

Основы практического использования нейронных сетей.

Лекция 10. Трансформеры.

Дмитрий Буряк.
к.ф.-м.н.
dyb04@yandex.ru

Задачи анализа естественного языка

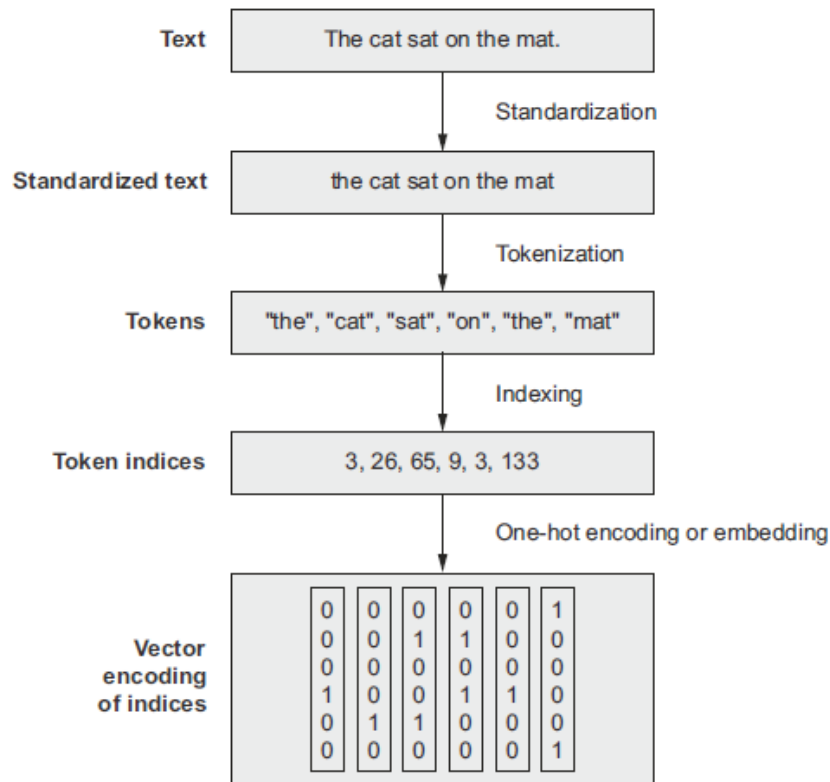
❑ Natural Language Processing (NLP) – применение методов распознавания образов в словах, предложениях, текстах.

❑ Виды задач NLP:

- Классификация текста (text classification)
- Фильтрация контента (content filtering)
- Классификация отзывов (sentiment analysis)
- Моделирование языка (language modeling)
- Перевод (translation)

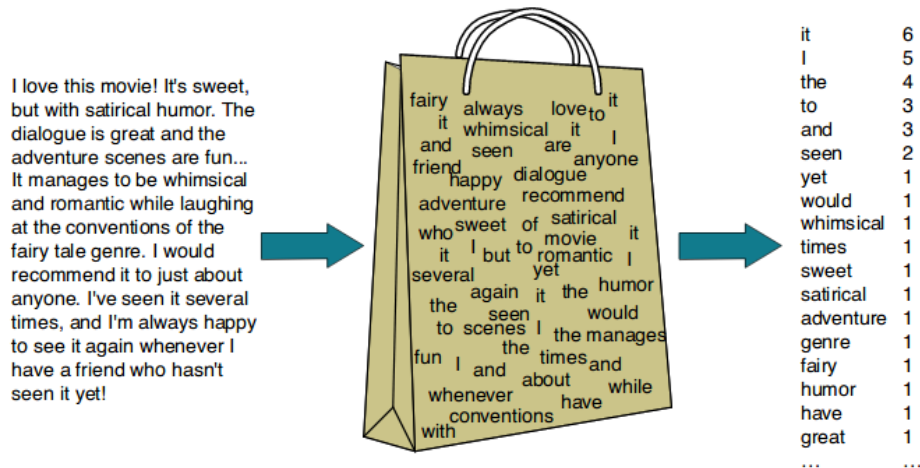
Предобработка текстовых данных

- ❑ Нормализация: приведение к прописным символам, удаление знаков пунктуации.
- ❑ Выделение токенов: символов, слов, групп слов (N-gram).
- ❑ Векторизация: преобразование токена в вектор (словарь → индекс слова в словаре → бинарный вектор/эмбединг).



Представление текста

- ❑ Текст – множество слов (N-gram) – bag-of-words.
 - сети прямого распространения, перцептроны.
- ❑ Текст – последовательность слов
 - Рекуррентные сети (LSTM, GRU)
 - Трансформеры



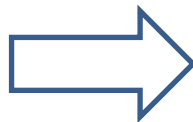
Представление типа bag-of-words

<https://koushik1102.medium.com/nlp-bag-of-words-and-tf-idf-explained-fd1f49dce7c4>

Векторизация. Дескрипторы слов.

❑ Бинарные вектора (on-hot):

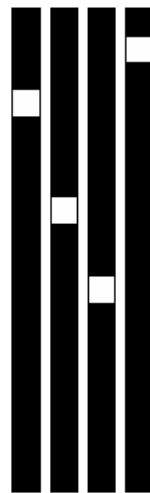
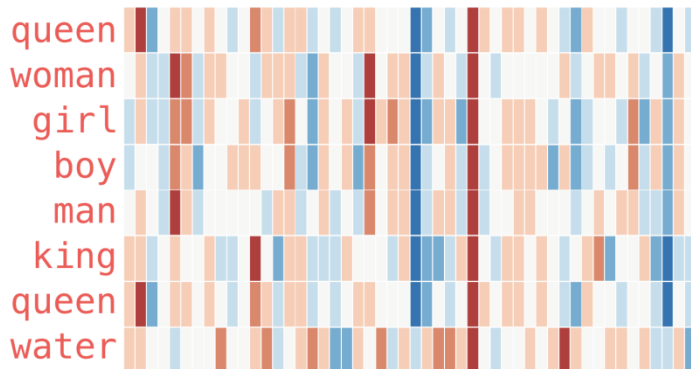
- высокая размерность
- ортогональны
- разрежены.



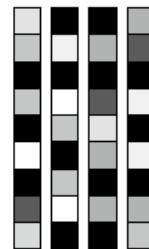
Не отражают свойств
естественного языка

❑ Формирование дескрипторов слов (эмбединг)

- вектора для близких по смыслу слов должны быть ближе, чем те, которые имеют разное значение
- получены в результате обучения
- word2vec
- GLoVe



One-hot word vectors:
- Sparse
- High-dimensional
- Hardcoded



Word embeddings:
- Dense
- Lower-dimensional
- Learned from data

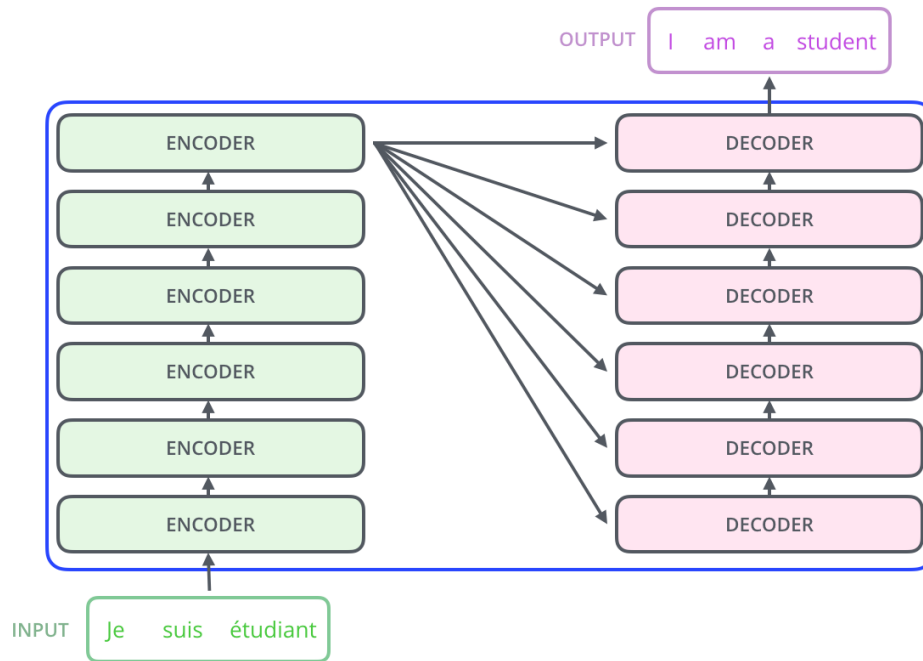
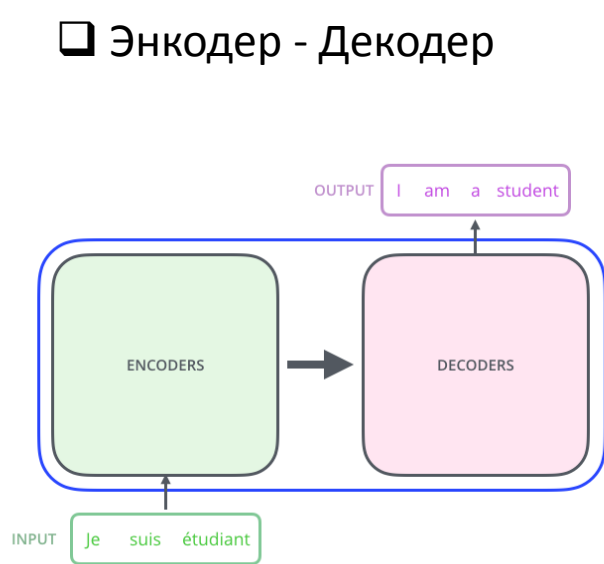
Трансформеры для обработки ТЕКСТОВ

- ❑ Vaswani et al. Attention is all you need, 2017
- ❑ Сеть прямого распространения, частично связанная, несколько параллельных ветвей, без сверточных слоев.
- ❑ Изначально предназначена для NLP, сейчас адаптирована для других задач, например, анализ изображений.



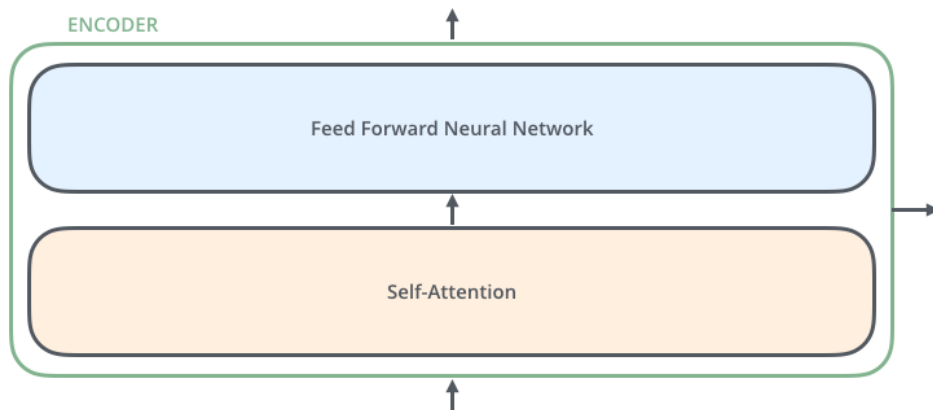
Структура трансформера

❑ Энкодер - Декодер



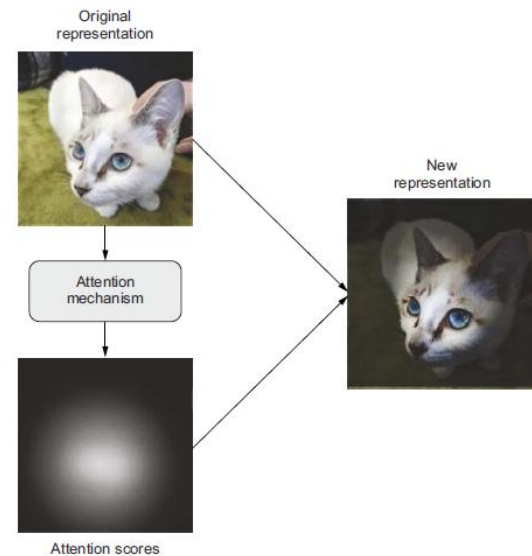
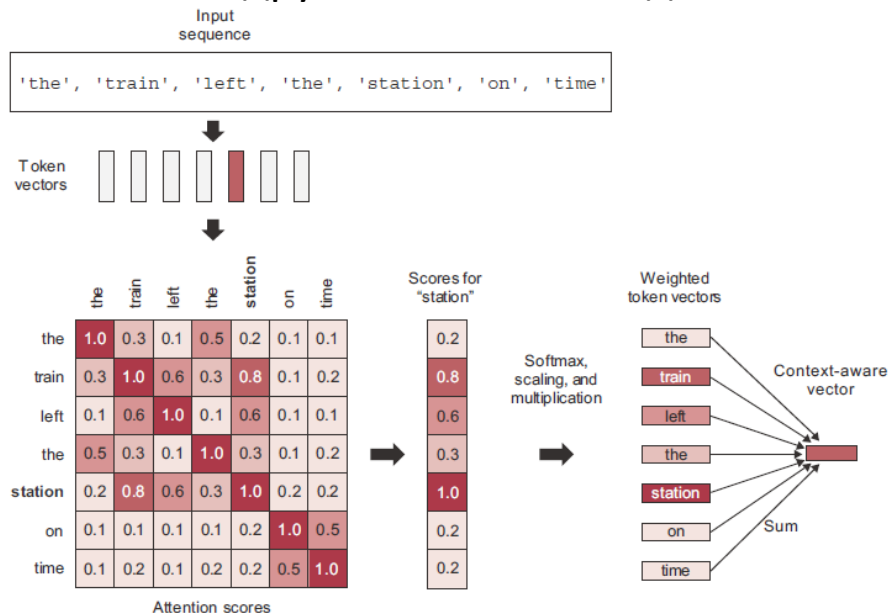
Энкодер

- ❑ Слой внимания: оценить взаимосвязь слов во входящей последовательности.
- ❑ Сеть прямого распространения для отображения результата слоя внимания



Механизм «Внимания»

- ❑ Слова могут иметь разное значение в зависимости от контекста
- ❑ Выделение признаков, зависящих от контекста.
- ❑ Цель – сформировать новое признаковое представление в зависимости от других слов в последовательности.

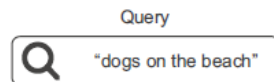


Идея реализации механизма «внимания» в глубоком обучении

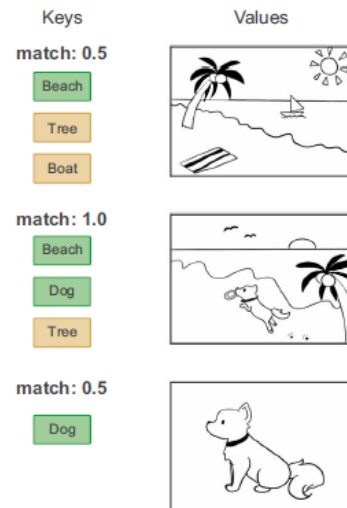
Модель “Query-Key-Value”

- ❑ Запрос (query) сравнивается с известными элементами данных (key) → коэффициент схожести.
- ❑ Для каждого известного элемента сопоставлен вектор значений (value).
- ❑ Результат сумма элементов значений, взвешенных коэффициентами схожести.

```
outputs = sum(values * pairwise_scores(query, keys))
```

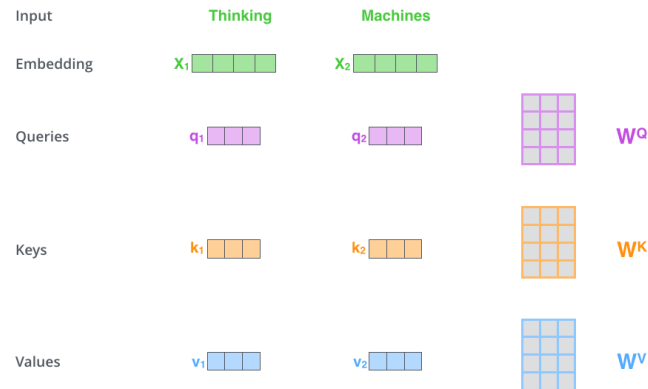


Применение модели “query-key-value”
для ранжирования изображений по их
релевантности текстовому запросу

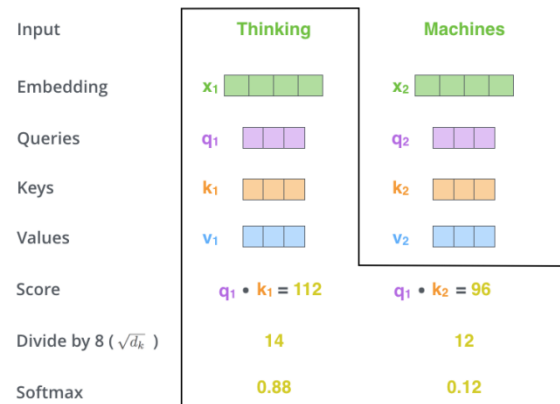


Пример “Query-Key-Value” (1)

❑ Элементы запрос, ключ и значение – результат умножения входного эмбединга на соответствующую матрицу

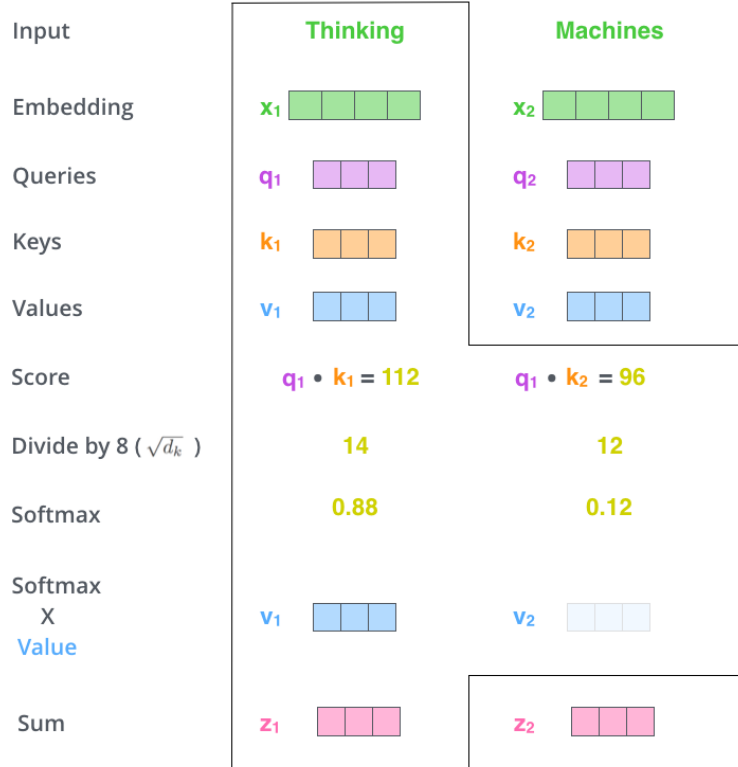


❑ Вычисление коэффициента схожести и взвешивающих коэффициентов (деление на 8 – для данного примера)



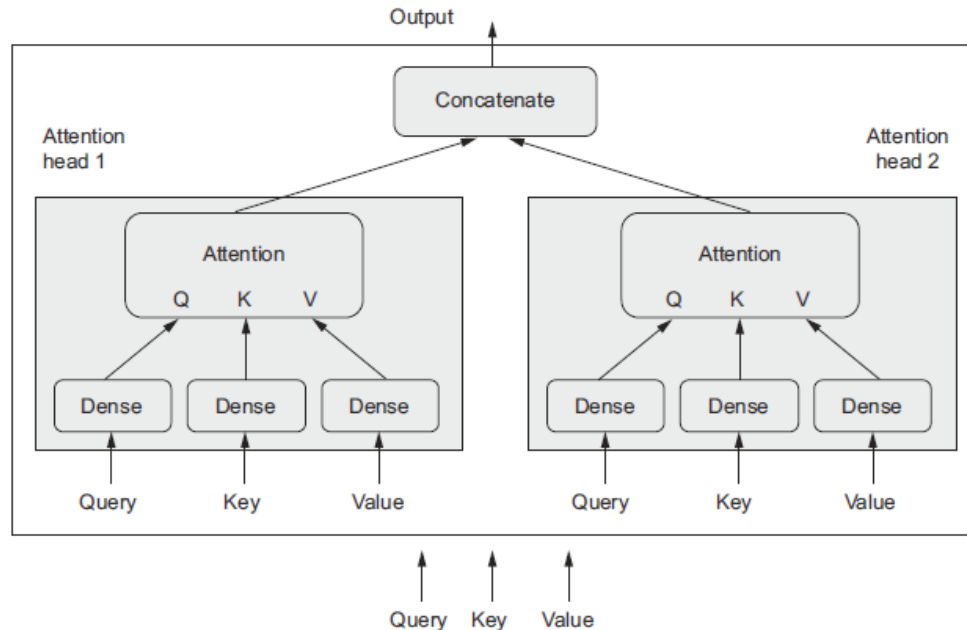
Пример “Query-Key-Value” (2)

□ Получение представления с учетом внимания для входного слова



Множественное внимание

- ❑ Формирование нескольких представлений (параллельно) для входной последовательности с последующей конкатенацией



Пример множественного внимания

1) This is our input sentence*

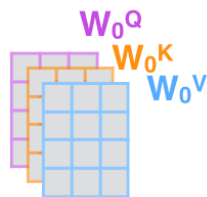
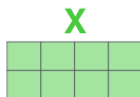
2) We embed each word*

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

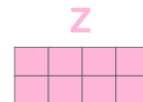
4) Calculate attention using the resulting $Q/K/V$ matrices

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

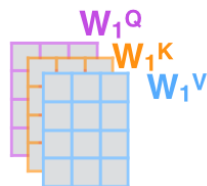
Thinking
Machines



W^O



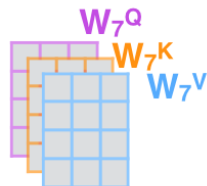
* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one



...

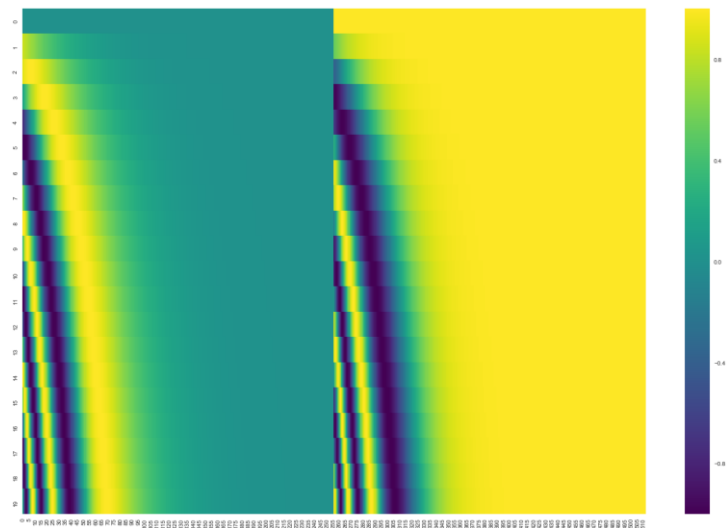
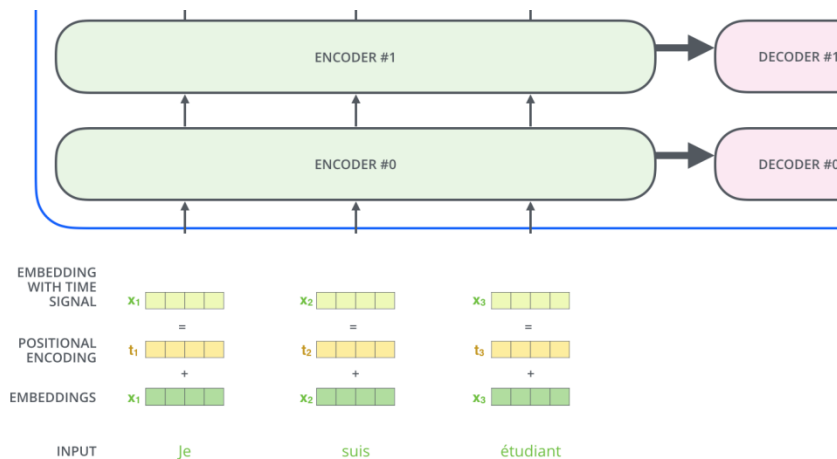
...

...



Позиционное кодирование

- ❑ Необходимо учитывать порядок слов в последовательности.
- ❑ К вектору эмбединг слова добавляется вектор, определяющий его позицию в предложении.



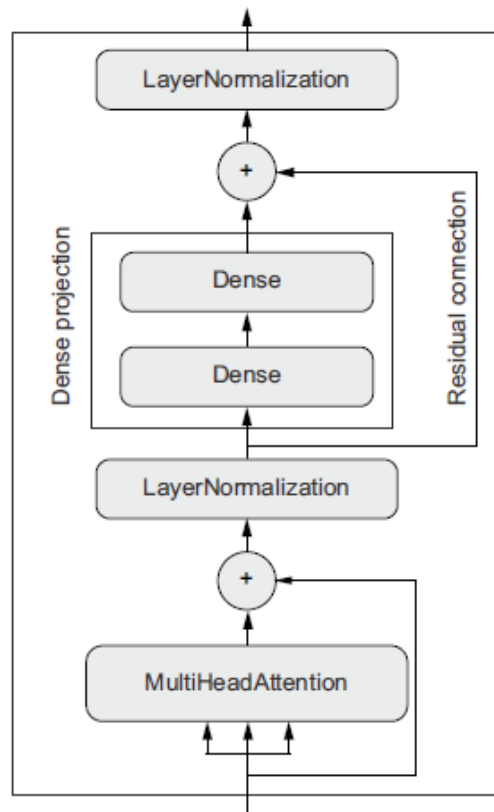
Пример позиционного кодирования для 20 слов. Длина вектора - 512

Архитектура энкодера

❑ Энкодер + классификатор = решение для классификации текстов.

❑ Энкодер + Декодер = модель для задач последовательность – последовательность (sequence-to-sequence)

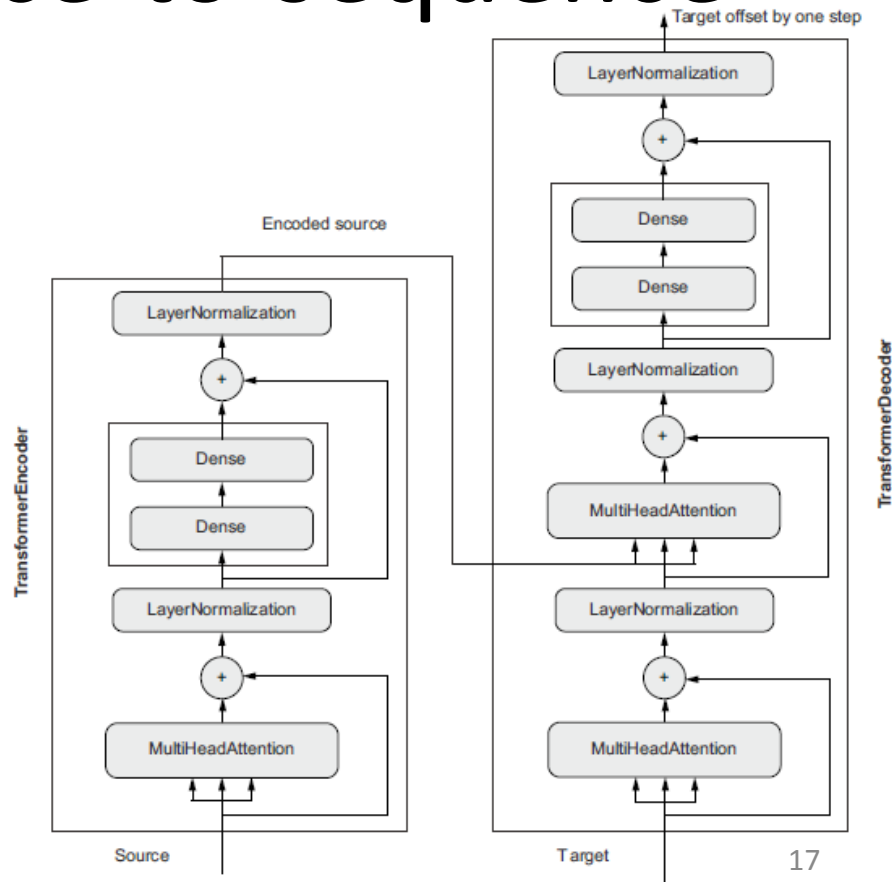
- перевод
- составление абстракта для текста
- ответ на вопрос
- генерация текста.



Модель sequence-to-sequence

❑ Декодер предсказывает выходное слово на основании его взаимосвязи со словами в исходной последовательности:

- выходное слово – запрос
- исходная последовательность – ключ, значение



Генерация выходного слоя

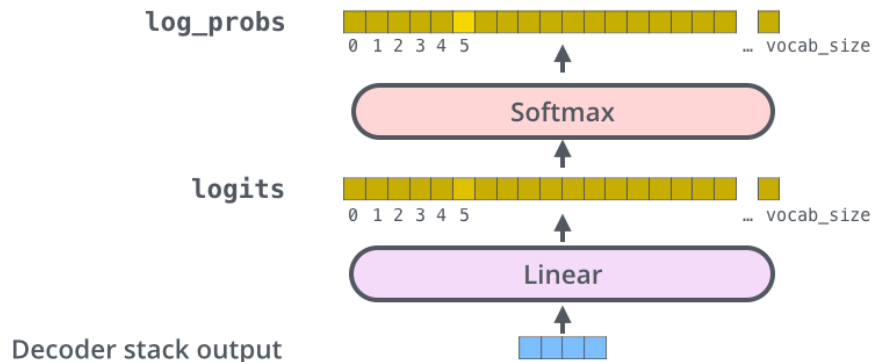
□ Линейный слой + softmax = вектор (размер словаря)

Which word in our vocabulary
is associated with this index?

am

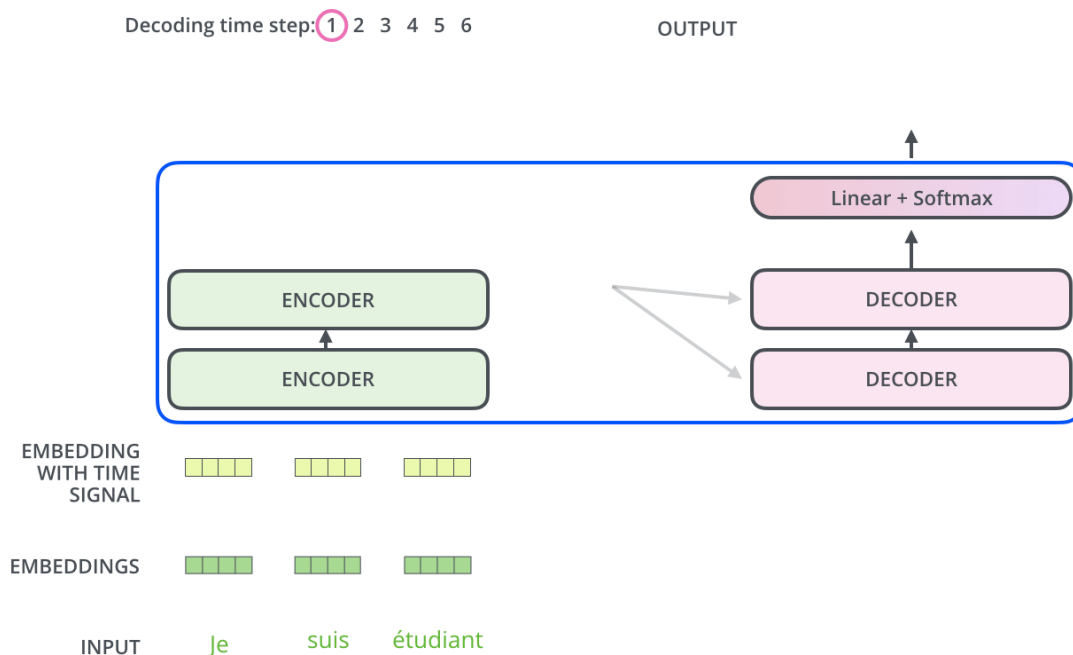
Get the index of the cell
with the highest value
(argmax)

5



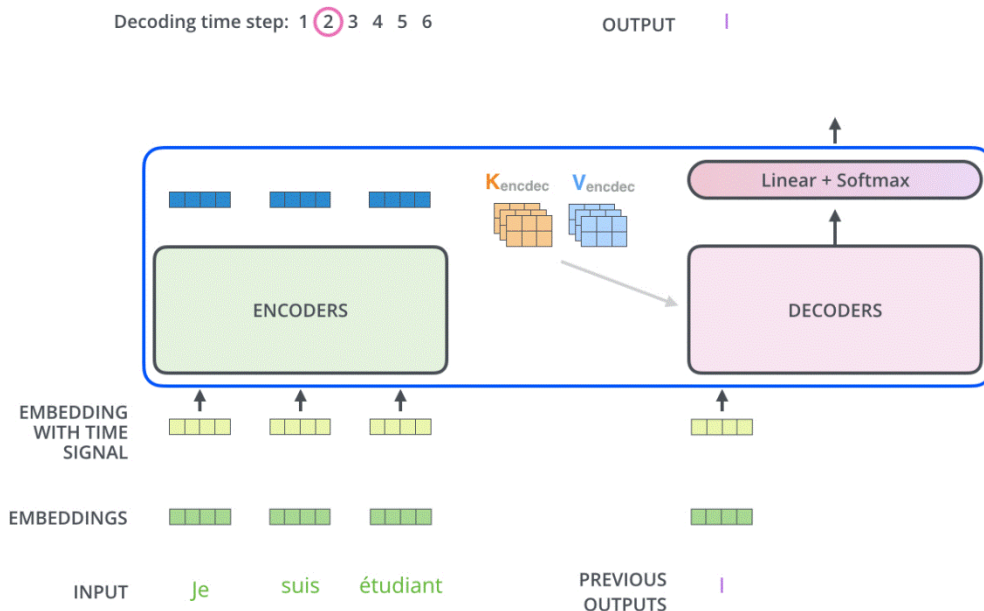
Пример sequence-to-sequence(1)

- ❑ Кодирование входной последовательности, получение первого выходного слова.



Пример sequence-to-sequence(2)

❑ Вызов декодера с текущим сгенерированным фрагментом пока не появится флаг окончания генерируемой последовательности.



Литература

- ❑ François Chollet. Deep Learning with Python, Second Edition, 2021
- ❑ Transformer в картинках (<https://habr.com/ru/post/486358/>)