

Лекция 11

Продвинутое глубокое обучение: трансформеры

Дополнительные главы
машинного обучения
Андрей Фильченков

28.05.2021

План лекции

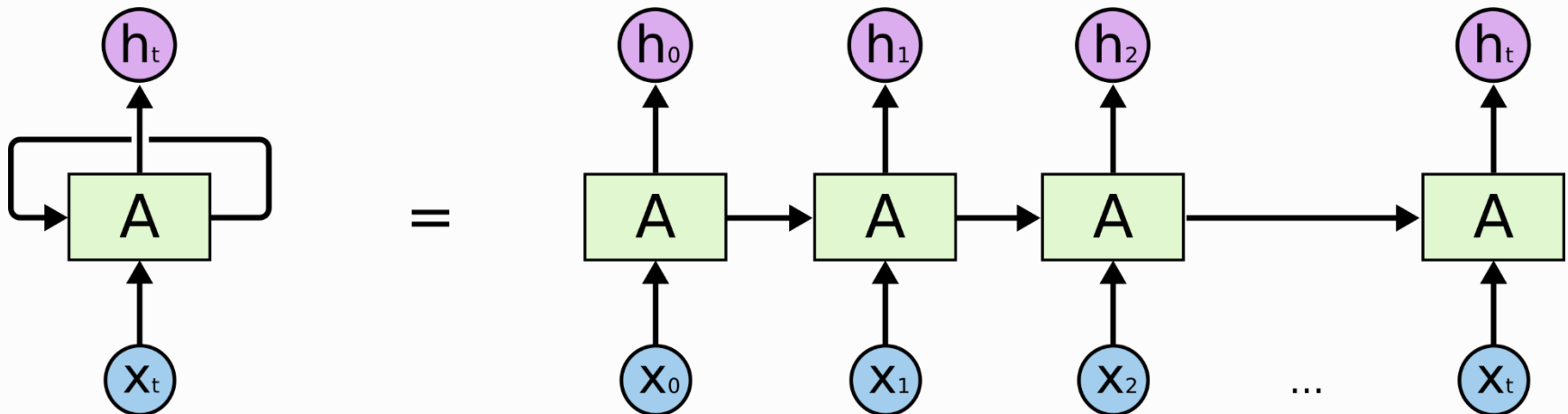
- Внимание (напоминание)
- Трансформер
- Другие трансформеры
- Семейства BERT и GPT
- В презентации используются материалы:
 - Выступления С.И. Орешина на семинаре лаборатории машинного обучения
 - openAI blog
- Слайды доступны: shorturl.at/wGV59
- Видео доступны: shorturl.at/ovBTZ

План лекции

- Внимание (напоминание)
- Трансформер
- Другие трансформеры
- Семейства BERT и GPT

Рекуррентная нейронная сеть (напоминание)

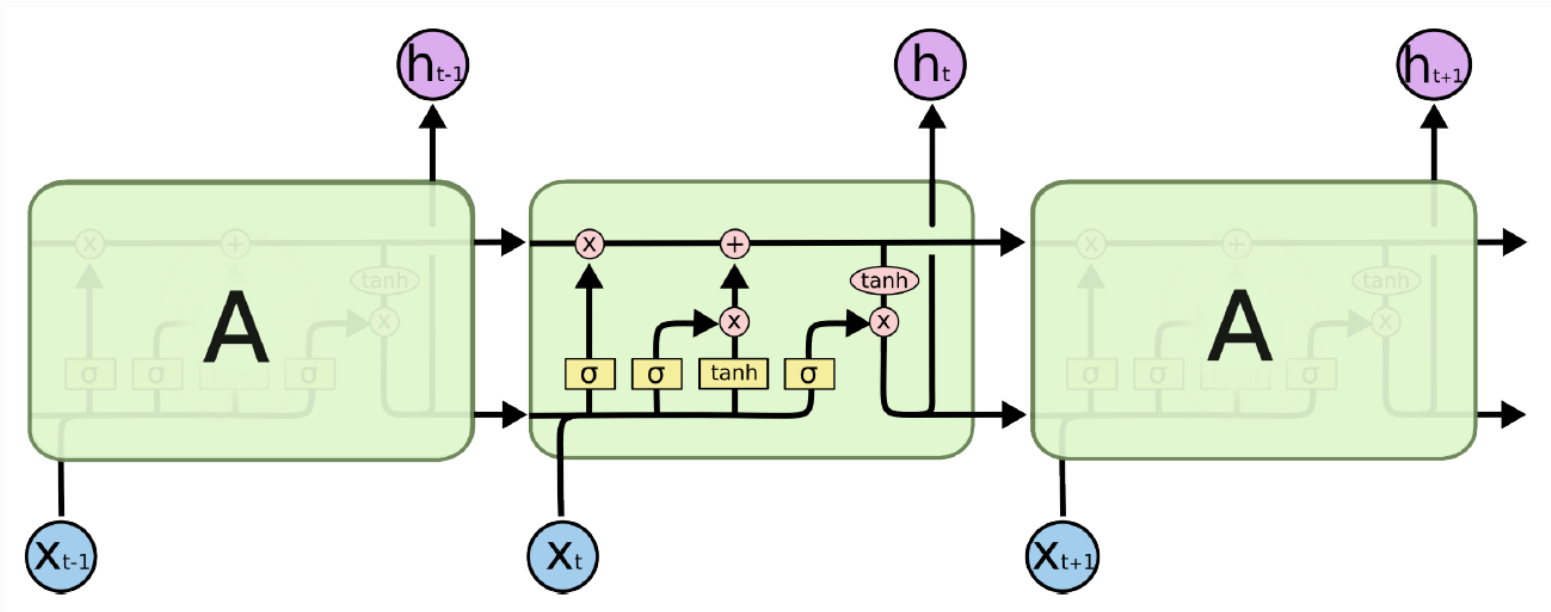
Сеть с петлями или развернутая сеть без петель



Модуль LSTM (напоминание)

Блок памяти используется в LSTM для хранения глобального состояния

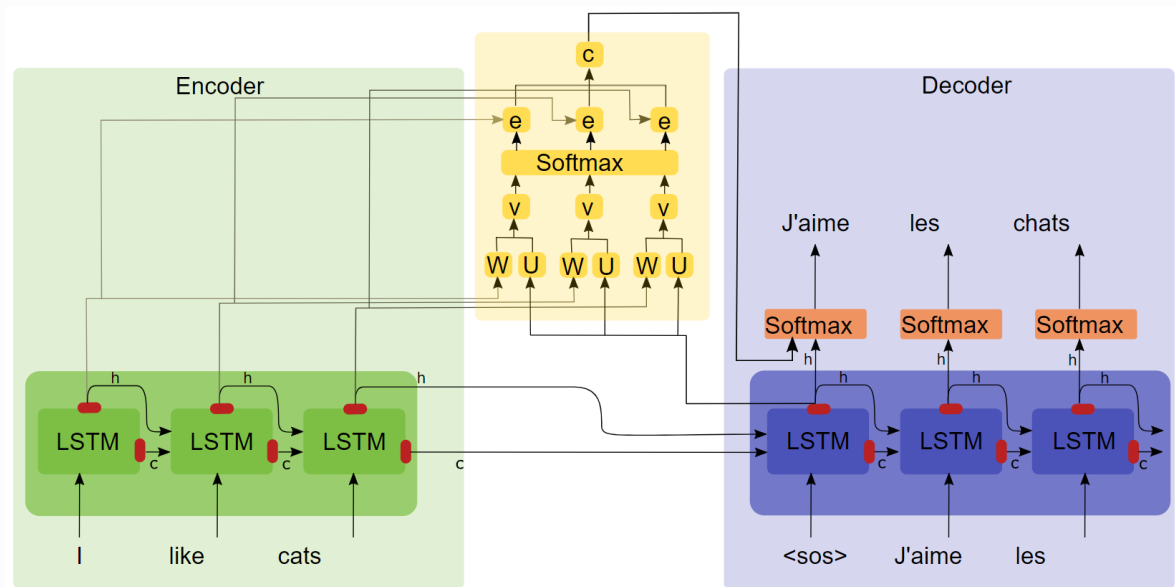
Ячейка параметрическая



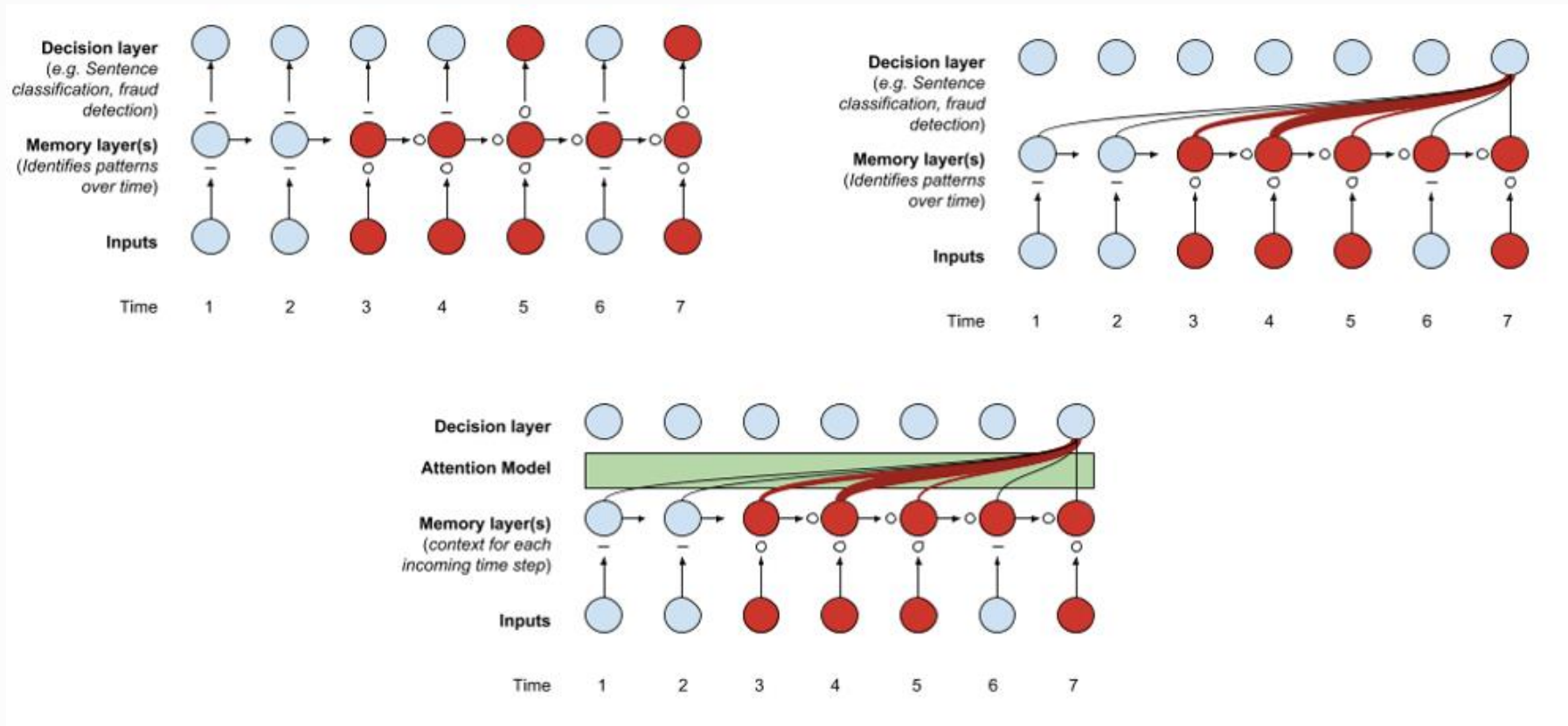
Предпосылка для механизма внимания (напоминание)

Предложение очень сложно (невозможно)
закодировать одним вектором.

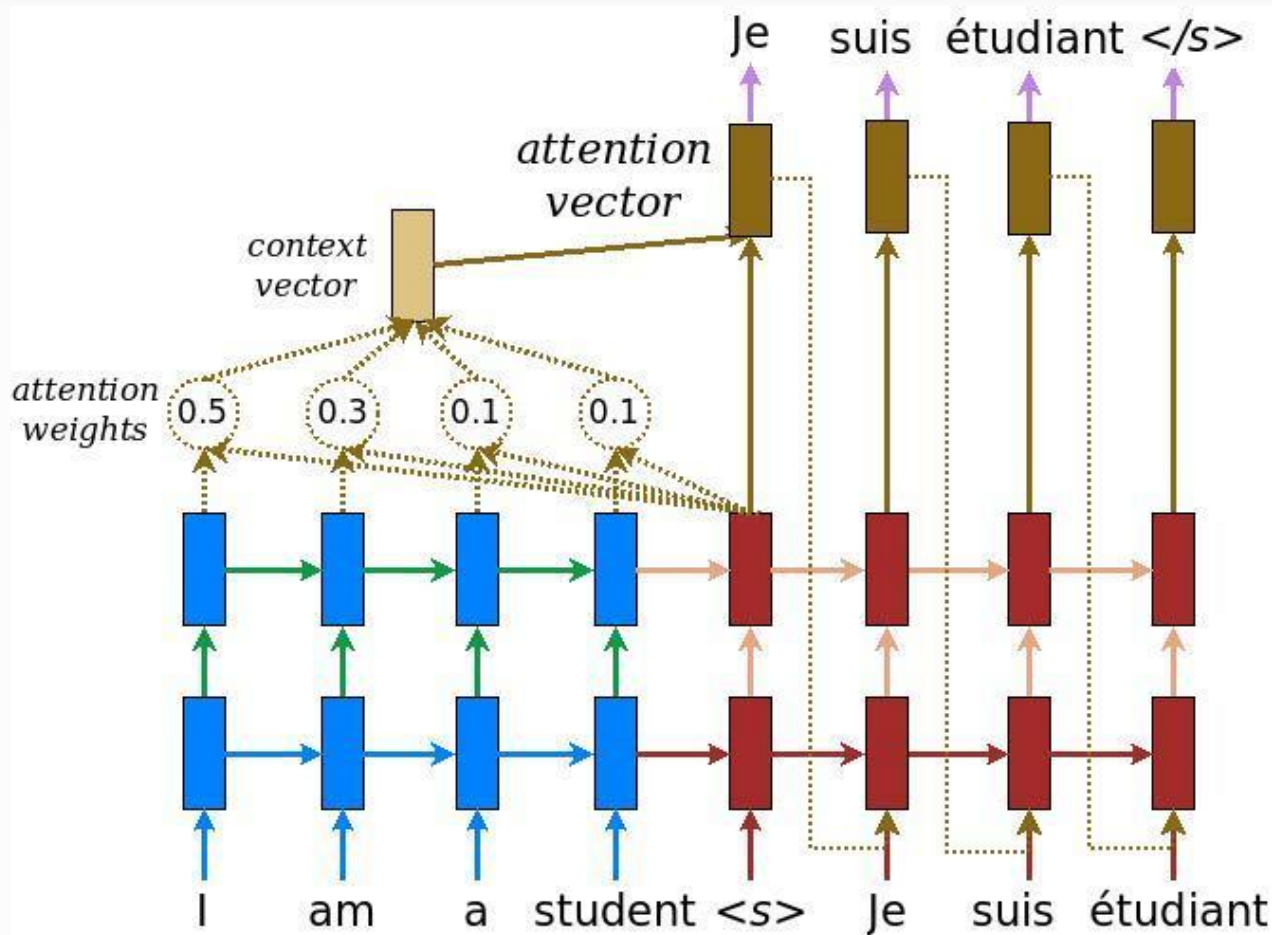
Для того, чтобы
его обработать,
можно посмотреть
не на одно
скрытое
состояние, а на
все скрытые
состояния.



Использование векторов скрытых состояний (напоминание)

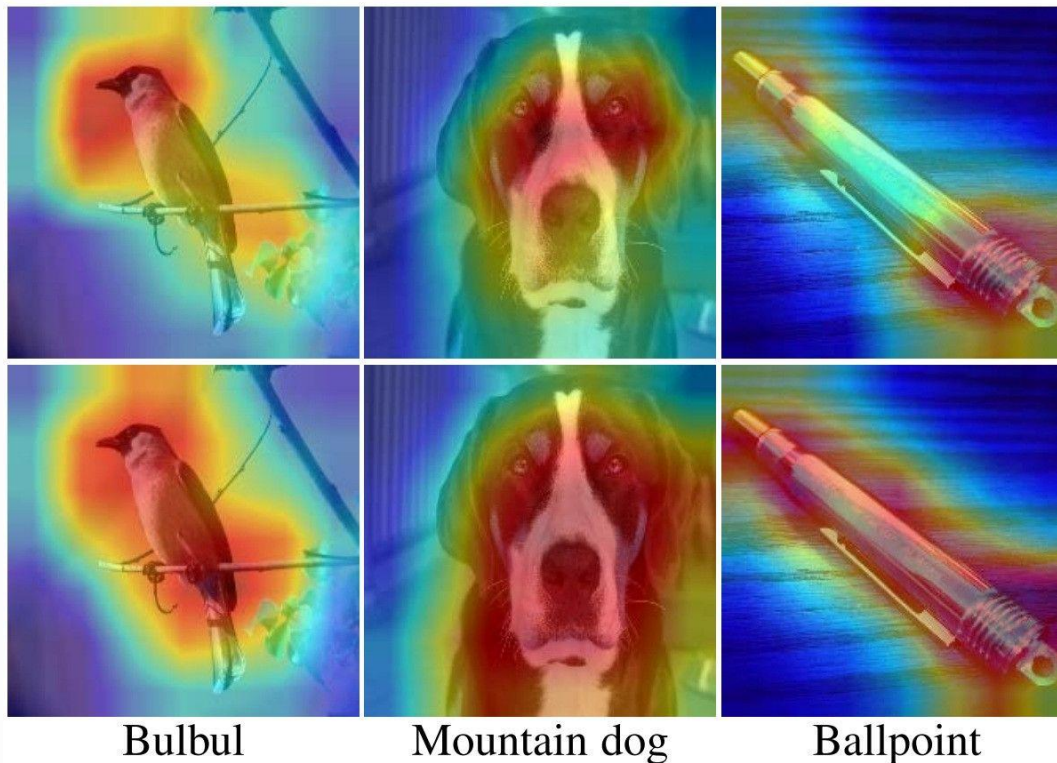


Веса, контекст и внимание (напоминание)

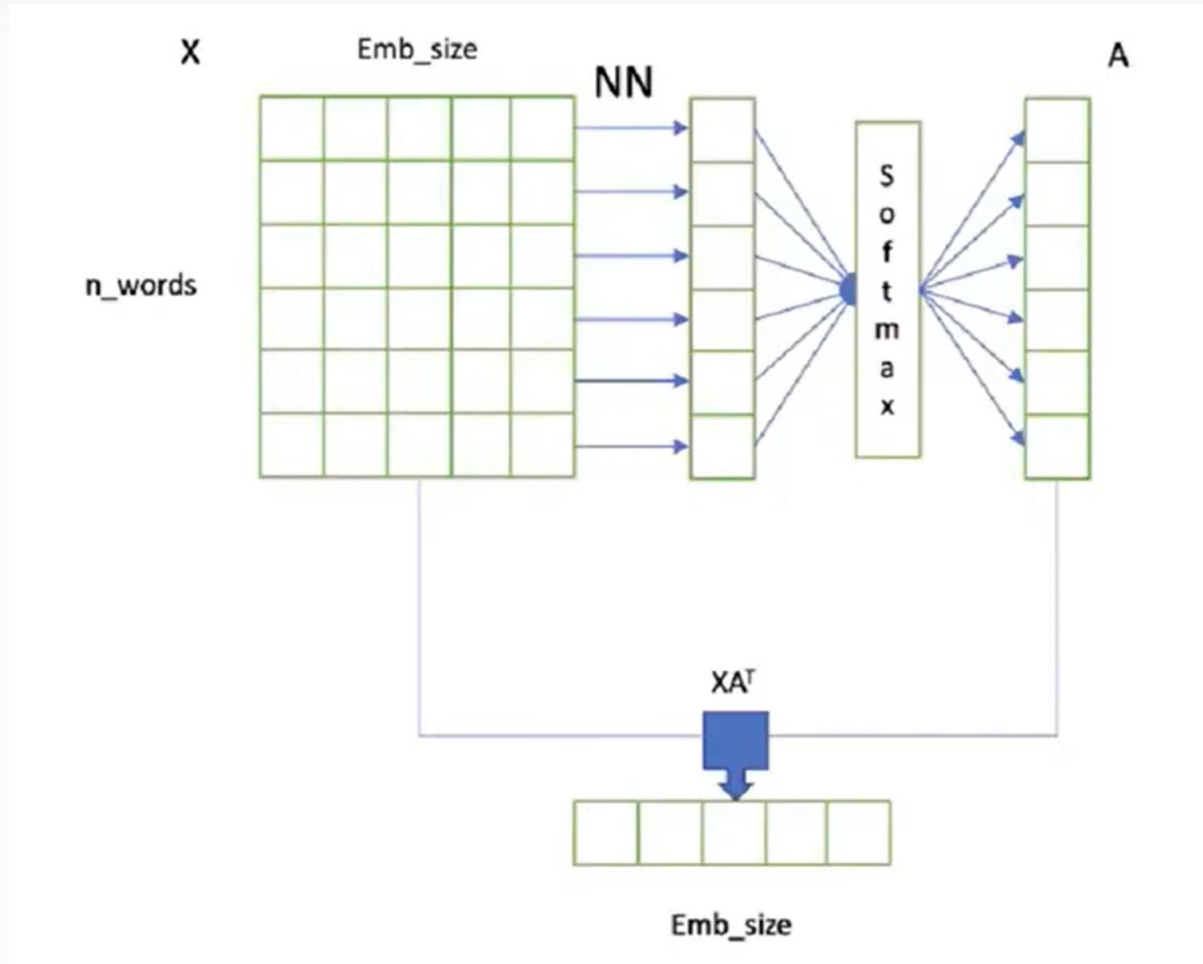


Внимание как комбинируемый слой

Блок внимания можно добавлять в произвольные сети. Дальше пытаться интерпретировать результаты.



Матричное представление



Преимущества

- Лучший учет контекста (чем при пуллинге)
- Адаптивное вычисление весов
- Большая вариативность конфигураций
- Распараллеливаемость
- Отсутствие взрыва / затухания градиента

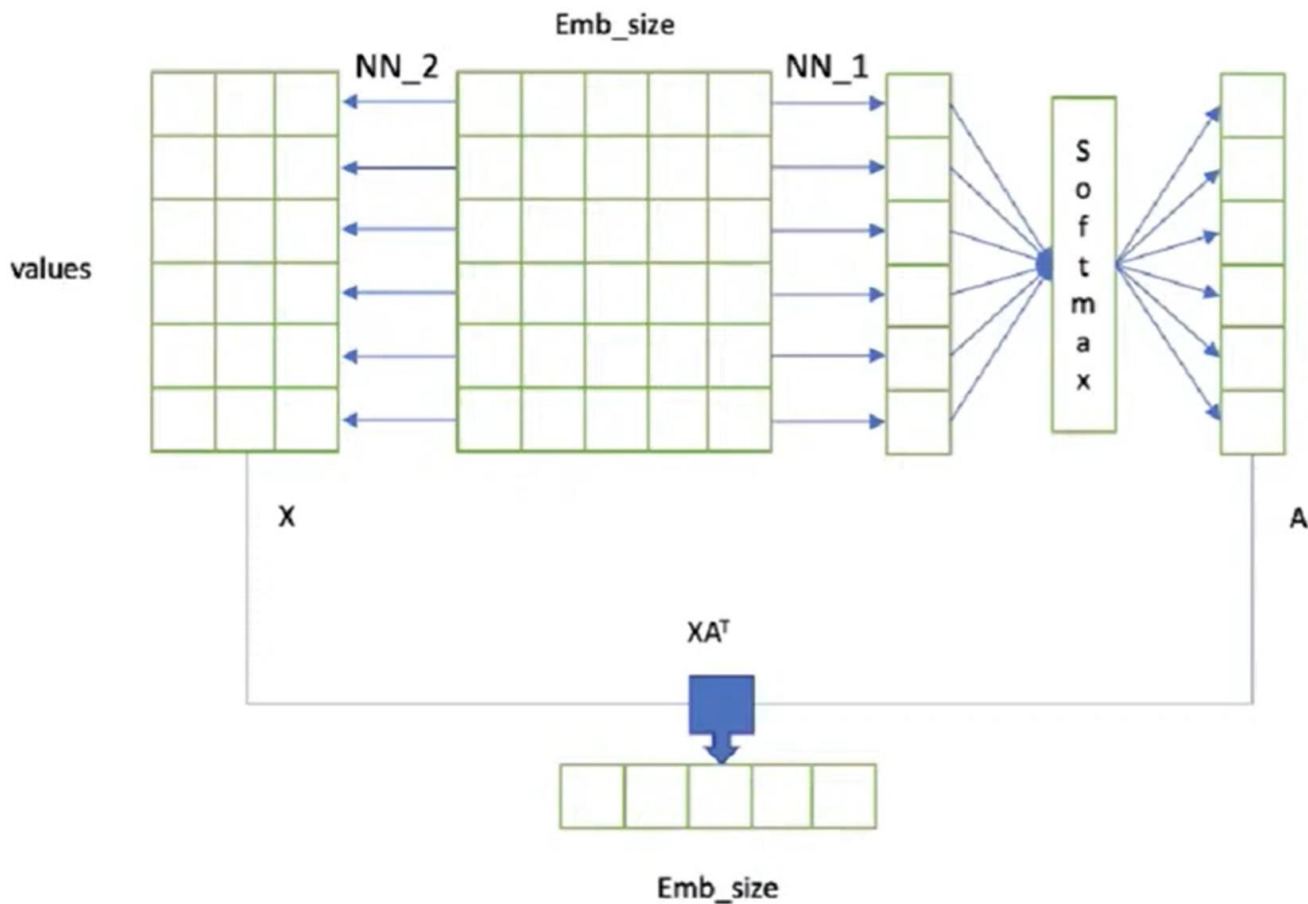
План лекции

- Внимание (напоминание)
- Трансформер
- Другие трансформеры
- Семейства BERT и GPT

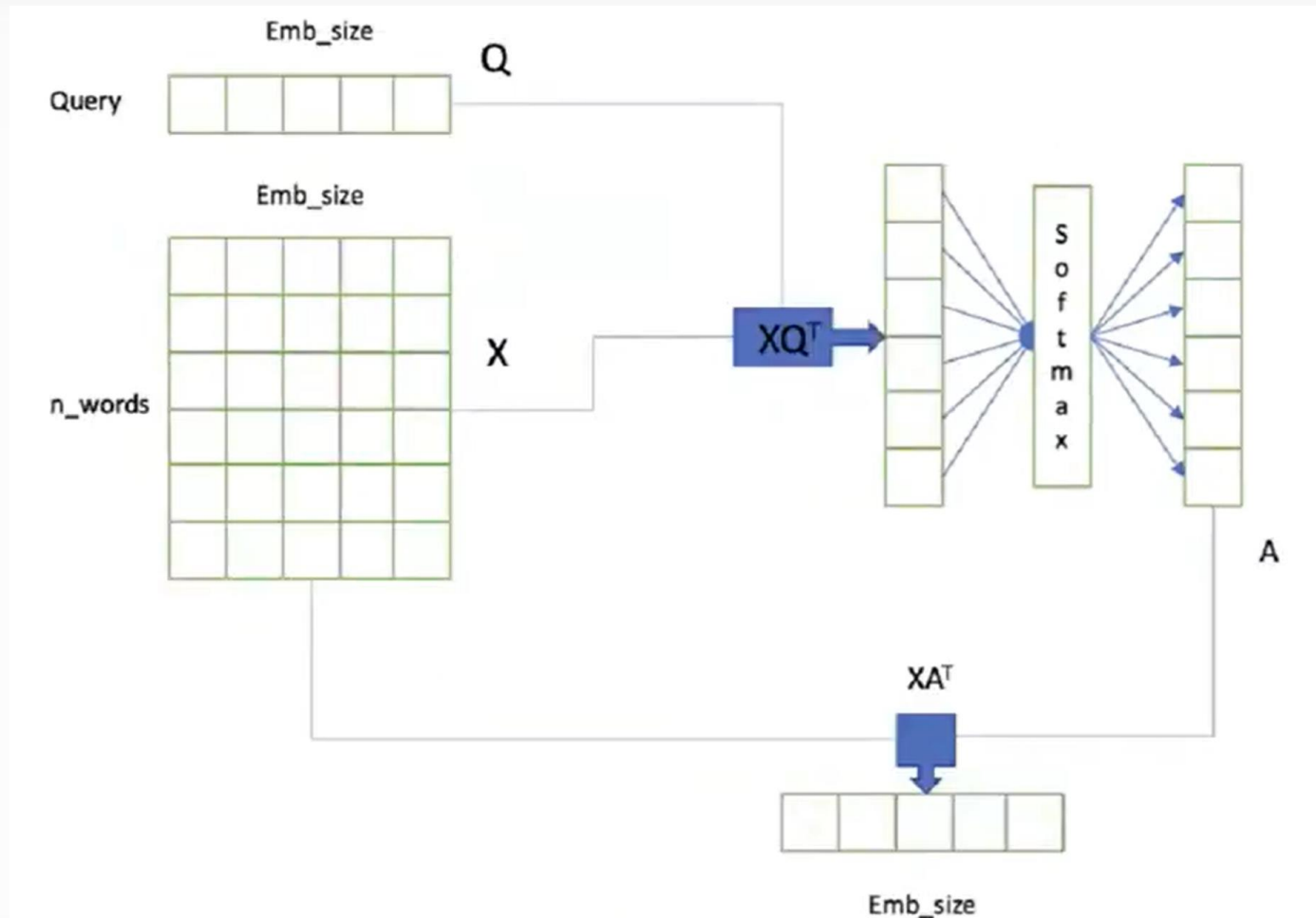
Основная идея

Основная идея: будем использовать только внимание как способ преобразования входного описания в выходное.

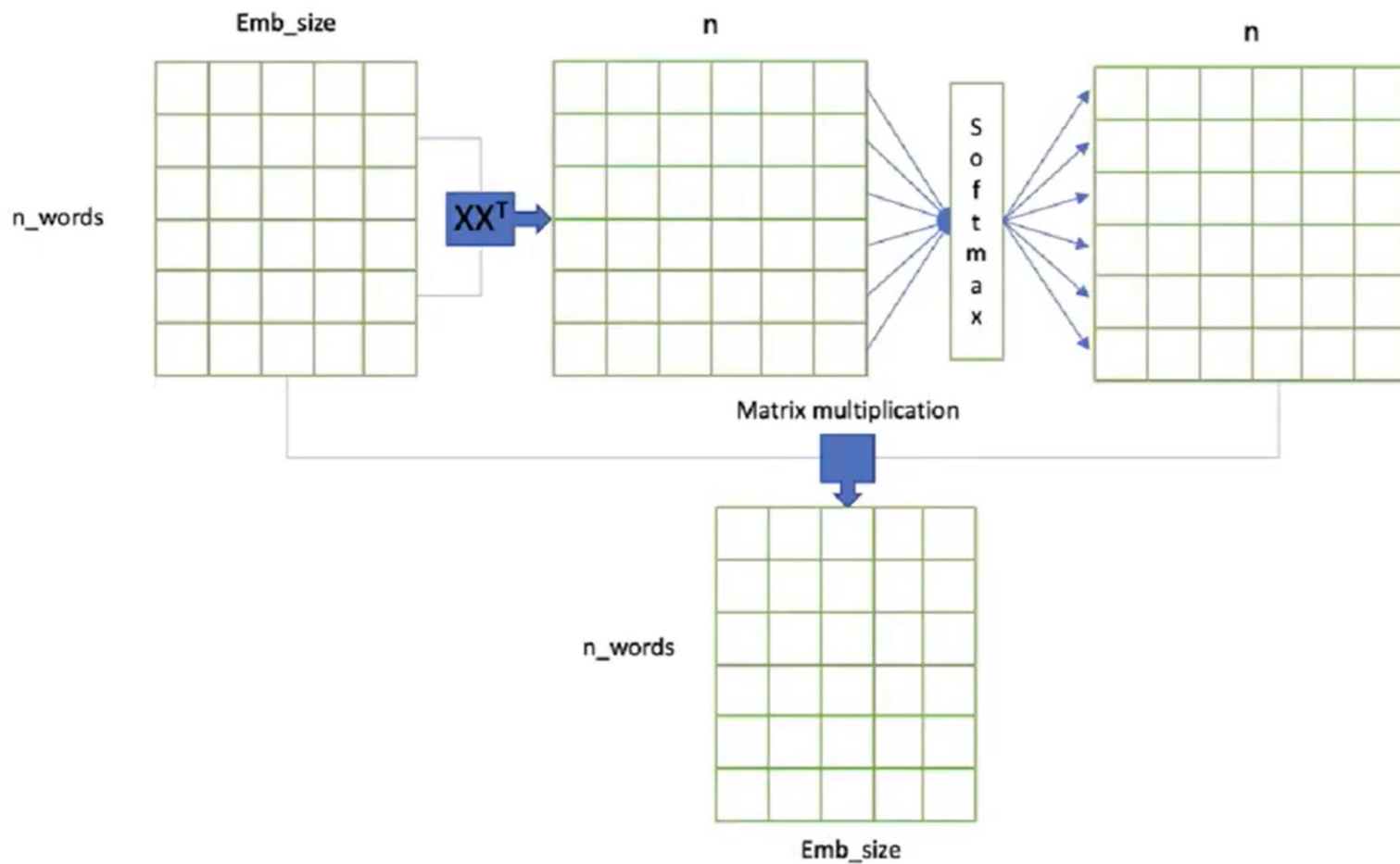
Дополнительное преобразование



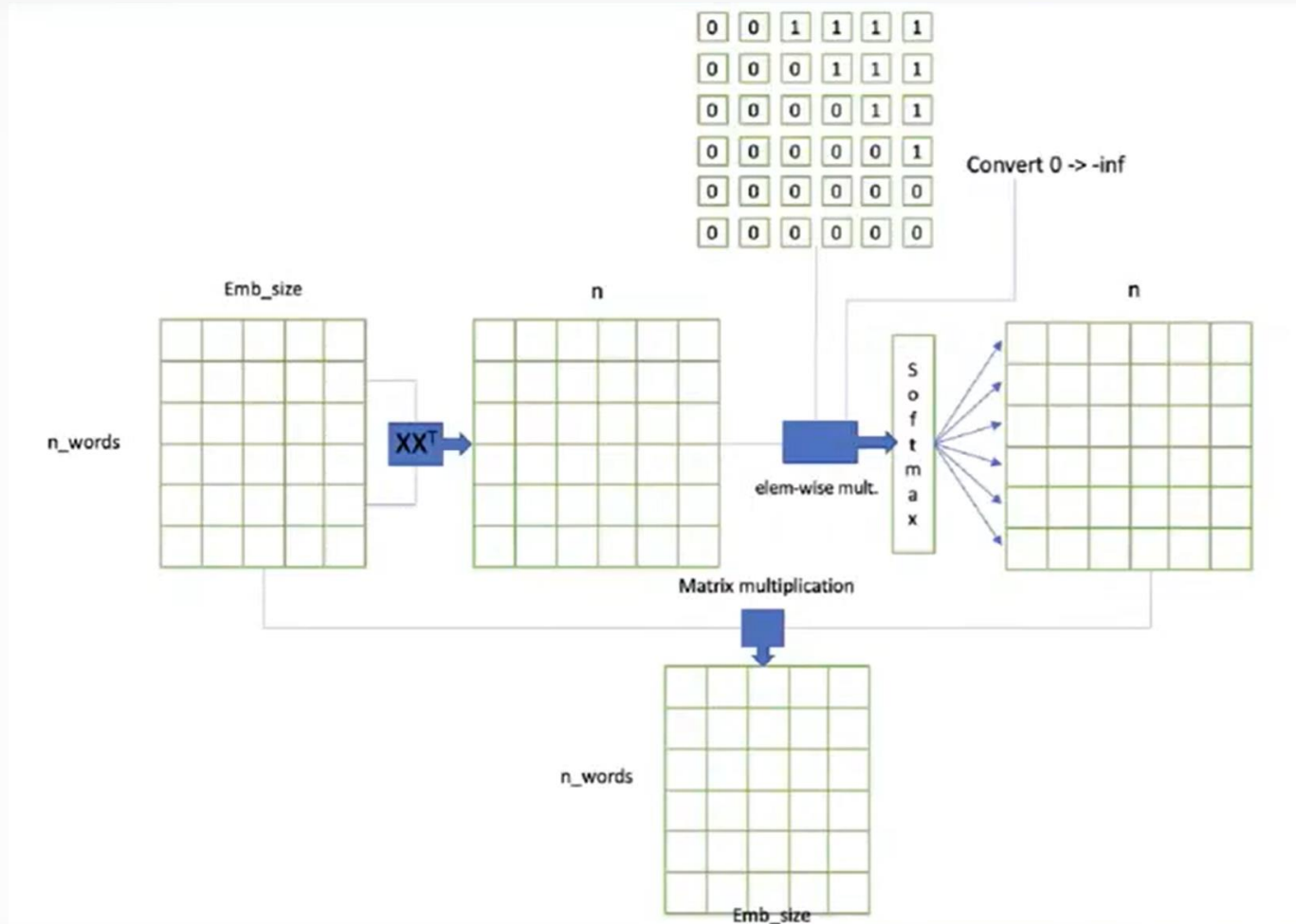
Добавление запроса



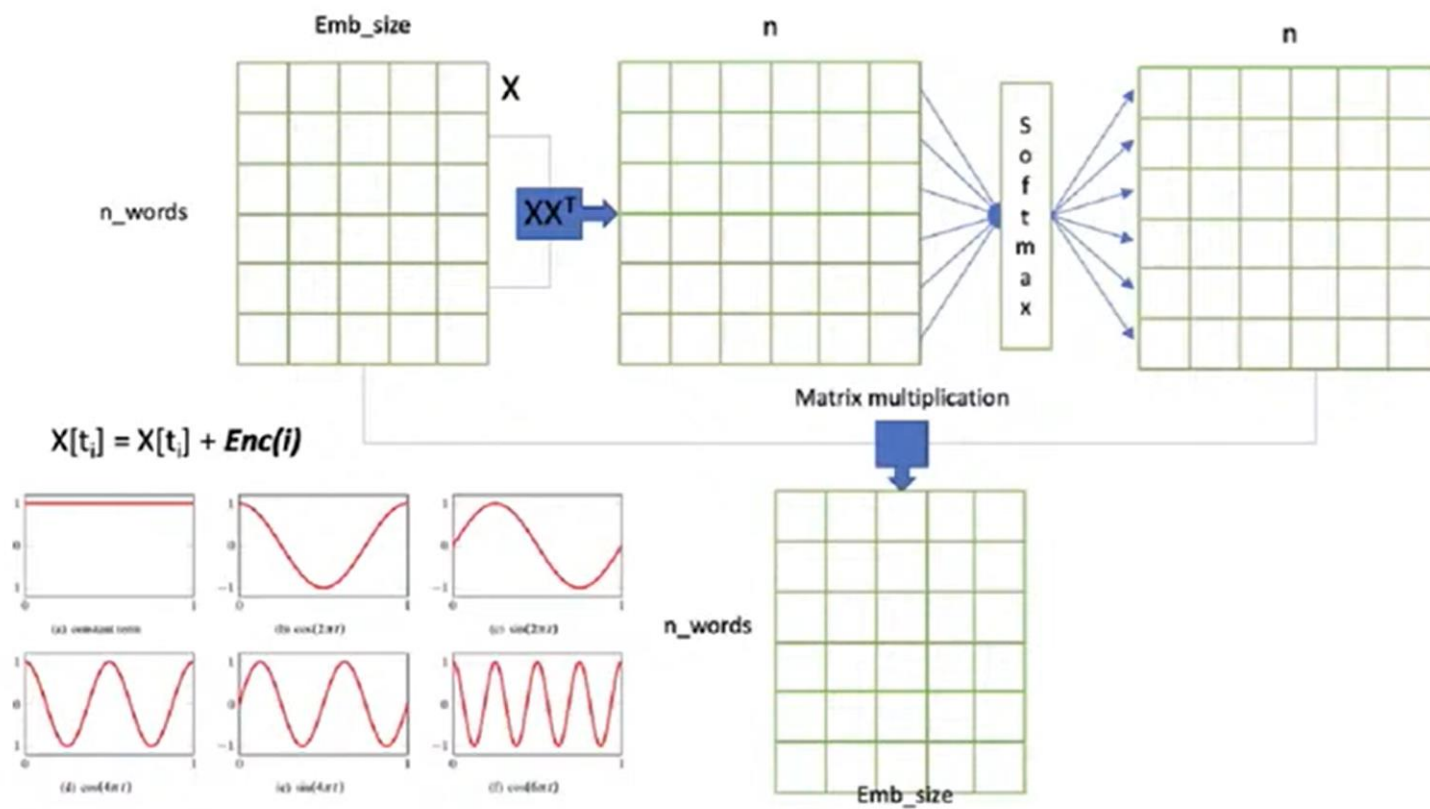
Self-attention



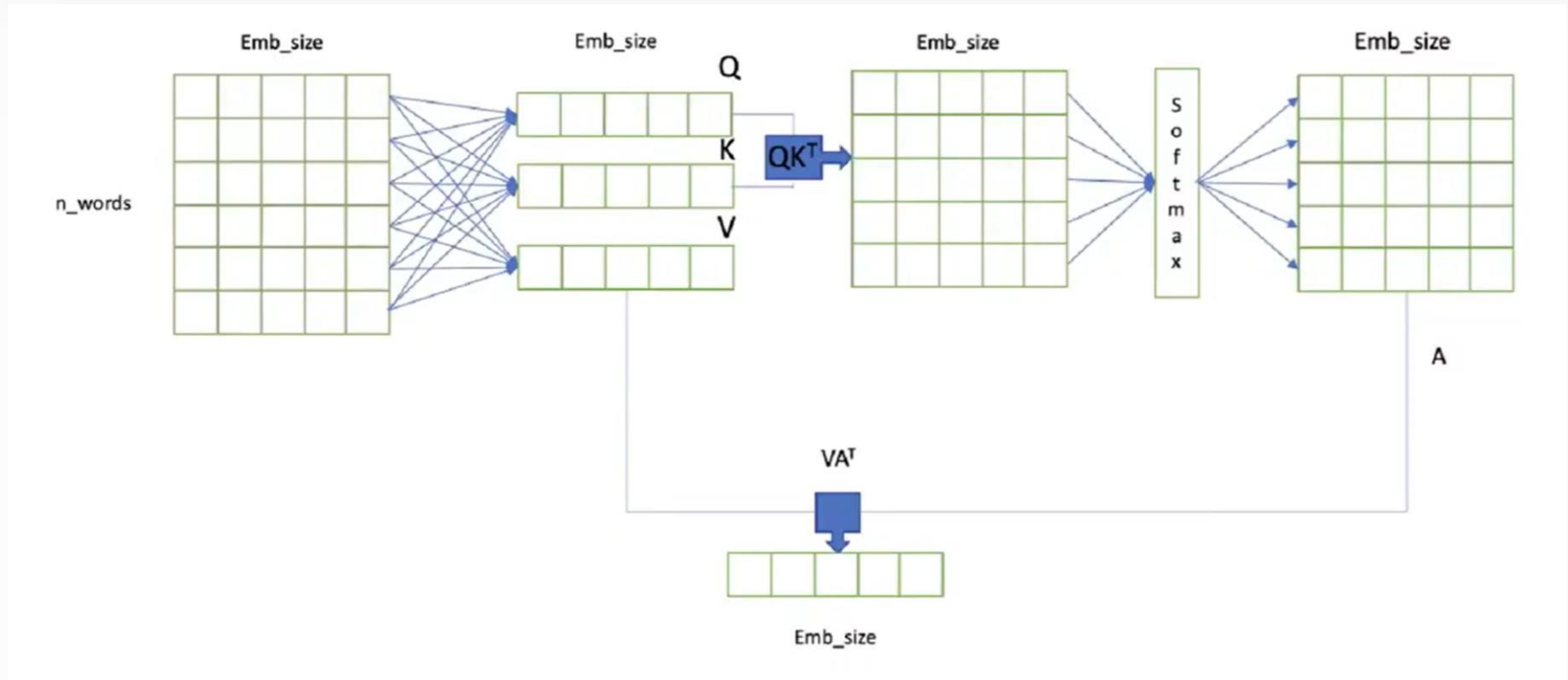
Учет неизвестности будущего



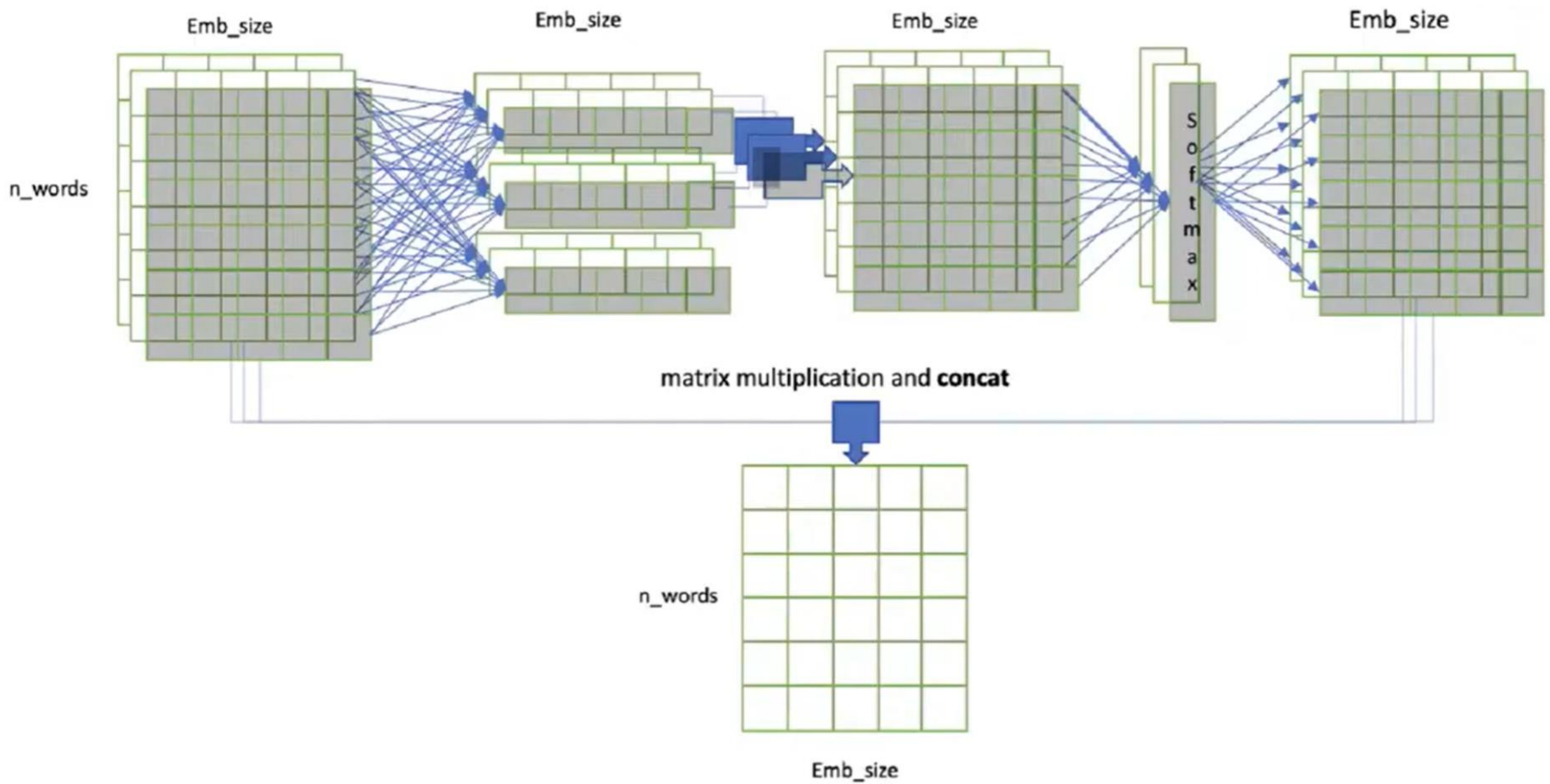
Учет позиции



Векторы Q, K, V

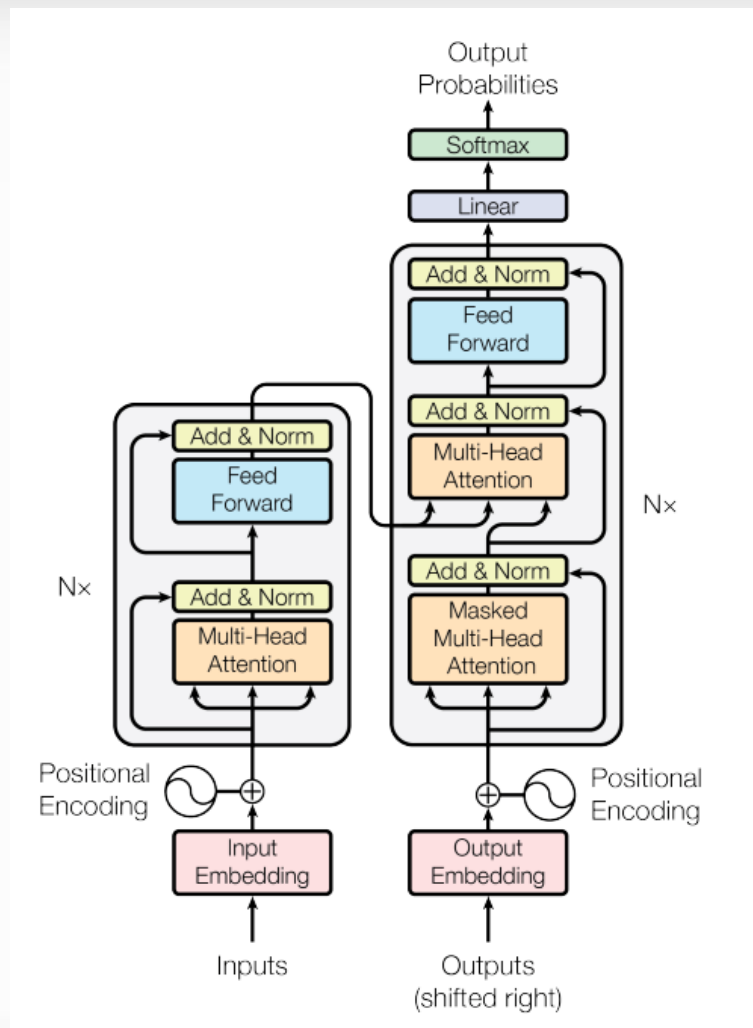


Множество головок

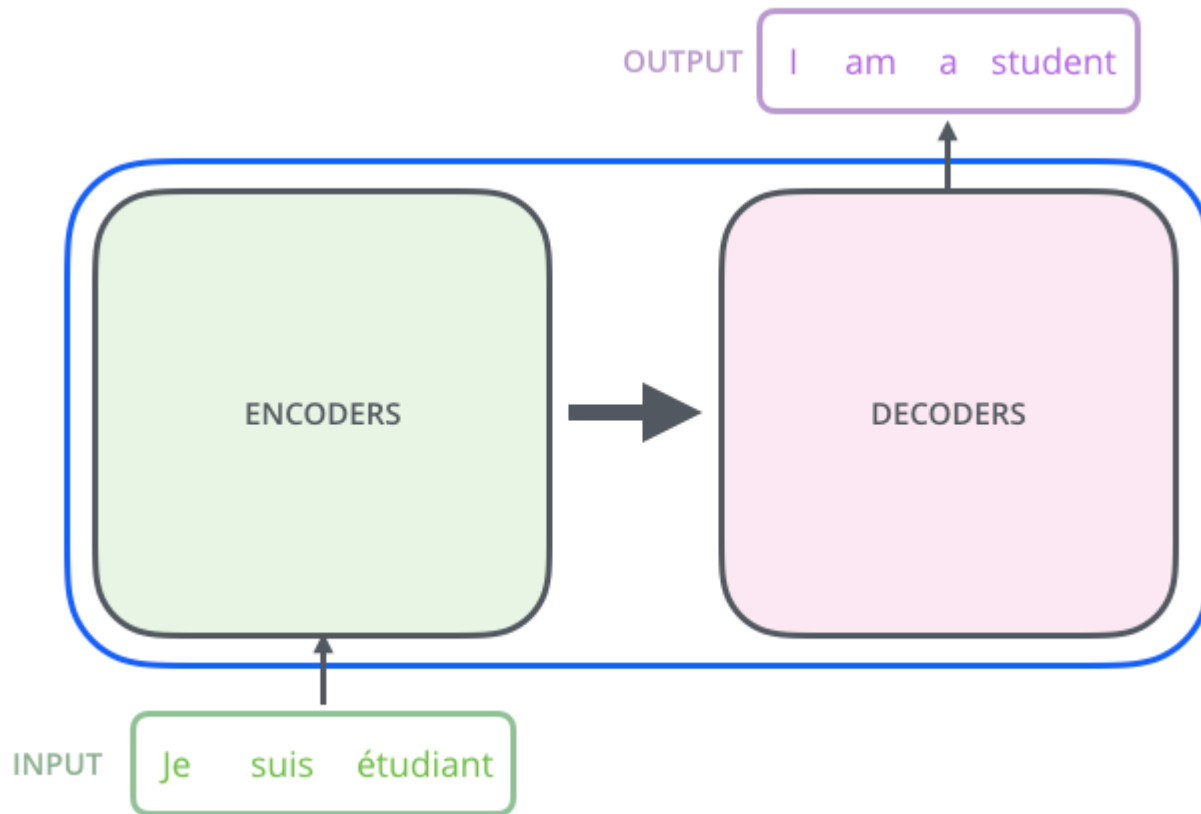


Трансформер

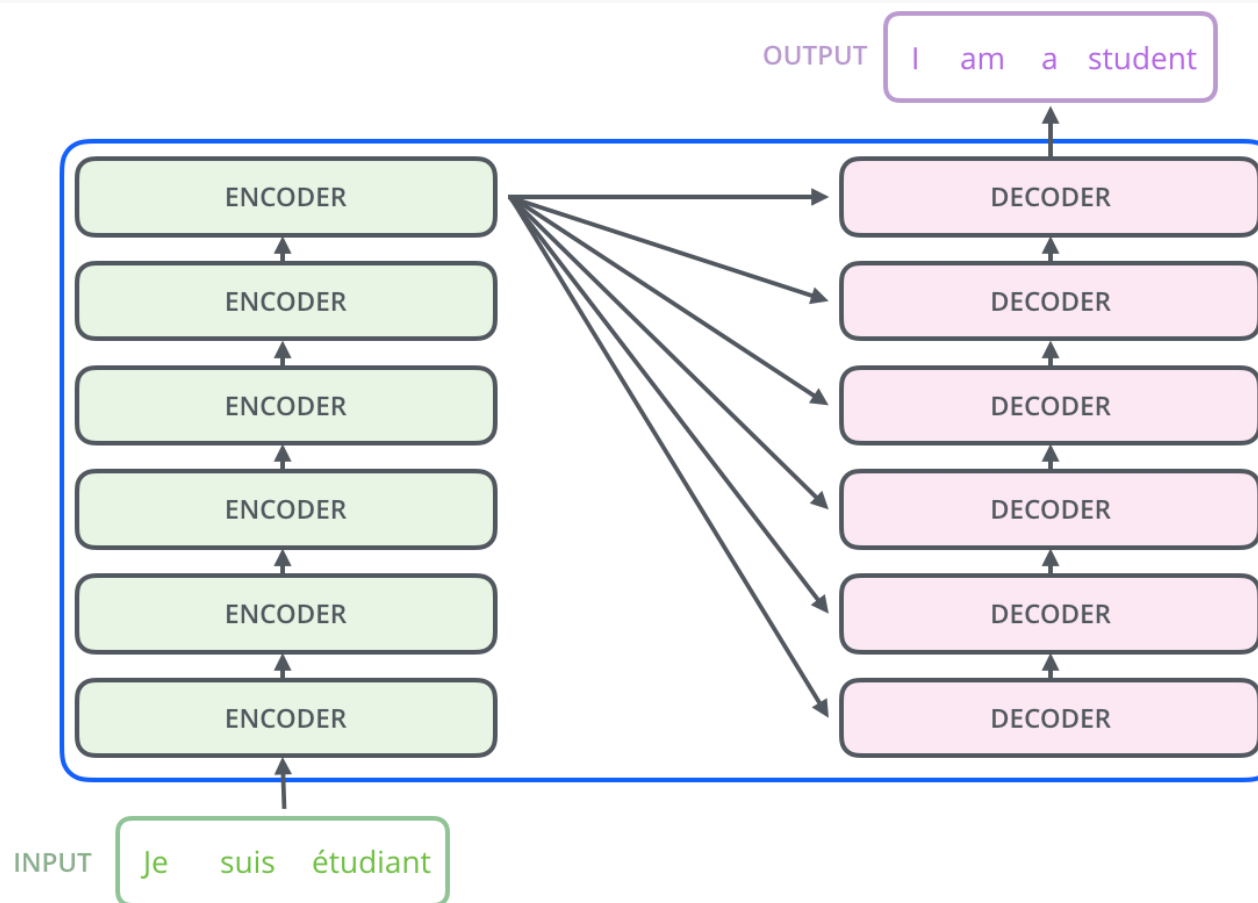
Теперь объединим это
в encoder-decoder-
подобную архитектуру
и добавим несколько
трюков



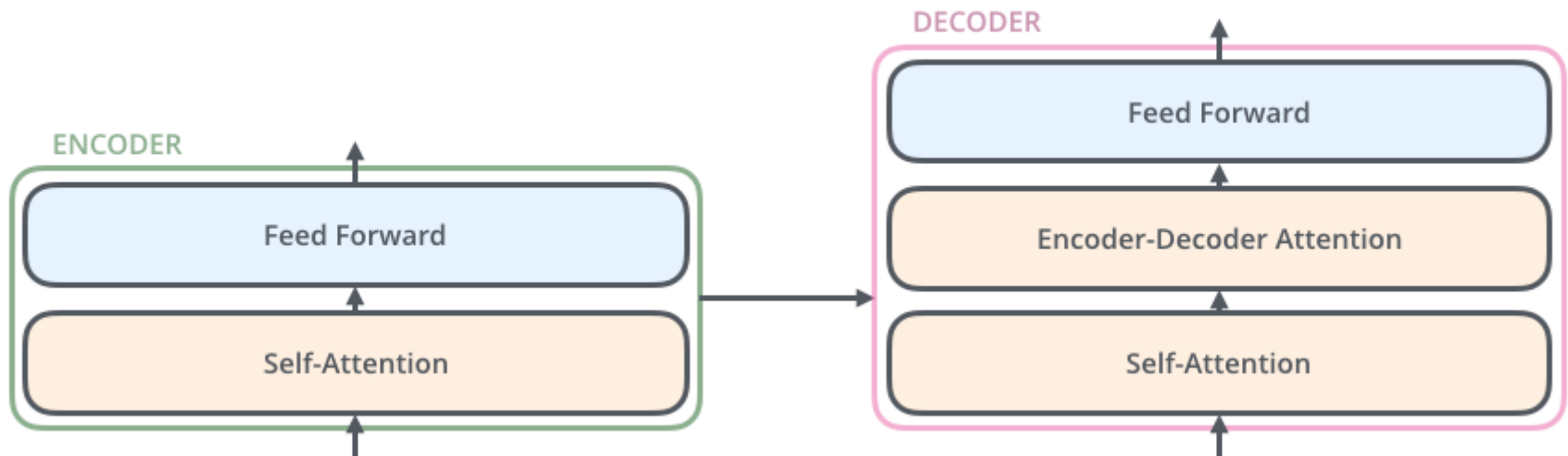
Encoder-decoder архитектура



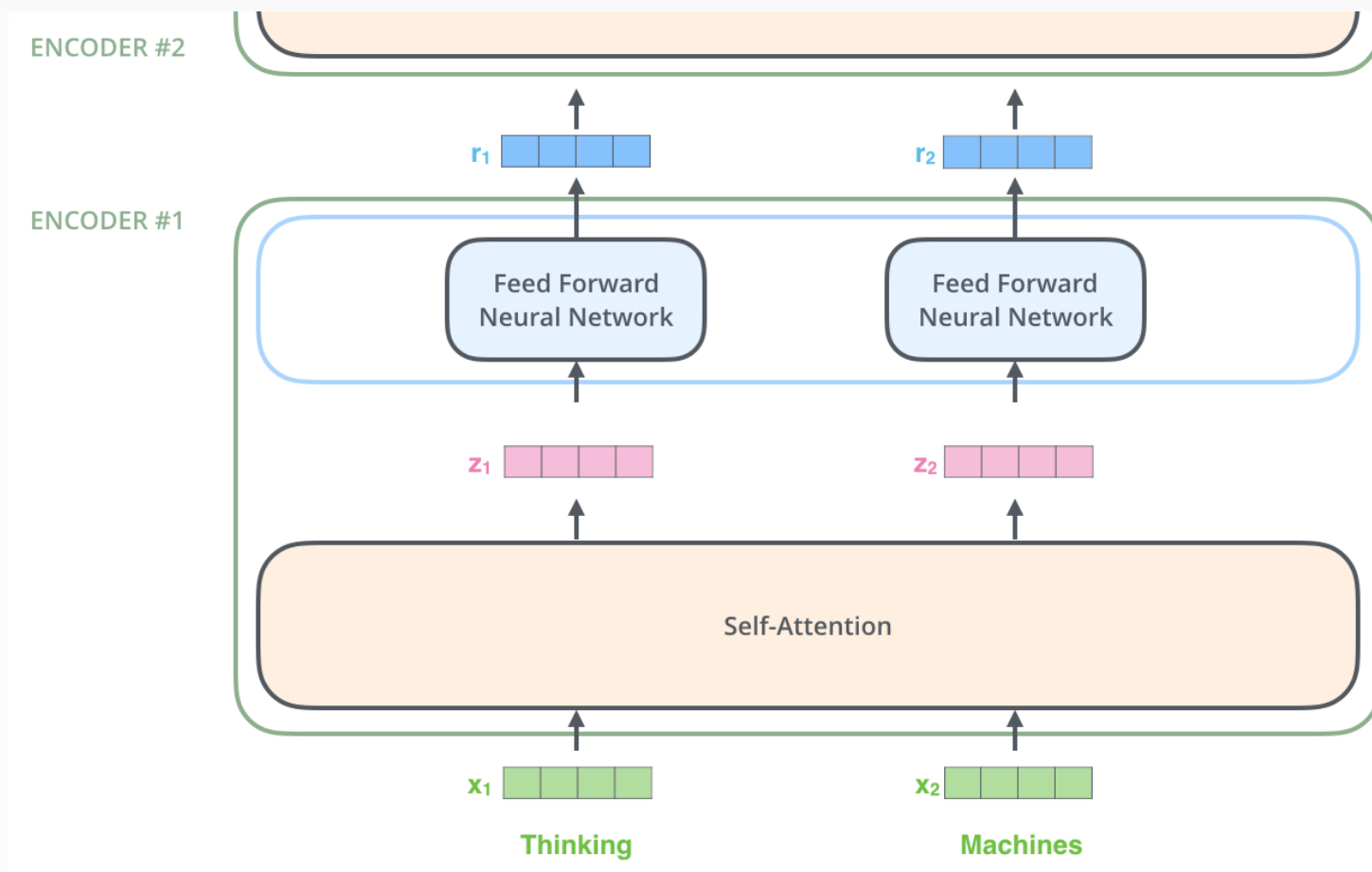
Шесть слоев



Устройство кодировщика и декодировщика



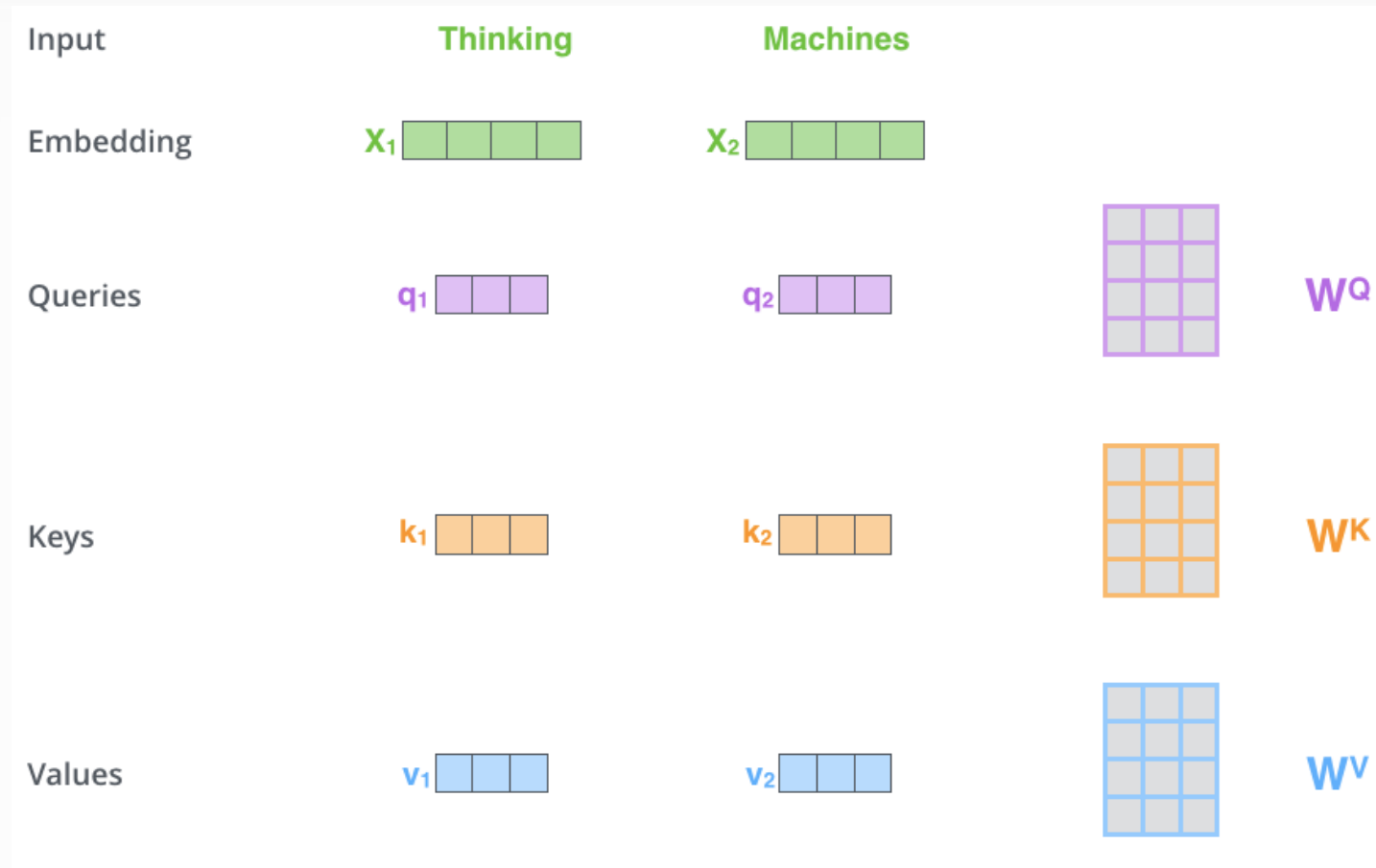
Трансформация векторных представлений



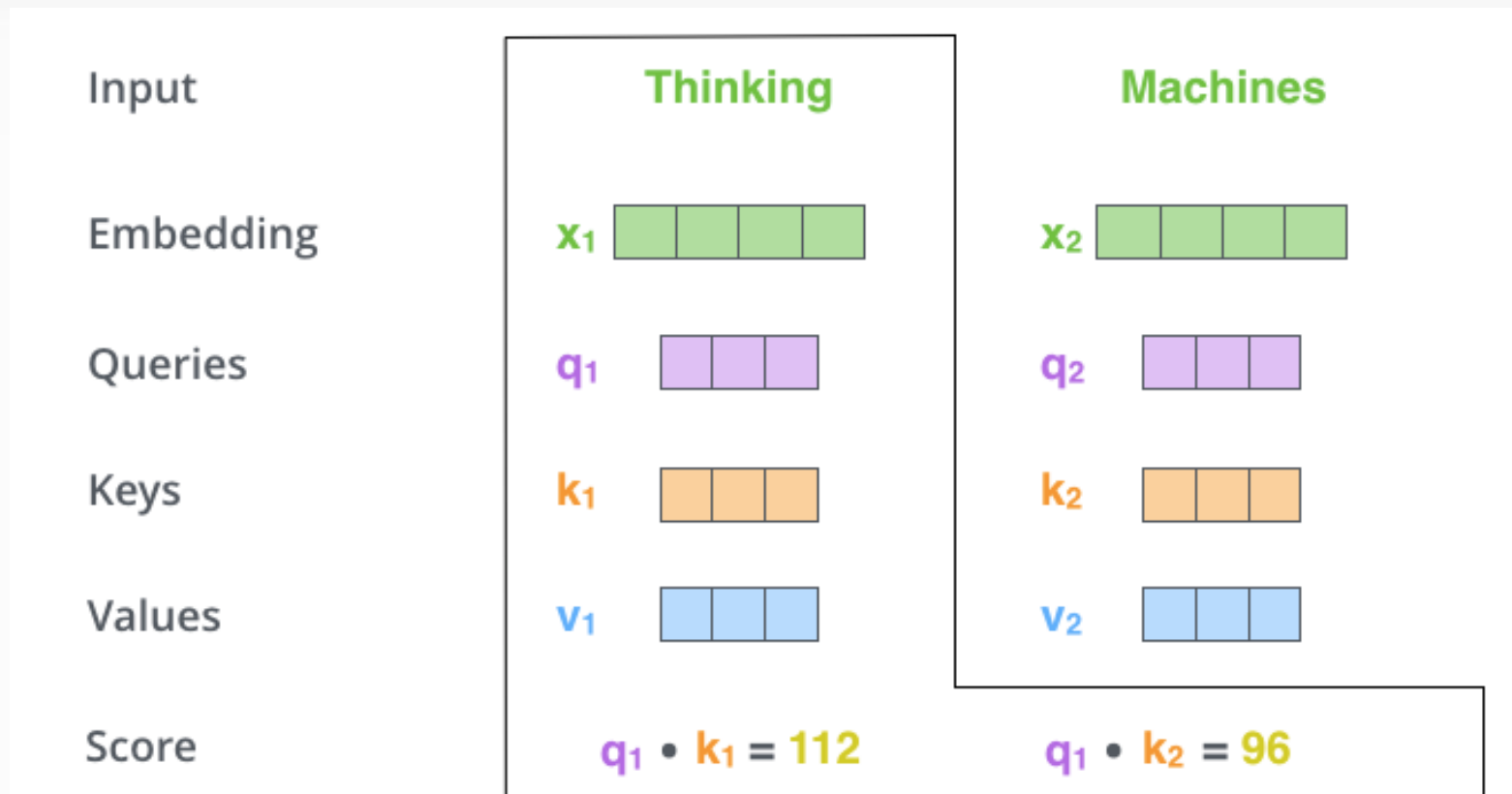
Алгоритм кодировщика

1. Вычисление значений векторов Q , K и V
2. Вычисление значения score от Q и K
3. Вычисление результата Softmax от дисконтированных значений score
4. Вычисление итогового значения как средневзвешенной суммы результата Softmax и V

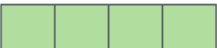
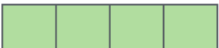

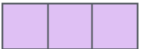

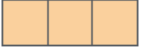

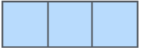
1. Векторы Q, K и V



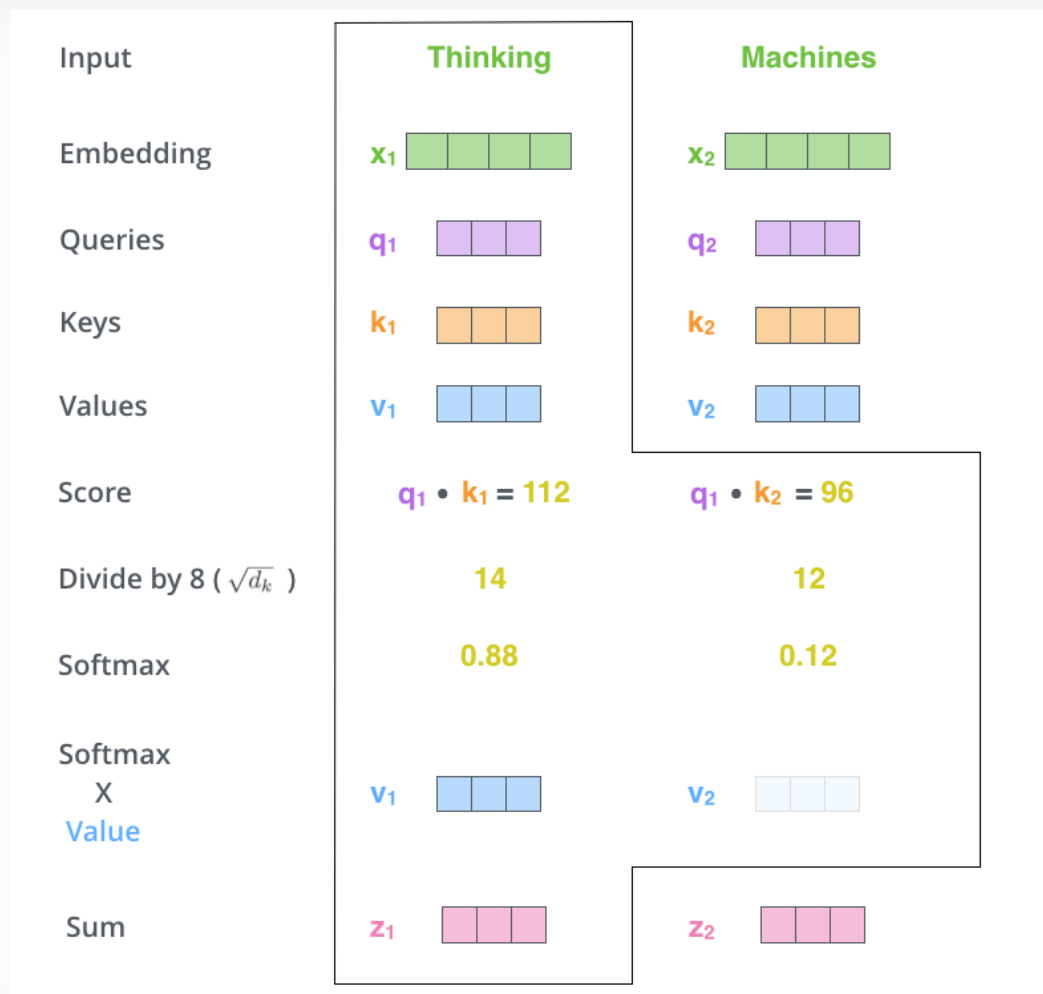
2. Score



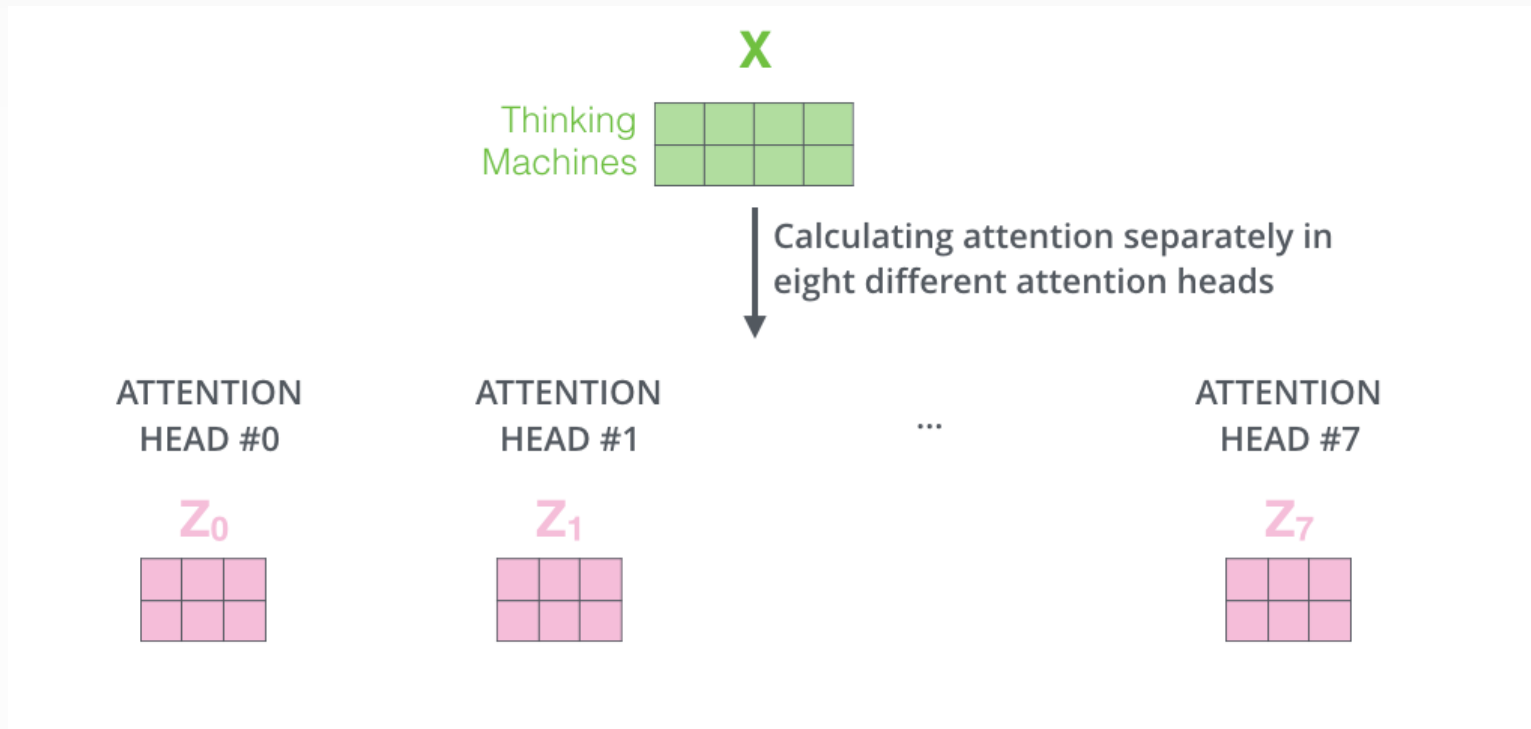
3. Softmax

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

4. Итоговое значение

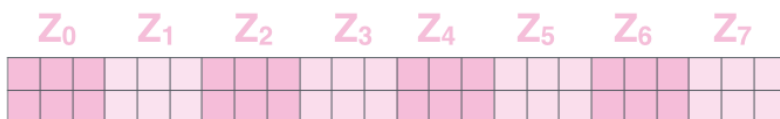


Множество головок (1/2)



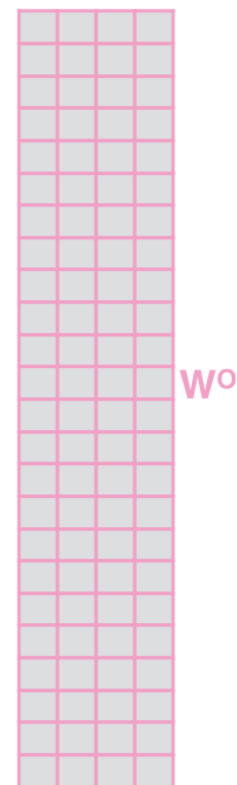
Множество головок (2/2)

1) Concatenate all the attention heads

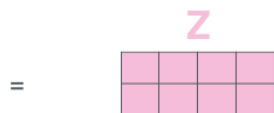


2) Multiply with a weight matrix W^O that was trained jointly with the model

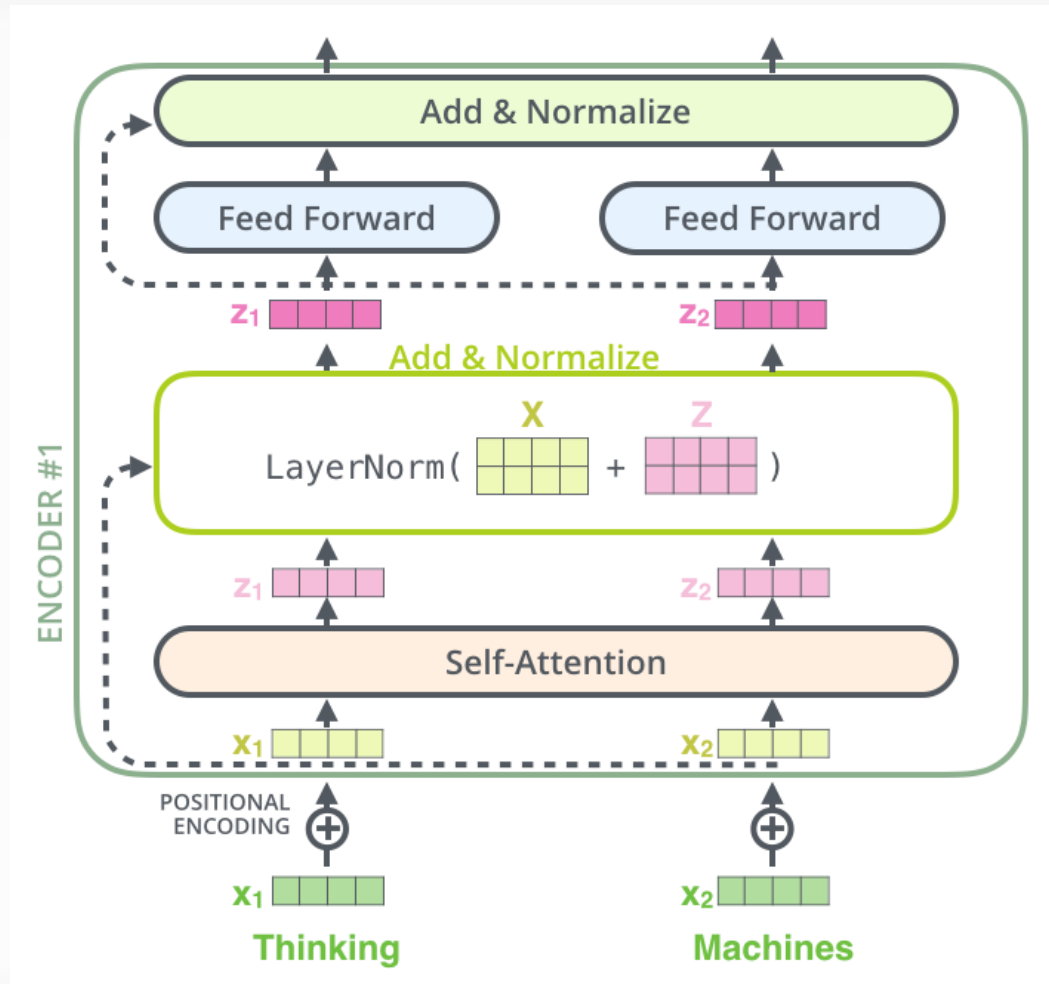
\times



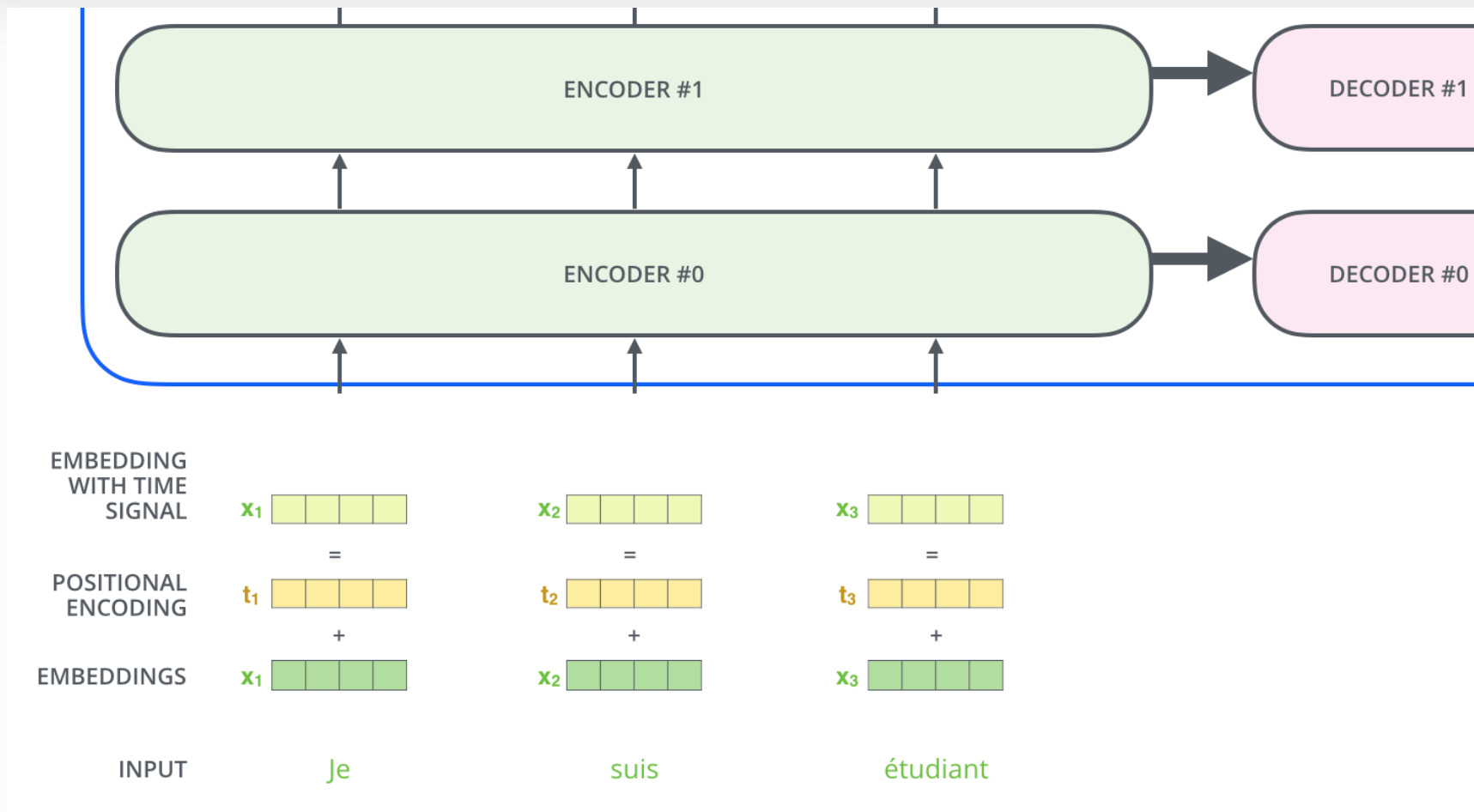
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN



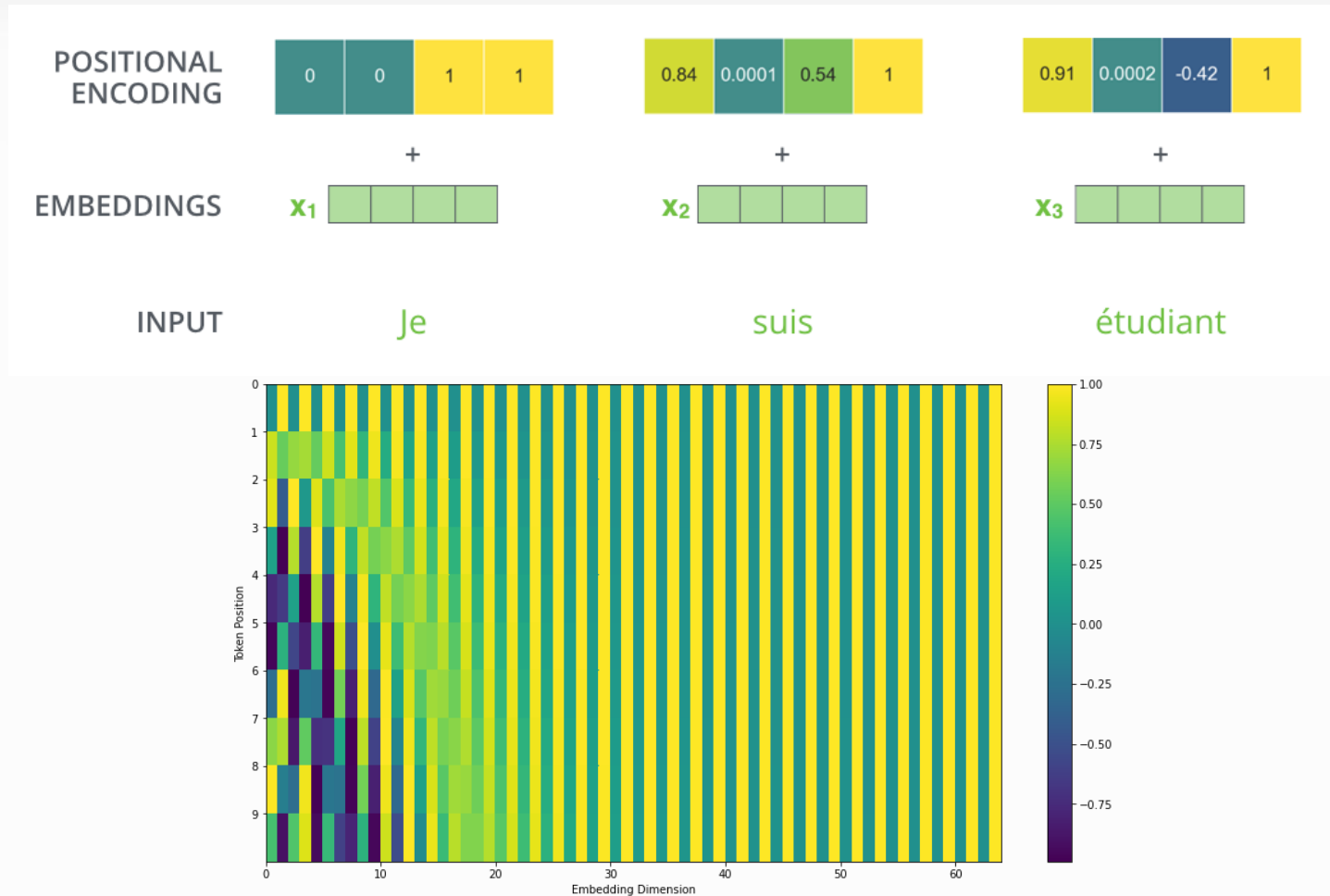
Послойная нормализация и остаточные соединения



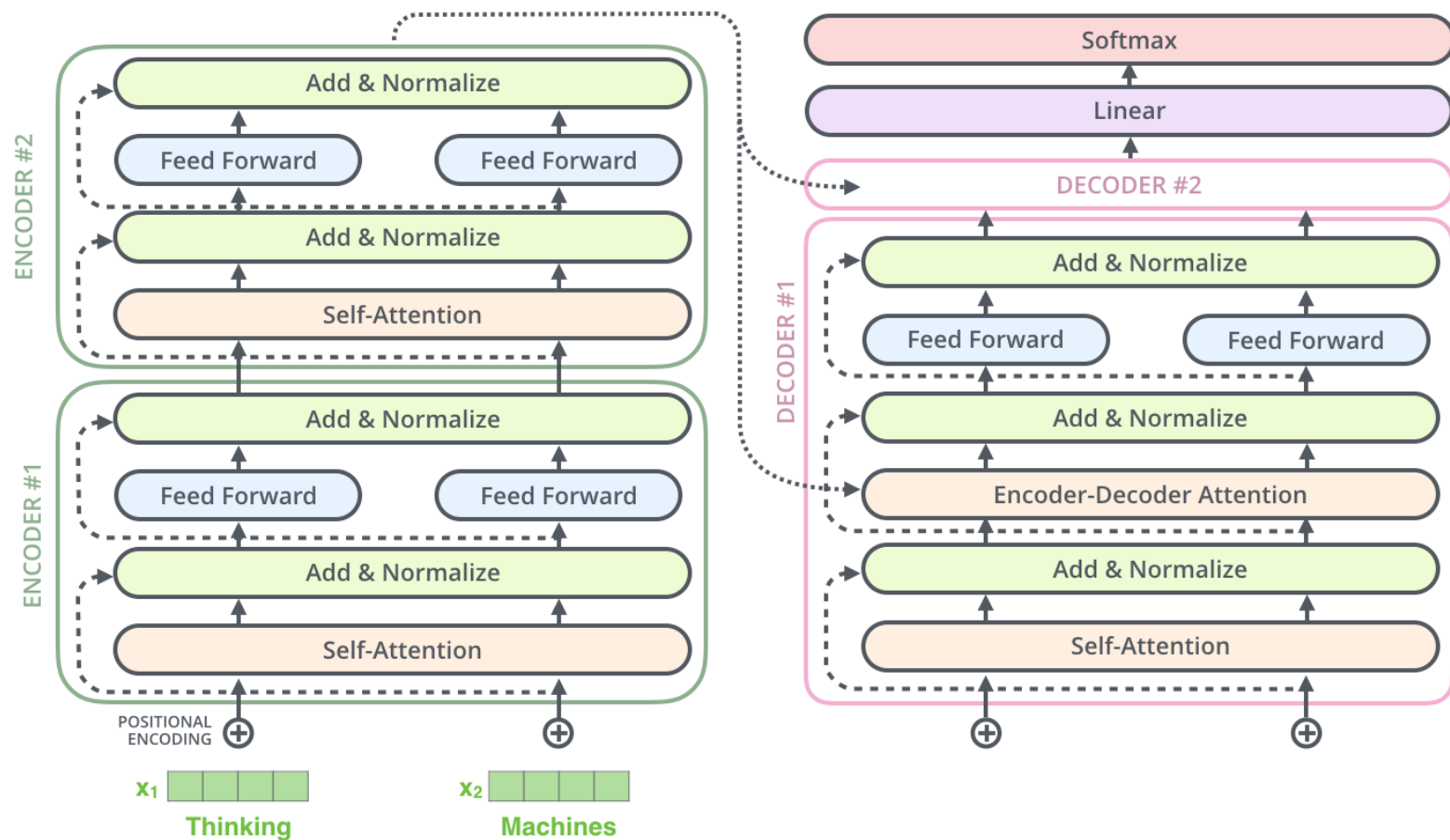
Позиционное кодирование (1/2)



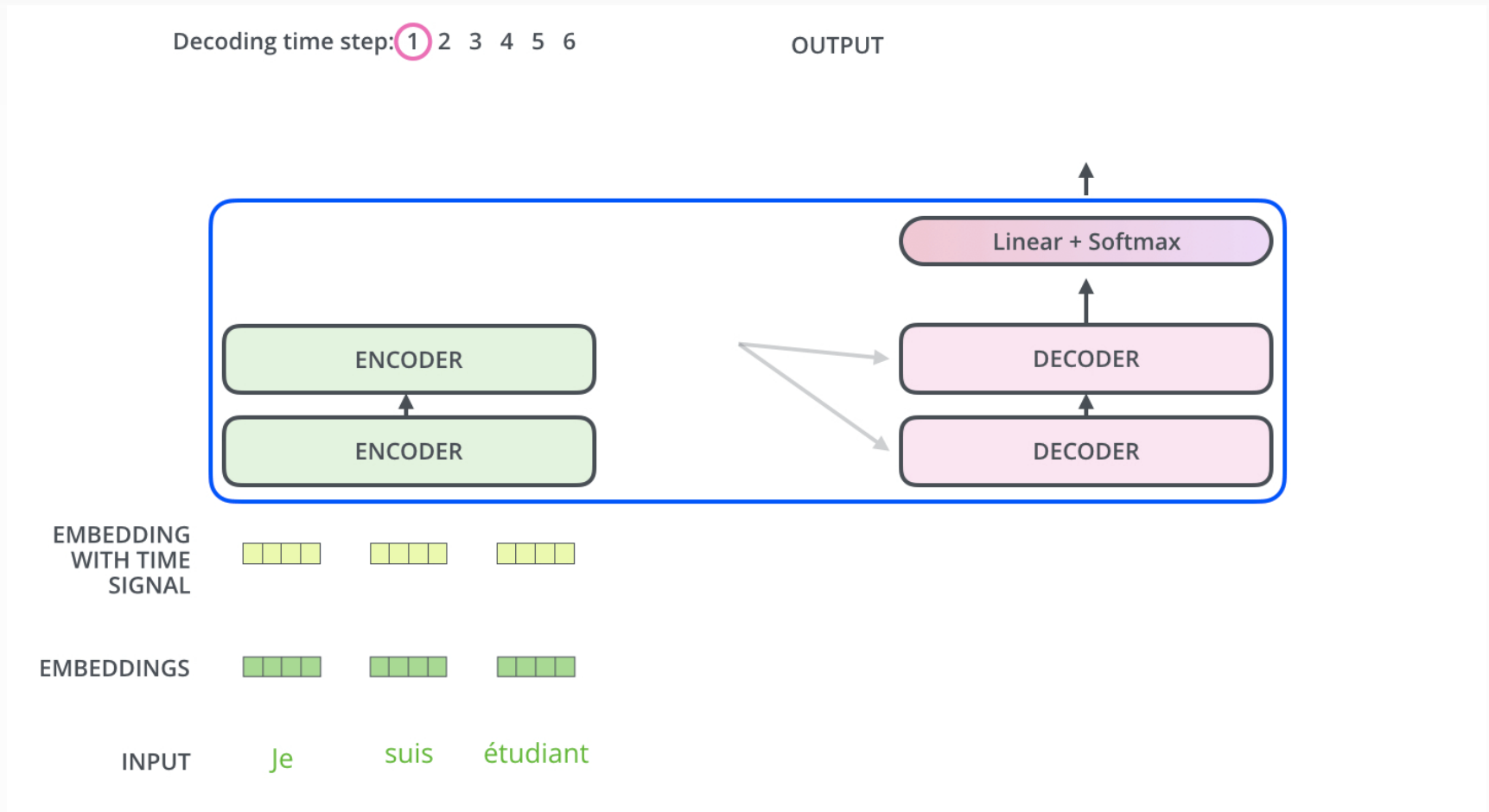
Позиционное кодирование (2/2)



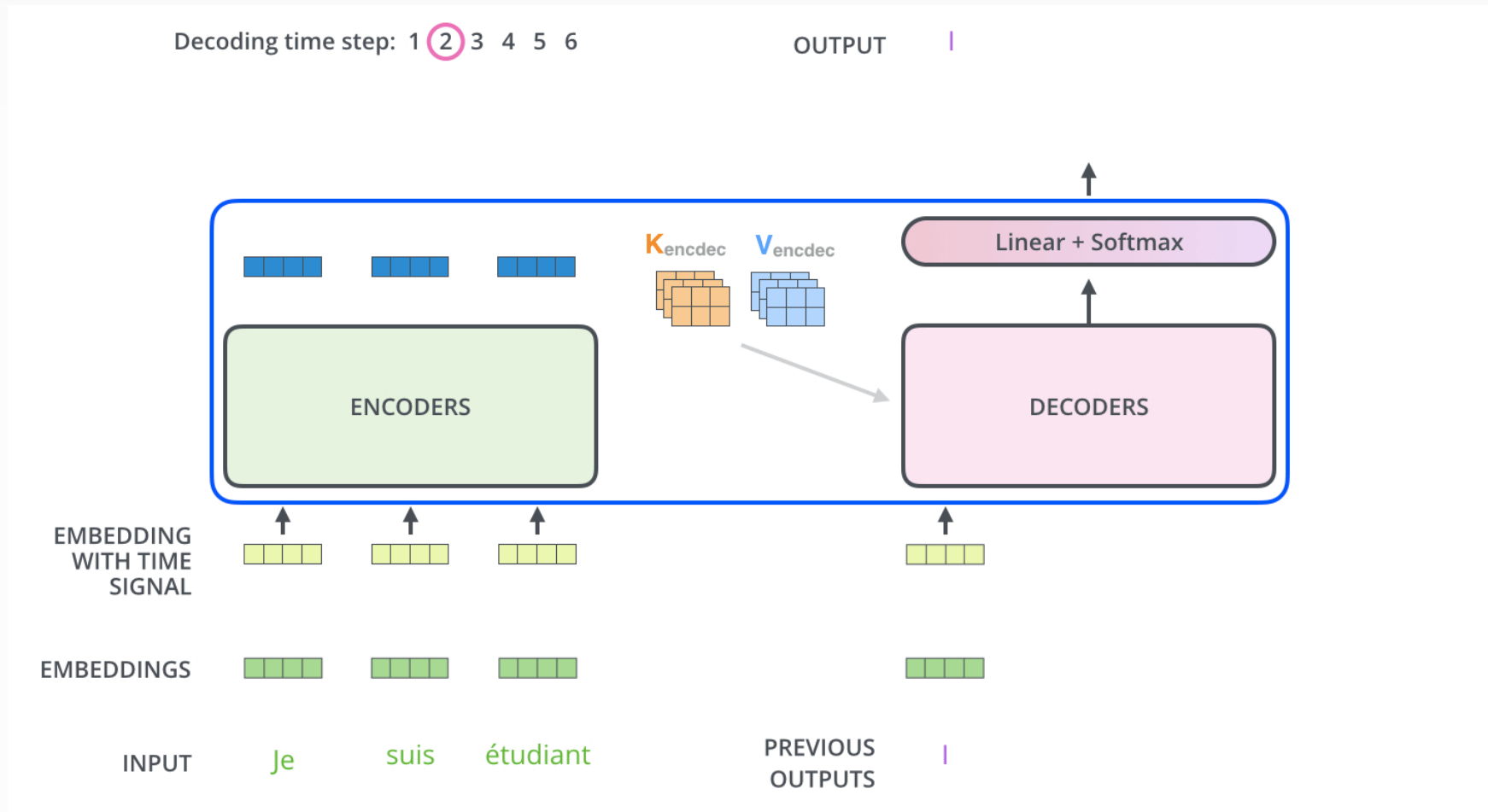
Декодер



Последовательное декодирование (1/2)

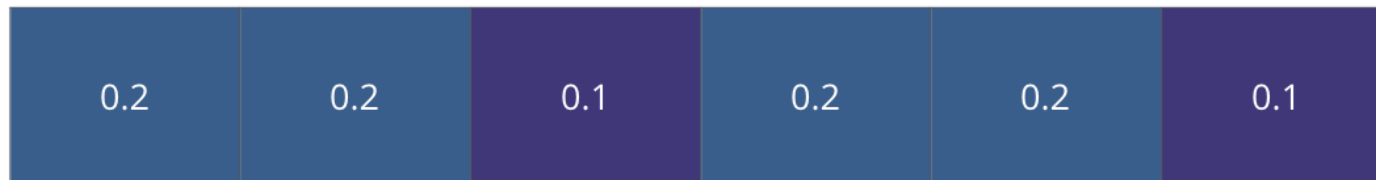


Последовательное декодирование (2/2)



Обучение модели по KL

Untrained Model Output



Correct and desired output



a am I thanks student <eos>

Особенности трансформеров

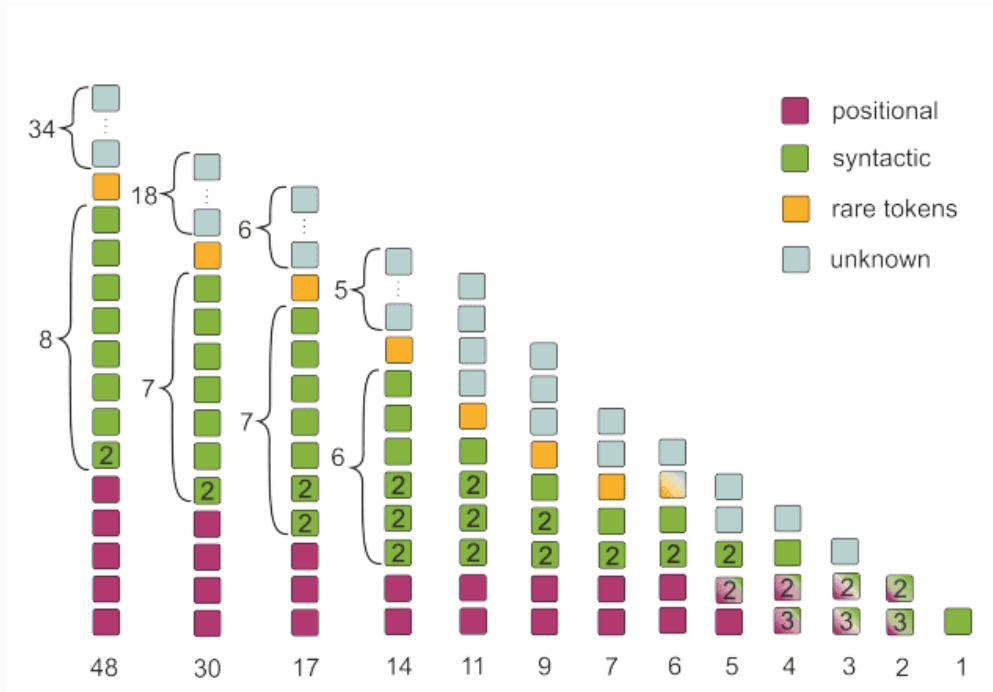
Некоторые наблюдения:

- Головки получают свою спецификацию
- Самые важные гиперпараметры это гиперпараметры нормализации (0,1% от всех)
- Без остаточных соединений матрицы внимания вырождаются в матрицы ранга 1

Специализация головок

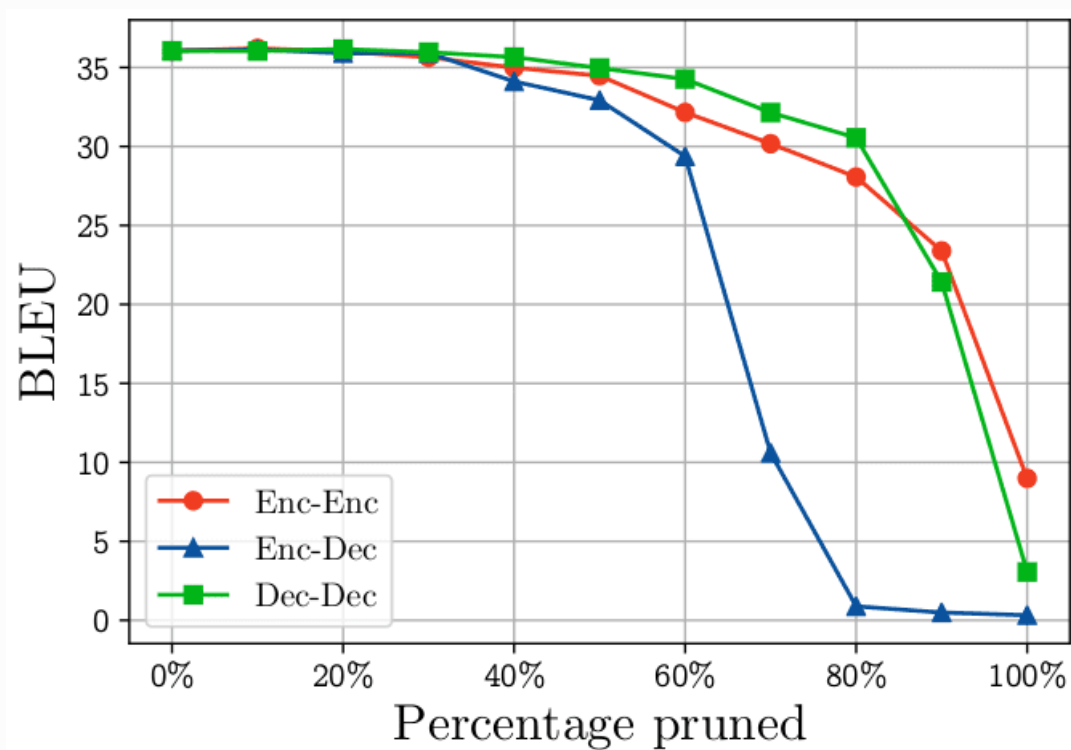
Оказывается, головки в итоге специализируются на:

- позиции слов
- синтаксисе
- редких словах
- чем-то еще



Удаление головок

Головки можно удалять, но до определенного предела



План лекции

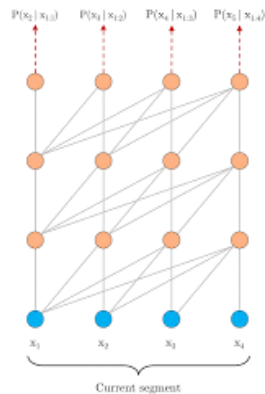
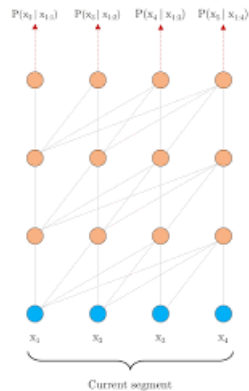
- Внимание (напоминание)
- Трансформер
- Другие трансформеры
- Семейства BERT и GPT

Ограниченность контекста

Трансформер использует фиксированный размер контекста, то есть обрабатывает текст посегментно, но обращая внимания на границы предложений.

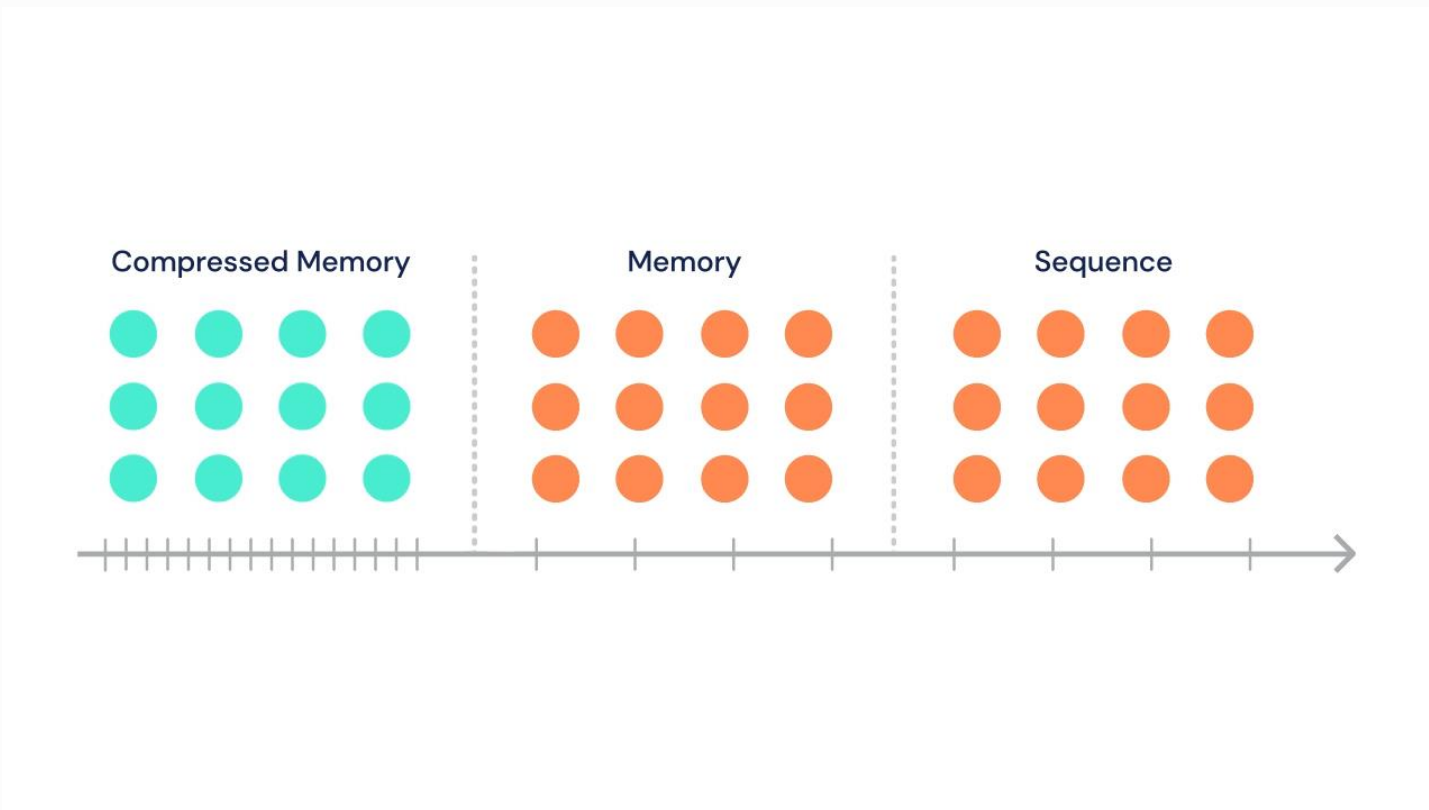
Идея: будем сохранять состояния для предыдущих сегментов и использовать их для обработки следующих сегментов.

Transformer XL



Compressive Transformers

Идея: будем хранить в памяти сжатые активации вообще всех предыдущих сегментов.



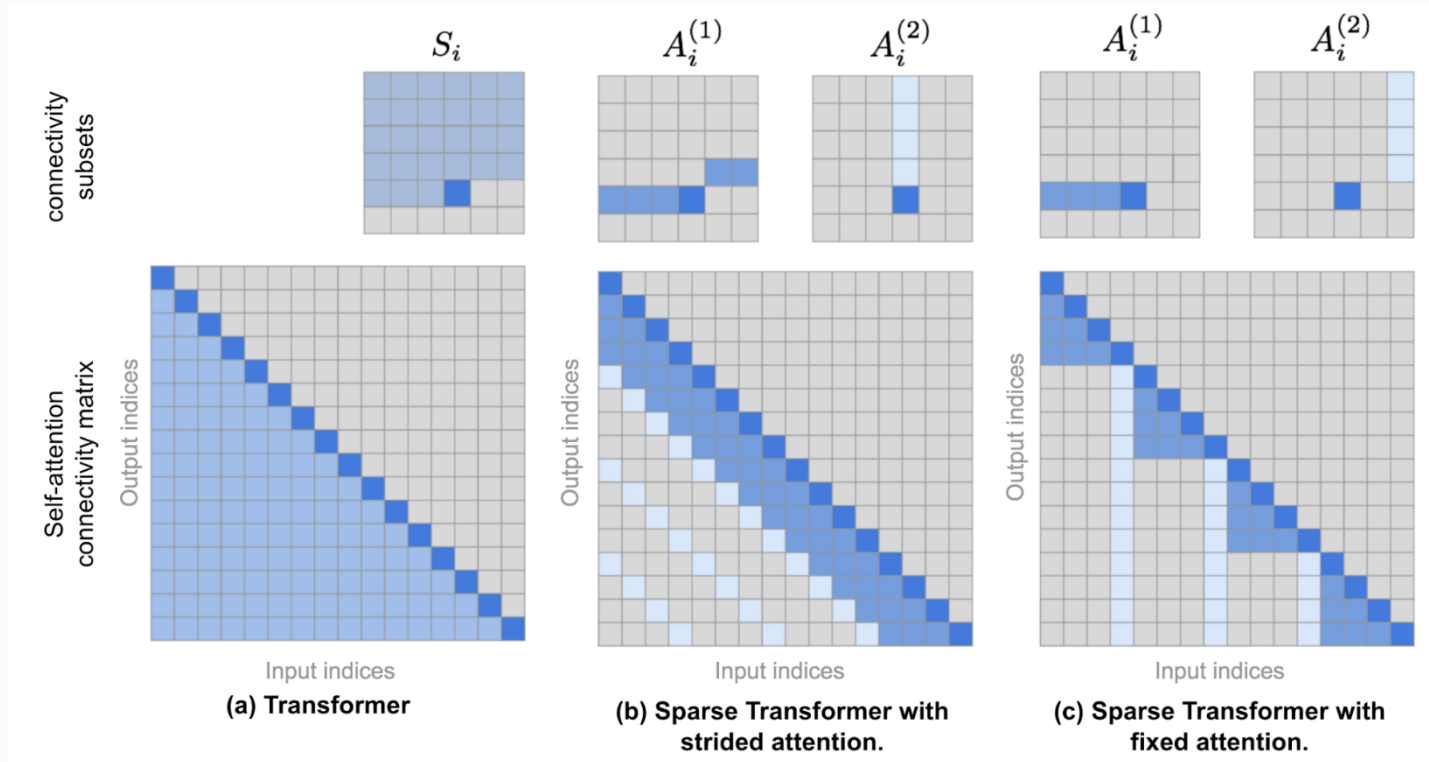
Поэлементное перемножение

Трансформер использует на каждом шаге попарное перемножение всех входов, что дает $O(N^2)$. Все такие матрицы для каждого слоя и каждой головки нужно хранить в памяти.

Идея: использовать разреженные матрицы внимания.

Sparse Transformer

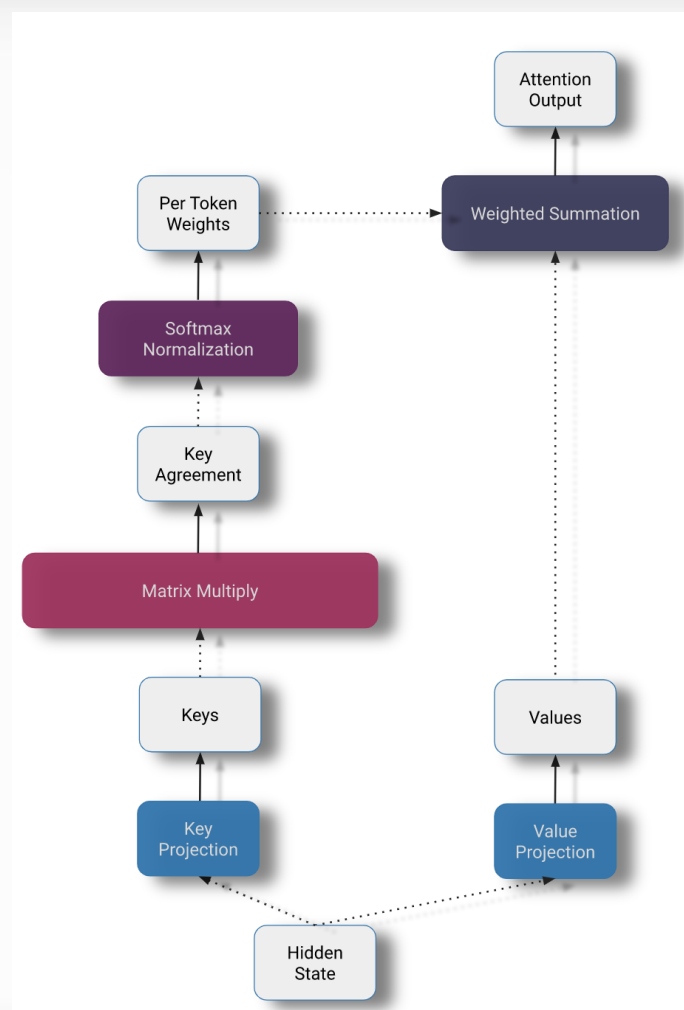
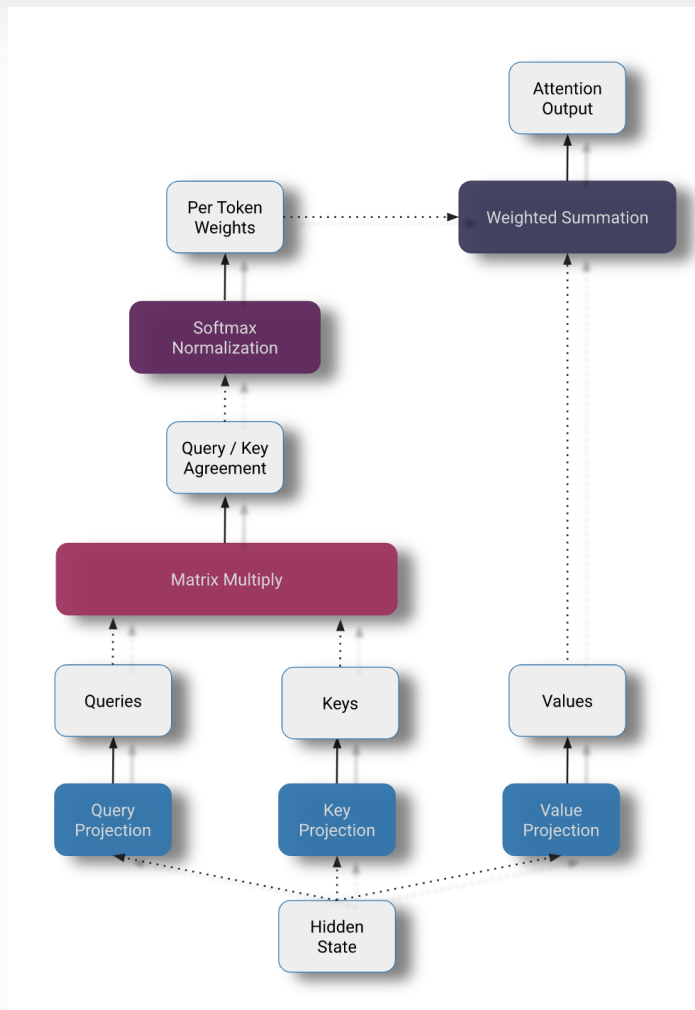
Последовательное (stride) и фиксированное (fixed) разложение внимания



Reformer

Основная идея: уменьшим требования к памяти и времени за счет снижения размерности векторов

Отбрасывание Q



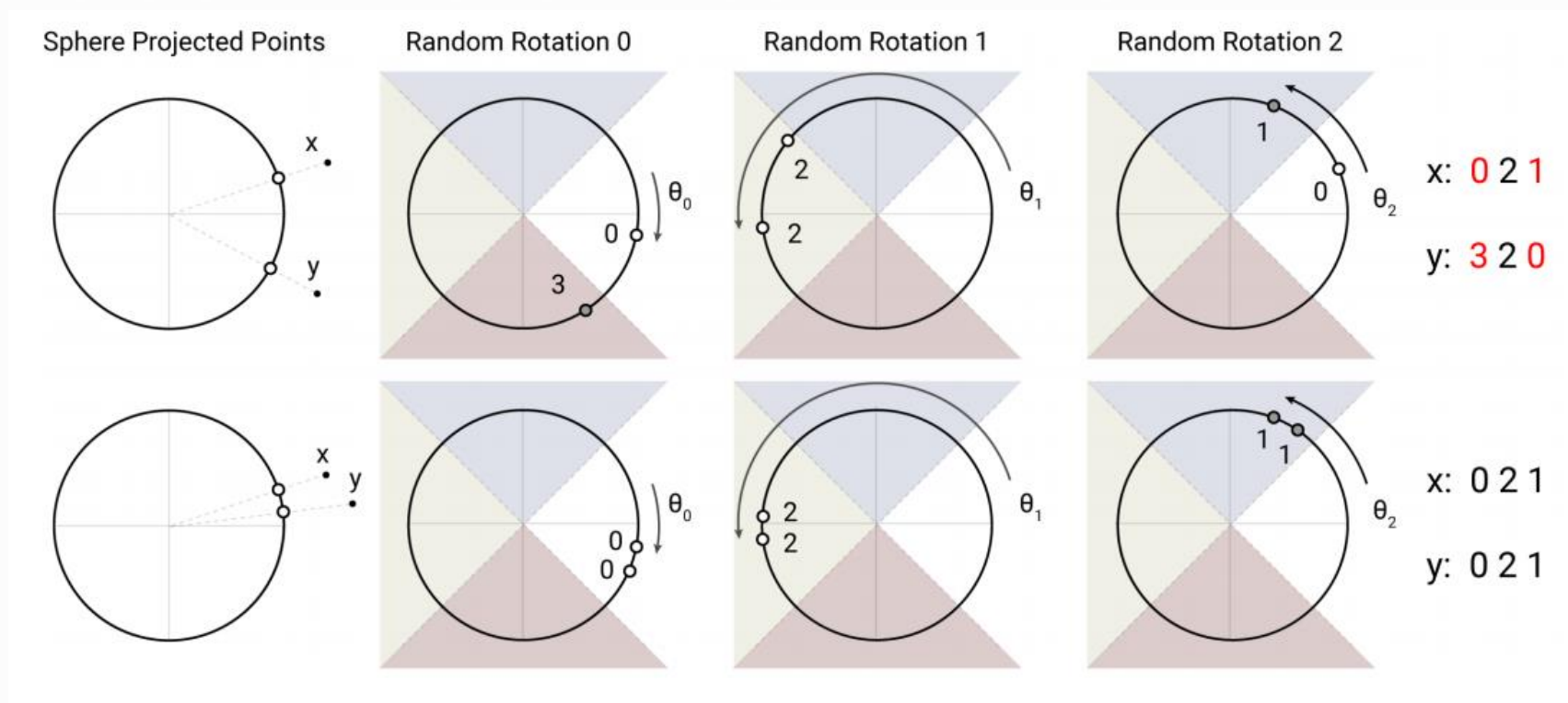
Добавление LSH

Хэширование с учетом положения (locally sensitive hashing, LSH) — метод хэширования высокоразмерных данных

Идея: будем использовать его, чтобы уменьшить размерность векторов

LSH случайными поворотами

Нормируем и будем случайно поворачивать



Кластеризация векторов

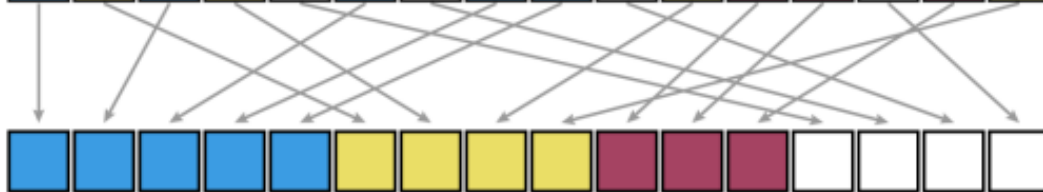
Sequence
of queries=keys



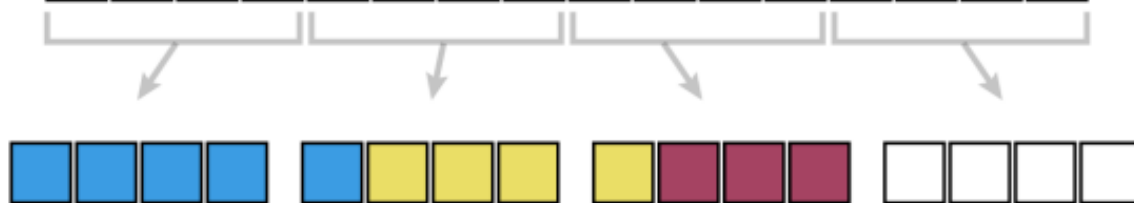
LSH bucketing



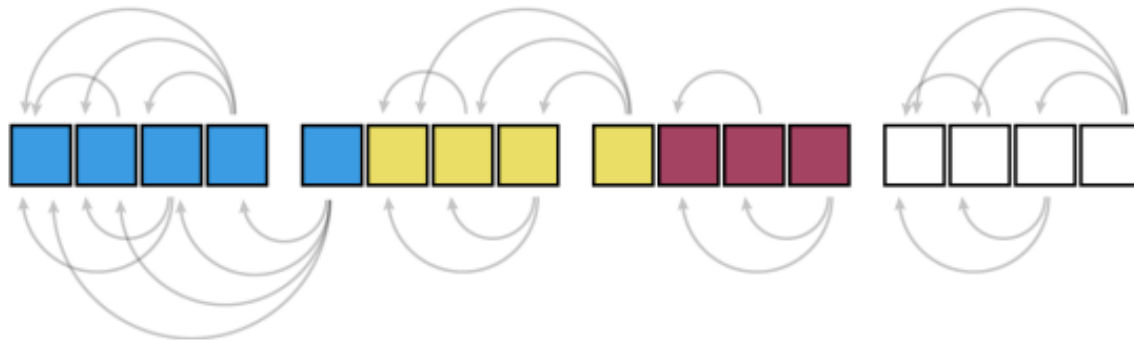
Sort by LSH bucket



Chunk sorted
sequence to
parallelize



Attend within
same bucket in
own chunk and
previous chunk



LSH-блок

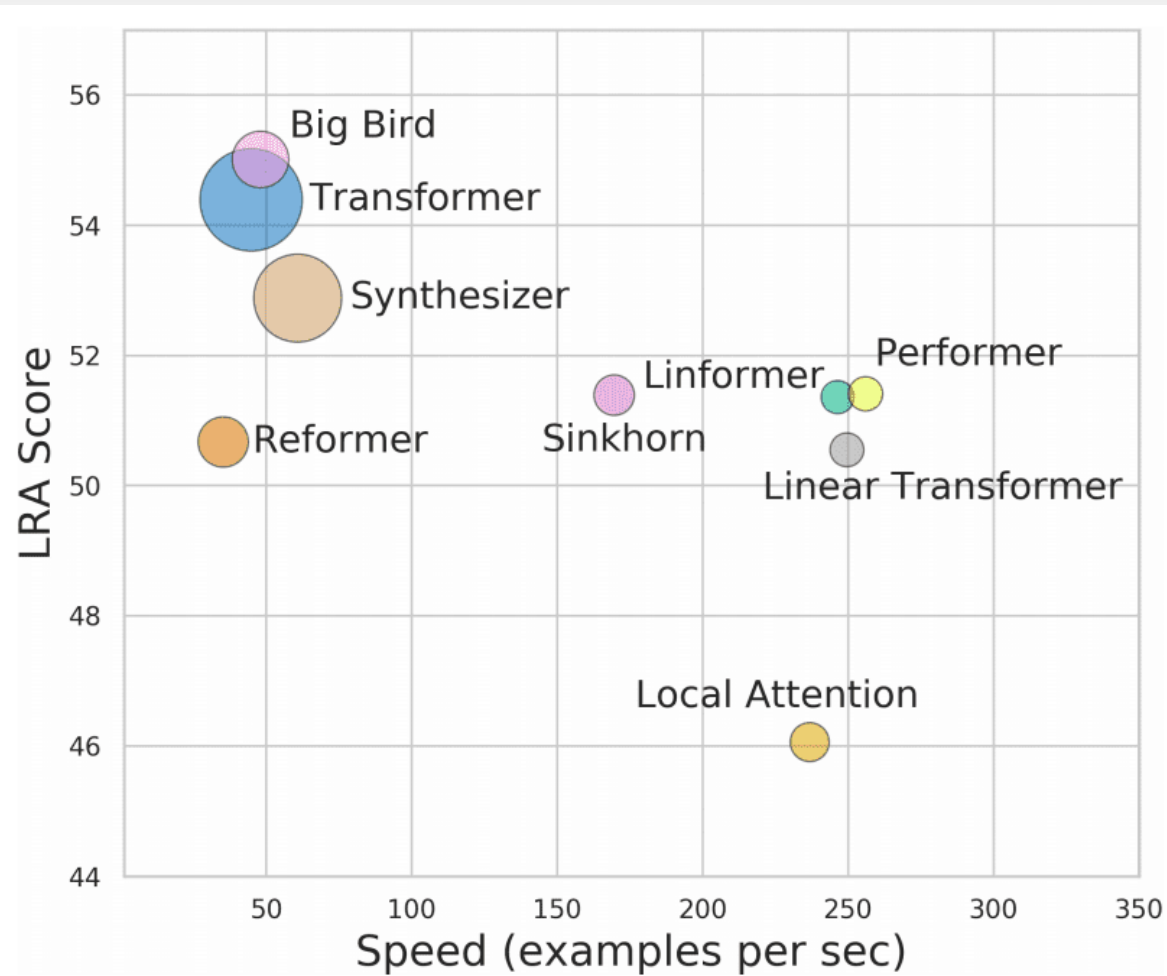
На практике, LSH-блок идет как стандартный блок в нейронной сети

Еще идеи

Как варианты:

- Перейти к задаче маршрутизации (Routing Transformer)
- Перейти к задаче оптимальной транспортировки (Sinkhorn Transformer)
- Приближать матрицу матрицей с маленьким рангом (Linformer)

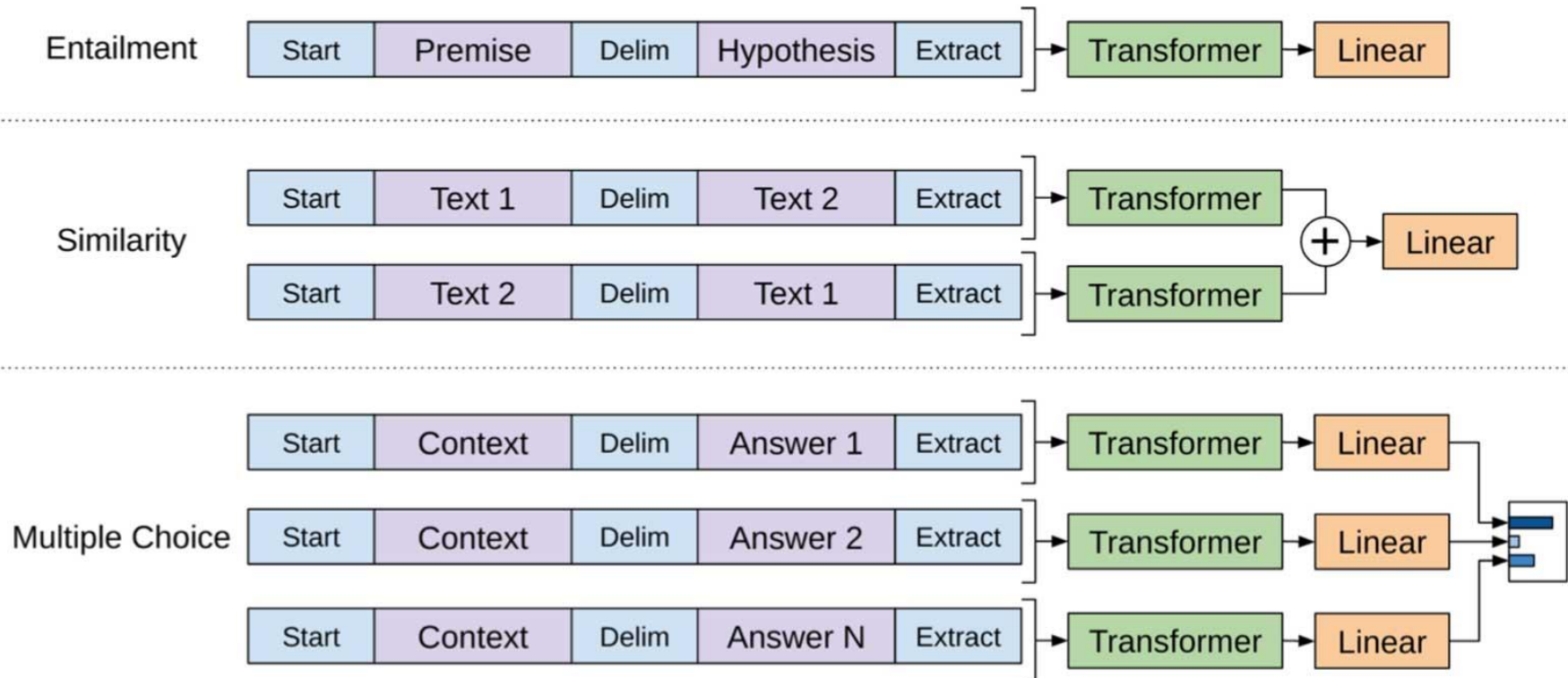
Сравнение трансформеров



План лекции

- Внимание (напоминание)
- Трансформер
- Другие трансформеры
- Семейства BERT и GPT

OpenAI Transformer

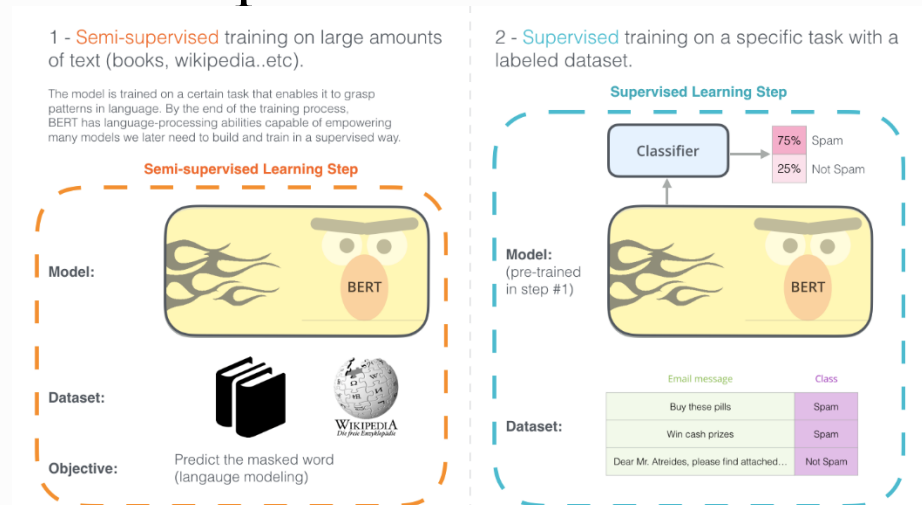


BERT

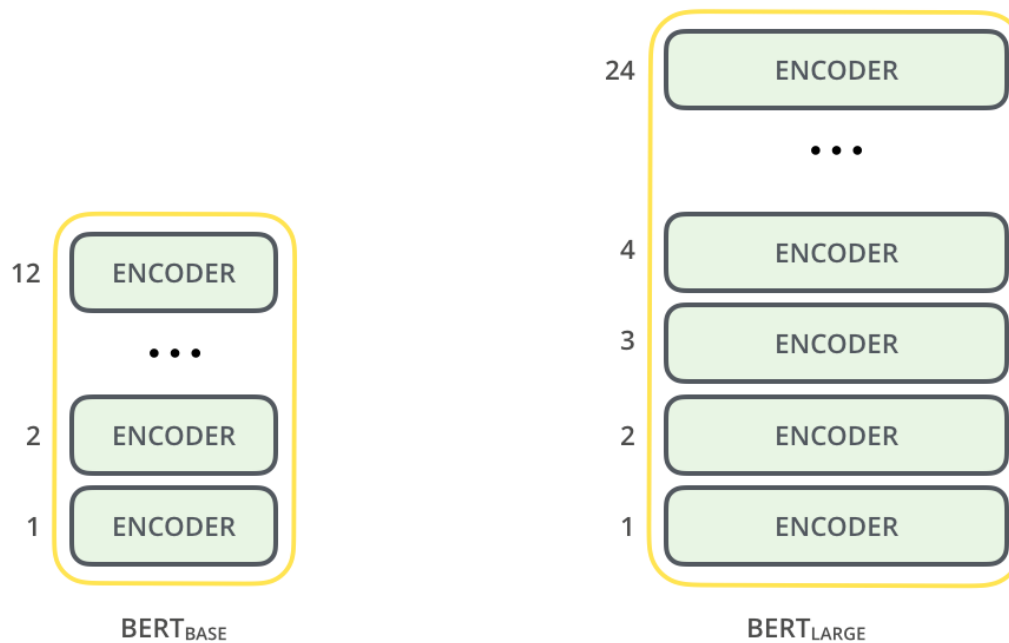
Bidirectional encoders representation of transformers

Решаем задачу двунаправленного моделирования языка на двух (трех) зачах:

- предсказание слов по контексту (15%)
- предсказание продолжения предложения
- (бонус) предсказать правильный порядок слегка перепутанных слов



Архитектура BERTa



Контекстуализированные векторные представления

Значение слова часто определяется контекстом:

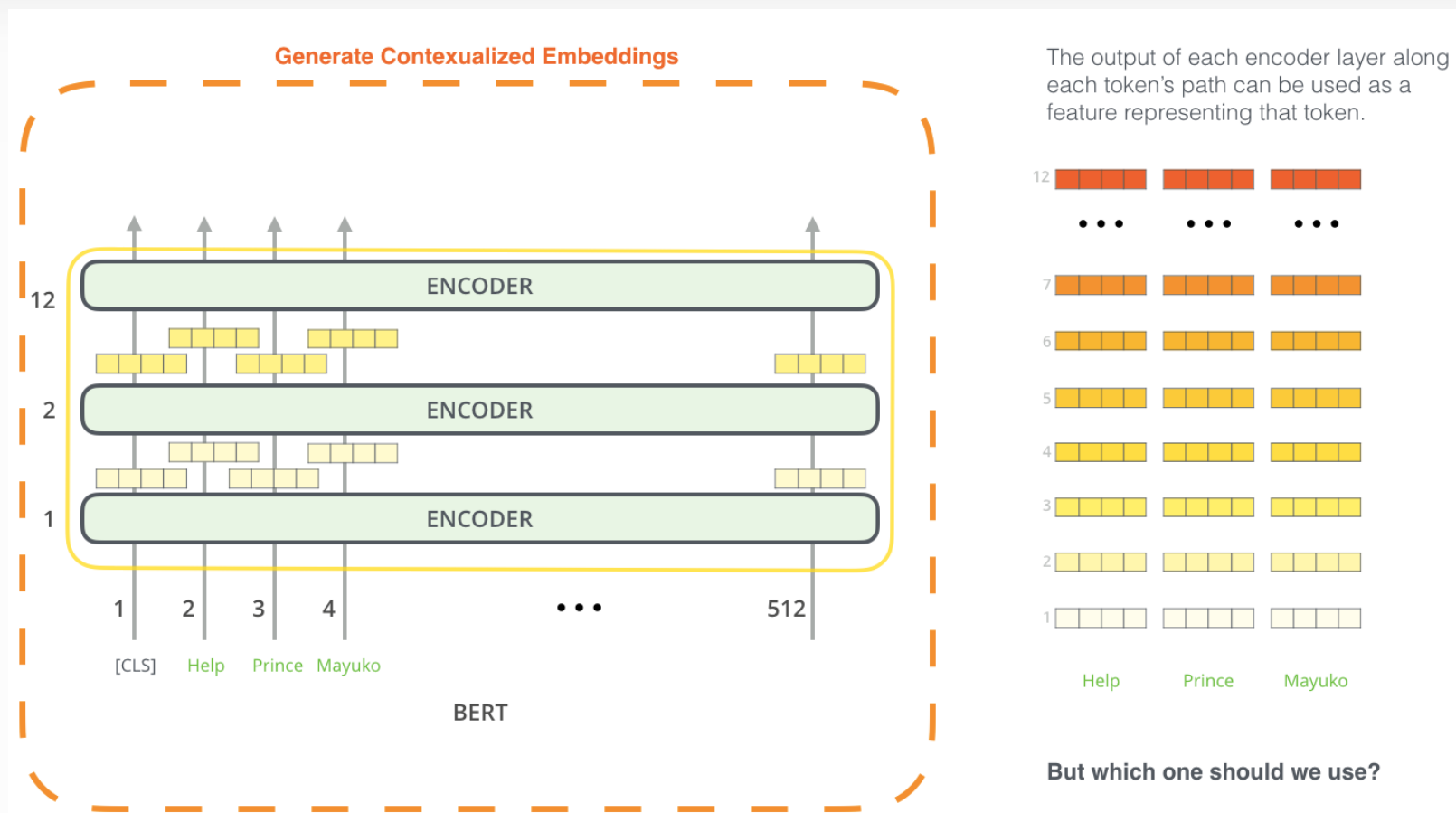
коса (крестьянская)

коса (девичья)

Коса (Балтийская)

Идея: будем строить векторное представление слова по предложению, в котором оно стоит

BERT для контекстуализации (1/2)



BERT для контекстуализации (2/2)

What is the best contextualized embedding for “Help” in that context?
For named-entity recognition task CoNLL-2003 NER

12



...

7



6



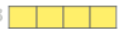
5



4



3



2



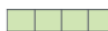
1



Help

First Layer

Embedding



Dev F1 Score

91.0

Last Hidden Layer

12



94.9

Sum All 12 Layers

12

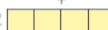


+

...

+

2



+

1



=



95.5

Second-to-Last Hidden Layer

11



95.6

Sum Last Four Hidden

12



+

11



+

10



+

9



=



95.9

Concat Last Four Hidden

9

10

11

12



96.1

XLNet

Соединяем BERT и Transformer XL

RoBERTa

Robustly Optimized BERT Approach

1. Использовали в 10 раз больше данных для обучения
2. Кратно увеличили число шагов обучения
3. Улучшили стратегию маскирования
4. Убрали задачу предсказания продолжения предложения

ALBERT

A Lite BERT

1. Общие параметры у слоев
2. Размерность скрытого слоя не совпадает с размерностью слов, а матрица слово \times слово раскладывается на две маленькие матрицы

GPT

Generative Pre-Trained Transformer
использует декодер трансформера,
способный порождать произвольные
последовательности.

Семейство GPT

GPT-1: 12-слойный декодер с 12 головками. Размерность эмбединга 1600.

GPT-2: вход содержит задачу, 48-слойный декодер с 24–48 головками с некоторыми изменениями архитектуры. Размерность эмбединга 1600.

GPT-3: 96-слойный декодер с 96 головками. Размерность эмбединга 12888.

Сравнение моделей

