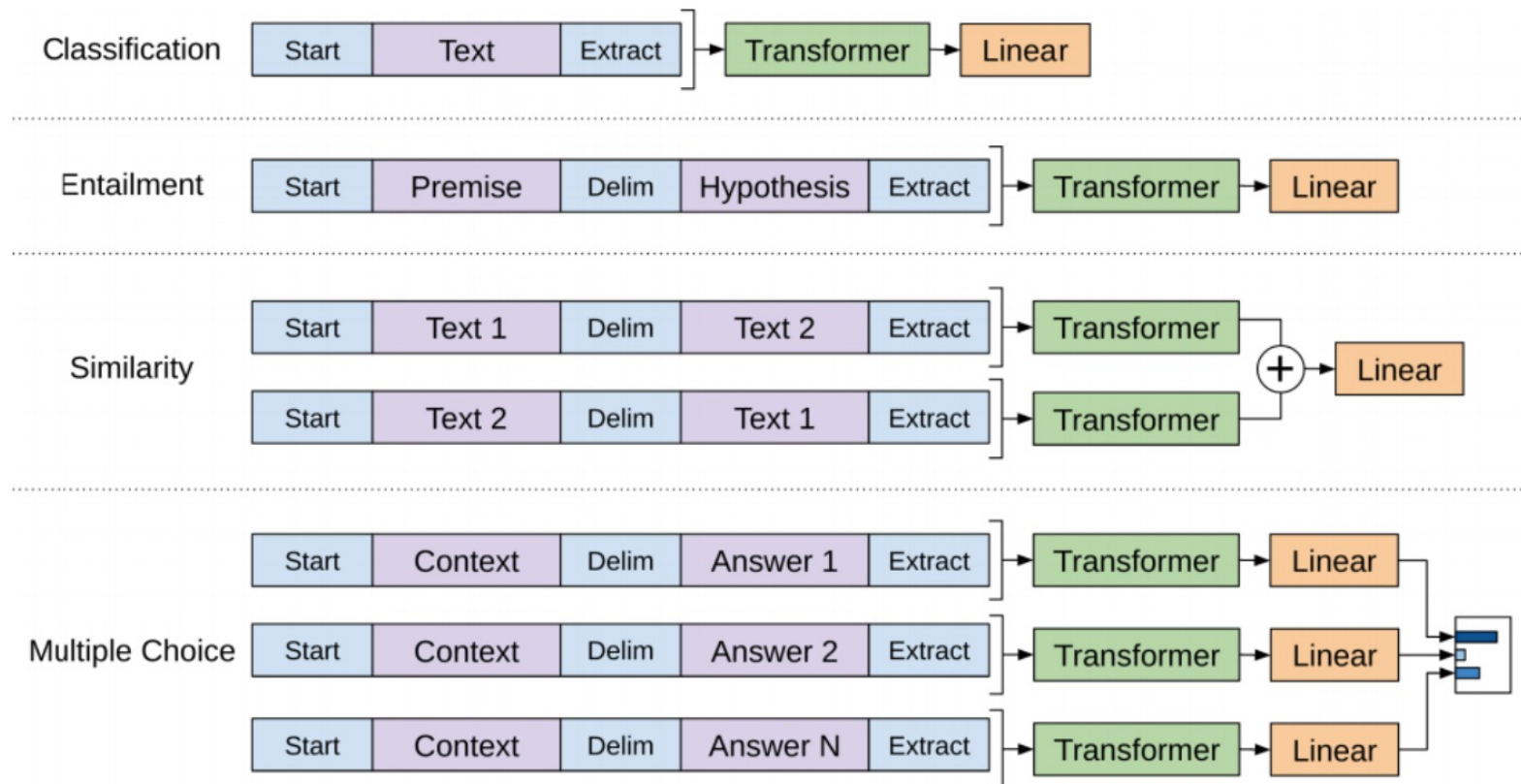
BERT

- Если раньше в Word2vec мы создавали один и тот же вектор для одного слова
- BERT теперь дает нам эмбединг для текста, мы можем кодировать текст в вектор и сравнивать их (RAG)

Применение трансформеров

- Статья от OpenAI указывает на ряд трансформаций, делающих возможным обработку входов при решении разнообразных задач

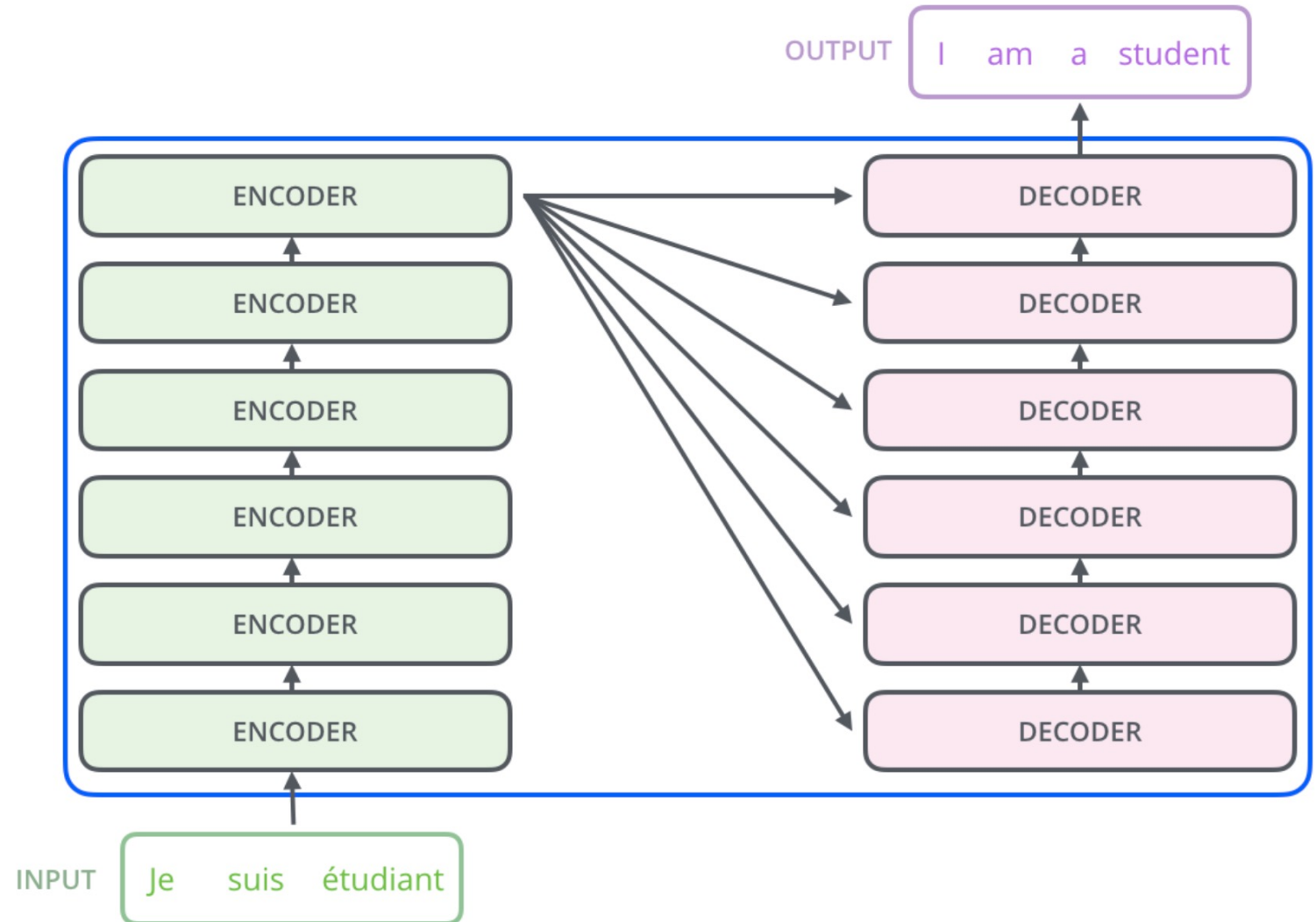
- Пример использования BERT: мы получаем вектор из текста
- Теперь тексты можно сравнить, классифицировать и тд
- Мы можем обучить свою очень простую модель на выходе – transfer learning



Архитектура Transformer

Рассмотрим знакомую задачу:

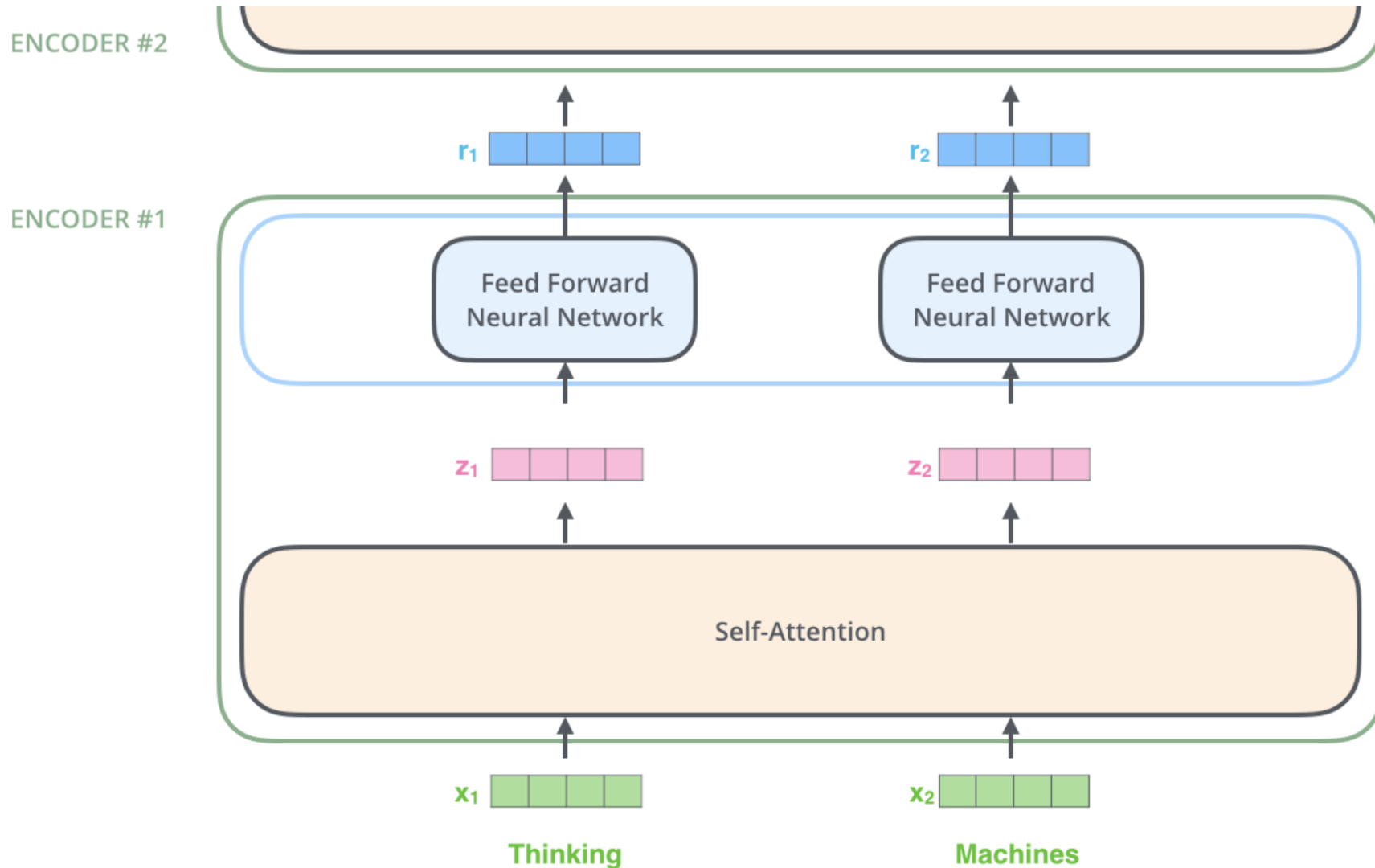
- У нас есть последовательность слов
- Мы должны сгенерировать новую последовательность слов (на другом языке)



Кодировщик

С помощью
специальность слоя
внутреннего
внимания мы из
эмбеддинга слова
будем получать
новый вектор z

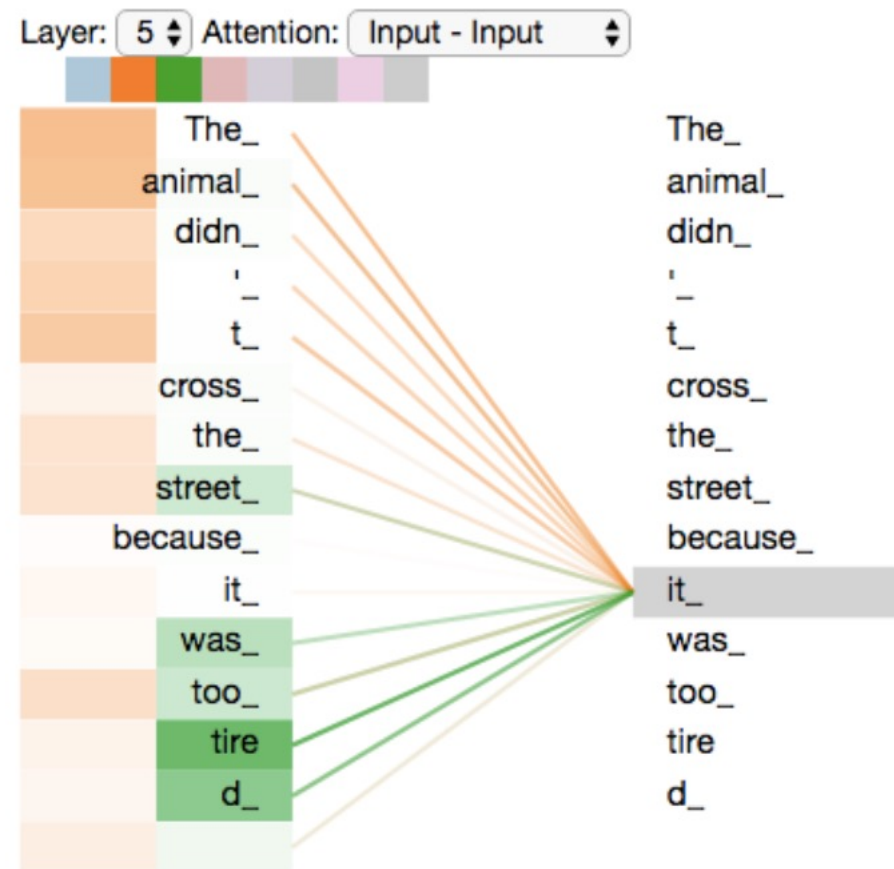
Так повторяем
несколько раз



Внутреннее внимание

Когда мы кодируем слово «it», одна голова внимания больше всего фокусируется на «животном», в то время как другая фокусируется на «усталом»

Представление модели слова «it» сочетает в себе часть представления как «животного», так и «усталого»



Внутреннее внимание

Перемножая x_1 на матрицу весов W^Q получаем q_1 , вектор "query", связанный с этим словом.

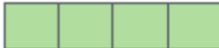
В итоге мы создаем проекцию "query", "key" и "value" каждого слова во входном предложении.

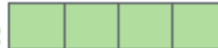
Input

Thinking

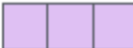
Machines

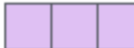
Embedding

x_1 

x_2 

Queries

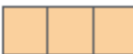
q_1 

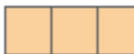
q_2 



W^Q

Keys

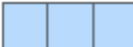
k_1 

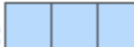
k_2 



W^K

Values

v_1 

v_2 



W^V

Query, Key, Value

- Каждая строка в матрице **X** соответствует слову во входном предложении
- Мы определяем какой в итоге результат будет для данного слова: это его вклад и вклад других слов в предложении

Input

Embedding

Queries

Keys

Values

Score

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

Softmax

Softmax

X

Value

Sum

Thinking

Machines

x_1

x_2

q_1

q_2

k_1

k_2

v_1

v_2

$q_1 \cdot k_1 = 112$

$q_1 \cdot k_2 = 96$

14

12

0.88

0.12

v_1

v_2

z_1

z_2

Несколько слов

Каждая строка в матрице **X** соответствует слову во входном предложении

А количество строк – это размер эмбединга для каждого слова

Запишем в **матричной форме**:

$$\text{softmax} \left(\frac{\begin{matrix} \text{Q} \\ \begin{matrix} \text{2x3 grid} \end{matrix} \end{matrix} \times \begin{matrix} \text{K}^T \\ \begin{matrix} \text{3x2 grid} \end{matrix} \end{matrix}}{\sqrt{d_k}} \right) \begin{matrix} \text{V} \\ \begin{matrix} \text{2x2 grid} \end{matrix} \end{matrix} = \begin{matrix} \text{Z} \\ \begin{matrix} \text{2x3 grid} \end{matrix} \end{matrix}$$

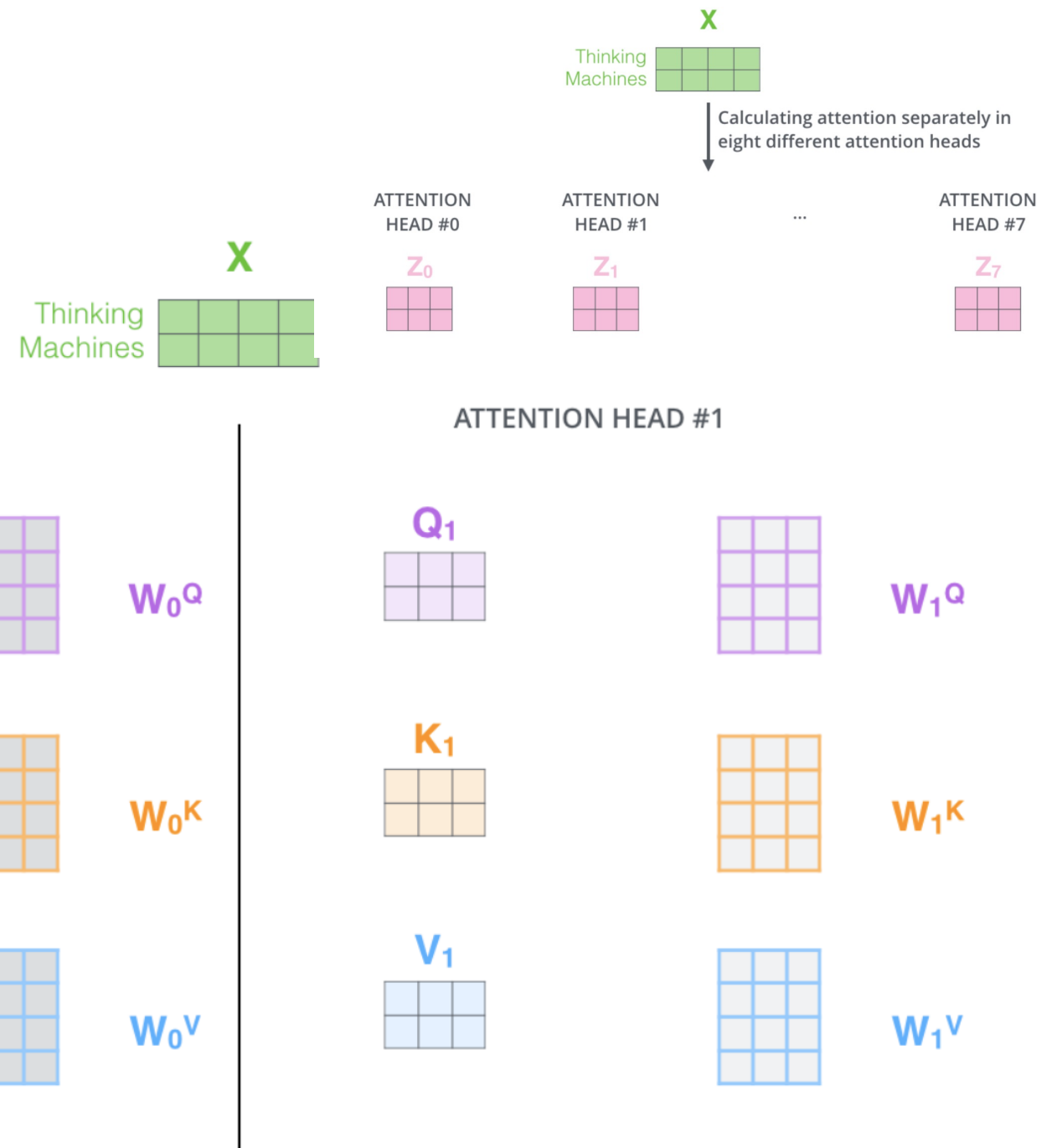
$$\begin{matrix} \text{X} \\ \begin{matrix} \text{2x4 grid} \end{matrix} \end{matrix} \times \begin{matrix} \text{W}^Q \\ \begin{matrix} \text{4x3 grid} \end{matrix} \end{matrix} = \begin{matrix} \text{Q} \\ \begin{matrix} \text{2x3 grid} \end{matrix} \end{matrix}$$

$$\begin{matrix} \text{X} \\ \begin{matrix} \text{2x4 grid} \end{matrix} \end{matrix} \times \begin{matrix} \text{W}^K \\ \begin{matrix} \text{4x3 grid} \end{matrix} \end{matrix} = \begin{matrix} \text{K} \\ \begin{matrix} \text{2x3 grid} \end{matrix} \end{matrix}$$

$$\begin{matrix} \text{X} \\ \begin{matrix} \text{2x4 grid} \end{matrix} \end{matrix} \times \begin{matrix} \text{W}^V \\ \begin{matrix} \text{4x3 grid} \end{matrix} \end{matrix} = \begin{matrix} \text{V} \\ \begin{matrix} \text{2x3 grid} \end{matrix} \end{matrix}$$

Attention heads

- При многоголовом внимании мы обрабатываем отдельные матрицы весов $Q/K/V$ для каждой головы
- Это приводит к разным матрицам $Q/K/V$
- Как и раньше, мы умножаем X на матрицы $WQ/WK/WV$, чтобы получить матрицы $Q/K/V$



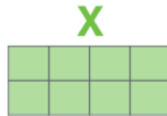
Attention result

- Когда мы получили результат для каждой головы нам нужно его объединить
- Используем еще одну матрицу W (которую тоже обучаем)
- Это пример просто линейной регрессии

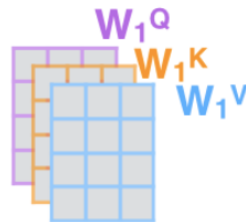
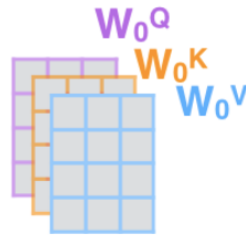
1) This is our input sentence*

Thinking
Machines

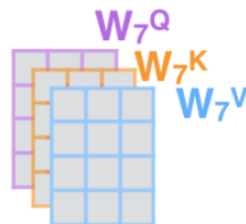
2) We embed each word*



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



...



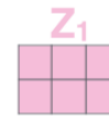
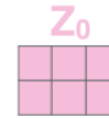
4) Calculate attention using the resulting $Q/K/V$ matrices



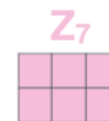
...



5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer



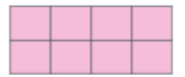
...



W^O

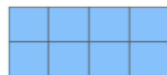


Z

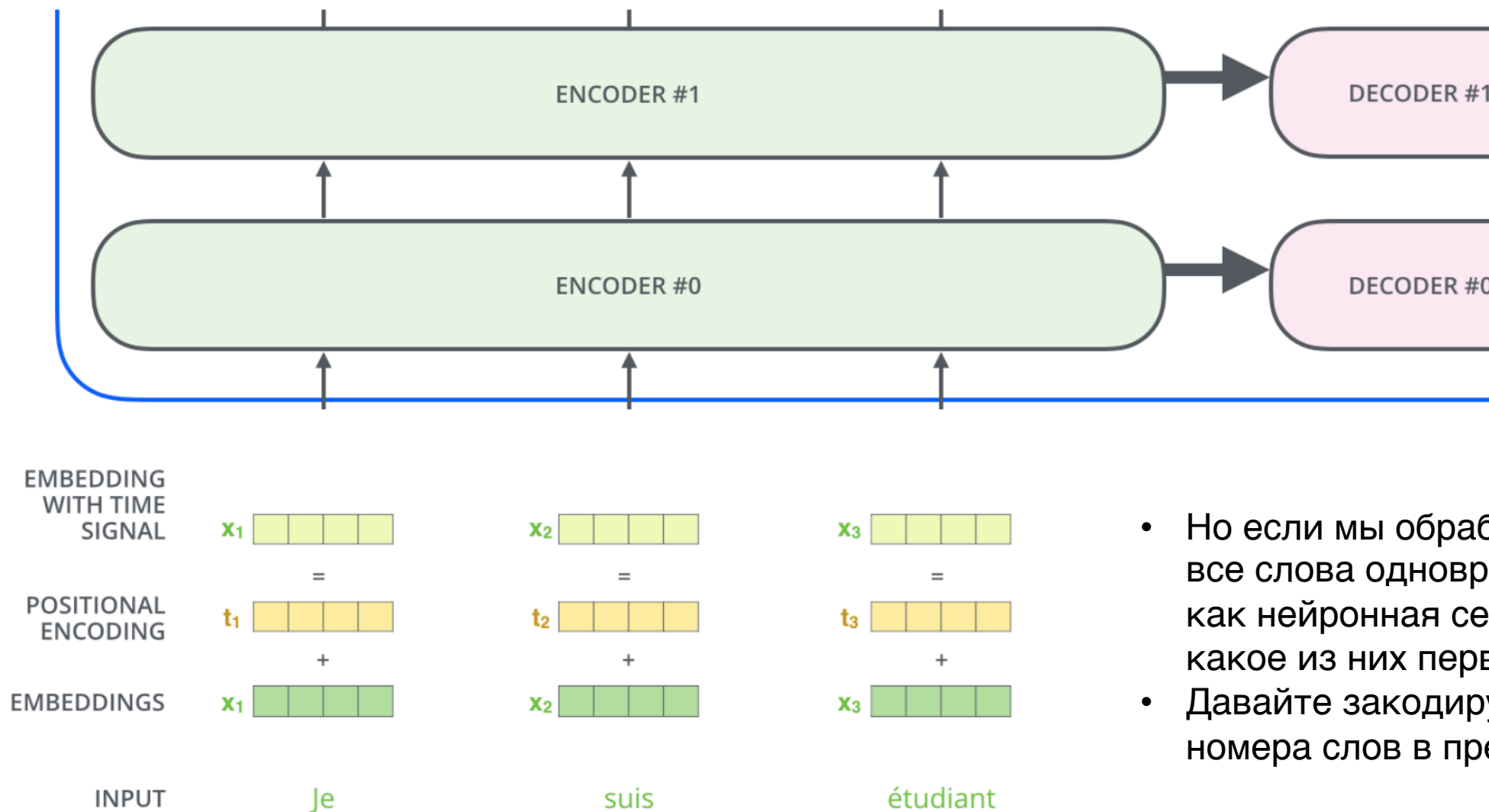


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

R



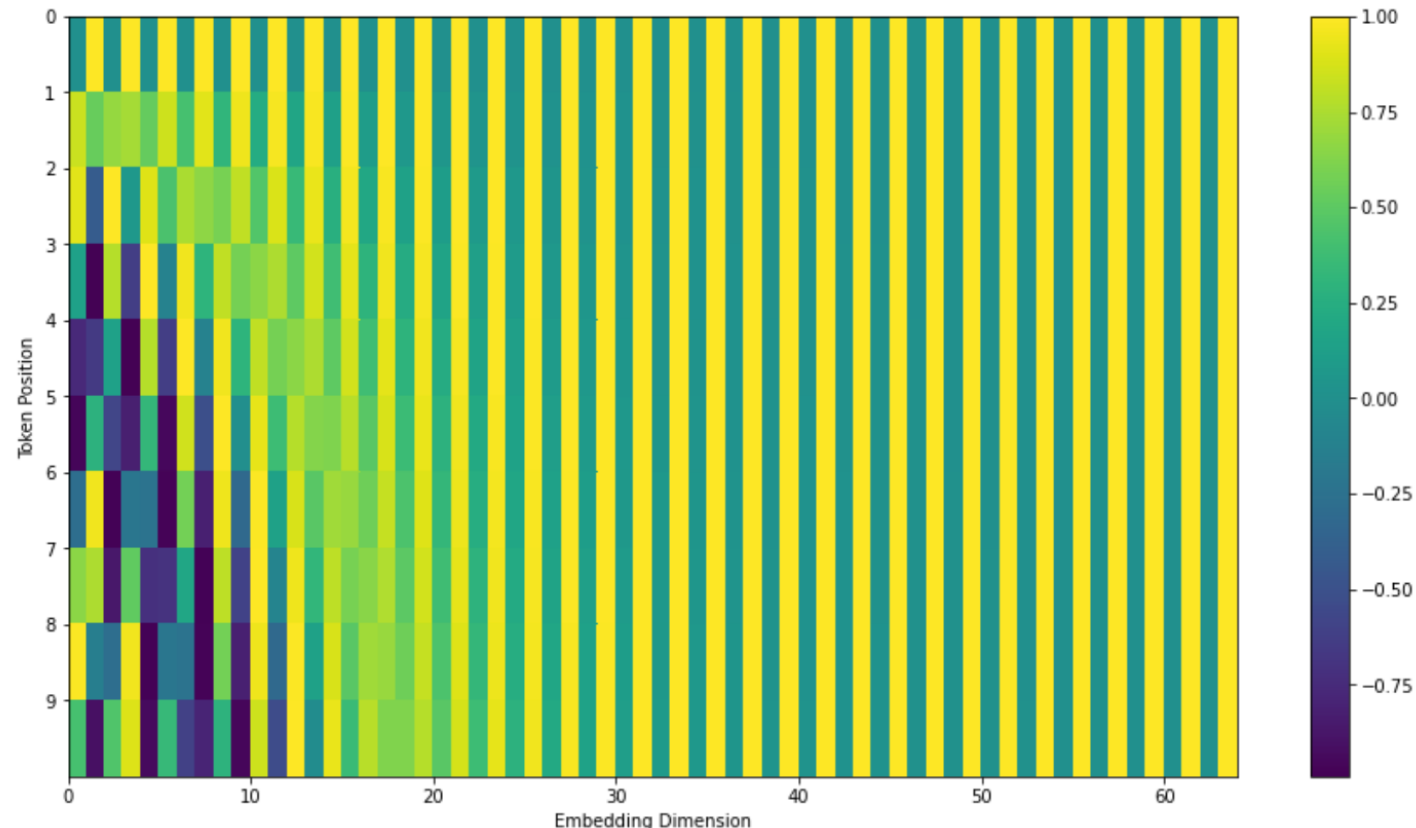
Позиции



- Но если мы обрабатываем все слова одновременно, как нейронная сеть поймет, какое из них первое?
- Давайте закодируем номера слов в предложении

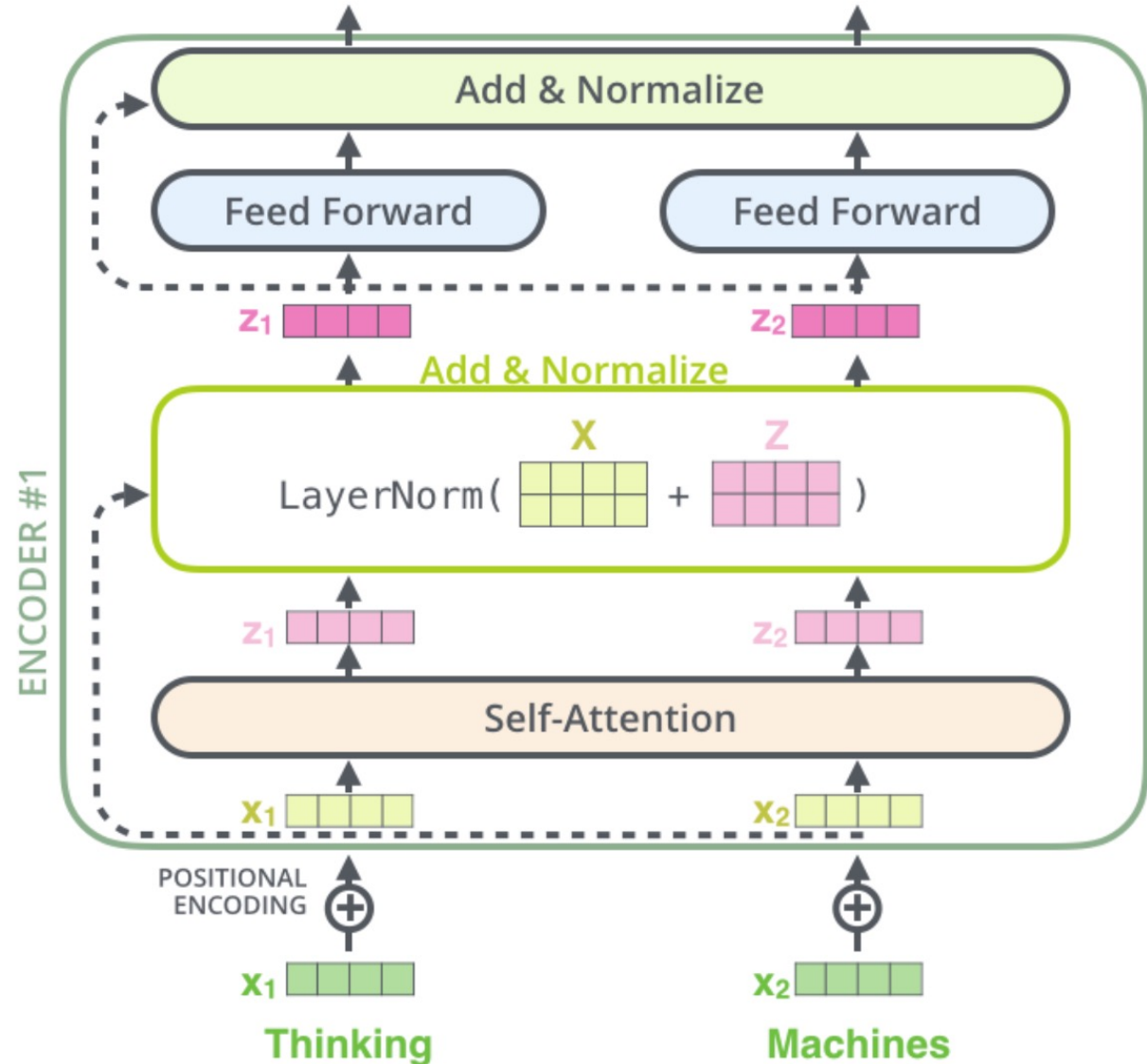
Positional encoding

- Пример positional encoding с игрушечным размером эмбединга 4
- Используем sin/cos чтобы посчитать позицию слова
- Как стрелки на часах и бинарный код вместе



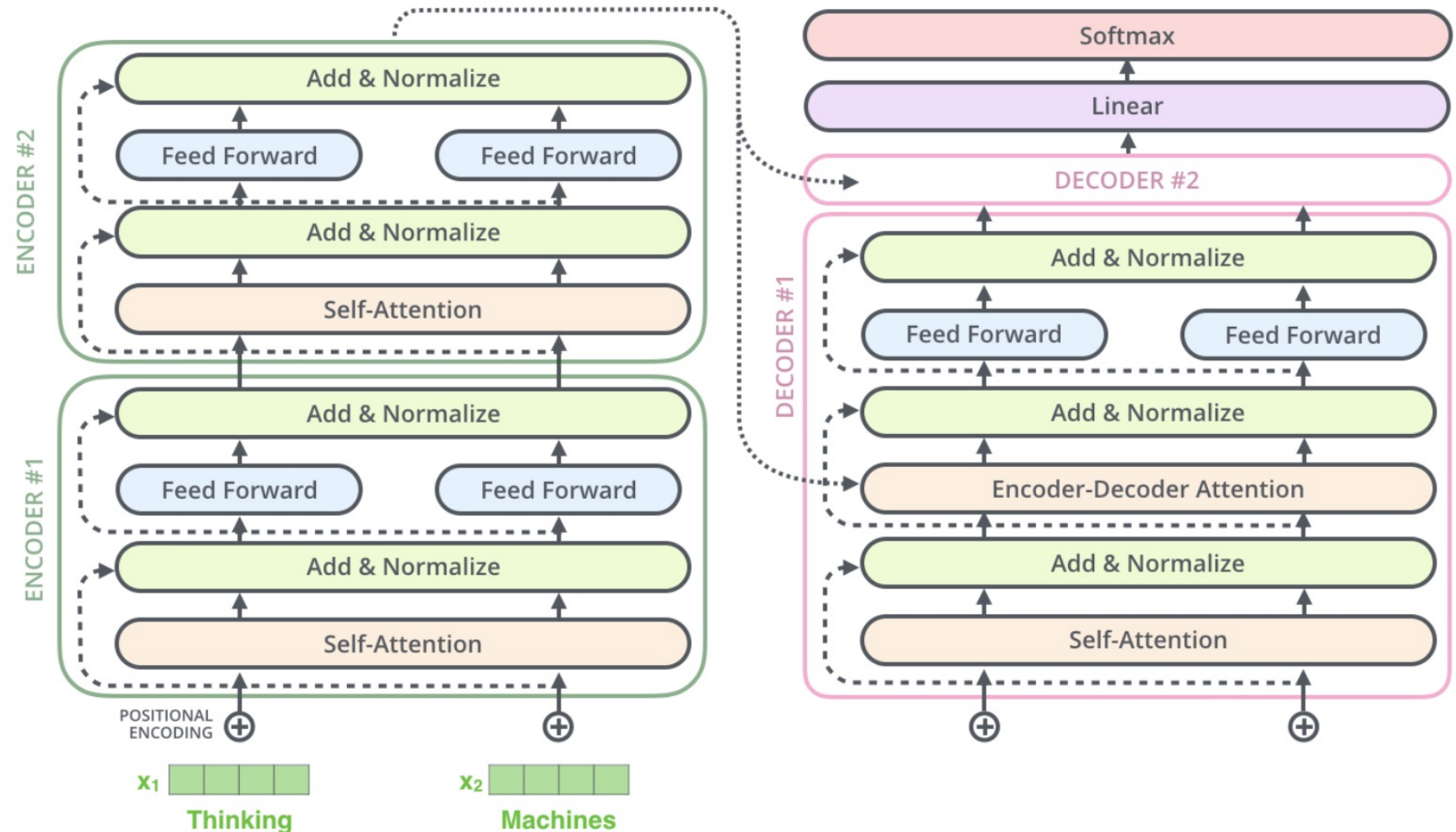
Add and Normalize

- Весь процесс вычислений в одном слое энкодера
- Все входные слова мы можем посчитать вместе и параллельно (как в CNN)
- В RNN мы и входной текст подаем по одному слову (формируем память)
- В трансформере нет скрытой памяти – новый сгенерированный текст это вся наша «память»



Encoder-decoder connection

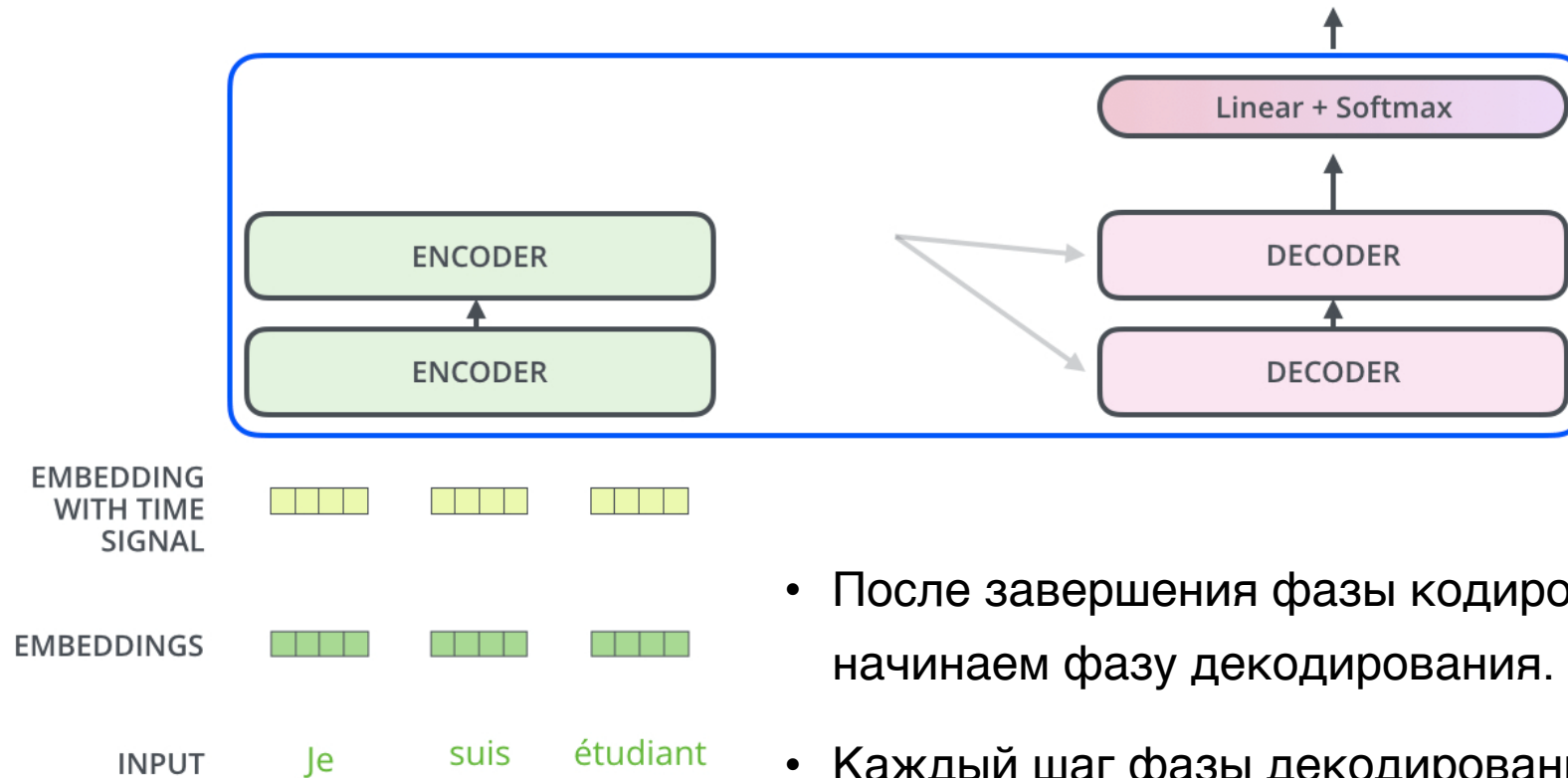
- Вместе два слоя энкодера и два слоя декодера для двух входных слов
- Все входные данные мы уже имеем в тексте (как в авторегрессии)
- Нам НЕ нужно подавать текст только по одному слову как в RNN на вход нейросети
- Но генерация только по одному слову



Decoder

Decoding time step: 1 2 3 4 5 6

OUTPUT



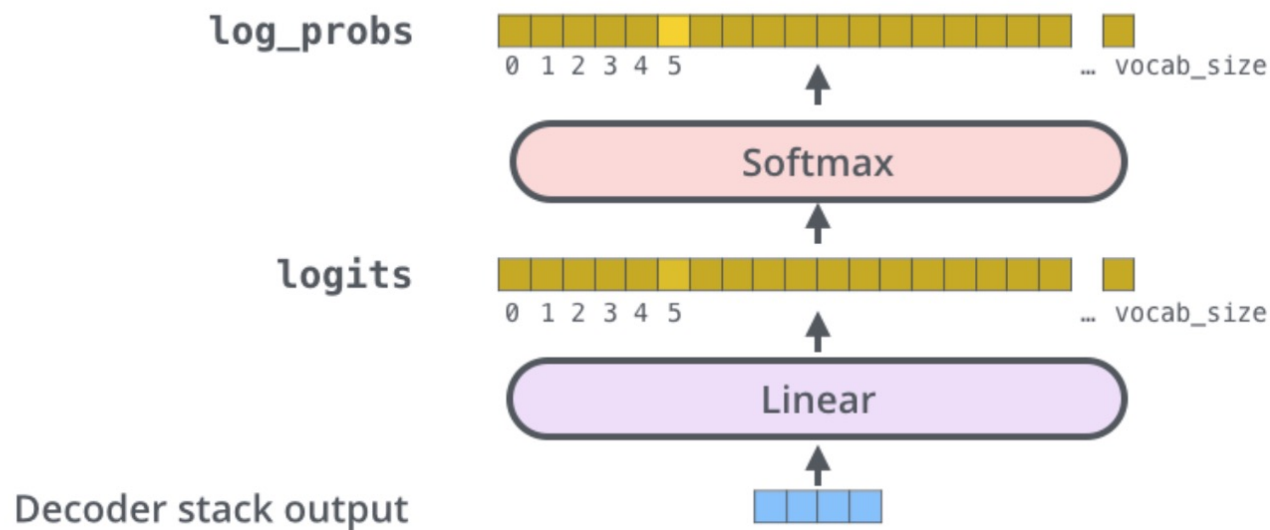
- После завершения фазы кодирования мы начинаем фазу декодирования.
- Каждый шаг фазы декодирования выводит элемент из выходной последовательности (в данном случае предложение перевода на английский язык)

Итоговое слово

- Модель знает 10,000 уникальных English слов (output vocabulary) , которому мы обучили на training dataset
- Чтобы вывести нужно слово нам нужен вектор из 10,000 чисел – One-hot encoding, каждое число вероятность нужного слова.
- Это является выходом модели в Linear layer.

Which word in our vocabulary
is associated with this index?

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



Model output

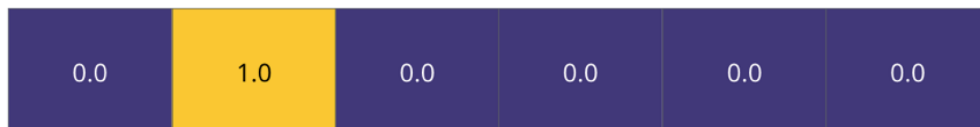
- Наша модель генерирует по очереди следующие слова
- В итоге у нас получается все предложение:

I am a student <eos>

Output Vocabulary

WORD	a	am	I	thanks	student	<eos>
INDEX	0	1	2	3	4	5

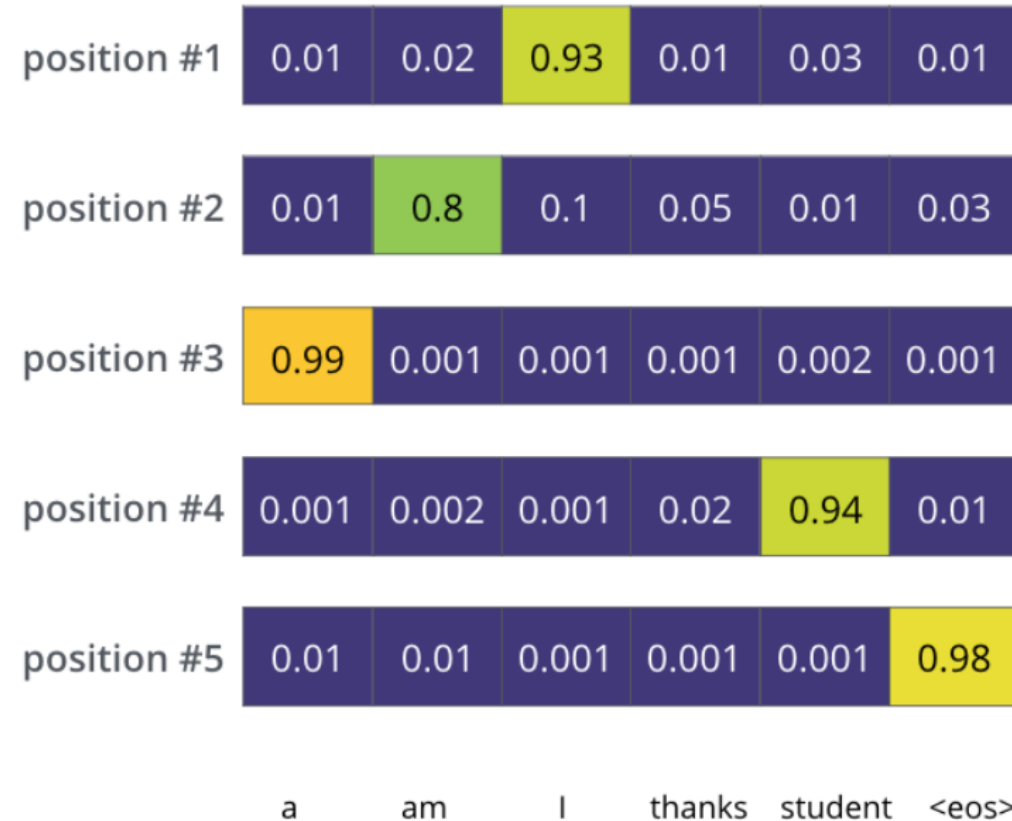
One-hot encoding of the word "am"



Example: one-hot encoding of our output vocabulary

Trained Model Outputs

Output Vocabulary: a am I thanks student <eos>

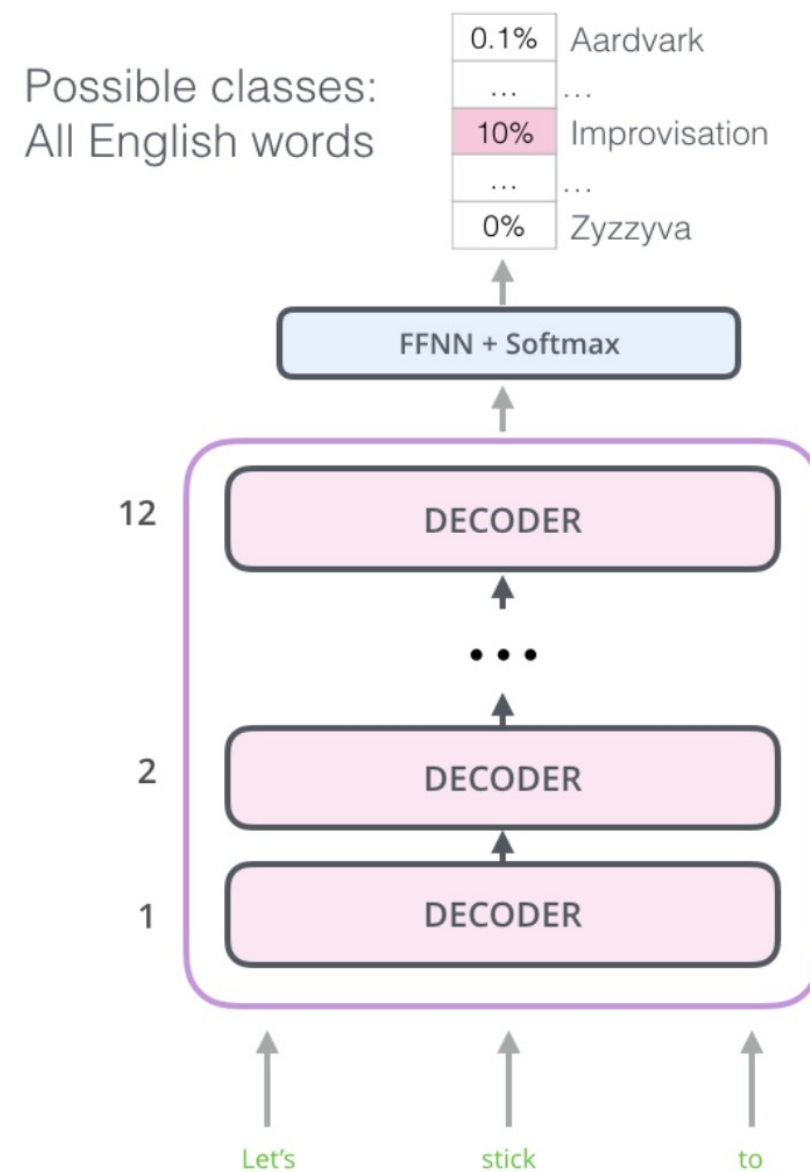


GPT

- GPT 2018 (117 млн параметров), 4.5 Гб текста
- GPT2 2019 (1.5 млрд параметров), обучен на основе Reddit (40Гб)
- GPT3 2020 (до 175 млрд параметров), 570 Гб текста
- GPT3.5 (InstructGPT) март 2022, обратная связь от человека
- В ноябре 2022 на основе InstructGPT был создан сервис ChatGPT
- Февраль 2023 представлена **открытая** модель Llama
- GPT4 март 2023
- Ноябрь 2024 компания Anthropic представила протокол MCP
- Январь 2025 Deepseek представила модель R1

GPT

- С помощью подобной структуры мы можем переходить к обучению модели для все той же задачи языкового моделирования: предсказать следующее слово, используя большой неразмеченный набор данных.
- Достаточно просто загрузить 7 тысяч книг и обучить на них модель.
- Книги для данного рода задач подходят отлично, т.к. они позволяют модели научиться находить связанные по смыслу фрагменты текста, даже если они значительно отстоят друг от друга
- Этого нельзя достигнуть, если обучать модель на твитах или новостных заметках.



Открытые LLM модели

Часть популярных моделей являются открытыми и каждый желающий, может их запустить у себя (Edge Device).

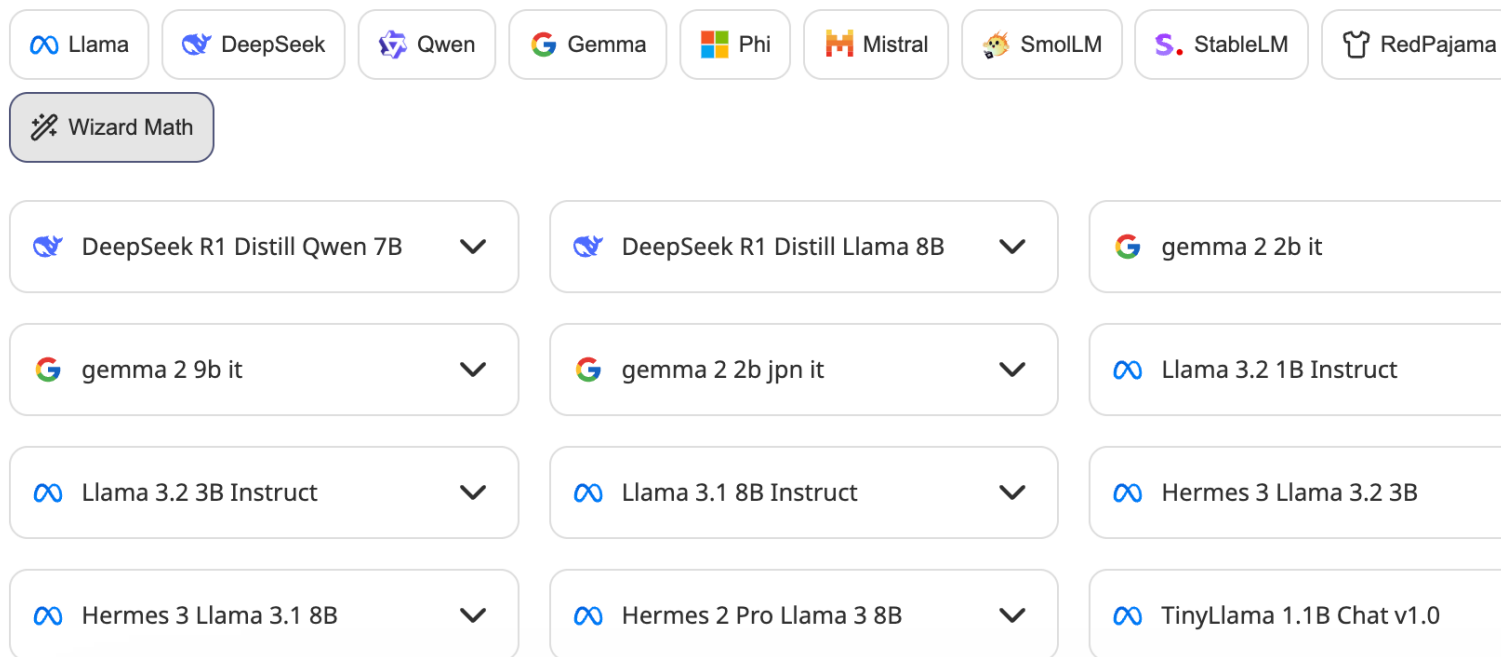
Список от mlc.ai:

Llama

Mistral от Mistral AI

Qwen от Alibaba Cloud

и тд



Также почитать:

https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard#/

Квантизация и дистилляция

Для запуска моделей на собственном устройстве (Edge Device) нам нужно уменьшить размер модели

Большие модели имеют миллиарды параметров, а значит требуют гигабайтов памяти

- Дистилляция – обучение более легковесной модели, ее учителем является большая модель с хорошими метриками
- Квантизация – использование 16 битных значений вместо 32-битных: меньшая точность, но меньший размер модели

WebLLM

- WebLLM – пакет JavaScript, который позволяет запускать LLM прямо в браузере
- Использует WebGPU
- Вам требуется GoogleChrome
- Можете использовать разные модели, настроить температуру
- Температура – параметр вариативности ответа: низкая температура оставляет только токены с высокими вероятностями

Model Type	WebLLM Models
Модель	Llama-3.2-1B-Instruct-q4f32_1-MLC (Meta)
Context Window Length The maximum number of tokens for the context window	2K
Температура Чем выше значение, тем более случайный вывод	0.1
Top P Do not alter this value together with temperature	0.9
Максимальное количество токенов Максимальная длина вводных и генерируемых токенов	2000
Штраф за повторения Чем выше значение, тем больше вероятность общения на новые темы	0.3
Штраф за частоту Большее значение снижает вероятность повторения одной и той же строки	0.5

<https://chat.webllm.ai/>

Retrieval Augmented Generation (RAG)

- LLM часто генерируют галлюцинации
- Лучше добавить в запрос подходящие данные (аналог Google)
- BERT превращает запрос и документы в векторы
- Мы сравниваем векторы и похожие подаем вместе с запросом в LLM

