Математические методы анализа текстов

Лекция

Sequence-to-sequence, механизмы внимания и transformer на примере задачи машинного перевода

Мурат Апишев (mel-lain@yandex.ru)

15 октября, 2019

Задача машинного перевода

- Задача перевода – массовая, её было бы здорово решать автоматически
- Идея машинного перевода зародилась ещё в 1947 году и к середине 60-х уже появились первые системы
- Несмотря на сложности, область развивается до сих пор, особенно сильный рост качества произошёл в последние годы
- Изначально задача решалась с помощью статистических методов (IBM Model)
- В 2013 году появилась первая полностью нейросетевая модель, с 2016 нейросетевой машинный перевод стал индустриальным стандартом
- Современный машинный перевод хорош в ситуациях, где тексты формализованы или же достаточно грубого перевода
- С художественной литературой до сих пор всё плохо

Сразу про метрики

- Стандартной автоматической метрикой в MT является BLEU
- На вход подаётся перевод, сделанный машиной и перевод, который считается правильным
- Рассмотрим на примере:

  Правильный ответ: *E-mail was sent on Tuesday*

  Ответ системы: *The letter was sent on Tuesday*

- Посчитаем для заданного N (обычно 3-4) число N-грамм в ответе системы, которые присутствуют в правильном ответе (1 к 1):

$$N = 1 \Rightarrow \frac{4}{6} \qquad N = 2 \Rightarrow \frac{3}{5} \qquad N = 3 \Rightarrow \frac{2}{4} \qquad N = 4 \Rightarrow \frac{1}{3}$$

Сразу про метрики

- Посчитаем для заданного N (обычно 3-4) число N-грамм в ответе системы, которые присутствуют в правильном ответе (1 к 1):

$$N = 1 \Rightarrow \frac{4}{6} \qquad N = 2 \Rightarrow \frac{3}{5} \qquad N = 3 \Rightarrow \frac{2}{4} \qquad N = 4 \Rightarrow \frac{1}{3}$$

- Теперь посчитаем среднее геометрическое по всем N:

$$\text{score} = \sqrt[4]{\frac{4}{6} \cdot \frac{3}{5} \cdot \frac{2}{4} \cdot \frac{1}{3}}$$

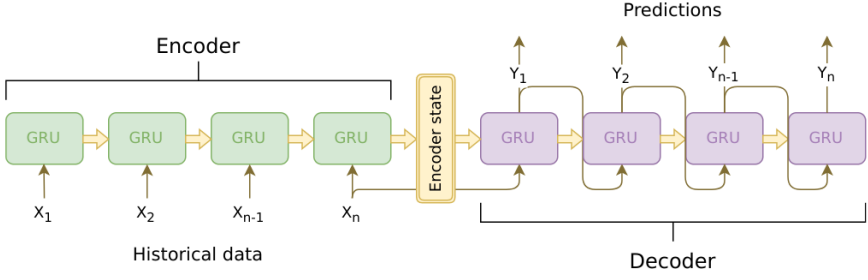
- Вместо подсчёта recall вводится штраф за краткость (Brevity penalty):

$$BP = \min(1, 6/5)$$

- Итоговая значение метрики BLEU: $BP \cdot score \approx 0.5081$

Sequence-to-sequence

- Архитектура из нейросетевых кодировщика и декодировщика
- Показывает хорошие результаты в задачах машинного перевода, суммаризации и генерации подписей к картинкам
- Кодировщик получает на вход последовательность входных элементов и генерирует числовой вектор контекста h_n
- На базе этого вектора декодировщик генерирует выходную последовательность



https://jeddy92.github.io/JEddy92.github.io/ts_seq2seq_intro

Seq-to-seq для MT

- И кодировщик, и декодировщик – рекурентные сети
- Входные слова x_i ($i \in [1, n]$) представляются в виде эмбеддингов (например, w2v)
- Узким местом данного подхода является вектор контекста h_n, представляющий собой вектор состояния RNN-кодировщика
- Очевидно, что последние слова входной фразы будут оказывать на него большее влияние, чем первые
- Как следствие, такой подход позволяет переводить только короткие фразы
- Одно из возможных решений – *механизм внимания* (attention)

---

https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention

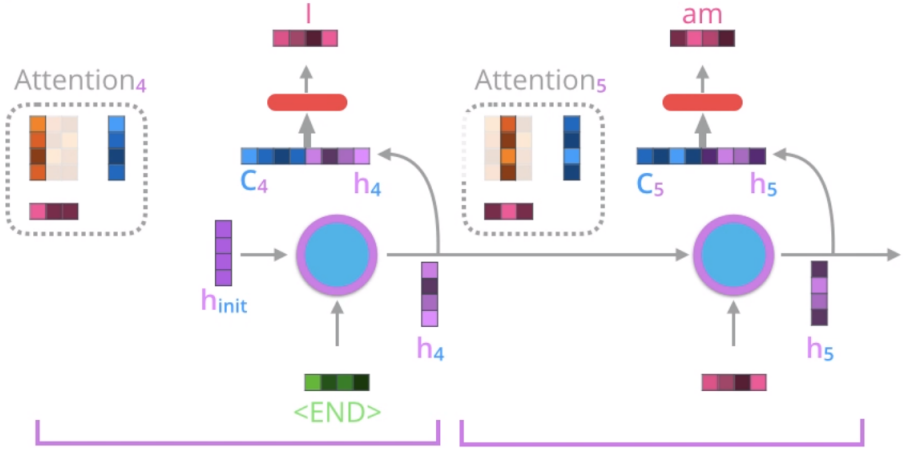
Seq-to-seq для MT + attention

- Одно из возможных решений – *механизм внимания* (attention):
  - при ручном переводе слова в предложении мы смотрим не только на само слово, но и на релевантный контекст
  - в то же время, мы игнорируем те части предложения, которые к текущему переводимому слов не относятся
- Кодировщик передаёт декодировщику не последнее значение своего вектора состояния h_n, а все $h_i, i \in [1, n]$
- Каждый вектор в наибольшей степени отражает влияние того слова, при обработке которого он был получен

Seq-to-seq для MT + attention

При генерации очередного слова декодировщик:

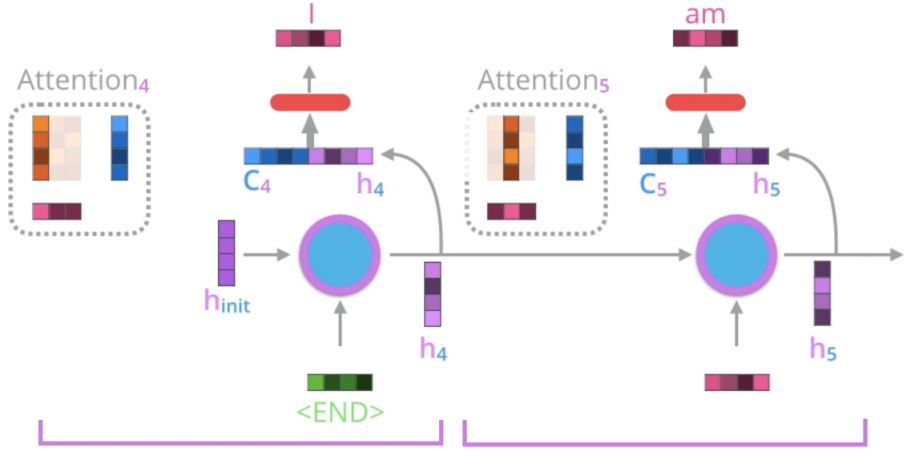
- получает на вход предшествующий вектор своего состояния $h_{j-1}, j \in [n+1, m]$ и выходной вектор последнего сгенерированного слова (или метки старта) s_{j-1}
- выдаёт новый вектор своего состояния h_j и вектор ответа – ответ игнорируется



Seq-to-seq для MT + attention

При генерации очередного слова декодировщик:

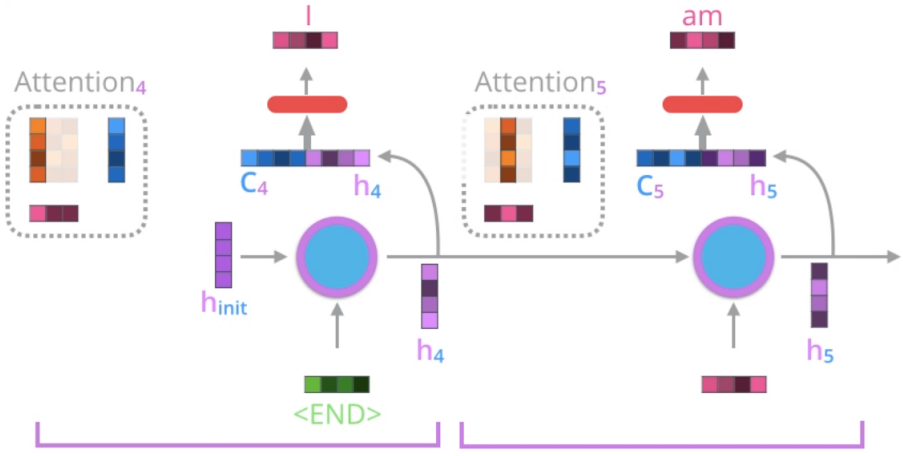
- считает для каждого вектора состояния кодировщика h_i некоторый вес α_{ij}, отражающий его важность при генерации текущего слова
- берёт взвешенную сумму векторов h_i и конкатенирует с вектором своего состояния h_j



Seq-to-seq для MT + attention

При генерации очередного слова декодировщик:

- подаёт результат в общую для модели полносвязную сеть, которая возвращает вектор s_j для текущего слова, а из него после применения softmax генерируется слово перевода y_j



Подсчёт весов attention

- Вес α_{ij} вектора состояния кодировщика h_i при генерации слова j зависит от самого h_j и выходного вектора для предыдущего слова s_{j-1}:

$$a_{ij} = \frac{\exp(\mathrm{sim}(h_i, s_{j-1}))}{\sum_k \exp(\mathrm{sim}(h_k, s_{j-1}))}$$

- Считать функцию близости sim можно по-разному:
  - Простое скалярное произведение:

$$\mathrm{sim}(h, s) = h^T s$$

  - Аддитивное внимание:

$$\mathrm{sim}(h, s) = w^T \tanh(W_h h + W_s s)$$

  - Мультипликативное внимание:

$$\mathrm{sim}(h, s) = h^T W s$$

- Параметры весовых функций (при наличии) обучаются вместе с основной сетью

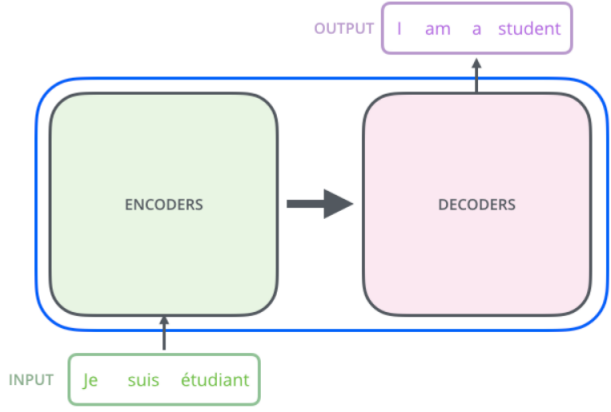
Attention Is All You Need

- До последнего времени основой для seq-to-seq сетей служили рекурентные сети
- Основная проблема в их использовании – большие затраты времени и вычислительных ресурсов на обучение
- В 2017 году была предложена архитектура Transformer, основной идея которой состоит в полном отказе от рекурентных слоёв в кодировщике и декодировщике
- Вместо этого предлагается использовать новый тип слоя – multi-head self-attention, работающий исключительно на основе механизма внимания
- Transformer превзошёл имеющиеся на тот момент архитектуры на основе LSTM и GRU как по качеству решения (в т.ч. и в переводе), так и по скорости обучения

https://arxiv.org/pdf/1706.03762.pdf

Transformer сверху вниз

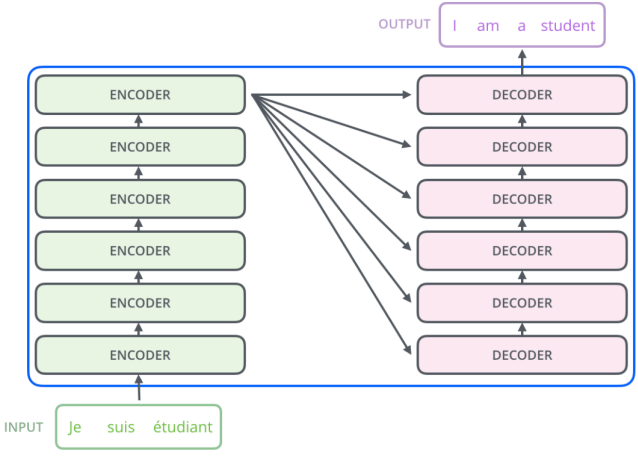
- По ссылке ниже находится одно из наиболее подробных и доступных объяснений трансформера, будем следовать ему
- Верхнеуровнево это всё тот же кодировщик-декодировщик



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

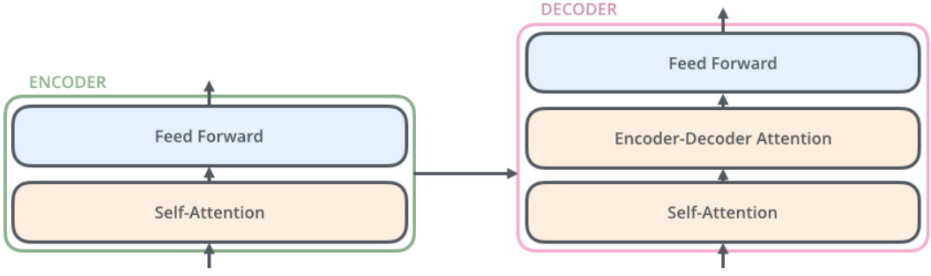
Transformer сверху вниз

- Кодировщик и декодировщик состоят из своих наборов одинаковых блоков, блоки стекаются друг за другом
- В оригинальной статье блоков 6, но это не принципиально
- Веса у каждого блока свои (т.е. неразделяемые)



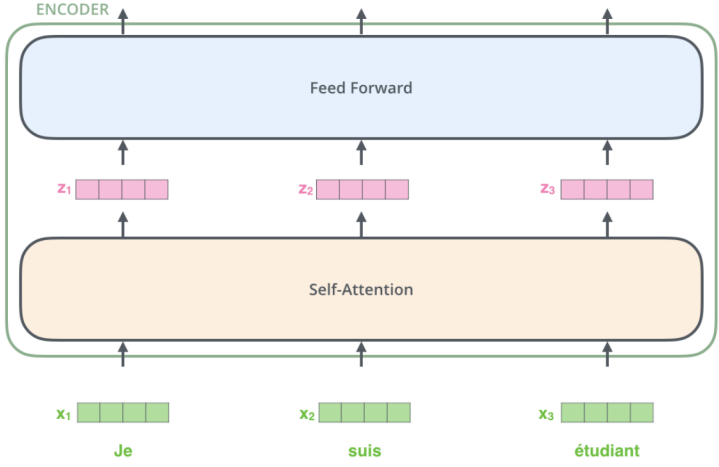
Transformer сверху вниз

- Первый слой кодировщика – self-attention, который помогает кодировать каждое слово последовательности с учётом остальных (рассмотрим далее)
- Далее выход self-attention для каждого элемента последовательности проходит через одну и ту же полносвязную сеть
- Декодер дополнительно к этим слоям имеет слой обычного attention для концентрации на нужных частях последовательности векторов

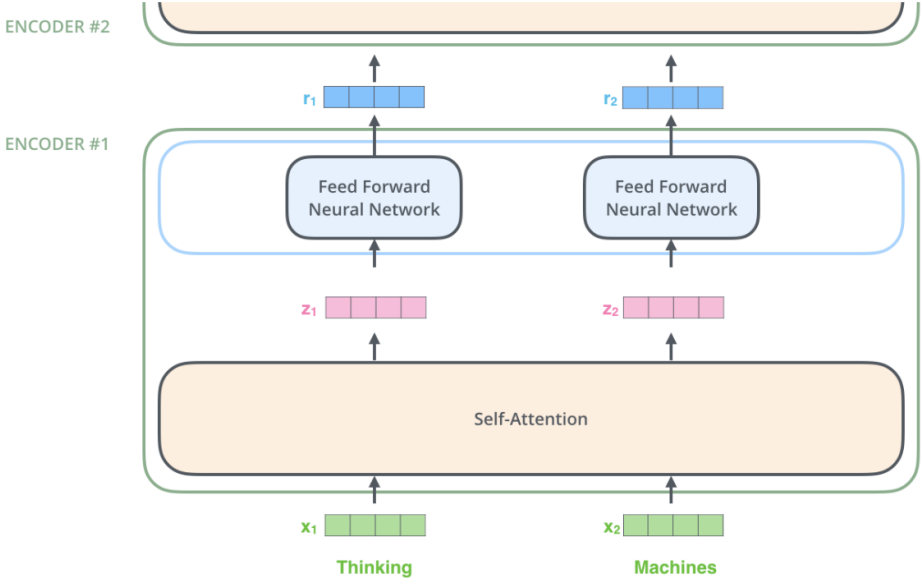


Transformer сверху вниз

- На вход первого кодировщика приходят эмбеддинги слов, остальные получают выходы предшественников
- Слова последовательности обрабатываются взаимнозависимо в слое self-attention, но независимо в полносвязном (можно параллелить)

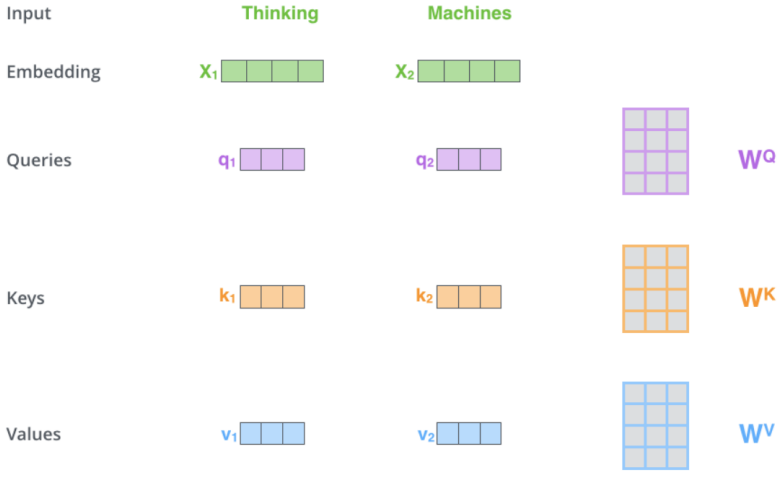


Transformer сверху вниз



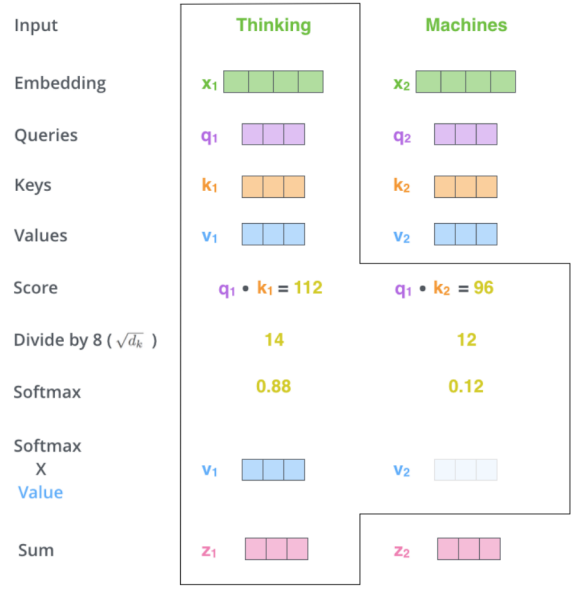
Self-attention

- Для каждого входного считаются три вектора:`Key`, `Value` и `Query`
- Матрицы преобразований обучаются вместе с сетью

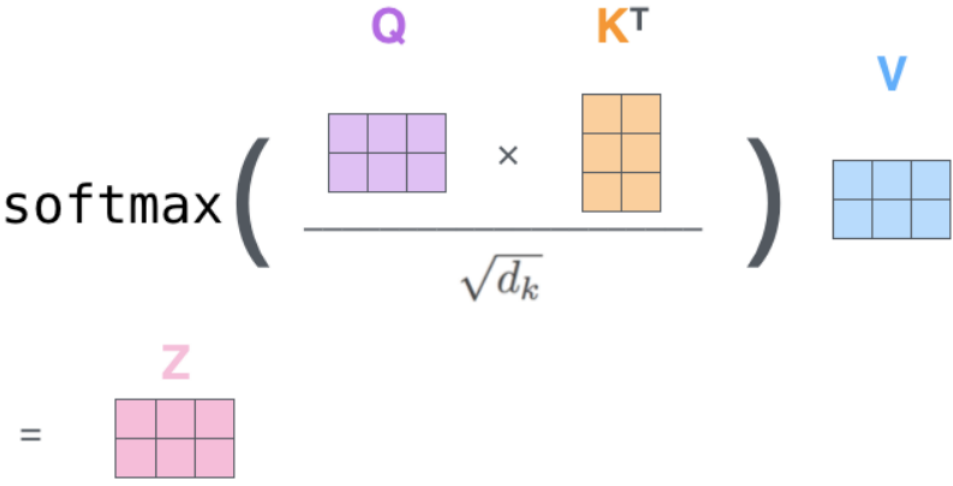


Self-attention

- Как и в обычном attention, задача в том, чтобы сгенерировать вектор для слова с учётом других слов в последовательности
- Для очередного слова процесс такой:
  - вектор `Query` для текущего слова скалярно умножаем на векторы `Key` всех входных слов, получаем веса
  - делим все веса на некоторую константу (для стабильности градиентов), пропускаем через softmax
  - складываем все векторы `Value` с полученными весами, получаем итоговый вектор для слова в контексте остальной последовательности

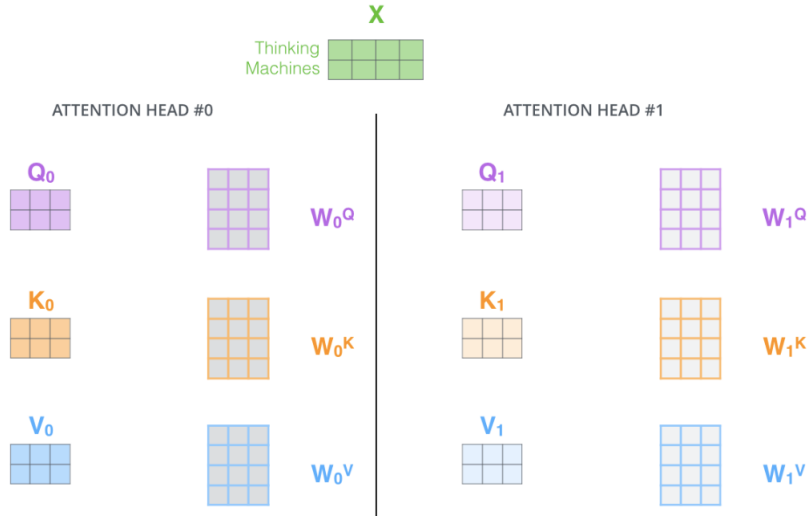


Self-attention в матричном виде



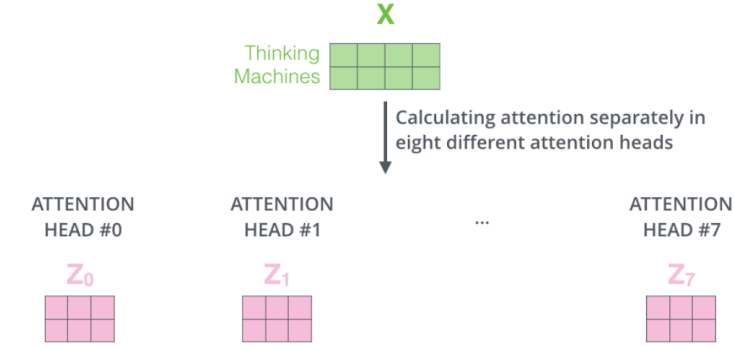
Multi-head self-attention

- **Идея:** будем параллельно считать не один вектор self-attention, а несколько

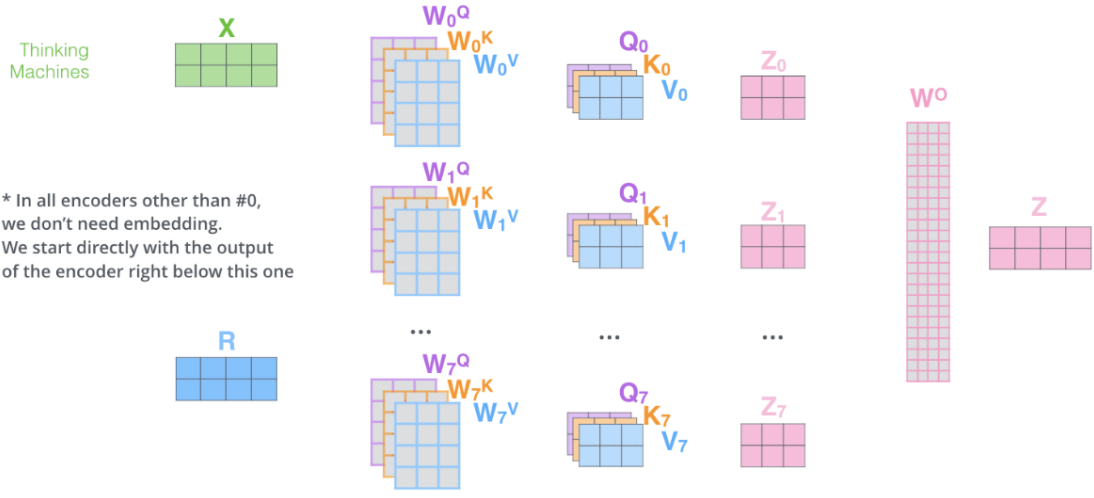


Multi-head self-attention

- Эксперименты показывают, что матрицы весов, инициализированные по-разному, выделяют различные аспекты слова в последовательности
- На выходе получается несколько матриц векторов для одно последовательности
- Перед подачей в полносвязную сеть они конкатенируются и умножаются на промежуточную весовую матрицу (W_0) для сохранения размерности

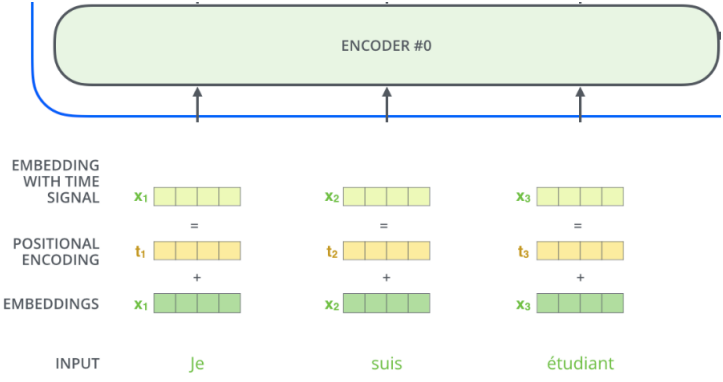


Общая схема multi-head self-attention



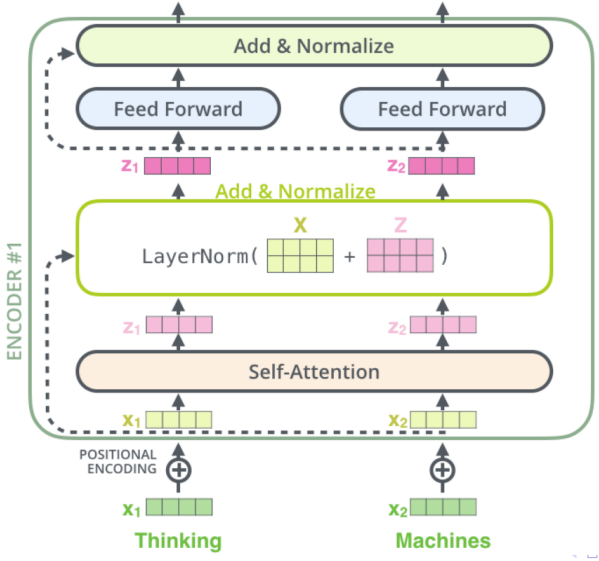
Positional encoding

- *Позиционное кодирование* (positional encoding) – способ передачи информации о взаимном расстоянии между словами в последовательности через их эмбеддинги
- Для этого к эмбеддингу слова прибавляется вектор, представляющий собой набор значений синусов и косинусов с разными периодами от позиции слова в предложении



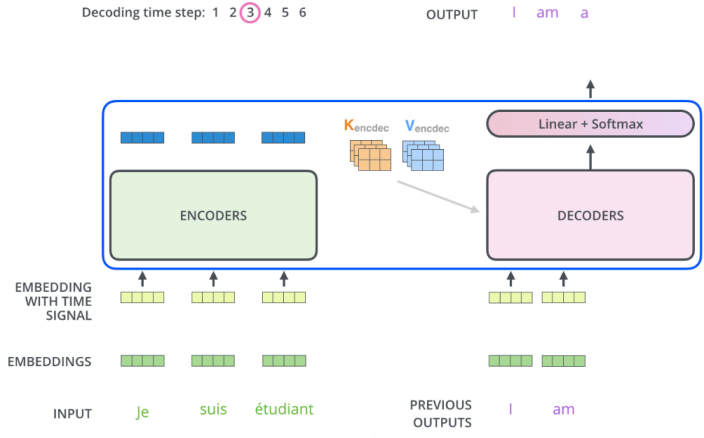
Детали устройства кодировщика

- Для борьбы с затуханием градиента добавляются residual connections
- Для ускорения обучения и повышения качества используется Layer normalization



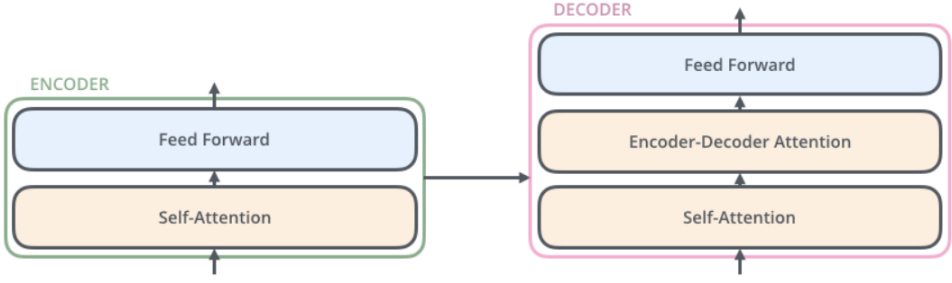
Декодировщик

- После окончания работы последнего кодировщика его выходы для всей последовательности преобразуются в два матрицы – `Key` и `Value`
- Эти матрицы передаются в каждый из декодировщиков и обрабатываются там слоем Encoder-Decoder Attention (обычный multi-head self-attention)
- Но для работы слоя не хватает векторов `Query`



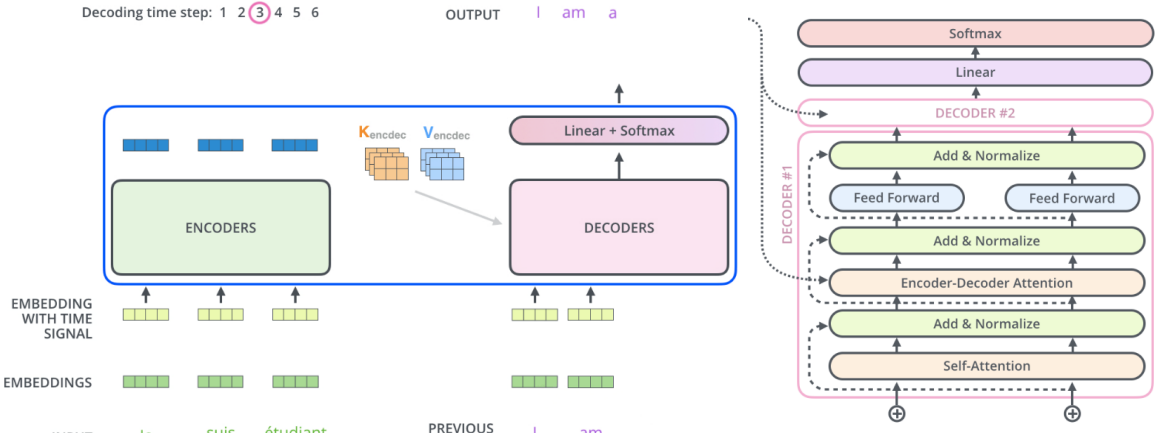
Декодировщик

- Но для работы слоя не хватает векторов `Query`
- Они получаются следующим образом:
  - подаём на вход нижнему слою multi-head self-attention векторы уже сгенерированных слов
  - на выходе получим преобразованные векторы, которые и будем использовать как `Query` в Encoder-Decoder Attention
  - учитываться при этом будут только позиции, соответствующие уже сгенерированным словам



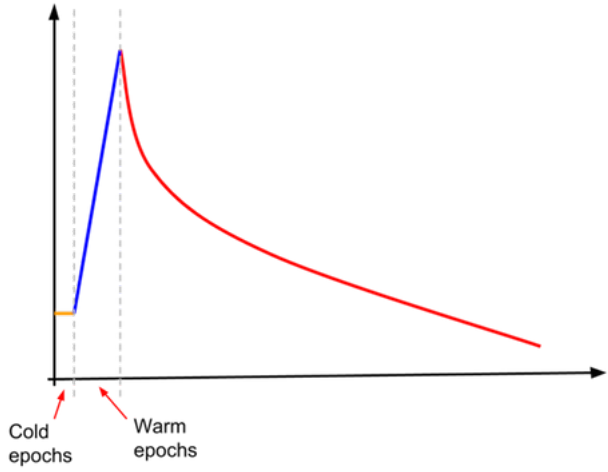
Transformer: последние штрихи

- Слева – полная схема трансформера, справа – внутренности декодировщика
- Выходы последнего декодировщика конкатенируются в один вектор фиксированной длины и проходят через softmax (дальше argmax или beam search)



Warm-up learning rate

- Техника подбора траектории изменения learning rate, без которой обучить трансформер проблематично
- Позволяет бороться с эффектом сильного влияния первых обучающих объектов
- На первом батче используется небольшое значение
- С каждым следующим батчем оно растёт линейно (warm-up period)
- С некоторого момента learning rate начинает убывать
- Warm-up period для сильно отличных от батча к батчу данных должен быть длиннее, чем для более равномерных



Проблема OOV-слов

- Большинство способов обработки новых слов связано с использованием эмбеддингов символов или символьных N-грамм
- Иногда модель эмбеддингов строится до обучения основной сети (например, FastText)
- А иногда эмбеддинги обучаются одновременно с сетью
- Например, можно подавать на вход вспомогательной CNN или BiLSTM one-hot векторы символов последовательности слов, и выходы вспомогательной сети подавать в основную как эмбеддинги слов
- Для выделения символьных N-грамм можно использовать byte-pair encoding

Byte-pair encoding

- Метод сжатия данных (неоптимальный), который используется для регулирования длины словаря и борьбы с OOV-словами
- Кодируем каждый символ байтом и итеративно считаем по корпусу статистики совстречаемости
- На каждой итерации сливаем два наиболее часто встречающихся рядом байта в один новый
- Останавливаемся при достижении нужного размера словаря
- Каждый байт кодируется своим эмбеддингом, слово получается как сумма входящих в него байтов ($\Rightarrow$ снижается воздействие морфологии)
- В чистом виде такая реализация плохая (квадратичная сложность), можно сделать лучше (см. ссылку)

---

https://narsil.github.io/2019/05/16/model-based-bpe-encodings.html

Пример смежной задачи: семантический парсинг

- Задача *семантического парсинга* (Semantic parsing) предложения состоит в извлечении его смысла и представлении в формальном виде
- **Пример:**

  «Who was the first person to walk on the moon?»

  Результат парсинга в виде SQL-запроса:

```
SELECT name FROM Person
WHERE moon_walk = true
ORDER BY moon_walk_date
FETCH first 1 rows only
```

---

https://medium.com/explorations-in-language-and-learning/trends-in-semantic-parsing-part-1-ba11888523cb

Подходы к решению

- В семантическом парсинге можно переводить предложения из естественного языка в формальный – это MT
- Тогда можно использовать весь инструментарий MT, который рассмотрен выше
- Подзадачей семантического парсинга является Semantic role labeling – выявление в тексте объектов, субъектов, действий и отношений
- Эта задача более простая, поскольку известна структура выявляемых сущностей
- Semantic role labeling можно рассматривать как стандартную задачу разметки последовательности, для которой хорошо подходят рекурентные и свёрточные сети