Задача преобразования последовательности. Машинный перевод.

Математические методы анализа текстов осень 2021

Попов Артём Сергеевич

19 октября 2021

Задача преобразования последовательности (Seq2Seq, sequence to sequence)

Дано D — множество пар последовательностей (x, y):

- $x = \{x_1, \ldots, x_n\}$, $x_i \in X$ — входная последовательность
- $y = \{y_1, \ldots, y_m\}$, $y_i \in Y$ — выходная последовательность

Необходимо по входной последовательности предсказать элементы выходной последовательности.

Отличия от задачи разметки:

- Длины x и y не совпадают
- Нет никаких заранее известных соответствий между x и y

Примеры задач преобразования последовательности

- Машинный перевод (machine translation)
- Абстрактная суммаризация — построение саммари по документу (abstractive summarization)
- Генеративная диалоговая система
- Преобразование текста на естественном языке в код (например, Text2SQL)
- Предсказание результата химической реакции
- Транскрибация аудио в текст

В сегодняшней лекции рассматриваем Seq2Seq на примере машинного перевода.

Задача машинного перевода

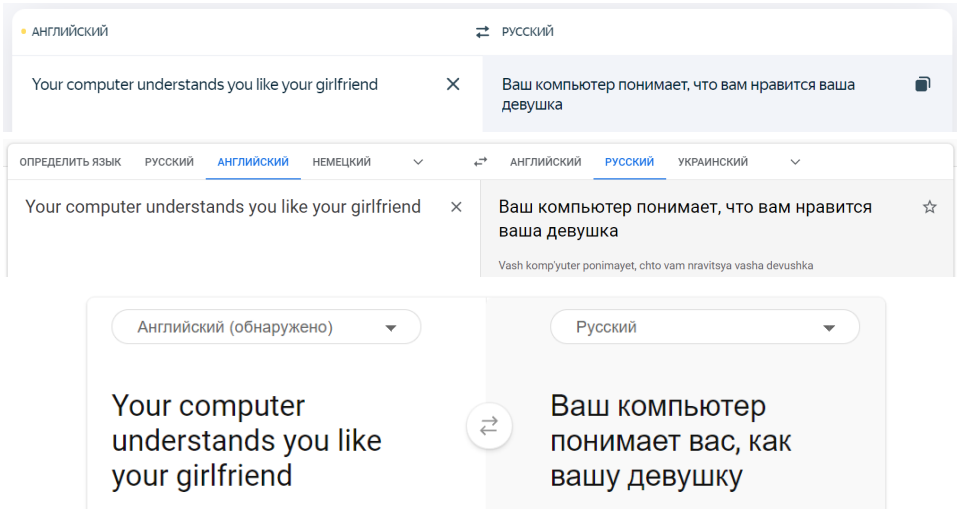
Как вы переведёте «Your computer understands you like your girlfriend»?

Задача машинного перевода

Как вы переведёте «Your computer understands you like your girlfriend»?

- Ваш компьютер понимает вас так же, как и ваша девушка.
- Ваш компьютер понимает вас так же, как и вашу девушку.
- Ваш компьютер понимает, что вам нравится ваша девушка.
- Твой компьютер понимает тебя так же, как и твоя девушка.

Что предлагают онлайн-переводчики?



Оценивание качества модели машинного перевода

Экспертная оценка — исходное предложение и перевод модели оцениваются специалистами по выбранной шкале:

+ Оценка очень точная
− Получать оценку дорого и медленно

Сравнение с правильным ответом — на тестовом корпусе сравниваем полученный ответ с одним из возможных переводов

+ Оценка получается быстро (если есть готовый корпус)
− Сложно построить корректный способ сопоставления результата модели с эталонным ответом

BLEU (bilingual evaluation understudy)

BLEU — метод сравнения последовательностей на основе пересечениях их n-грамм:

$$BLEU_N(\hat{y}, y) = BP(\hat{y}, y) * \exp\left(\frac{1}{N}\sum_{i=1}^{N}\log p_n(\hat{y}, y)\right)$$

p_n — число n-грамм в ответе модели, присутствующих в эталонном ответе (аналог точности)

Brevity Penalty — штраф за краткость (аналог полноты):

$$BP = min\left(1, \exp\left(1 - \frac{len(y)}{len(\hat{y})}\right)\right)$$

[1] Papineni et al; BLEU: a Method for Automatic Evaluation of Machine Translation (ACL 2002)

WER (word error rate)

WER — минимальное число операций, нужное для преобразования полученного перевода в правильный

Допустимые операции: замена, вставка, удаление слова

Значение рассчитывается по формуле

$$WER = \frac{\#\text{insetions} + \#\text{deletions} + \#\text{replacements}}{\#\text{words in translated sentence}}$$

Особенности оценивания машинного перевода

Особенности метрик BLEU и WER:

- \+ Легко считаются
- \+ Неплохо коррелируют с экспертными оценками
- – Оперируют короткими фрагментами, не оценивают общую корректность
- – Не позволяют оценить жанровую специфику
- – Не дифференцируемы (можно оптимизировать через RL)

Данные для машинного перевода

Обычно данные для NMT представляют собой наборы пар фрагментов на разных языках:

- пары предложений-переводов с выравниванием по словам
- просто пары предложений-переводов
- пары абзацев/документов-переводов

Примеры популярных мультиязычных корпусов (parallel corpus):

- Europarl: параллельные предложения на 21 языке
- Wikipedia: параллельные предложения на 20 языках
- Global Voices: параллельные тексты на 57 языках

История машинного перевода

- 1950–1960: rule-based подходы
- 1990–2010: статистический машинный перевод (SMT, statistical machine transltation)
- 2010–н.в.: нейросетевой машинный перевод (NMT, neural machine translation)

Современный машинный перевод хорош в ситуациях, где тексты формализованы или же достаточно грубого перевода.
С художественной литературой до сих пор всё плохо.

Статистический машинный перевод[1][2]

Для перевода используется модель шумного канала:

$$\hat{y} = \arg\max_y p(y|x) = \arg\max_y p(x|y)p(y)$$

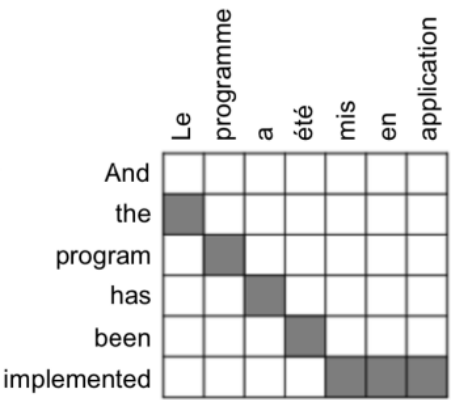
$p(y)$ — языковая модель

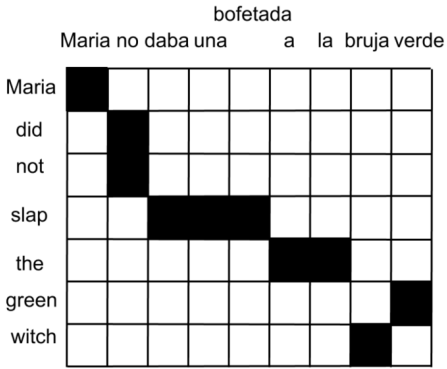
$p(x|y)$ — модель перевода (translation model), оценивается при помощи скрытых переменных выравниваний (alignments)

[1] Brown et al. The mathematics of statistical machine translation: Parameter estimation (Computational linguistics 1993)

[2] Collins. Statistical Machine Translation: IBM Models 1 and 2 (2011)

Примеры выравниваний





Архитектура кодировщик-декодировщик (encoder-decoder)

Задача решается методом максимизации правдоподобия:

$$\log p_\theta(y|x) = \log \prod_{i=1}^{m} p_\theta(y_i|x, y_{<i}) = \sum_{i=1}^{m} \log p_\theta(y_i|x, y_{<i}) \to \max_\theta$$

Кодировщик получает на вход последовательность входных элементов x и генерирует вектор контекста h_n.

Декодировщик по уже сгенерированным токенам и вектору контекста итеративно генерирует следующие токены.

Архитектуры кодировщика и декодировщика могут не совпадать.

[1] Cho et al; Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation (EMNLP 2014)

[2] Sutskever et al; Sequence to Sequence Learning with Neural Networks (NIPS 2014)

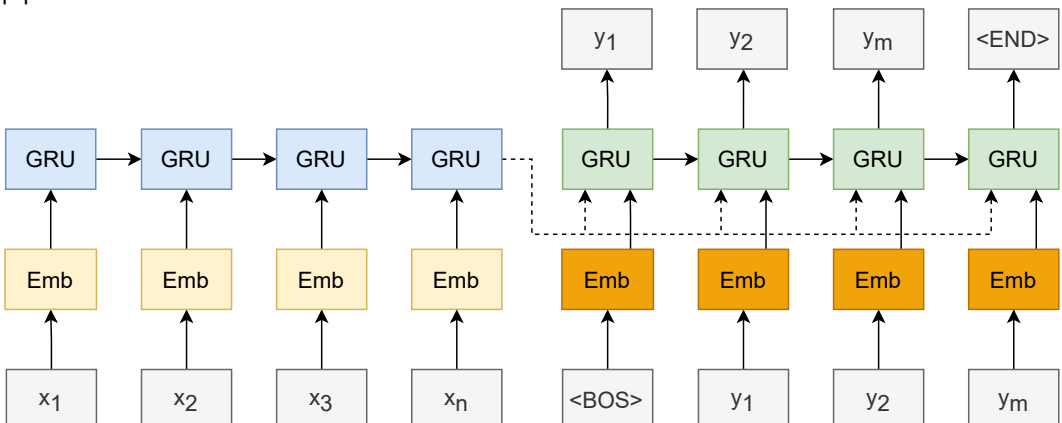
Кодировщик-декодировщик на основе RNN

Кодировщик и декодировщик можно задать рекуррентными сетями (например, GRU).

- Входные слова каждой из сетей кодируются эмбеддингом
- Вектор контекста может использоваться при вычислении первого скрытого состояния декодировщика (по обычным формулам GRU)
- Вектор контекста можно конкатенировать с предыдущим состоянием декодировщика перед пересчётом скрытого состояния декодировщика
- Выходная последовательность дополняется <BOS> на старте и <EOS> токеном на конце

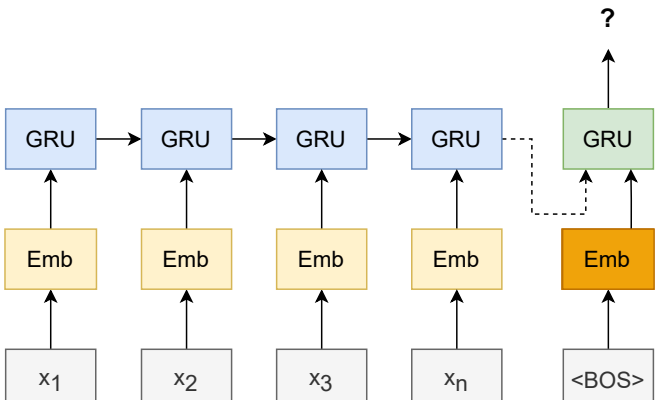
Обучение кодировщика

Для n = 4 и m = 2:



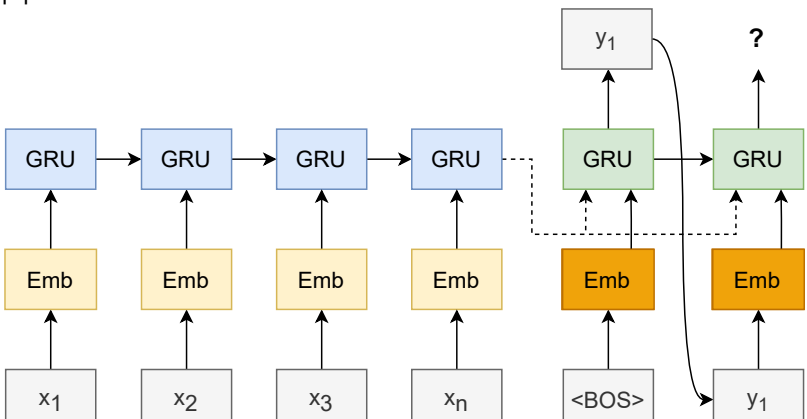
Применение декодировщика

Для n = 4 и m = 2:



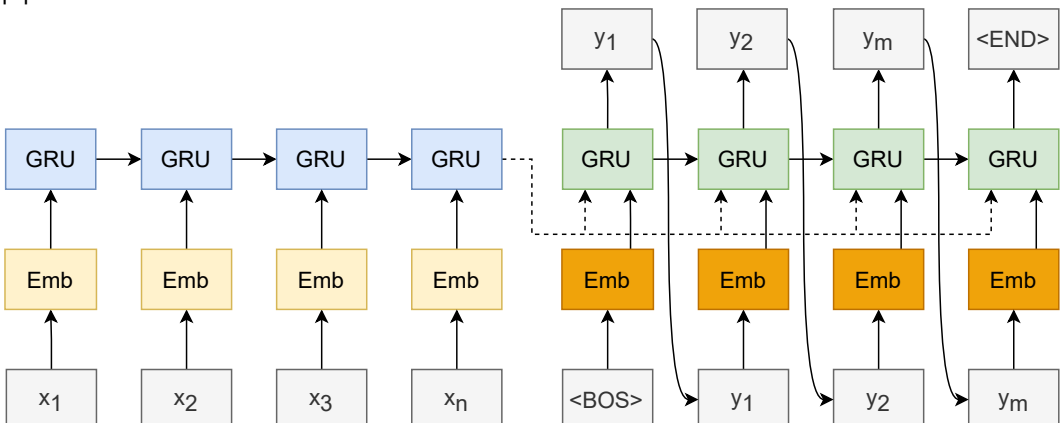
Применение декодировщика

Для n = 4 и m = 2:



Применение декодировщика

Для n = 4 и m = 2:



Разница в обучении и применении

Обучение

По последовательности $[x_1, \ldots, x_n, \texttt{<BOS>}, y_1, \ldots, y_n]$
восстанавливаем последовательность $[y_1, \ldots, y_n, \texttt{<EOS>}]$.

Применении

По последовательности $[x_1, \ldots, x_n, \texttt{<BOS>}, \hat{y}_1, \ldots, \hat{y}_k]$
предсказываем следующий токен $\hat{y}_{k+1}$, пока не получим <EOS>
или не превысим заданное максимальное число токенов.

В каких ситуациях разница повлияет на качество?

Разница в обучении и применении

Обучение

По последовательности $[x_1, \ldots, x_n, \texttt{<BOS>}, y_1, \ldots, y_n]$
восстанавливаем последовательность $[y_1, \ldots, y_n, \texttt{<EOS>}]$.

Применении

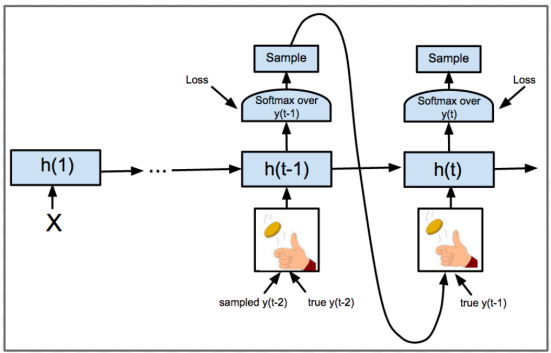
По последовательности $[x_1, \ldots, x_n, \texttt{<BOS>}, \hat{y}_1, \ldots, \hat{y}_k]$
предсказываем следующий токен $\hat{y}_{k+1}$, пока не получим <EOS>
или не превысим заданное максимальное число токенов.

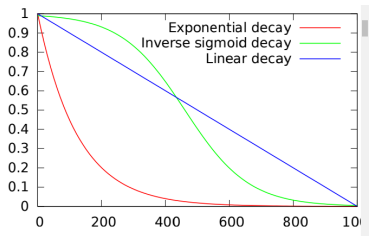
В каких ситуациях разница повлияет на качество?

Плохо генерируем слово для плохо сгенерированного предложения

Scheduled Sampling

Выбираем с вероятностью ϵ_i истинное слово, иначе сгенерированное:





ϵ_i убывает с течением итераций по одному из трёх законов.

[1] Bengio et al; Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks (2015)

Трюки при генерации

Все трюки, обсуждавшиеся для авторегрессионной генерации языковыми моделями, работают и для этой архитектуры!

Трюки при генерации

Все трюки, обсуждавшиеся для авторегрессионной генерации языковыми моделями, работают и для этой архитектуры!

- кэширование весов модели
- softmax с температурой
- topK и topP сэмплирование
- beam-search

Ещё один трюк: penalized sampling

Генеративные модели могут зацикливаться: повторять один и тот же токен много раз.

Если q_u — логиты модели на i-ой позиции соответствующие u-му токену, то вероятность пересчитывается так:

$$Z_u = \begin{cases} \theta, \text{ если } u \in \{\hat{y}_1, \ldots, \hat{y}_{i-1}\} \\ 1, \text{ иначе} \end{cases}$$

$$p(y_i = u) = \mathrm{softmax}(q_u / Z_u)$$

Обратите внимание, что q_u должны быть одного знака.

Проблемы архитектуры кодировщик-декодировщик

Узкое место всей архитектуры — вектор h_n:

- в векторе h_n необходимо закодировать всю информацию о входной последовательности
- h_n лучше помнит конец последовательности, чем начало
- чем длиннее исходное предложение, тем сложнее уместить его смысл в h_n
- чем больше токенов было сгенерировано, тем сложнее хранить информацию о входной последовательности

Как решать эту проблему?

Проблемы архитектуры кодировщик-декодировщик

Узкое место всей архитектуры — вектор h_n:

- в векторе h_n необходимо закодировать всю информацию о входной последовательности
- h_n лучше помнит конец последовательности, чем начало
- чем длиннее исходное предложение, тем сложнее уместить его смысл в h_n
- чем больше токенов было сгенерировано, тем сложнее хранить информацию о входной последовательности

Как решать эту проблему? При декодировании хотим уметь заглядывать в любое место входной последовательности

Механизм внимания (attention mechanism) в RNN

- При декодировании модель может подсмотреть в каждое из внутренних состояний кодировщика
- Влияние состояния кодировщика h_i на значение состояния декодировщика q_j зависит от значения близости $sim(h_i, z_{j-1})$

Обратите внимание. Идея очень похожа на идею выравниваний из статистического машинного перевода.

Общий механизм внимания

Дано: вектор запроса q, вектора контекста $c_1, \ldots, c_n$
Хотим пересчитать q, используя релевантный контекст

Общая формула внимания:

$$q_{new} = \sum_{i=1}^{n} norm(sim(Query(q), Key(c_i)))Value(v_i)$$

где $Query$, Key, $Value$ — преобразования вектора в вектор (могут иметь параметры и быть обучаемыми), sim — функция близости, $norm$ — функция нормировки по элементам контекста

Механизм внимания в RNN для машинного перевода

Строим эмбеддинги входных слов и пропускаем их через GRU:

$$v_i = Emb_e(x_i), \quad h_i = GRU(v_i, h_{i-1})$$

Строим эмбеддинги выходных слов и пропускаем их через GRU. Учитываем вектор контекста c_j (механизм внимания) для пересчёта состояния GRU:

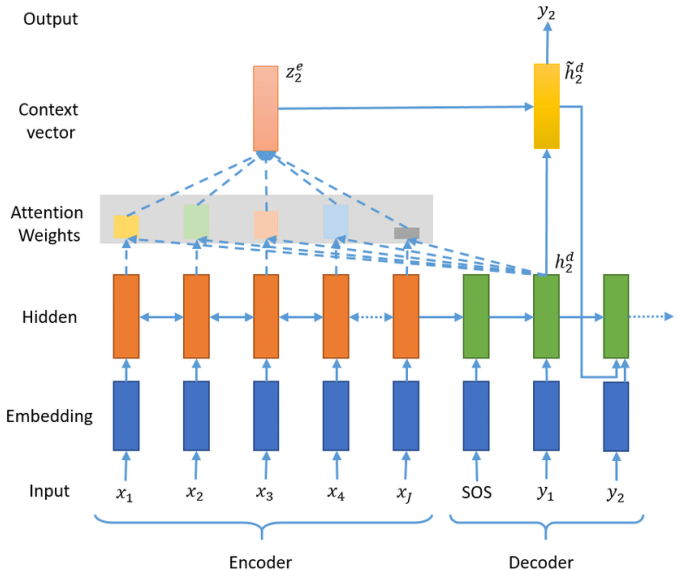
$$e_j = Emb_d(y_j), \quad z_j = GRU(e_j, [z_{j-1}, c_j])$$

$$\alpha_{ij} = \underset{i \in \{1,\dots,n\}}{\mathrm{softmax}}(sim(h_i, z_{j-1})), \quad c_j = \sum_{i=1}^{n} \alpha_{ij} h_i$$

Вероятность выхода также вычисляется с использованием c_j:

$$p(y_j = u | x, y_{<j}) = \underset{u \in Y}{\mathrm{softmax}}((W[c_j, z_j] + b)_u)$$

Механизм внимания в RNN для машинного перевода



Функция близости для внимания

- Скалярное произведение:

$$\mathrm{sim}(h_i, h_j) = h_i^T h_j$$

- Аддитивное внимание:

$$\mathrm{sim}(h_i, h_j) = w^T \tanh(W_{h_i} h_i + W_{h_j} h_j)$$

- Мультипликативное внимание:

$$\mathrm{sim}(h_i, h_j) = h_i^T W h_j$$

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

Вариации при работе с вниманием

Где использовать внимание?

- Для пересчёта следующего скрытого состояния
- Для вычисления вероятностей выходного слова

Что подавать в функцию близости?

- Выходы с любого слоя кодировщика
- Эмбеддинги входных слов

По каким позициям вычислять внимание?

- Global Attention — внимание по всем входным словам
- Local Attention — предсказываем центральную позицию внимания и работаем с словами из фиксированного окна

[1] Luong et al; Effective Approaches to Attention-based Neural Machine Translation (2015)

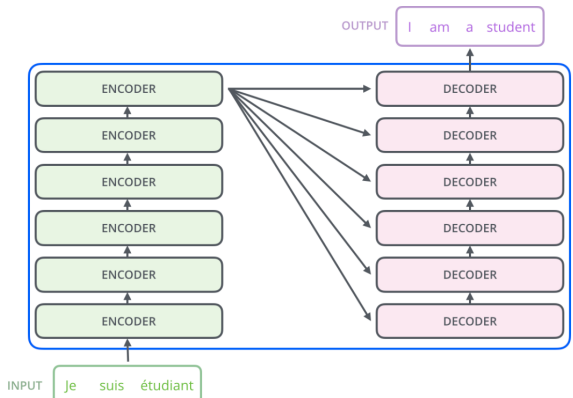
Недостатки модели RNN с вниманием

- Плохо распараллеливается и при обучении, и при применении
- Из-за затухания/взрыва градиентов есть сильные ограничения по количеству используемых слоёв

Идея. Избавиться от рекуррентности и создать модель, полностью основанную на внимании.

Модель трансформер для машинного перевода

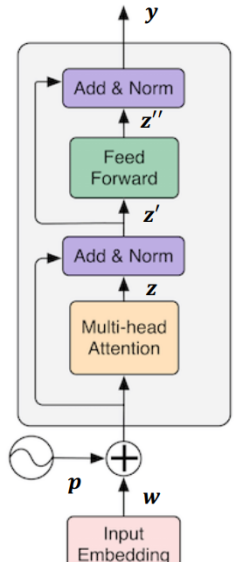
- Кодировщик и декодировщик состоят из своих наборов одинаковых блоков, блоки стекаются друг за другом.
- По-умолчанию, веса у каждого блока свои (неразделяемые).



Кодировщик трансформера: напоминание

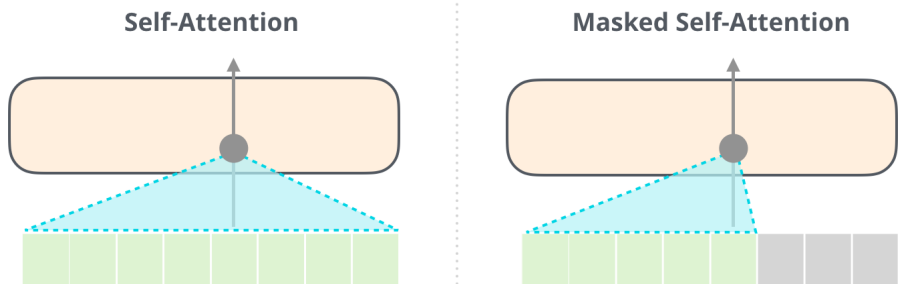
1. Перед первым слоём складываем представления токенов и позиций
$$x_i = Emb(w_i) + p_i$$
2. Применяем MHSA, l – номер слоя
$$z = MHSA(x; \theta_l)$$
3. Residual связи + нормализация слоя
$$z'_i = LN(z_i + x_i; \mu_1, \sigma_1)$$
4. Дополнительные Feed-Forward слои
$$z''_i = RELU(z'_i V_1 + b_1)V_2 + b_2$$
5. Residual связи + нормализация слоя
$$y_i = LN(z''_i + z'_i; \mu_2, \sigma_2)$$

На остальных слоях повторяем шаги 2-6.



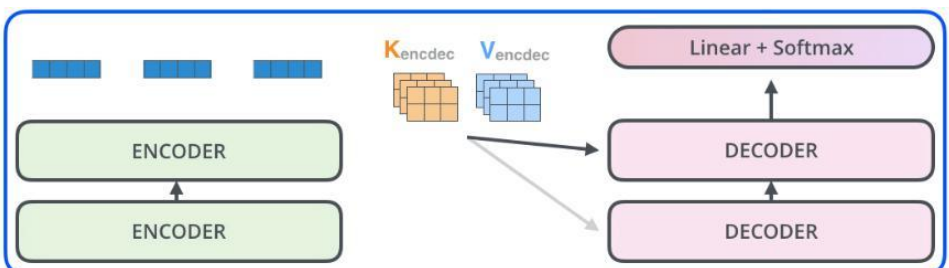
Декодировщик трансформера: masked self-attention

Внимание в декодировщике трансформера учитывает только предыдущие токены:



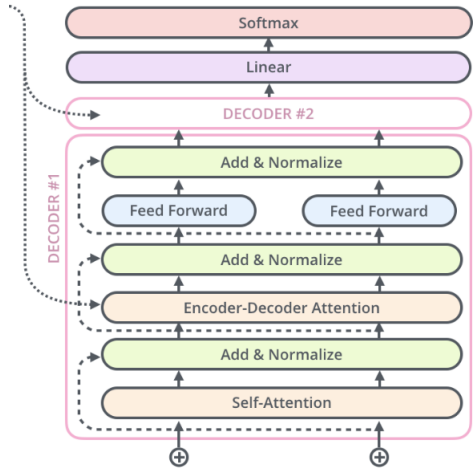
Декодировщик трансформера: связь с кодировщиком

- Декодировщик состоит из последовательных блоков-декодировщиков
- Выходы кодировщика преобразовываются обучаемыми весовыми матрицами в набор матриц `Key` и `Value`
- Эти матрицы передаются в каждый из блоков-декодировщиков



Архитектура декодировщика

1. Выходы первого слоя MHSA идут во второй — Encoder-Decoder Attention
2. Encoder-Decoder Attention — MHSA выходов первого слоя по выходам кодировщика:
   - `Key` и `Value` — выходы кодировщика
   - `Query` — первый слой декодировщика
3. На выходе блока — набор векторов, соответствующих токенам входной последовательности



Что ещё следует помнить про трансформеры?

- Обычно, мы работаем с BPE токенами на входе и выходе.
- Кодировщик и декодировщик не обязаны быть одного размера.
- При обучении можно использовать Adam, обычно используют warm-up расписание для темпа обучения.
- Сложность трансормера квадратичная как по длине последовательности, так и по размеру внутреннего слоя.
- Позиционные эмбеддинги могут быть фиксированными или обучаемыми. Также, их можно сделать относительными.

Неавторегрессионный машинный перевод

- До сих пор рассматривался авторегрессионный подход — генерация текущего токена зависит от предыдущих:

$$p(\mathrm{y}|\mathrm{x};\theta) = \prod_{j=1}^{m} p(\mathrm{y}_j|\mathrm{y}_{<j},\mathrm{x};\theta)$$

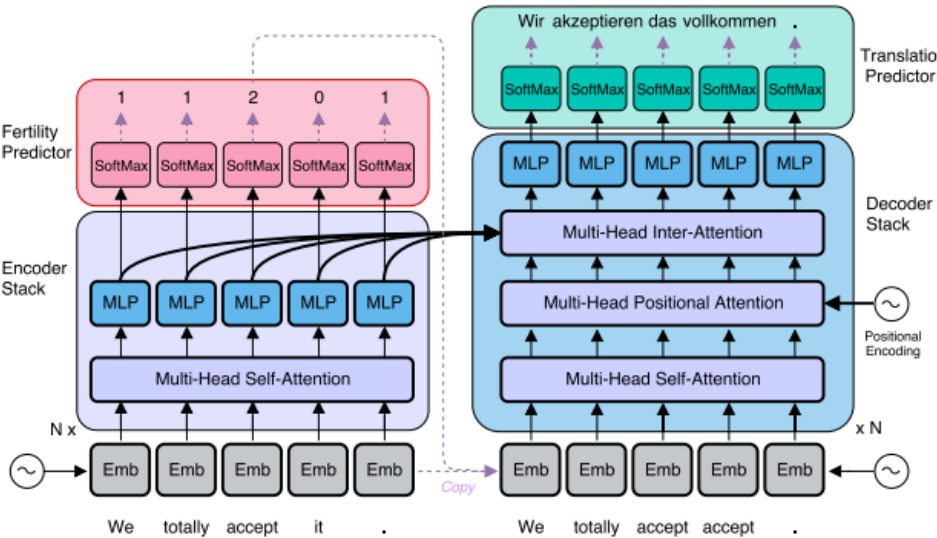
- Неавторегрессионный перевод предполагает параллельную генерацию всех токенов целевой последовательности
- Базовый вариант — модель без зависимостей выходов:

$$p(\mathrm{y}|\mathrm{x};\theta) = p(m|\mathrm{x};\theta)\prod_{j=1}^{m} p(\mathrm{y}_j|\mathrm{x};\theta)$$

Non-Autoregressive Transformer

- Сопоставим каждому токену входной последовательности целое число fertility — сколько токенам выходной последовательности он соответствует
- Будем предсказывать эти числа на выходе кодировщика
- Каждый токен исходной последовательности будем подавать на вход декодеровщика N раз (его значение fertility)
- В блок-декодировщика между двумя слоями self-attention добавляется + один:
  - Value — выходы первого слоя внимания
  - Key и Query — позиционные эмбеддинги входной последовательности
- Декодировщик является двунаправленным, но есть маскирование токена от самого себя.

Non-Autoregressive Transformer



Машинный перевод без учителя

Параллельные данные можно сгенерировать, для этого нужны:

- Алгоритм перевода между парой языков для некоторого числа популярных слов
- Языковая модель для целевого языка
- Векторные представления слов для обоих языков

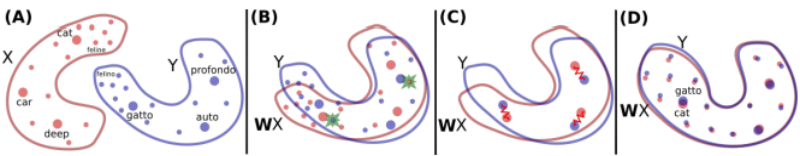
Основная идея:

- Составляем автоматический словарь между двумя языками (актуальный и устойчивый к опечаткам)
- Для исходного текста генерируем варианты переводов по токенам
- С помощью языковой модели выбираем наиболее вероятный перевод на целевом языке

Построение автоматического словаря

Обучаются поворот пространства векторов слов целевого языка и его наложение на пространство векторов исходного языка:

- Построение векторных пространств для каждого языка
- Выбор опорных точек — пар слов с известным переводом
- Поворот и растяжение пространства для совпадения опорных точек
- Растяжение плотных областей вокруг частых слов



[1] A. Conneau et al; Word Translation Without Parallel Data (ICLR 2018)

Резюме по лекции

- Машинный перевод — одна из флагманских задач обработки текстов, многие методы NLP появились в процессе её решения
- Текущим стандартом является архитектура Seq2Seq на основе Transformer, можно использовать RNN с вниманием
- Transformer сейчас одна из наиболее сильных и универсальных моделей, но обучается сложно, важны технические детали
- Обычно перевод авторегрессионный, но возможны и другие парадигмы
- Для обучения нужны параллельные корпуса, иногда их можно генерировать автоматическими методами