# Способы кодирования слов

one hot

word2vec

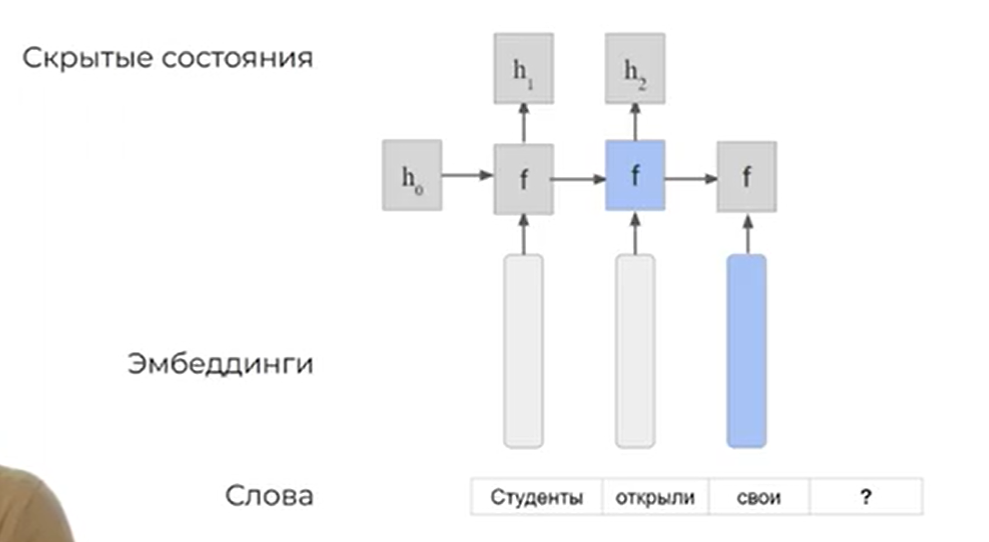
tf-idf

# RNN

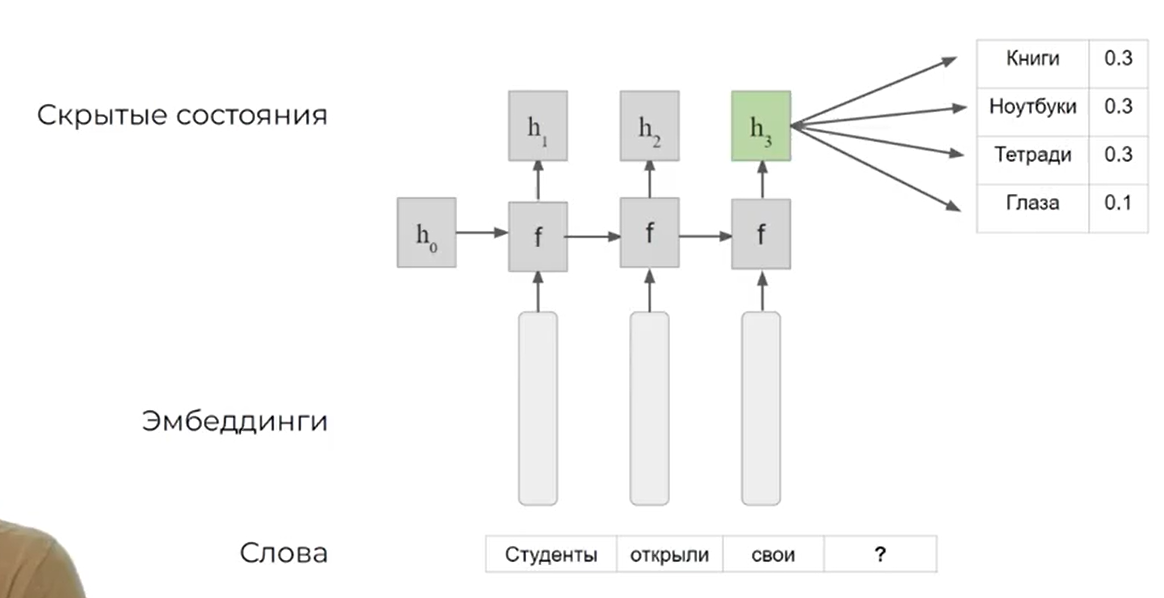
## Обучение

1. разбиение текста на последовательности небольшой длины
2. Последовательно подавать слово и рассчитывать предсказание

2.1) подаём слово и хидден стейт в rnn

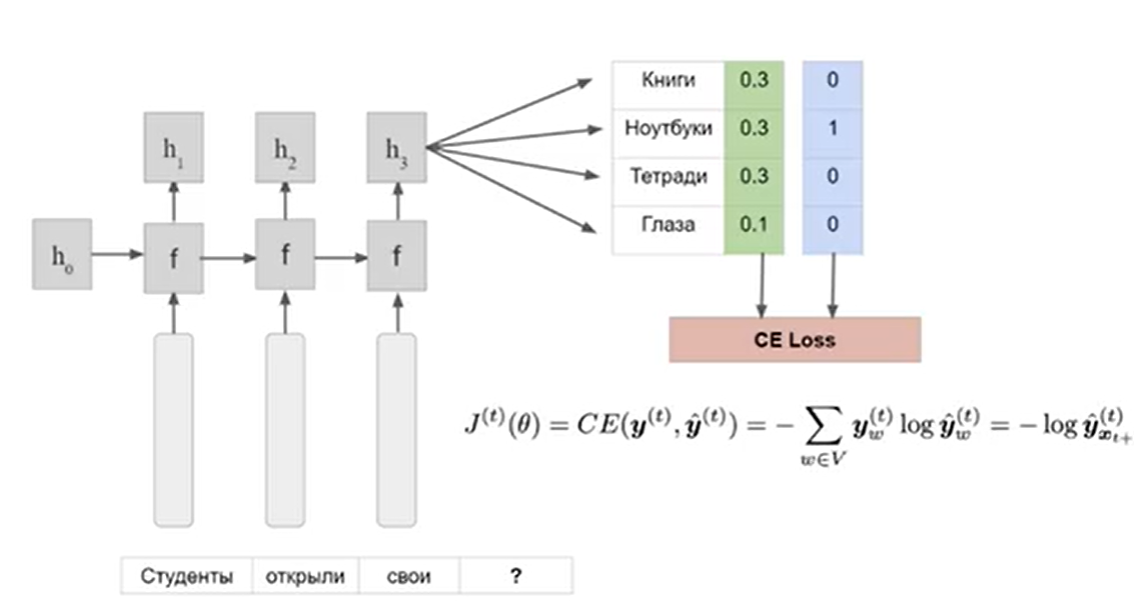


2.2) получаем предсказание следующего слова на выходе hi



2.3) прогоняем эмбеддинг hi через fcnn и софтмакс и получаем вероятности следующего слова

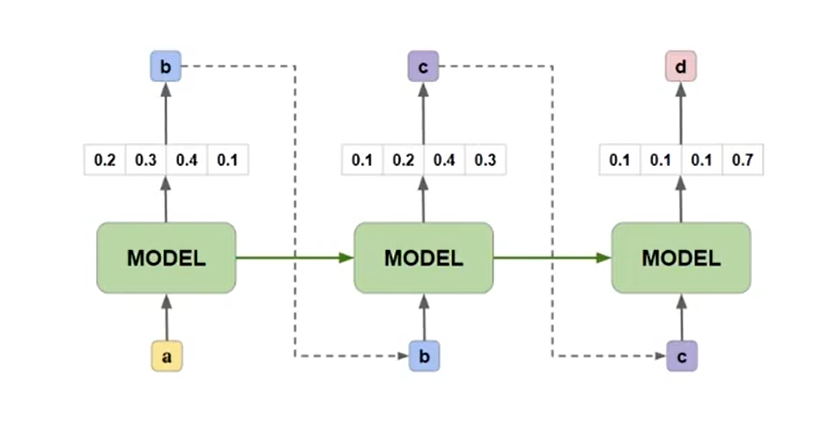
2.4) Вычисляем лосс



1. на полученном лоссе обучаем сеть

## Генерация текста

Авторегрессионная генерация



1. подается контекст и символ старта
2. получаем вектор вероятностей и выбираем наиболее хороший
3. подаём предсказанное слово снова в модель и получаем предсказание
4. итд пока не встретим символ конца.

# Трансформер

## Пайплайн трансформера

у нас есть исходный текст

энкодер:

Мы вычисляем ОДИН раз селф атеншн эмбединги исходного текста

1) переводим слова в эмбединги

2) делаем позиционное кодирование

3) получаем для каждого слова эмбединги из селф атеншн

4) делаем резидуал коннектион

5) нормализуем и получаем выход из энкодера, где каждый эмбединг слова содержит информацию о его "роли" в предложении с учетом порядка слов

6) сохраняем эти значения и готовимся подавать их в декодер

декодер:

декодер может предсказать только одно слово за раз

inference происходит как авторегрессия по методу teacher forcing

Поэтому обучение является рекуррентным и его нельзя распараллелить.

1) Передаём на вход энкодера токен начала предложения

2) Считаем позиционное кодирование

3) считаем маскт селф атеншн для поданного на вход декодера предложения

4) резкон + layer norm

5) передаем сохраненные эмбединги слов энкодера и считаем кросс атеншн для каждого слова декодера ОТНОСИТЕЛЬНО каждого эмбединга ЭНКОДЕРА

6) резкон + layer norm

7) полученные эмбединги передаём в fcnn, чтобы получить ЛОГИТЫ такой длины которой нам надо (которой у нас словарь другого языка)

8) передаём вектор логитов на выход

8) софтмакс

9) выбираем значение из словаря в соответствии с полученным вектором

10)ПОТОМ конкатенируем полученное значение с символом старта и СНОВА передаём на вход декодера

11) так пока не достигнем символа конца текста или нужного числа токенов

обучение

Обучение же можно легко распараллелить. То есть можно параллельно предсказывать следующее слово на любом месте предложения параллельно. Ведь у нас есть целиком предложение.

1) подаём исходное предложение целиком

2) настраиваем маску для маскт атеншн в зависимости от того какое слово мы хотим предсказать. то есть мы не должны считать селф атеншн относительно будущих слов, как будто их нет. и в итоге получаем эмбединги

3) резкон + layer norm

4) кросс атеншн относительно всех эмбедингов энкодера

5) резкон + layer norm

6) подаем на fcnn, чтобы получить логиты длиной длины величины словаря целевого языка или др

7) софт макс

8) выбираем слово на основе софтмакс

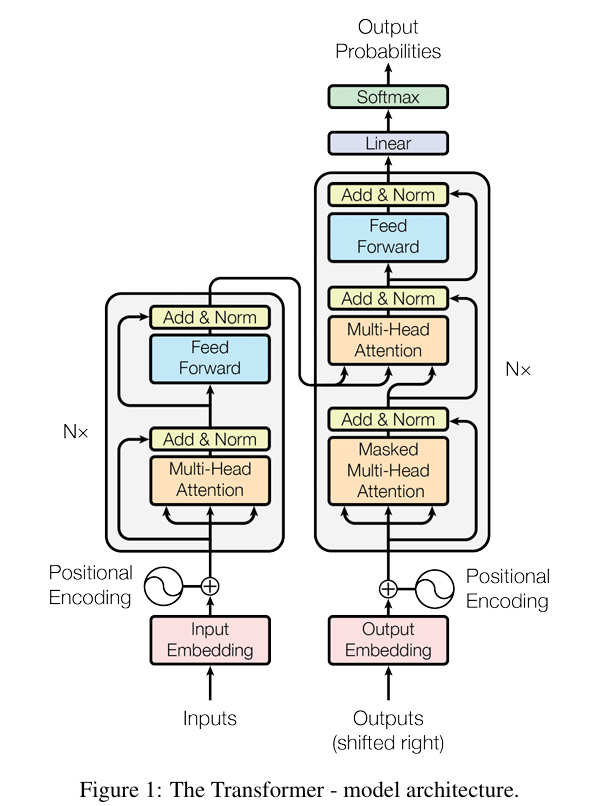
## Attention

[[1706.03762] Attention Is All You Need](https://arxiv.org/abs/1706.03762)

## Устройство трансформера

[Лекция. Архитектура Transformer. Введение, Transformer Encoder](https://www.youtube.com/watch?v=TBEwpgyoo20)

[Transformer Neural Networks, ChatGPT's foundation, Clearly Explained!!!](https://www.youtube.com/watch?v=zxQyTK8quyY&t=1316s)

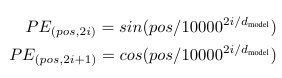


### Encoder

#### positional encoding

Очень важно учитывать порядок слов в предложении (какое слово за каким находится), для этого используют positional encoding

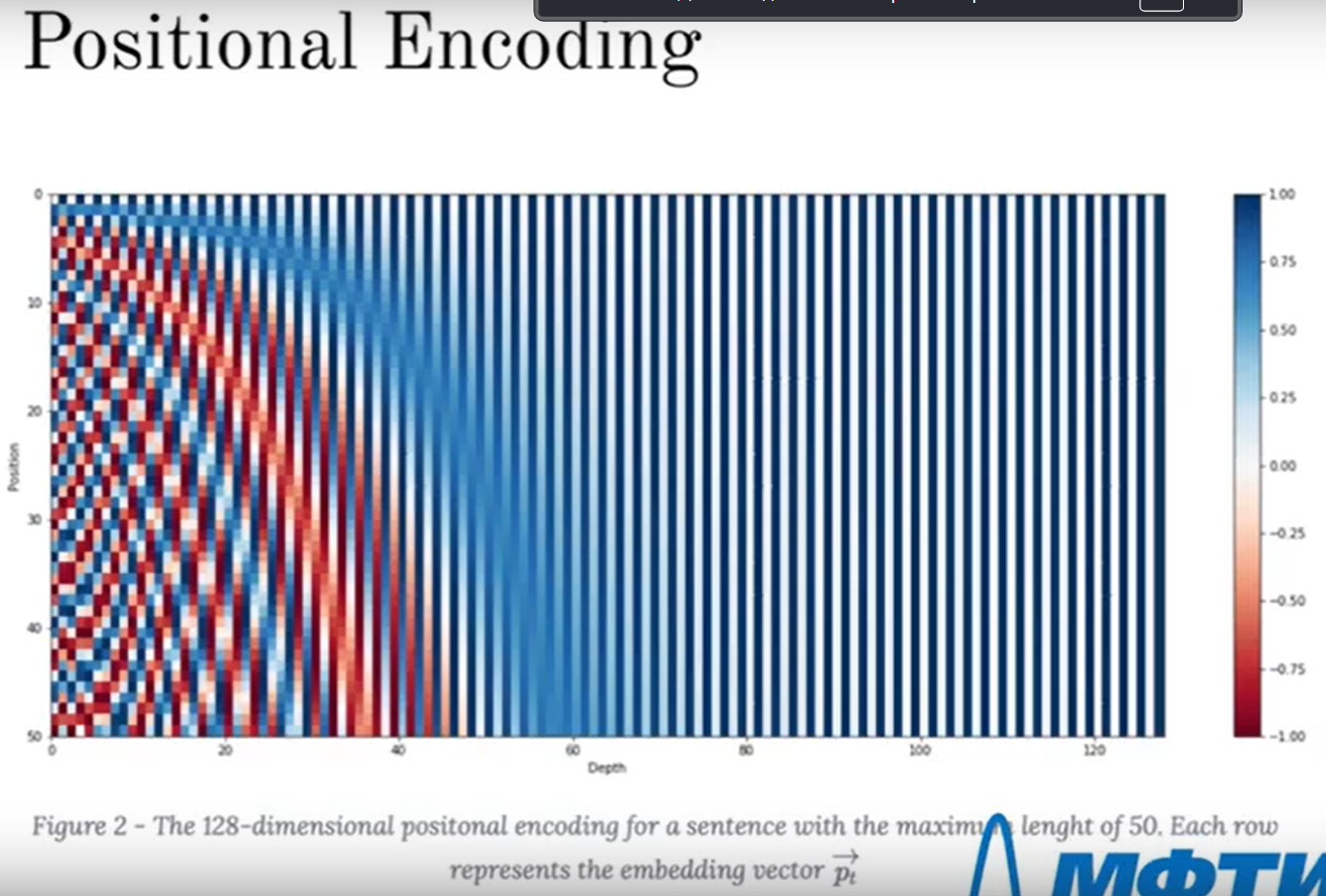
В оригинальной статье используется следующее преобразование



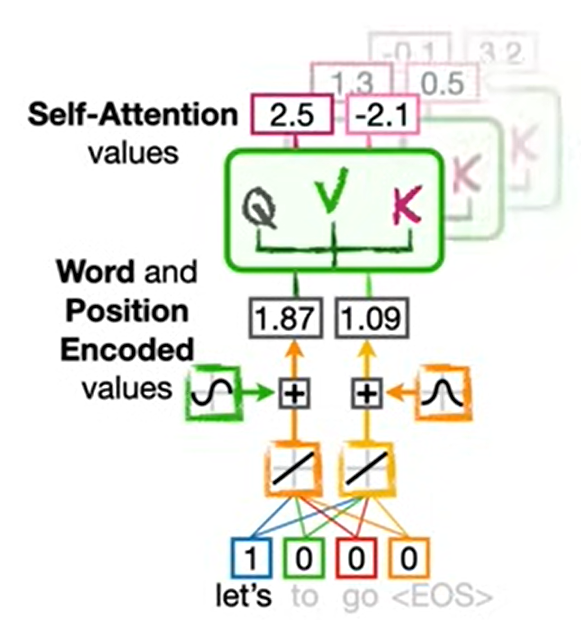
потому что dot product в селф атеншене с таким позициональным кодированием хорошо отражает конкретную разницу между позицией слов (что именно разница 20)

каким образом я не знаю

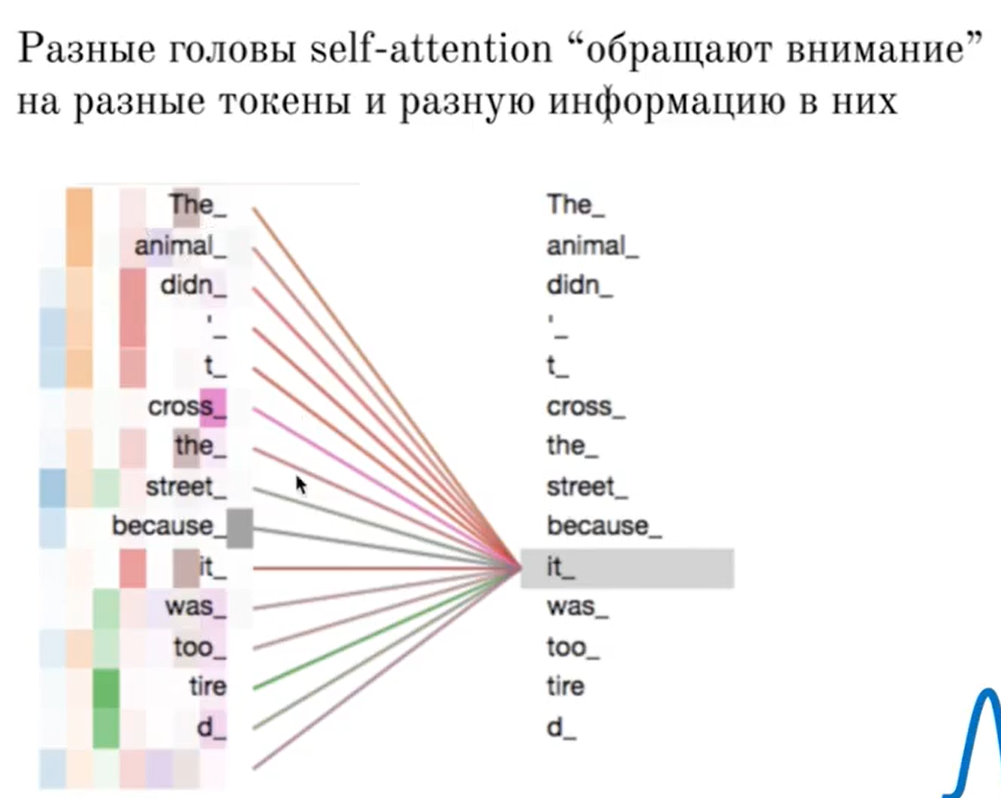
Визуализация этой формулы для эмбеддингов размером 128:



#### multi-head attention



мультихед атеншен - это несколько блоков селф атеншена, которые натренированы на нахождение РАЗНЫХ типов связей между словами в предложении, а на выходе эмбеддинги разных “голов” преобразовывают в новый итоговый эмбеддинг для слова, содержащий информацию про каждый вид отношений.



примеры:

sport is good, it helps a lot

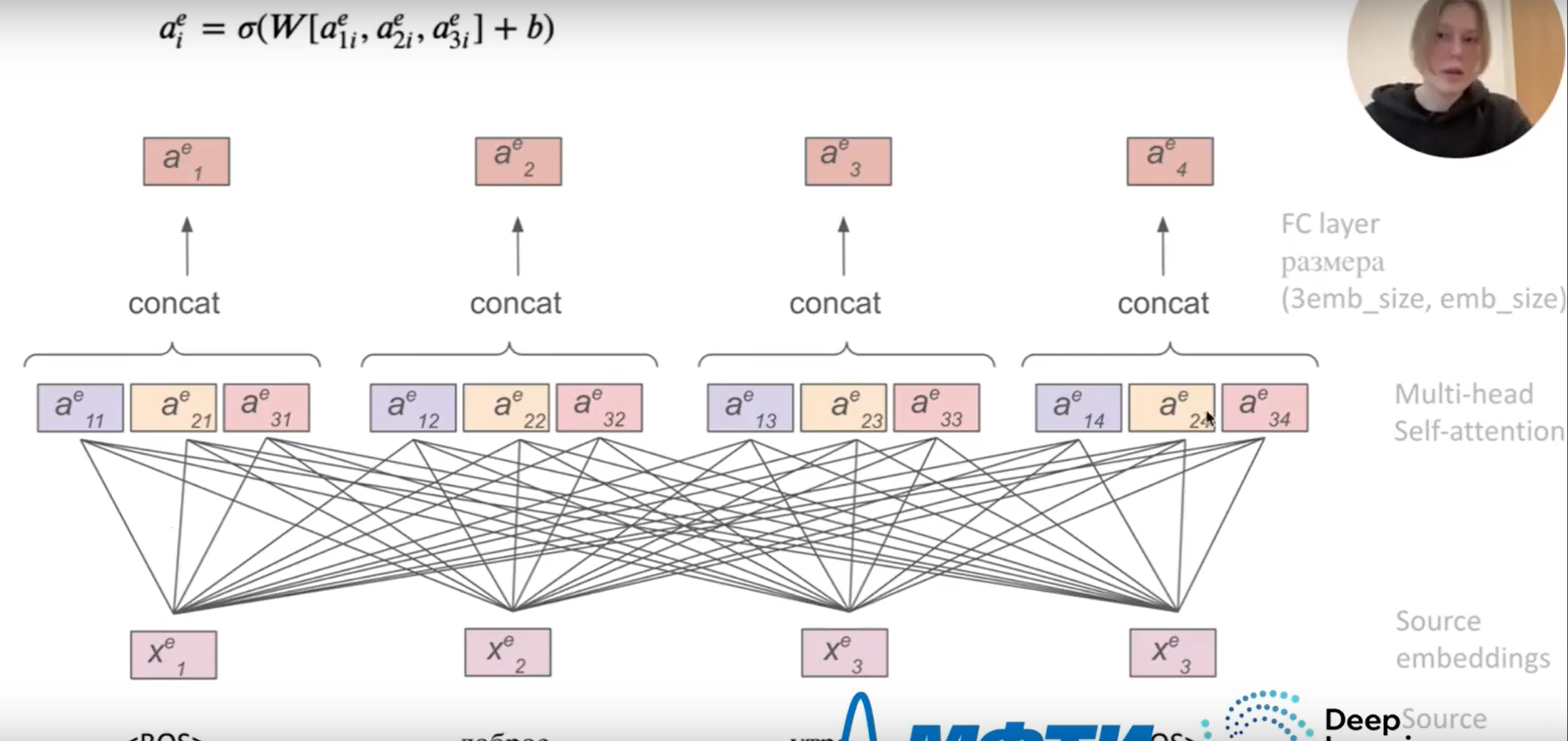
связь типа it - sport

boy wich do good tricks

связь типа boy - wich do

Преобразоание происходит так:

1. после 3 голов мы получаем три эмбеддинга длинной как длина оригинального вектора.
2. конкатенация по 1 оси ( продолжаем эмбединги как бы)
3. прогоняет с помощью fcnn чтобы получить вектор исходной длины



### Decoder

[Лекция. Архитектура Transformer. Decoder, QKV Attention](https://www.youtube.com/watch?v=tsee8mosj5U)

#### Обучение

Такая же идея как и для всех

Обучение:

Декодеру по методу teacher forcing подают несколько слов , когда для генерации нового токена используются предыдущие ИСТИННЫЕ токены.

inference

Генерация токенов происходит авторегрессивно