# Обработка естественного языка. Лекция 5
# Трансформер

Михаил Пожидаев

3 октября 2023 г.

# Трансформер

## Attention!

Не использует рекуррентные ИНС, (почти) и основан только на механизме внутреннего внимания.

## Функция

Трансформер — модель для машинного перевода текста (*transduction model*).

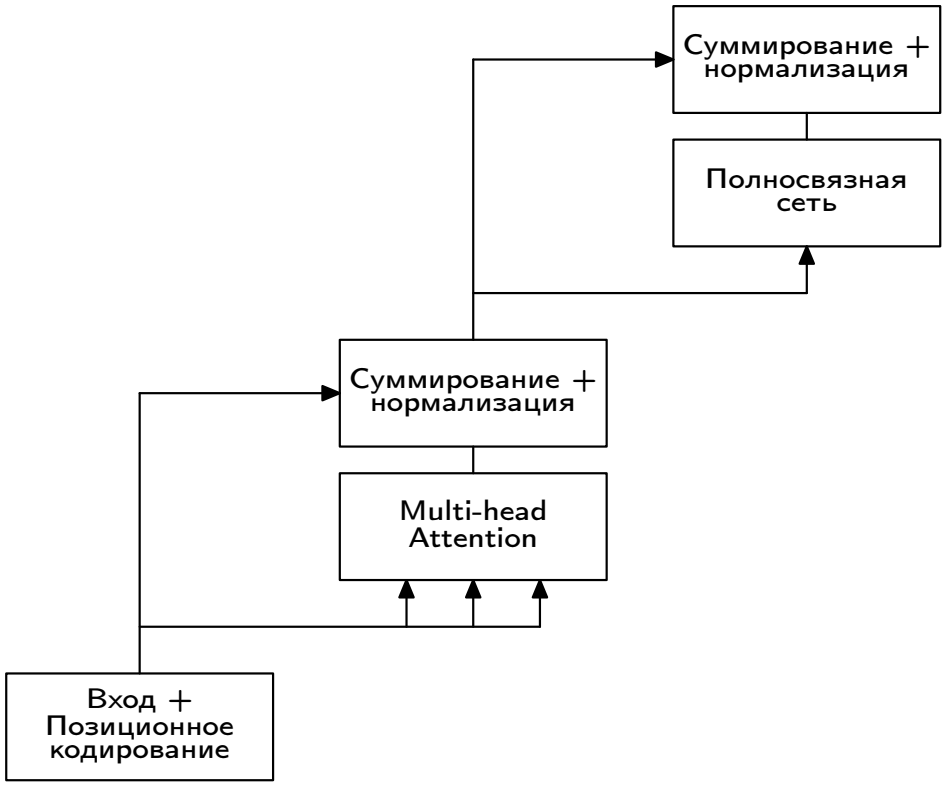
## Наконец!

Разные токены во входной последовательности можно обрабатывать параллельно.

# Блок кодера

1. Входная последовательность $(x_1, x_2, \ldots, x_n)$ обрабатывается ячейкой внимания и преобразуется в последовательность $(z_1, z_2, \ldots, z_n)$. Каждый $x_t$ представлен в виде word embedding.
2. Каждый вектор $z_t$ затем обрабатывается дополнительной линейной ячейкой, и это можно делать параллельно!

# Кодер

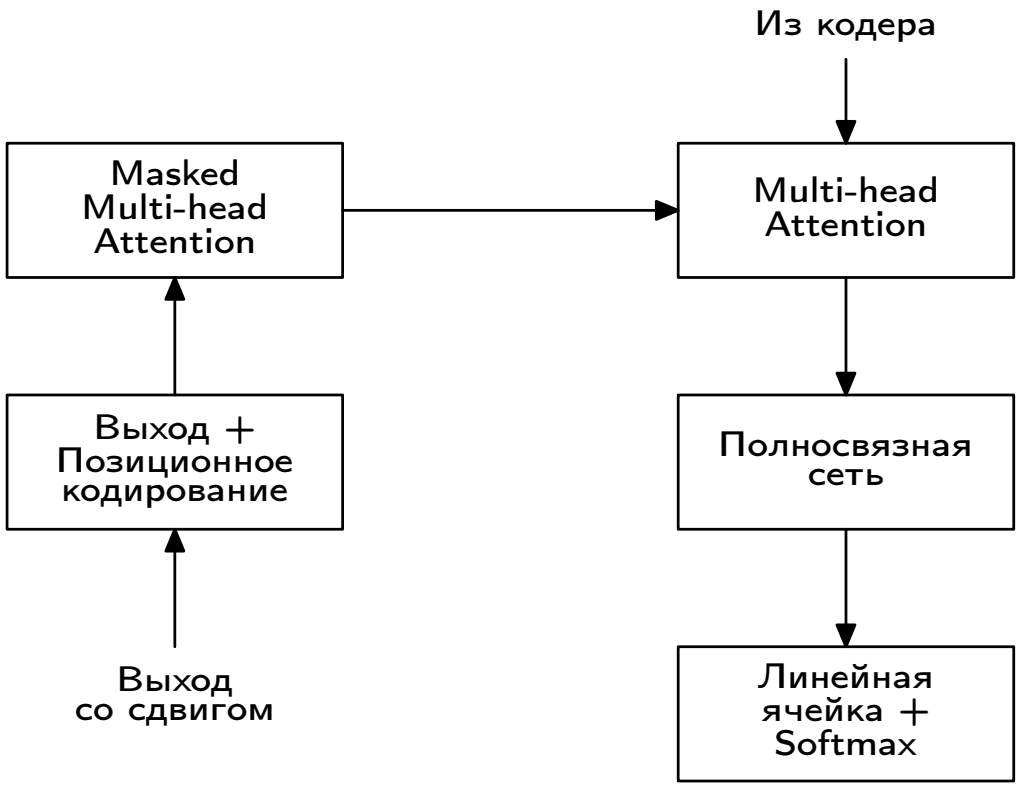

Суммирование + нормализация
Полносвязная сеть
Суммирование + нормализация
Multi-head Attention
Вход + Позиционное кодирование

# Блок декодера

Блок декодера похож на блок кодера, но содержит дополнительную ячейку и имеет следующую структуру:

1. Ячейка внимания (как в кодере).
2. Дополнительная ячейка внимания с модифицированными связями функционально близкая к вниманию в варианте Богданова.
3. Линейная ячейка (как в кодере).

# Декодер


Из кодера
Masked Multi-head Attention
Multi-head Attention
Выход + Позиционное кодирование
Полносвязная сеть
Выход со сдвигом
Линейная ячейка + Softmax

# Структура трансформера

Трансформер содержит $2k$ блоков:

- $k$ блоков кодера, каждый из которых включает:
  - одну ячейку внимания;
  - одну линейную ячейку;
- $k$ блоков декодера, каждый из которых содержит:
  - две ячейки внимания, одна из которых повторяет назначение внимания Богданова;
  - одну линейную ячейку как в кодере;
- ещё одна линейная ячейка для порождения выходных токенов.

$k$ — параметр модели, обычно равен 6.

# Три типа внимания

1. Обычный multi-head-attention в кодере, в котором векторы , *query*, *key* и *value* вычисляются обычным способом на текущей последовательности.
2. Multi-head-attention в декодере, похожий на ячейку в кодере, но не имеющий доступа к элементам последовательности справа от текущего.
3. Дополнительная ячейка в декодере, в котором *query* берутся с предыдущего шага декодера, а *key* и *value* — с выхода кодера.

# Схема работы декодера

## Уже было в Seq2seq!

Декодер работает по схеме, напоминающей работу декодера в Seq2seq, в которой совершается несколько шагов, на каждом из которых генерируется новый элемент выходной последовательности.

1. Новый элемент генерируется дополнительной линейной ячейкой.
2. Процесс продолжается, пока не будет получен токен, обозначающий конец сегмента.

# Кодирование положения токена

Непонятно, где это было!

Внутреннее внимание не сохраняет информацию о положении исходного элемента, поэтому добавим дополнительное кодирование (position encoding).

$$PE(pos, 2i) = \sin(pos/10000^{2i/d})$$
$$PE(pos, 2i+1) = \cos(pos/10000^{2i/d})$$

- $pos$ — индекс входного токена;
- $2i$ и $2i+1$ — индекс компоненты вектора;
- $d$ — размерность векторного пространства.

# Спасибо за внимание!

Всё о курсе: https://marigostra.ru/materials/nlp.html
E-mail: msp@luwrain.org
Канал в Телеграм: https://t.me/MarigostraRu