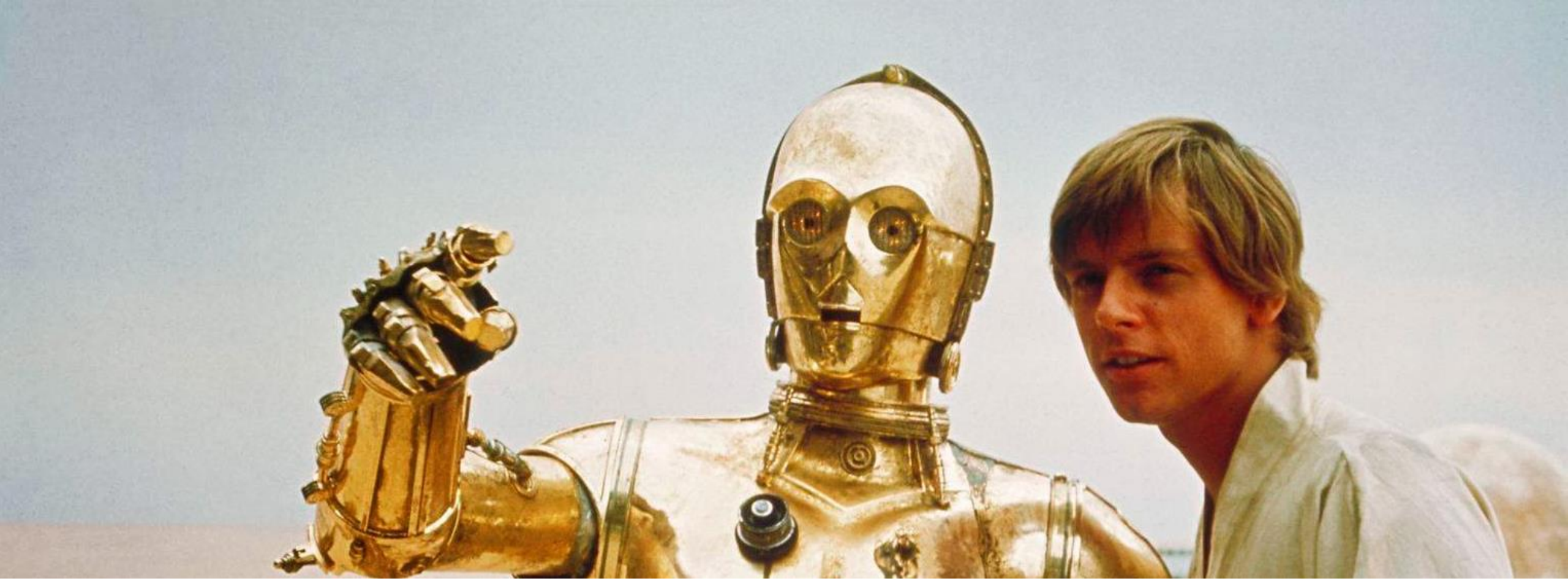


Задача преобразования последовательности. Машинный перевод. Трансформеры.

Попов Артём, OzonMasters, осень 2022

Natural Language Processing

Задача машинного перевода

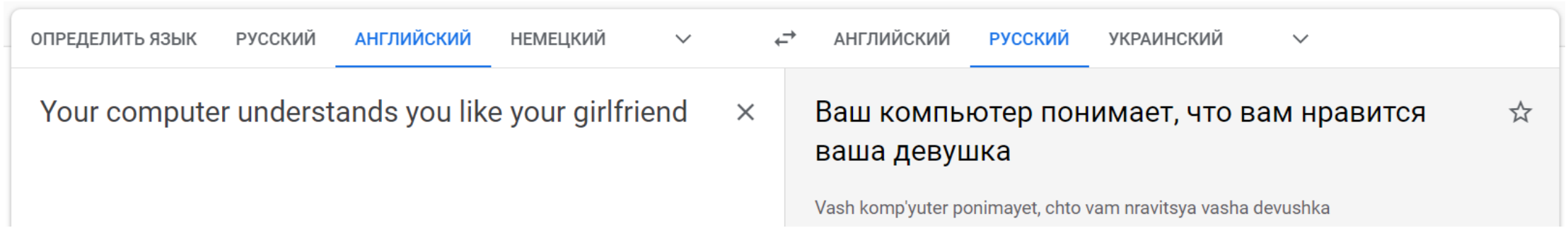
Как вы переведёте
«Your computer understands you like your girlfriend»?

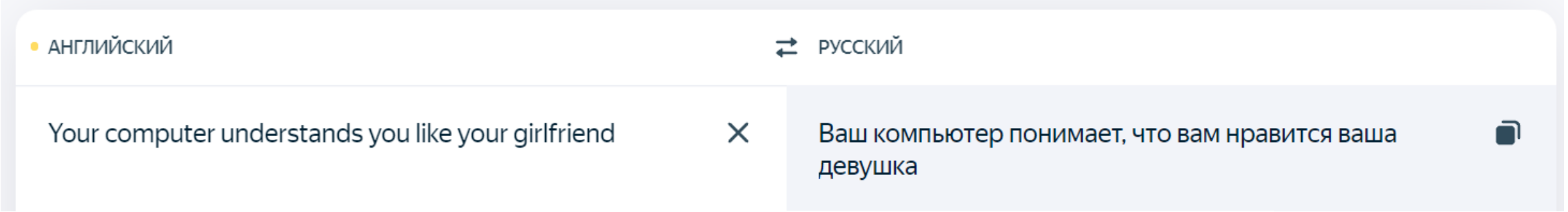
Задача машинного перевода

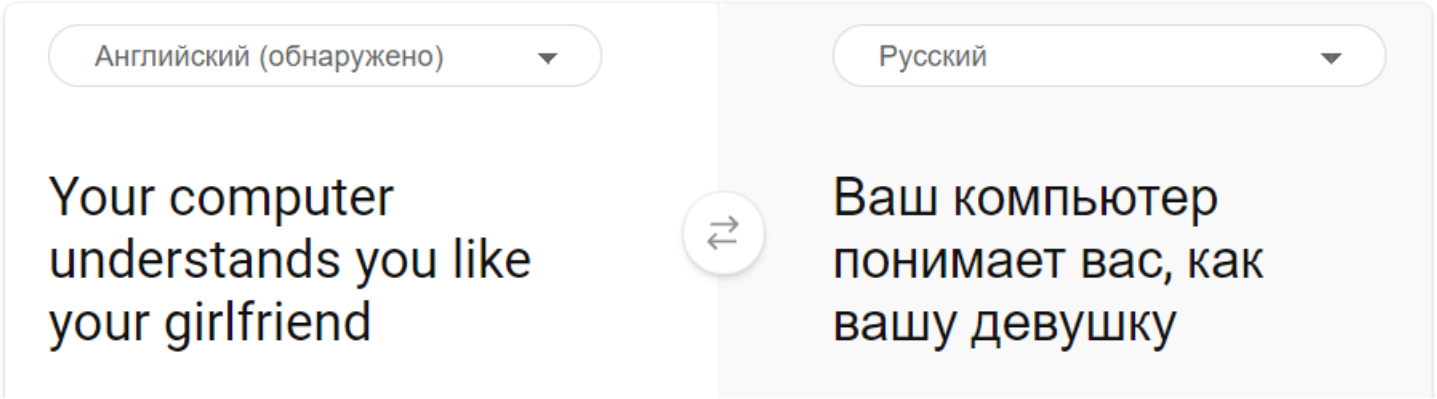
Как вы переведёте
«Your computer understands you like your girlfriend»?

- 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?







Задача преобразования последовательности

Дано 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.

1. Длины x и y не совпадают
2. Нет никаких известных связей между элементами x и y

Другие названия: sequence to sequence (Seq2Seq)

Примеры задач преобразования последовательности

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

В сегодняшней лекции рассматриваем Seq2Seq на примере машинного перевода.

Оценивание качества машинного перевода

Экспертная оценка — исходное предложение и ответ, полученный моделью 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 ответов

+ Оценка получается быстро

– Сложно сопоставлять результата модели с эталонным ответом

Оценивание через другие модели – измерение сторонних характеристик ответа (например, грамматической согласованности слов в переводе)

+ Оценка получается быстро

– Оценка не всегда коррелирует с экспертной

BLEU (bilingual evaluation understudy)

BLEU – метод сравнения последовательностей на основе пересечения их n-грамм:

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

p_n – доля n-грамм в ответе модели, присутствующих в эталонном ответе (аналог точности)

Brevity penalty – штраф за краткость (аналог полноты):

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

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

Пример вычисления BLEU3

Эталон: В среду вечером я отправил письмо

Модель: Я отправил письмо в четверг

$$p_1(\hat{y}, y) = \frac{4}{5}, \qquad p_2(\hat{y}, y) = \frac{2}{4}, \qquad p_3(\hat{y}, y) = \frac{1}{3}$$

$$\mathrm{B}P(\hat{y}, y) = \min\left(1, \exp\left(1 - \frac{6}{5}\right)\right) = \min\left(1, \exp\left(-\frac{1}{5}\right)\right) \approx 0.819$$

$$BLEU_3(\hat{y}, y) = 0.819 \times \exp\left(\frac{1}{3}\left(\log\frac{4}{5} + \log\frac{2}{4} + \log\frac{1}{3}\right)\right) \approx 0.418$$

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:

$$WER(\hat{y}, y) = \frac{\#insertions + \#deletions + \#replacements}{\#words\ in\ translated\ sentence}$$

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 (но можно оптимизировать через RL)

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

Для перевода используется модель шумного канала:

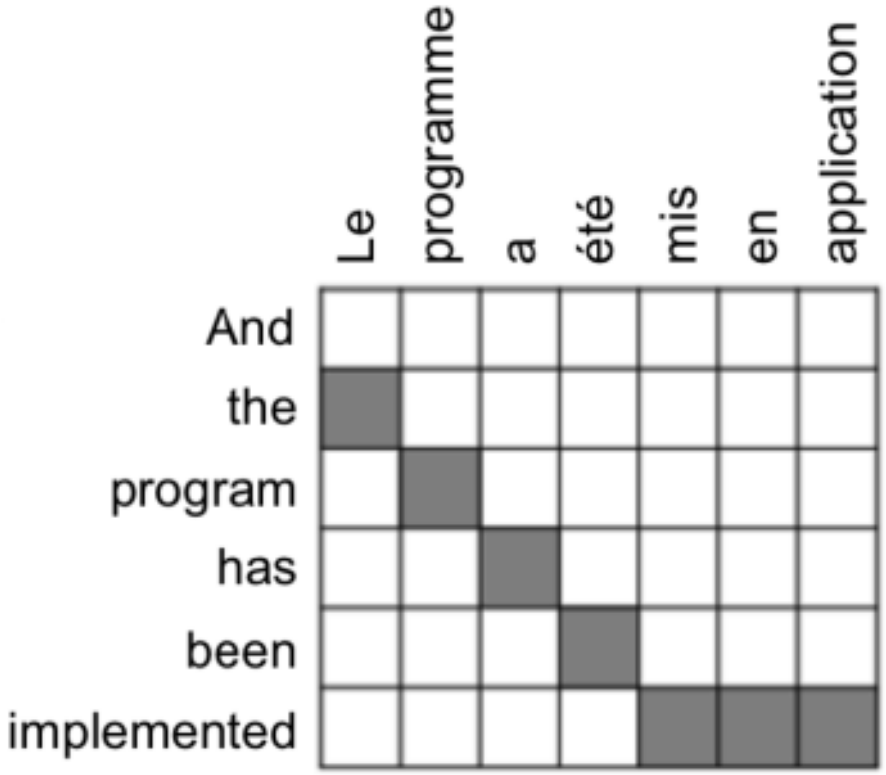
$$\hat{y} = \arg\max_y p(y|x) = \arg\max_y p(x|y)p(y)$$

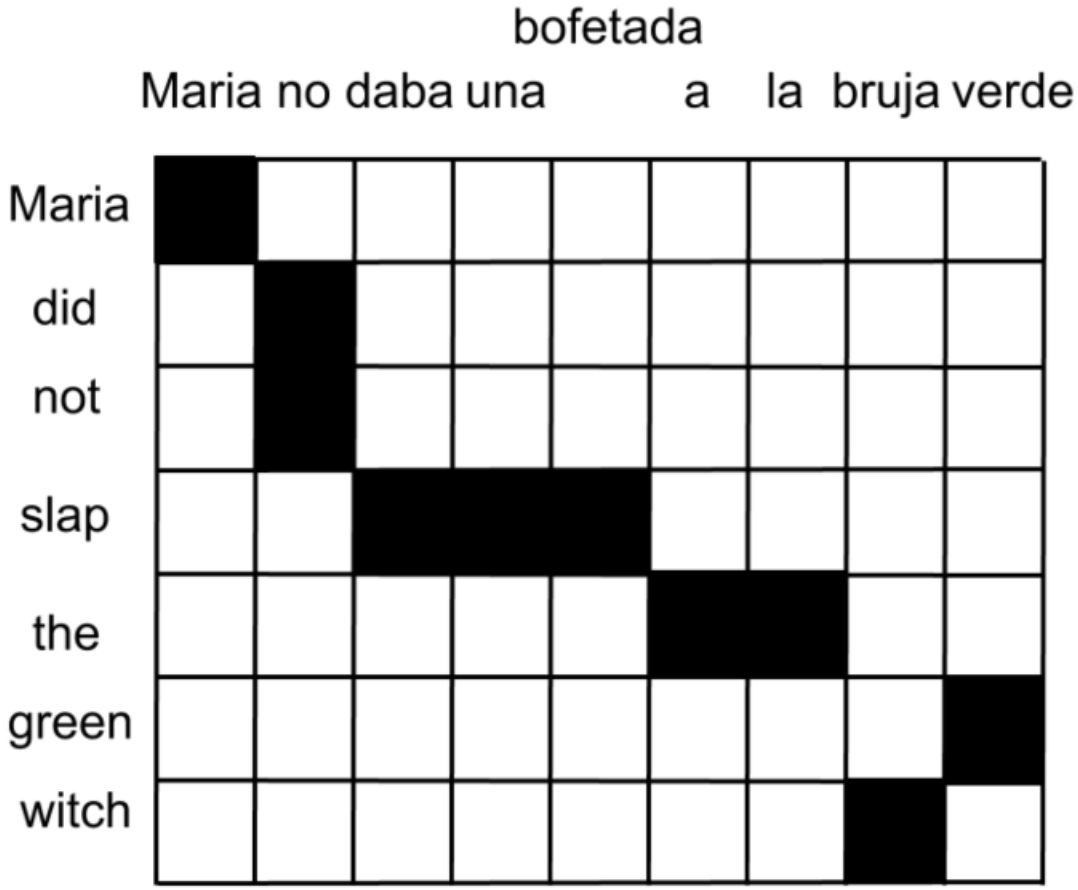
$p(y)$ – насколько y естественна для языка перевода (оценивается через языковую модель, следующая лекция)

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

Brown et al (1993). The mathematics of statistical machine translation: Parameter estimation
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.

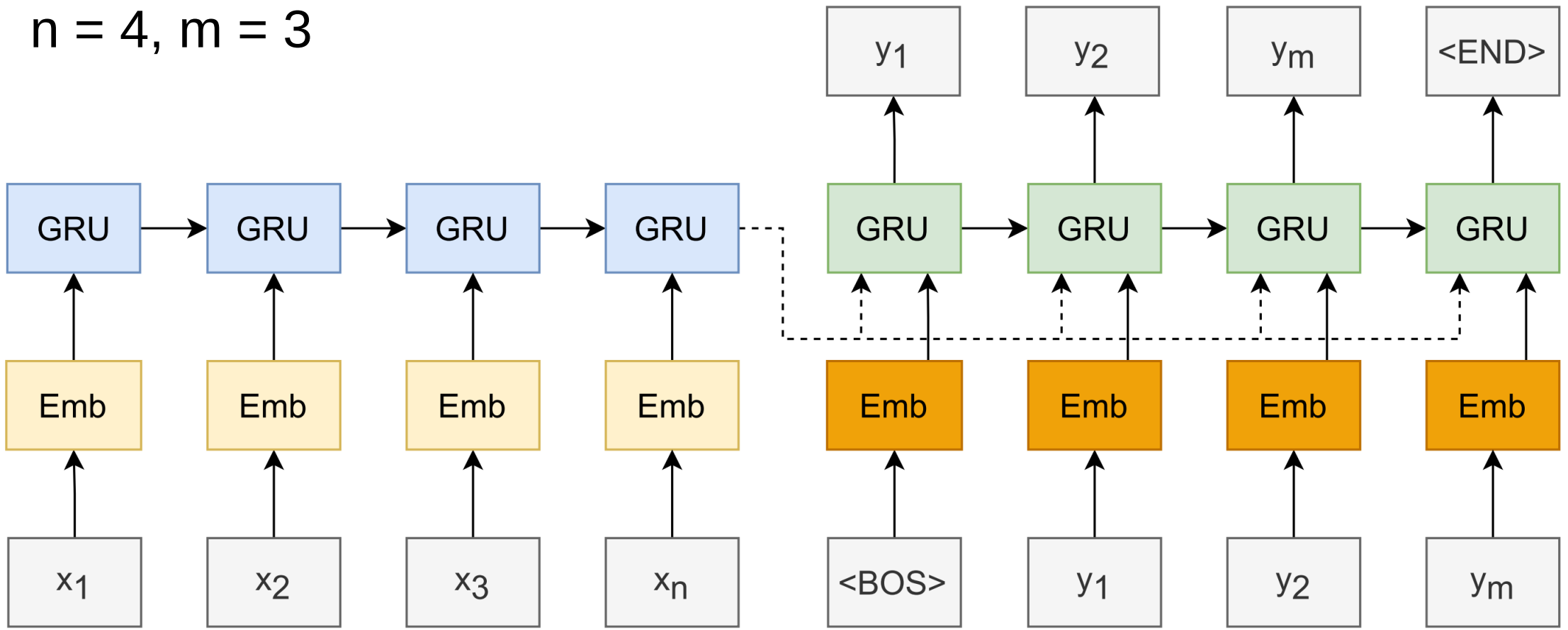
Sutskever et al (NIPS 2014); Sequence to Sequence Learning with Neural Networks
Cho et al (EMNLP 2014); Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

Кодировщик-декодировщик на основе RNN

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

- Входные слова каждой из сетей кодируются эмбеддингом
- Вектор контекста h_n можно конкатенировать с предыдущим состоянием декодировщика z_{i-1} перед пересчётом скрытого состояния декодировщика z_i
- Вектор контекста можно использовать при вычислении первого скрытого состояния декодировщика
- Выходная последовательность дополняется специальными токенами $y_0 = < START >$ и $y_{m+1} = < END >$
- Декодировщик не может быть двунаправленным

Обучение кодировщика-декодировщика



Применение кодировщика-декодировщика

Для получения таргет-последовательности для нового объекта, необходимо применять архитектуру итеративно:

Дано: $x = (x_1, \dots, x_n), y_0 = < START >$

$h_i = GRU_{enc}(Emb(x_i), h_{i-1}), \qquad i \in \{1, \dots, n\}$

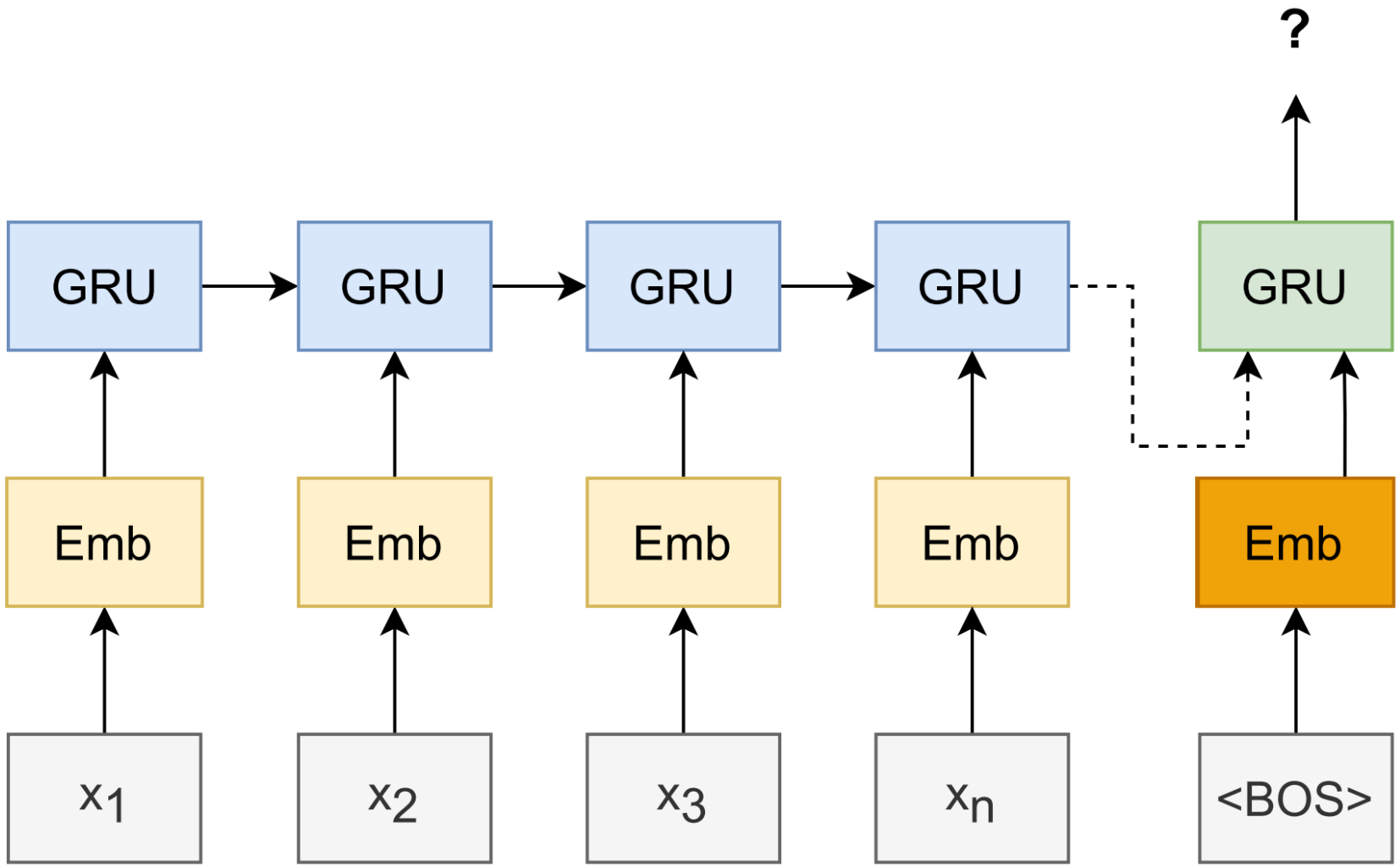
Пока $\hat{y}_j \neq < END >$ **или** $j \neq M$:

$$z_j = GRU_{dec}(Emb(y_j), [z_{j-1}, h_n])$$

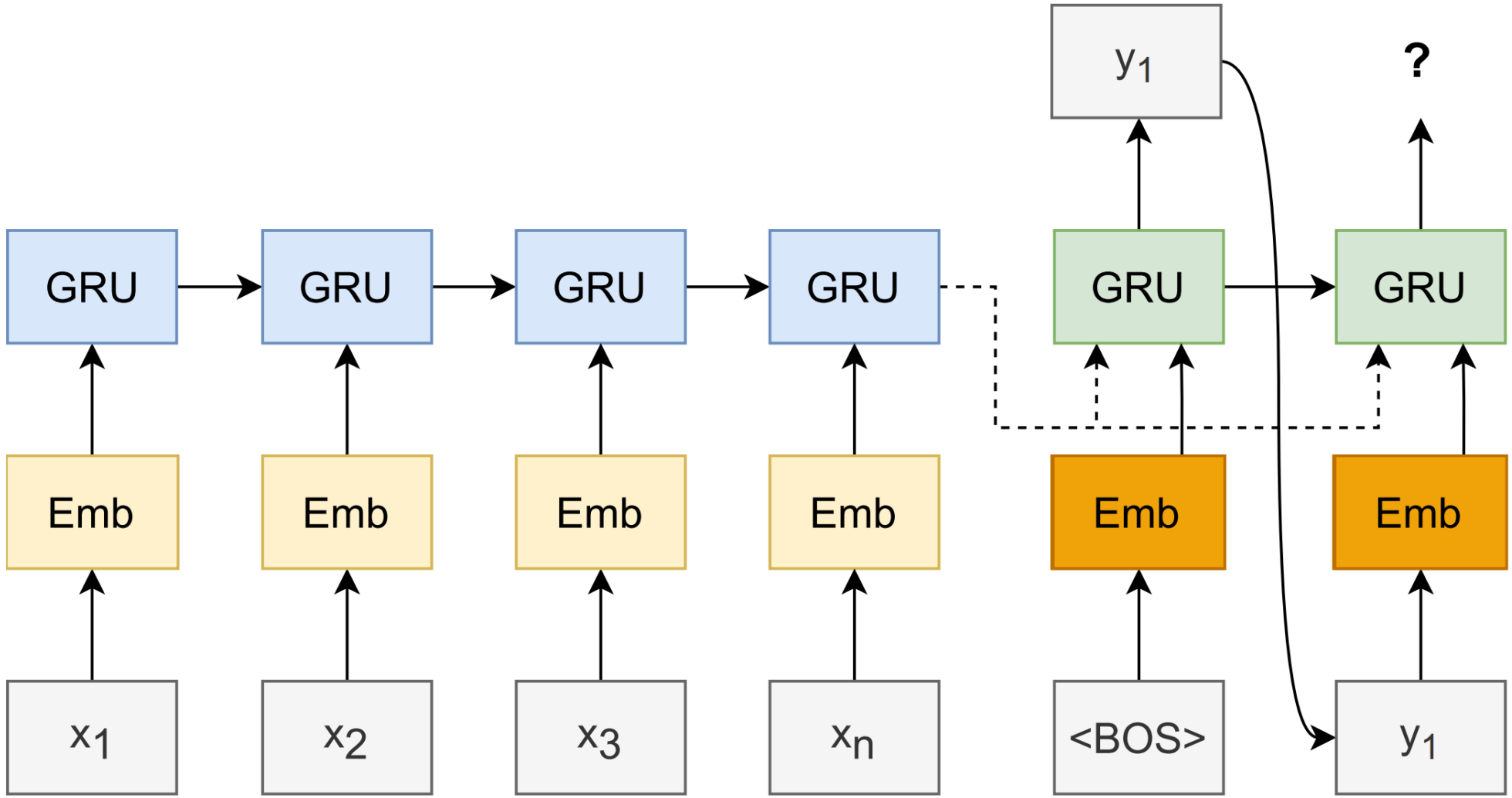
$$\hat{y}_{j+1} = \arg\max_{y \in Y}(Uz_j + b)$$

где M – максимальная длина выходной последовательности (гиперпараметр, заданные пользователем)

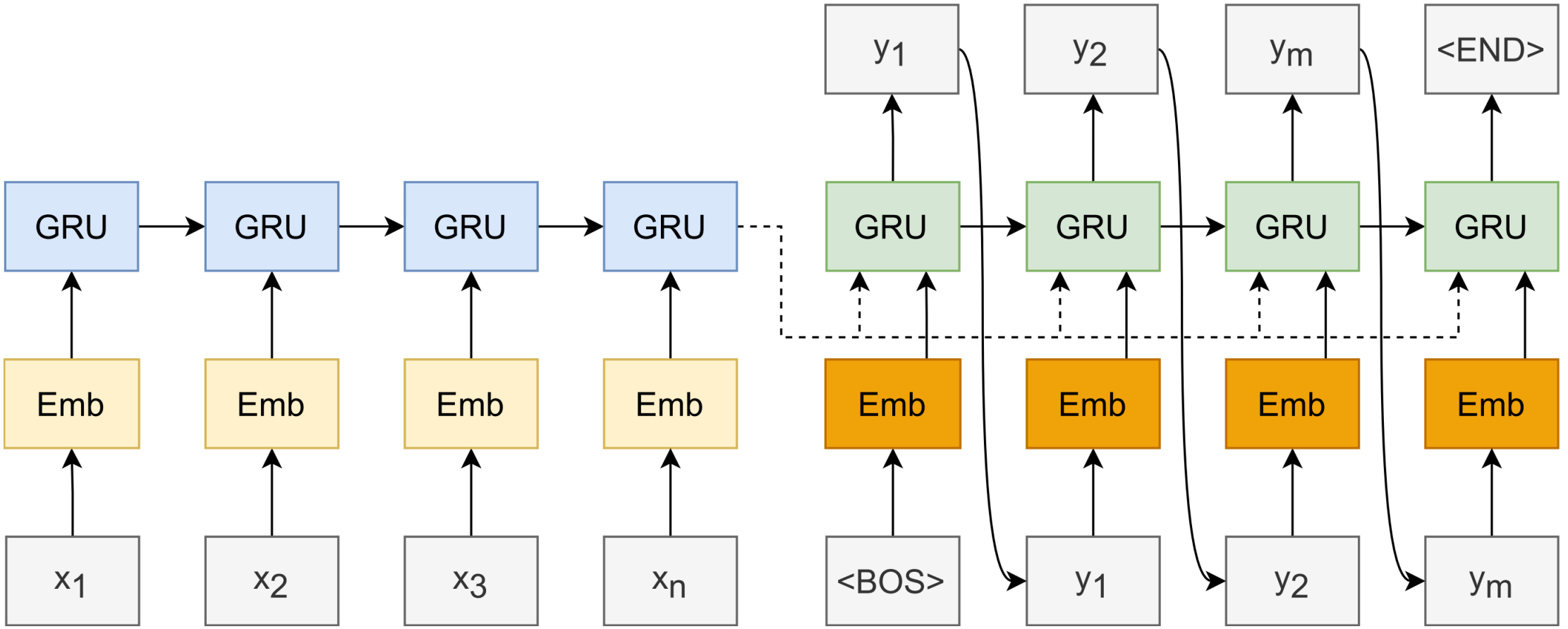
Применение кодировщика-декодировщика



Применение кодировщика-декодировщика



Применение кодировщика-декодировщика



Разница в обучении и применении

Обучение
По последовательности $[x_1, \dots, x_n, y_0, \dots, y_m]$ восстанавливаем последовательность $[y_1, \dots, y_{m+1}]$

Применении
По последовательности $[x_1, \dots, x_n, y_0, \hat{y}_1, \dots, \hat{y}_j]$ предсказываем следующий токен $\hat{y}_{j+1}$, пока не получим $< END >$ или не превысим заданное максимальное число токенов

В каких ситуациях разница повлияет на качество?

Разница в обучении и применении

Обучение
По последовательности $[x_1, \dots, x_n, y_0, \dots, y_m]$ восстанавливаем последовательность $[y_1, \dots, y_{m+1}]$

Применении
По последовательности $[x_1, \dots, x_n, y_0, \hat{y}_1, \dots, \hat{y}_j]$ предсказываем следующий токен $\hat{y}_{j+1}$, пока не получим $< END >$ или не превысим заданное максимальное число токенов

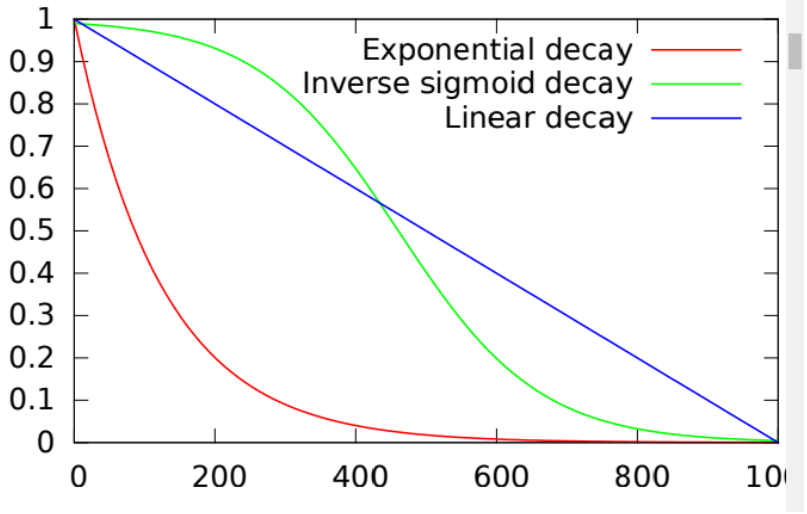
В каких ситуациях разница повлияет на качество?

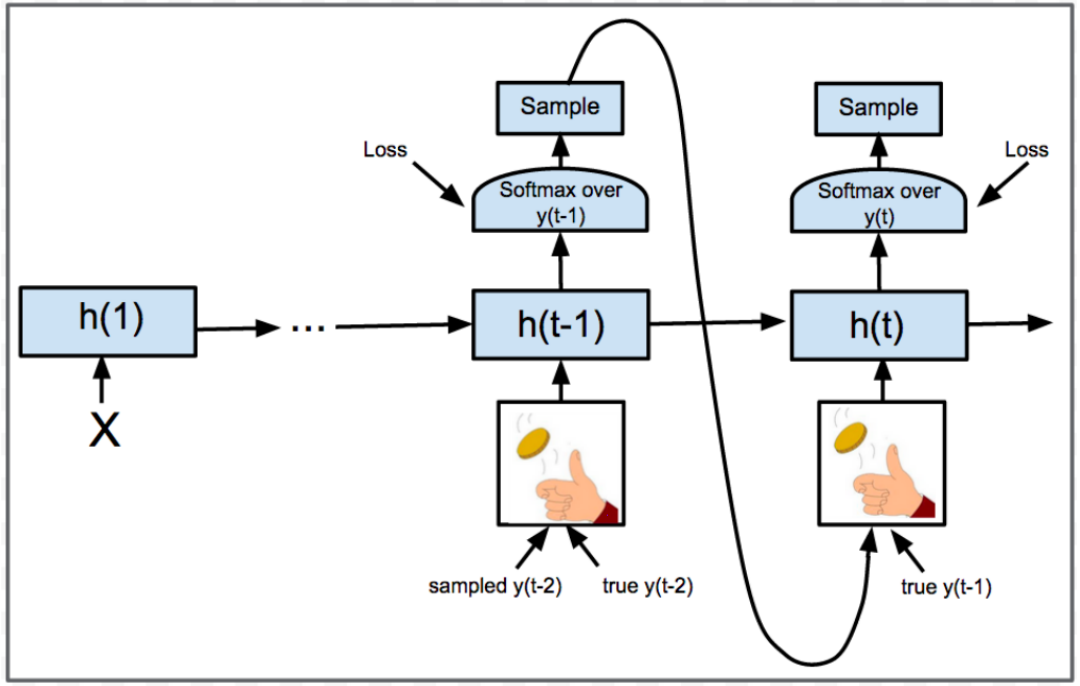
Плохо генерируем следующее слово для плохо сгенерированного предложения

Scheduled sampling

С вероятностью ϵ_{epoch} выбираем истинное слово, иначе сгенерированное.

ϵ_{epoch} убывает с течением итераций по одному из трёх законов:





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

Трюки при применении

На этапе применения модели обычно применяется много различных трюков и эвристик.

На следующих лекциях:

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

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
- если h_n используется только для инициализации: чем больше токенов 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
- если h_n используется только для инициализации: чем больше токенов сгенерировано, тем сложнее хранить информацию о входной последовательности

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

Общий механизм внимания

Дано: вектор запроса q, вектора контекста $c_1, \dots, c_n$

Хотим обновить q, используя релевантный контекст

Общая формула внимания:

$$Attn(q, c) = \sum_{i=1}^{n} norm\left(sim\big(Query(q), Key(c_i)\big)\right) Value(c_i)$$

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 (attention mechanism) в RNN

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

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

Механизм внимания в RNN для машинного перевода

Строим эмбеддинги входных слов и пропускаем их через GRU:

$$v_i = Emb_{enc}(x_i), \qquad h_i = GRU_{enc}(v_i, h_{i-1})$$

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

$$e_j = Emb_{dec}(y_j), \qquad z_j = GRU_{dec}(e_j, [z_{j-1}, c_j])$$

$$\alpha_{ij} = softmax_{i\in\{1,\dots,n\}}\left(sim(h_i, z_{j-1})\right), \qquad c_j = \sum_{i=1}^{n} \alpha_{ij} h_i$$

Вероятность выхода тоже можно вычислять с использованием c_j:

$$p(y_j = y | x, y_{<j}) = softmax_{y\in Y}(W[c_j, z_j] + b)$$

Общая формула внимания для RNN

Внимание в RNN – частный случай общего механизма внимания:

$$Attn(q, c) = \sum_{i=1}^{n} norm\left(sim\big(Query(q), Key(c_i)\big)\right) Value(c_i)$$

$q = z_j$ – состояние декодировщика

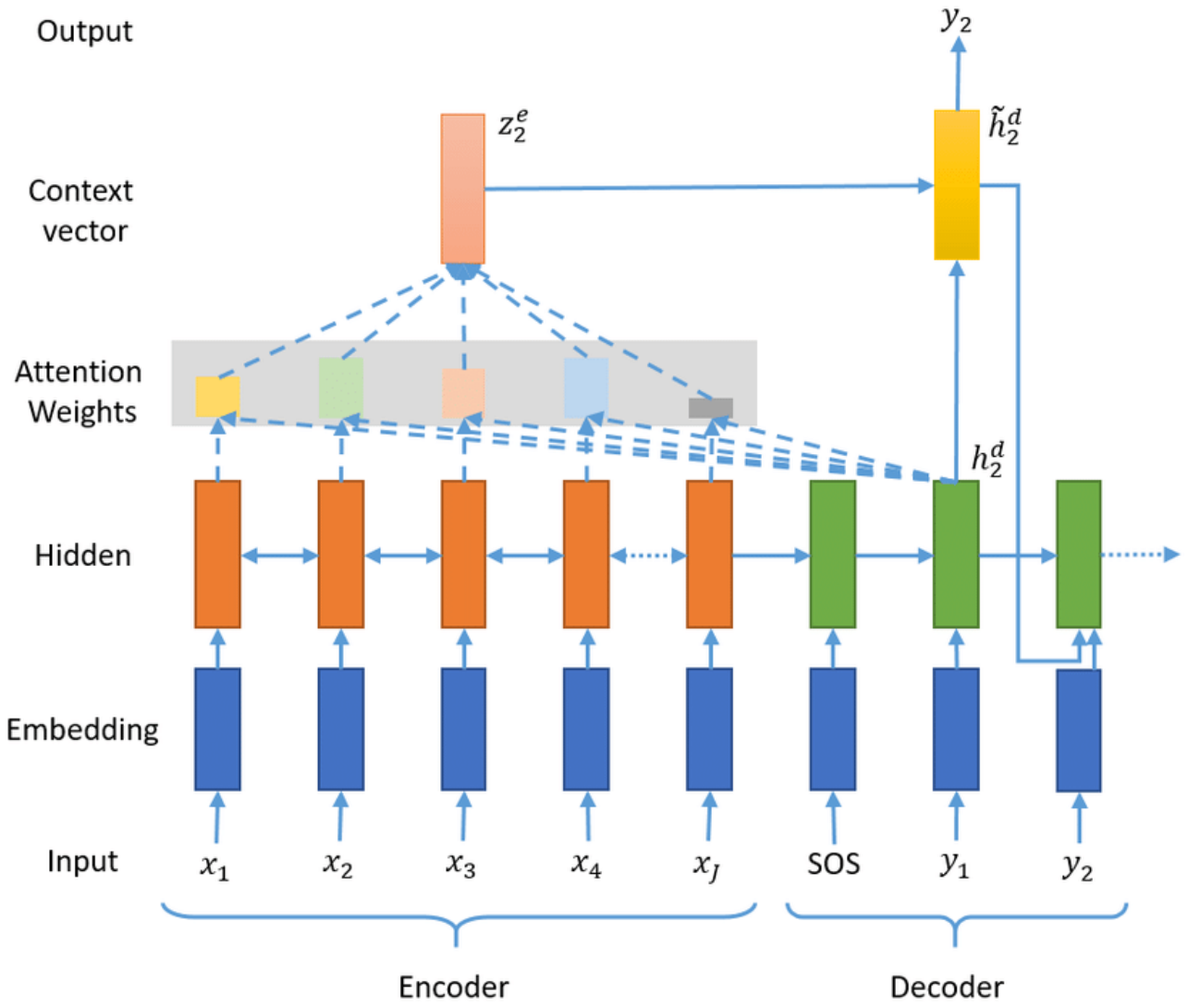
$\mathrm{c} = (h_1, \ldots, h_n)$ – выходы кодировщика

$Query(x) = Key(x) = Value(x) = x$

$norm(\alpha_i) = softmax_{i \in \{1,..,n\}}(\alpha)$

$sim(q, k) = \langle q, k \rangle$

Механизм внимания в RNN для машинного перевода



Функция близости для внимания

Скалярное произведение:

$$sim(h_i, z_j) = \langle h_i, z_j \rangle$$

Аддитивное внимание:

$$sim(h_i, z_j) = \langle w, \tanh(W h_i + U z_j) \rangle$$

Мультипликативное внимание:

$$sim(h_i, z_j) = \langle h_i, W z_j \rangle$$

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

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 (transformer)

Идея: в каждой позиции хотим давать сети информацию обо всех элементах последовательности.

Составляющие трансформера:

- механизм self-attention
- позиционные представления (positional encoding)
- нормализация

Сначала рассмотрим устройство кодировщика трансформера.

Vaswani et al (2017); Attention is all you need

Механизм self-attention (SA)

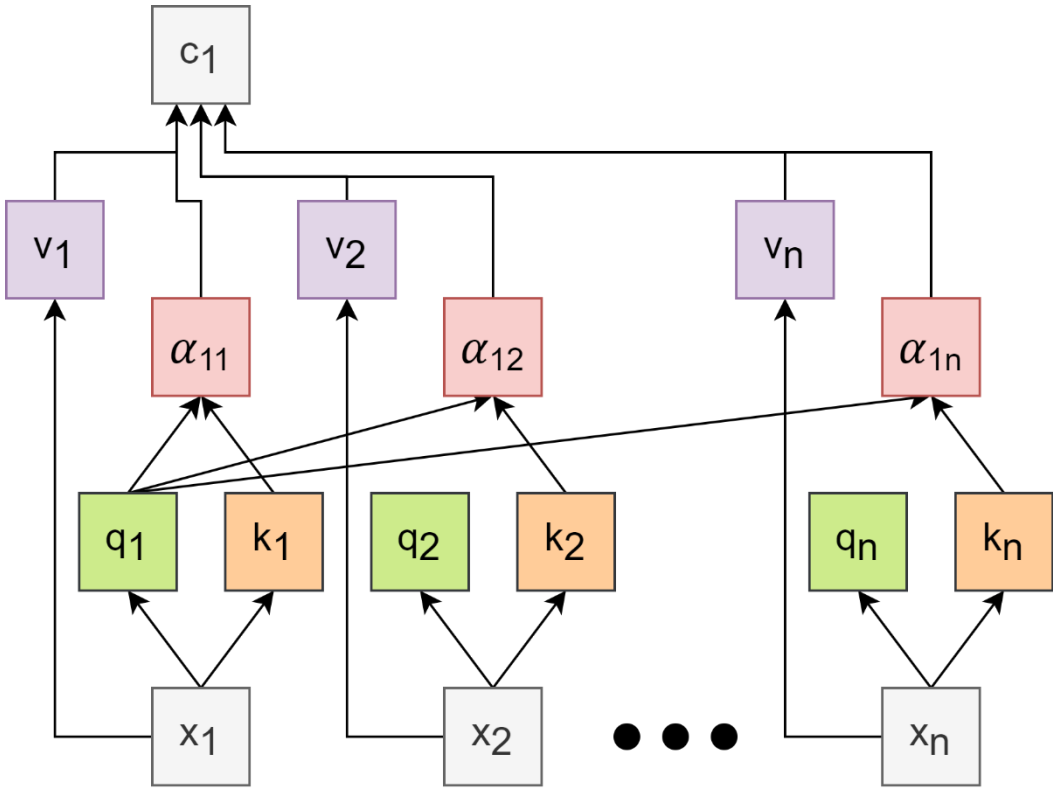
Вход:

последовательность $(x_1, \dots, x_n)$, $x_i \in \mathbb{R}^d$

Выход:

последовательность $(c_1, \dots, c_n)$, $c_i \in \mathbb{R}^m$

Параметры слоя:
матрицы преобразования
$W_k, W_q, W_v \in \mathbb{R}^{d \times m}$



Алгоритм работы self-attention (SA)

1. Переводим каждый элемент входа в три эмбеддинга более низкой размерности: запрос, ключ, значение.

$$q_i = W_q x_i, \qquad k_i = W_k x_i, \qquad v_i = W_v x_i$$

2. Считаем близости между “запросами” и “ключами”

$$sim(q_i, k_j) = \frac{\langle q_i, k_j \rangle}{\sqrt{m}}, \qquad \alpha_{ij} = \underset{j \in \{1,\dots,n\}}{softmax}\, sim(q_i, k_j) = \frac{exp\left(sim(q_i, k_j)\right)}{\sum_{s=1}^{n} exp\left(sim(q_i, k_s)\right)}$$

3. Вычисляем выпуклую комбинацию значений v

$$c_i = \sum_{j=1}^{n} \alpha_{ij} v_j\,, \qquad c = SA(x; W_q, W_k, W_v)$$

Алгоритм работы multi-head self-attention (MHSA)

Вход: последовательность $(x_1, \dots, x_n),\ x_i \in \mathbb{R}^d$

Выход: последовательность $(y_1, \dots, y_n),\ y_i \in \mathbb{R}^d$

Параметры слоя (θ): N преобразований $W_k^j, W_q^j, W_v^j \in \mathbb{R}^{d \times m}$, линейное преобразование $W \in R^{Nm \times \mathrm{d}}$

1. Вычисляем по всем наборам параметров self-attention
   $$c^j = SA\left(x; W_k^j, W_q^j, W_v^j\right)$$

2. Конкатенируем и пропускаем через ещё один слой
   $$y_i = MHSA(x;\theta)_i = \left[c_i^1, \dots, c_i^N\right] W$$

MHSA на практике

Часто, выбирают $N = d \,/\, m$. Какая вычислительная сложность модели в этом случае?

MHSA на практике

Часто, выбирают $N = d \,/\, m$. Какая вычислительная сложность модели в этом случае?

$O\left(n^2 d + n d^2\right)$

- Из-за квадратичной по длине последовательности сложности трансформер обычно применяется к последовательностям небольшой длины (≤ 512).

- В последние годы выпущено много модификаций архитектуры для обработки длинных последовательностей (возможно, обсудим в конце семестра)

Позиционные представления

Проблема. Механизм self-attention никак не учитывает порядок элементов в последовательности.

Решение. Добавить информацию о порядке при помощи позиционных эмбеддингов.

Способы реализации

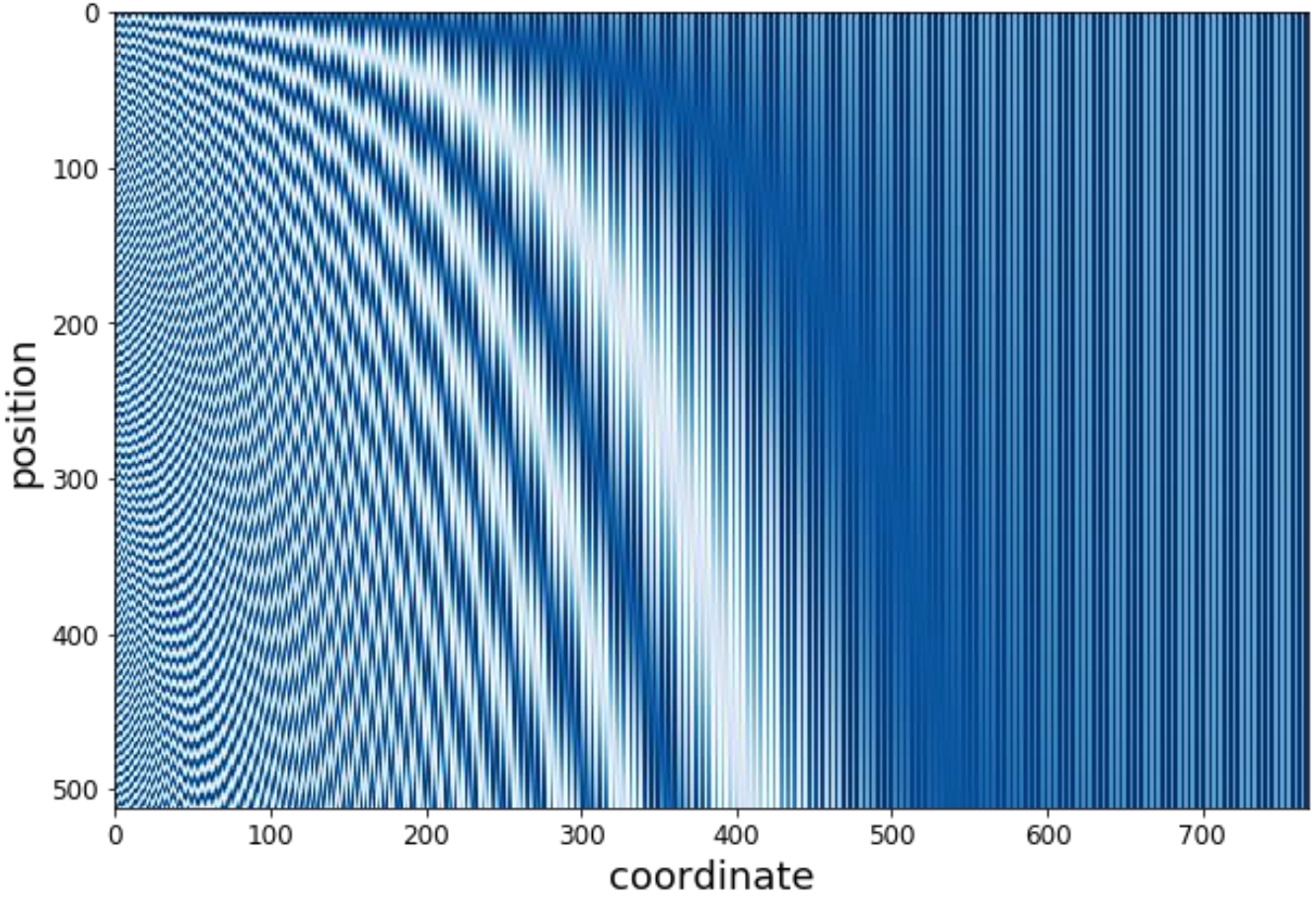
- фиксированные позиционные эмбеддинги
- обучающиеся позиционные эмбеддинги
- относительные позицонные эмбеддинги (relative encoding)

Фиксированные позиционные представления

Для i-ой позиции позиционный эмбеддинг можно задать так:

$$p_i = \begin{pmatrix} \sin(w_1 i) \\ \cos(w_1 i) \\ \cdots \\ \sin(w_{d/2} i) \\ \cos(w_{d/2}) \end{pmatrix}$$

$$w_j = \frac{1}{10000^{2j/d}}$$



Интуиция фиксированных представлений

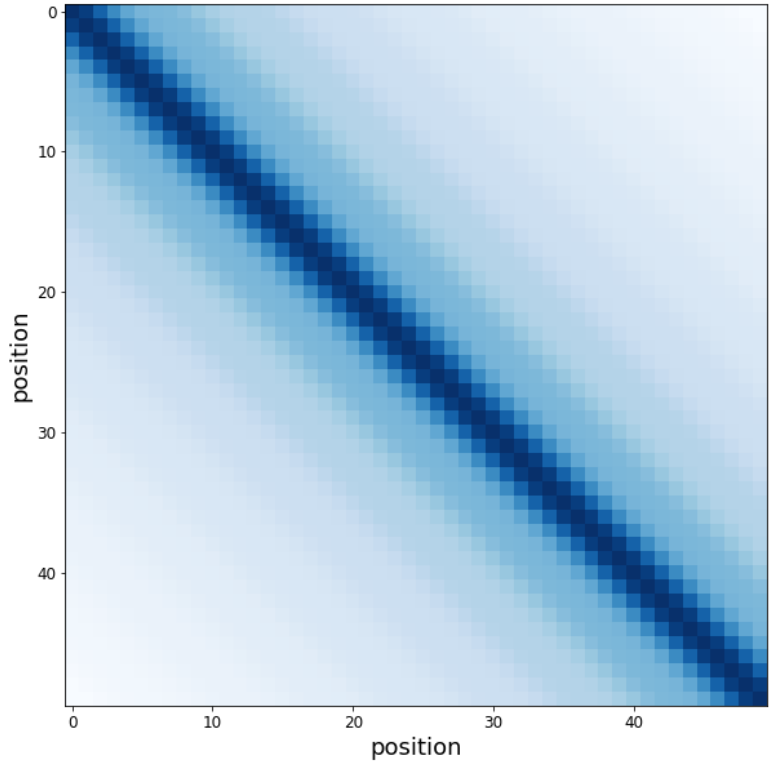
Теорема. Для любых векторов p_i и p_{i+k} верно $E_k p_i = p_{i+k}$.

$$E_k = \begin{pmatrix} \Phi_1^k & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \Phi_{d/2}^k \end{pmatrix}$$

$$\Phi_m^k = \begin{pmatrix} \cos(\lambda_m k) & \sin(\lambda_m k) \\ -\sin(\lambda_m k) & \cos(\lambda_m k) \end{pmatrix}$$

$$\lambda_m = 10000^{-2m/d}$$

Матрица Грама для позиционных эмбеддингов



Обучающиеся позиционные представления

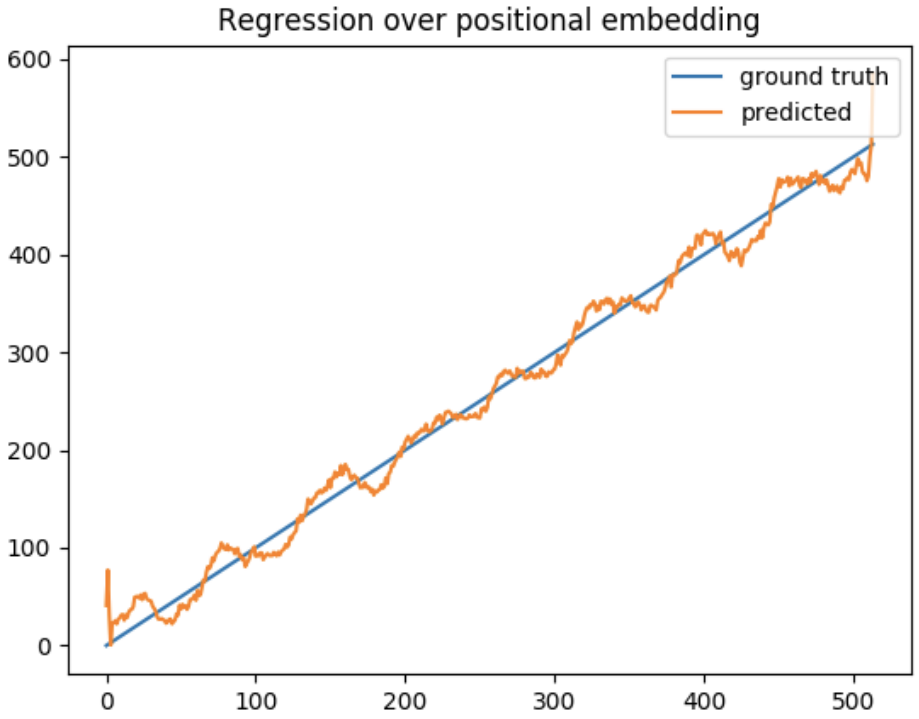
Идея. Вместо использования фиксированного вектора, будем учить представление для каждой позиции.

Плюсы и минусы

– Больше параметров в модели

\+ Можем выучить некоторые нетривиальные позиционные зависимости

Результат предсказания позиции по обученному позиционному эмбеддингу



Относительные позиционные представления

Эмбеддинг может зависеть от разности позиций элементов:

$$sim(q_i, k_j) = \frac{\langle q_i, k_j + W_p^k p_{ij} \rangle}{\sqrt{d}}, \qquad p_{ij} = Embedding(i - j)$$

$$c_i = \sum_{j=1}^{n} \alpha_{ij}(v_j + W_p^v p_{ij})$$

Плюсы и минусы:

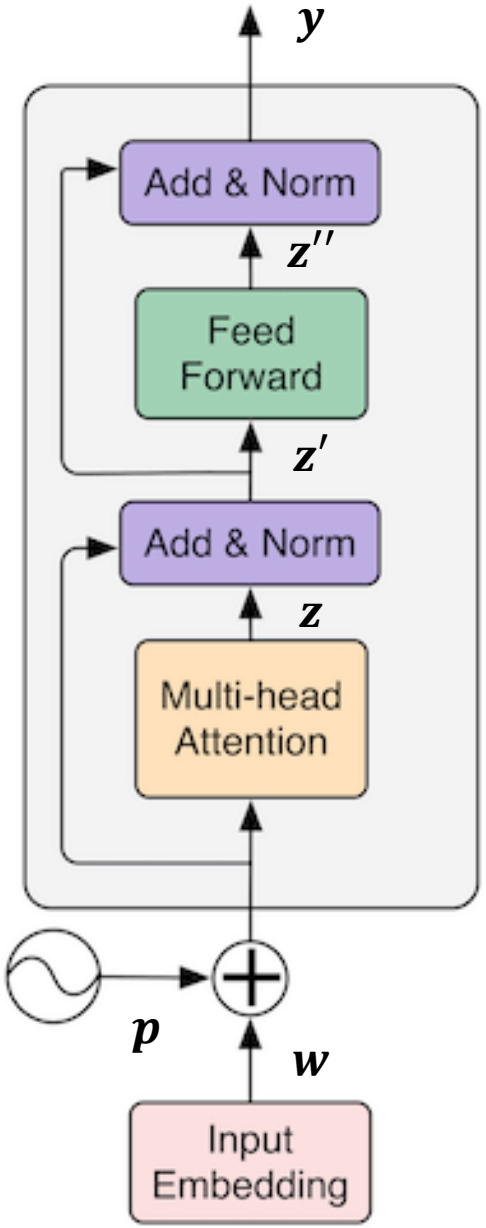
– Требует большего количества операций

\+ Можно моделировать сложные взаимоотношения (матрица смежности графа)

Энкодер трансформера

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

На остальных слоях повторяем шаги 2-6.



Layer normalization (нормализация слоя)

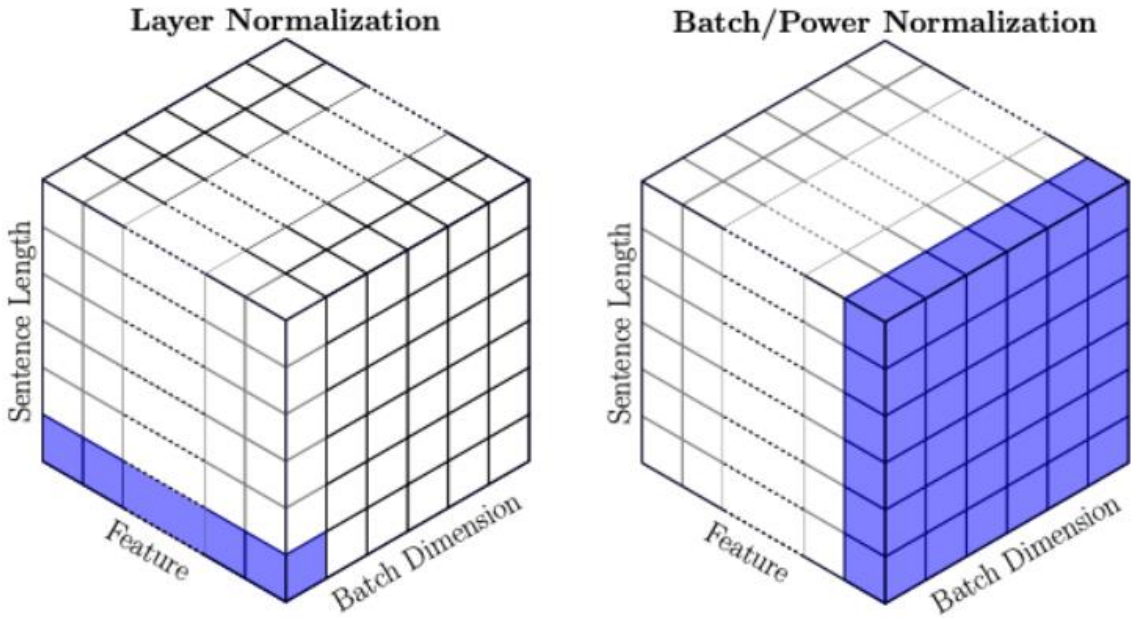
$$LN(x;\mu,\sigma) = \left\{\sigma_j \frac{x^j - \mu_x}{\sigma_x} + \mu_j\right\}_{j=\overline{1..d}}$$

$$\mu_x = \frac{1}{d}\sum_{j=1}^{d} x_j$$

$$\sigma_x^2 = \frac{1}{d}\sum_{j=1}^{d}\left(x_j - \mu_x\right)^2$$

$$x, \mu, \sigma \in \mathbb{R}^d$$

Почему LN, а не BN?



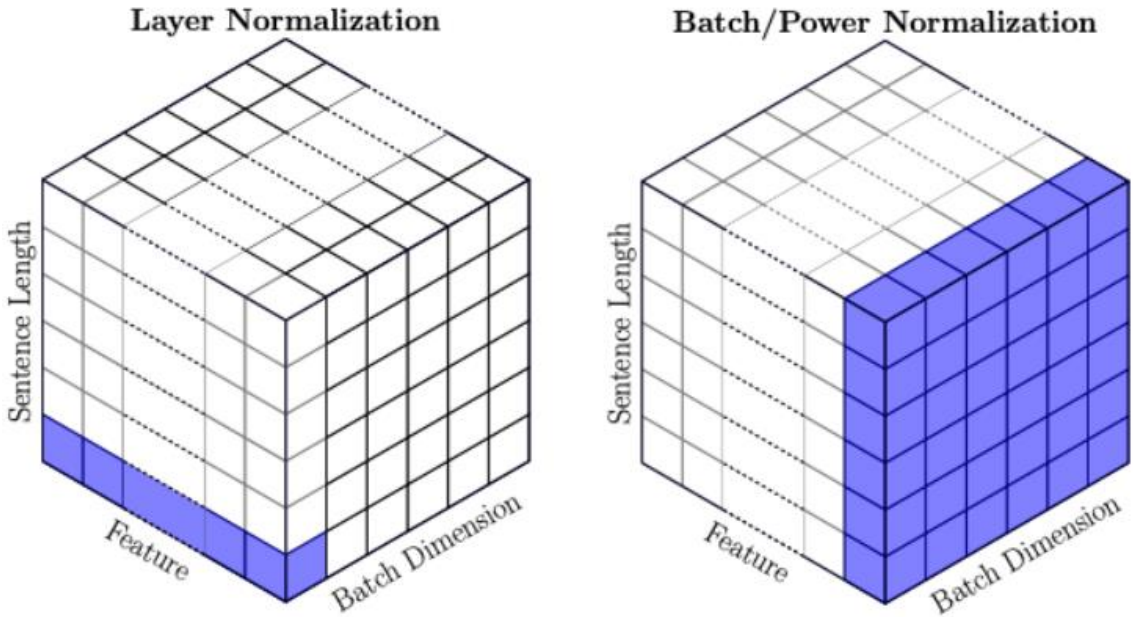
Layer normalization (нормализация слоя)

$$LN(x;\mu,\sigma) = \left\{\sigma_j \frac{x^j - \mu_x}{\sigma_x} + \mu_j\right\}_{j=\overline{1..d}}$$

$$\mu_x = \frac{1}{d}\sum_{j=1}^{d} x_j$$

$$\sigma_x^2 = \frac{1}{d}\sum_{j=1}^{d}\left(x_j - \mu_x\right)^2$$

$$x, \mu, \sigma \in \mathbb{R}^d$$



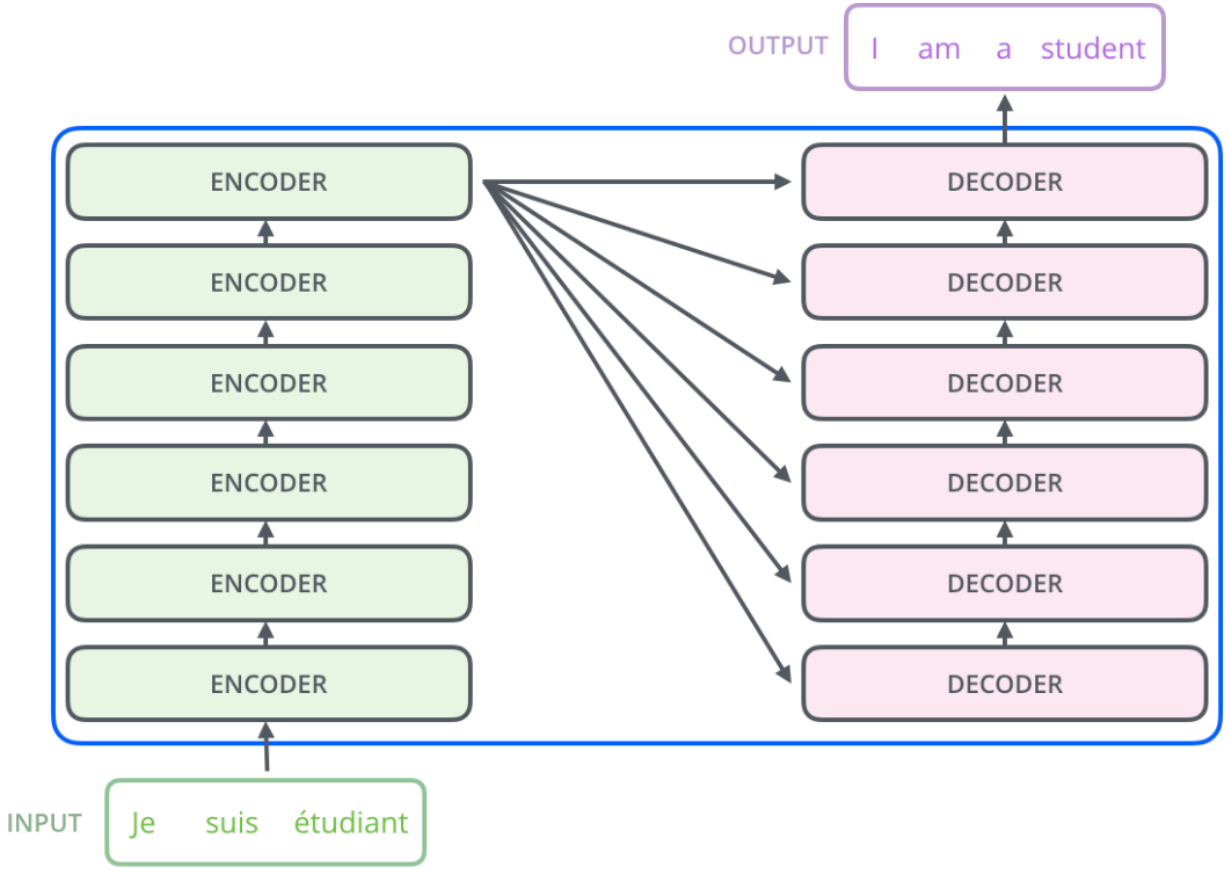
Почему LN, а не BN?

Для распараллеливания по элементам последовательности.

Теперь мы готовы перейти к кодировщику-декодировщику!

Декодировщик трансформера: связь с кодировщиком

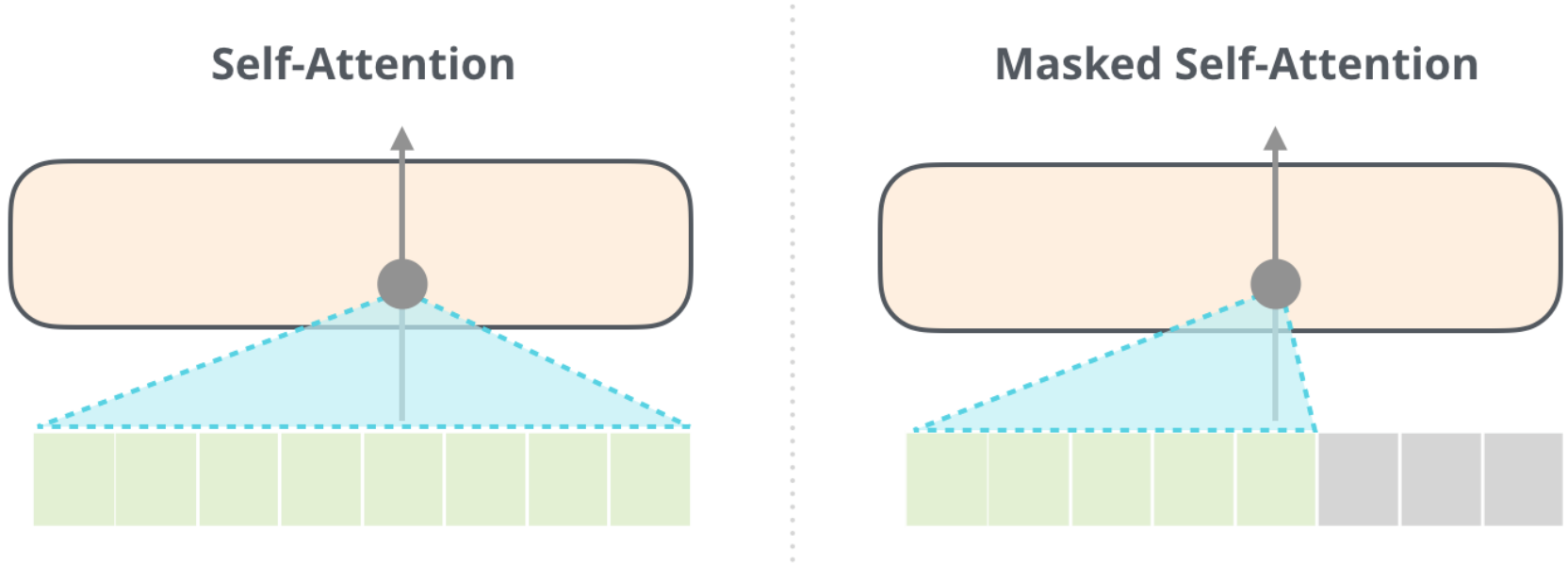
Кодировщик и декодировщик состоят из своих наборов одинаковых блоков, блоки стекаются друг за другом.
По-умолчанию, веса у каждого блока свои (неразделяемые).



Masked self-attention

Внимание в декодировщике трансформера учитывает только предыдущие токены:

$$\alpha_{ij} = \underset{j \in \{1,\ldots,i\}}{softmax}\ sim(q_i, k_j)$$

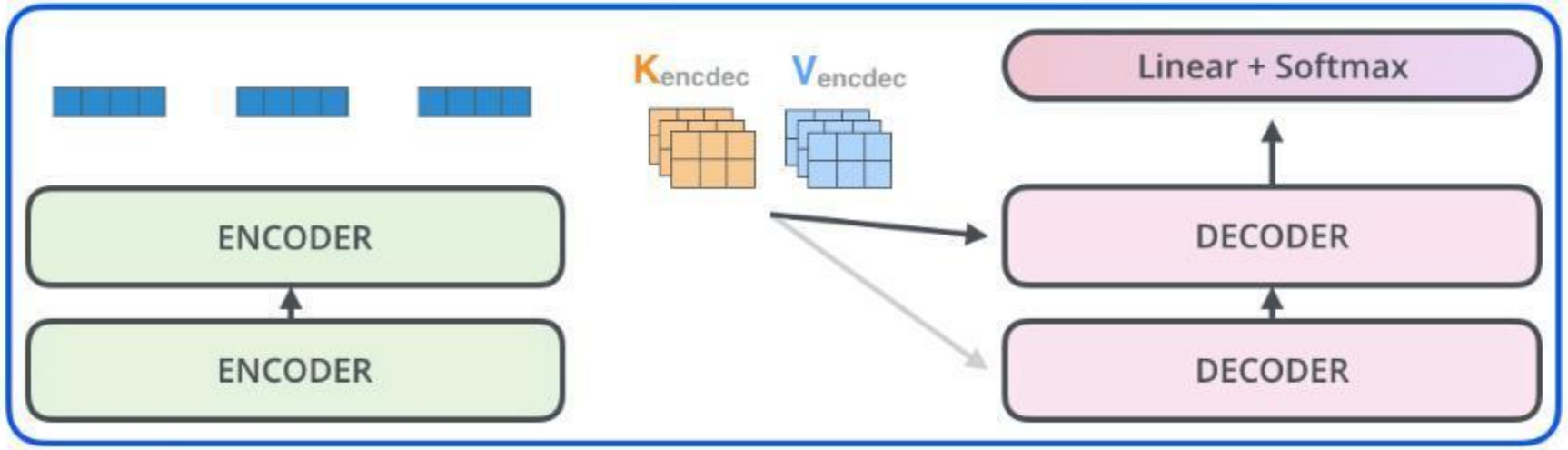


Декодировщик: связь с кодировщиком

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



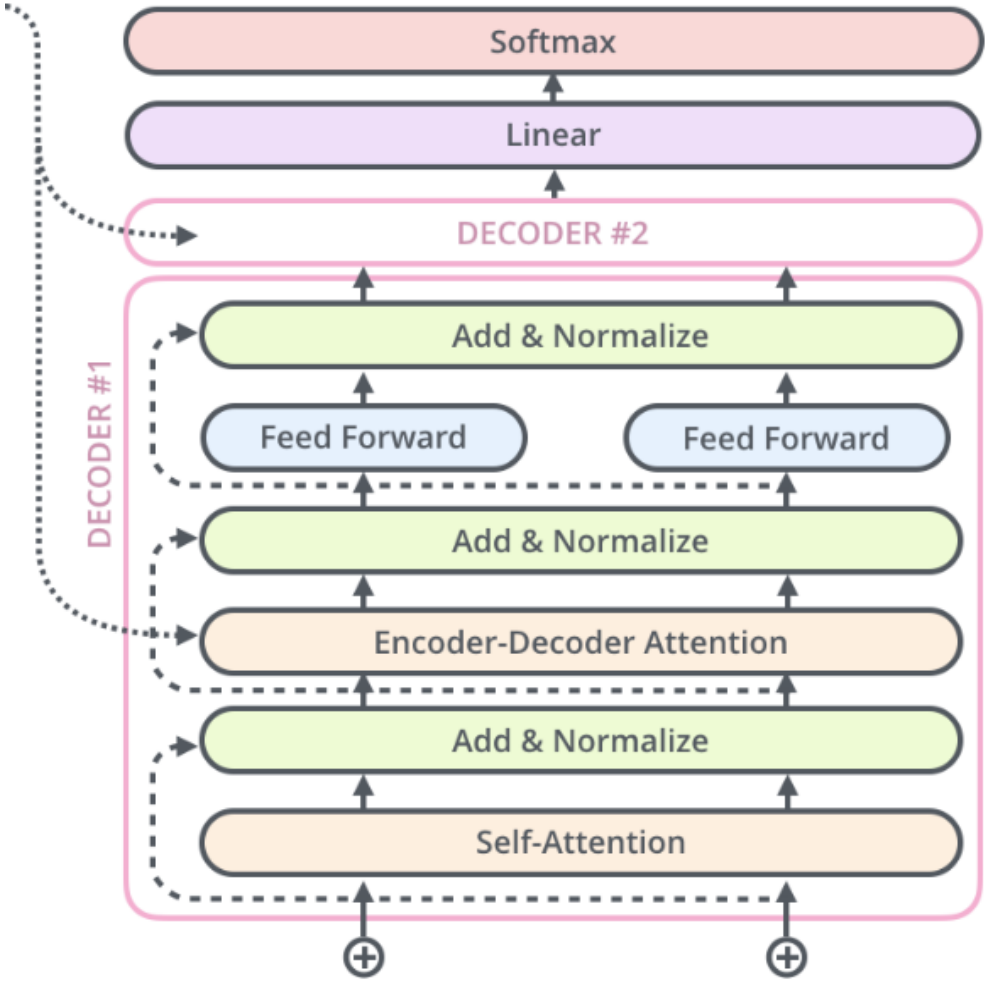
Архитектура декодировщика

Выходы первого слоя MHSA идут во второй – Encoder-Decoder Attention

Encoder-Decoder Attention — MHSA выходов первого слоя по выходам кодировщика:

- Key и Value — выходы кодировщика
- Query — первый слой декодировщика

На выходе блока — набор векторов, соответствующих токенам входной последовательности.



Обучение трансформеров: warmup

Проблема 1. В первые несколько итераций сеть адаптируется к данным, а только потом начинает обучаться.

Проблема 2. На первых эпохах сложная сеть переобучается под простые объекты в данных.

Решение. Использование warmup scheduler (изменение темпа обучения с разогревом).

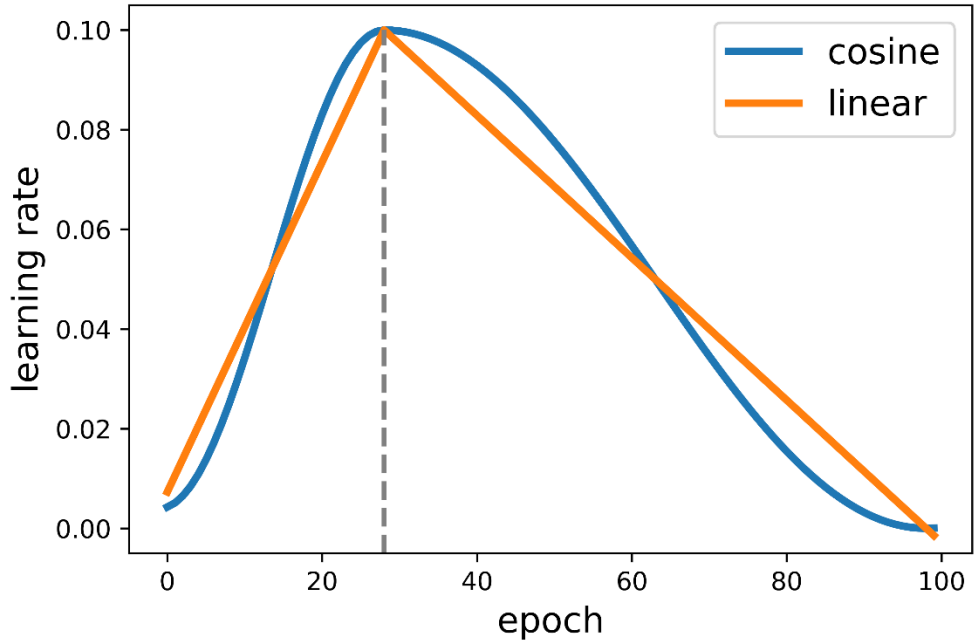
Warmup sheduler: основные стратегии

Два основных типа:

- Cosine annealing
- Linear annealing

Важные параметры:

- доля разогрева (pct)
- общее число эпох
- первый, максимальный и последний темп обучения



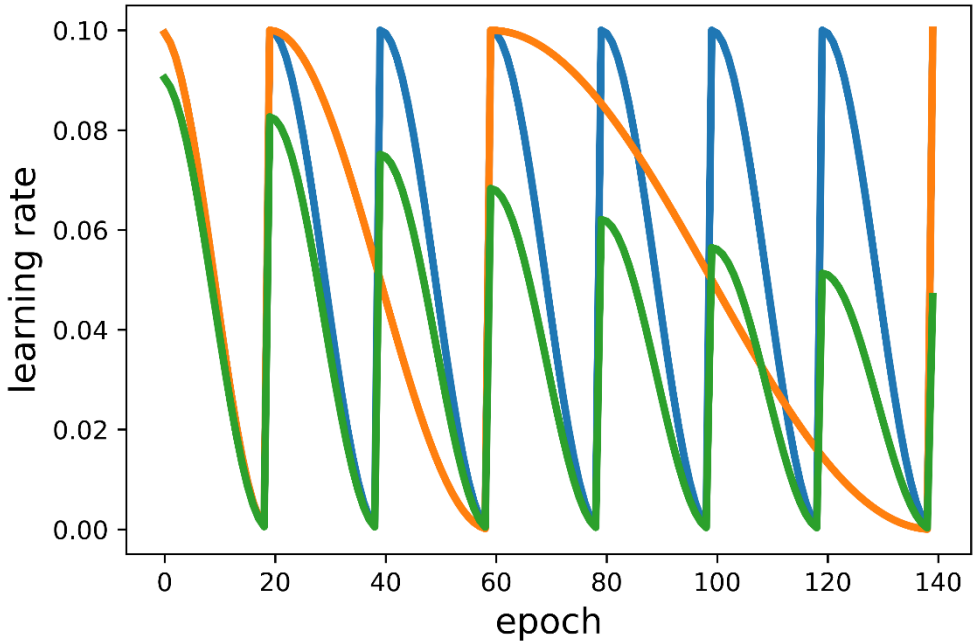
Pytorch: OneCycleLR

Smith et al. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates

Warmup: циклические стратегии

Стратегии могут быть циклическими:

- Длина циклов может увеличиваться
- Максимальный темп обучения может уменьшаться
- К первой точке нового цикла может не быть плавного перехода



Зачем это может быть надо?

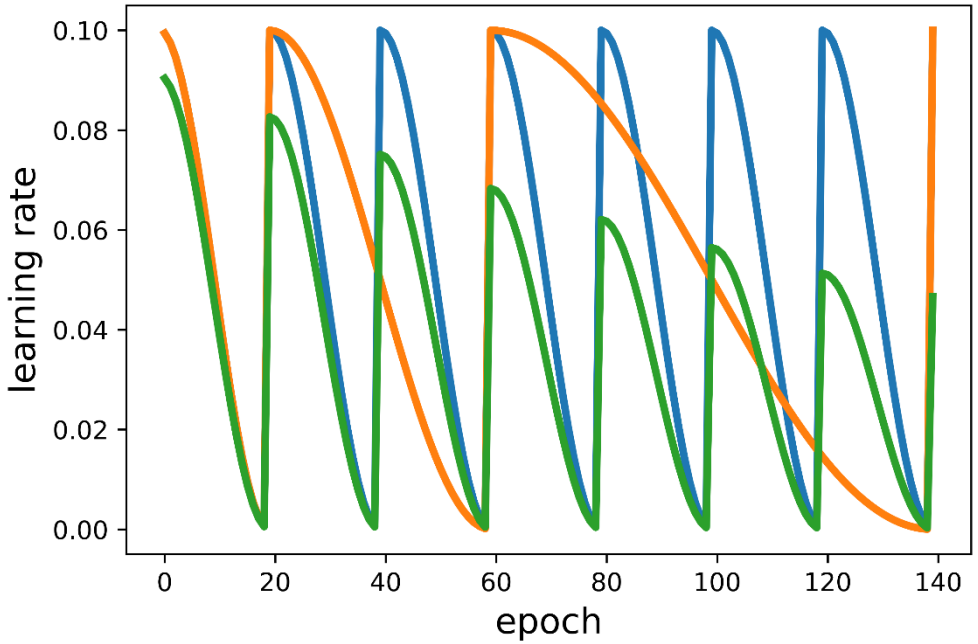
Smith. Cyclical Learning Rates for Training Neural Networks

Loshchilov et al. SGDR: Stochastic Gradient Descent with Warm Restarts

Warmup: циклические стратегии

Стратегии могут быть циклическими:

- Длина циклов может увеличиваться
- Максимальный темп обучения может уменьшаться
- К первой точке нового цикла может не быть плавного перехода



Зачем это может быть надо?

Можем ансамблировать модели из нижних точек графиков.

Smith. Cyclical Learning Rates for Training Neural Networks

Loshchilov et al. SGDR: Stochastic Gradient Descent with Warm Restarts

Что ещё следует помнить про трансформеры?

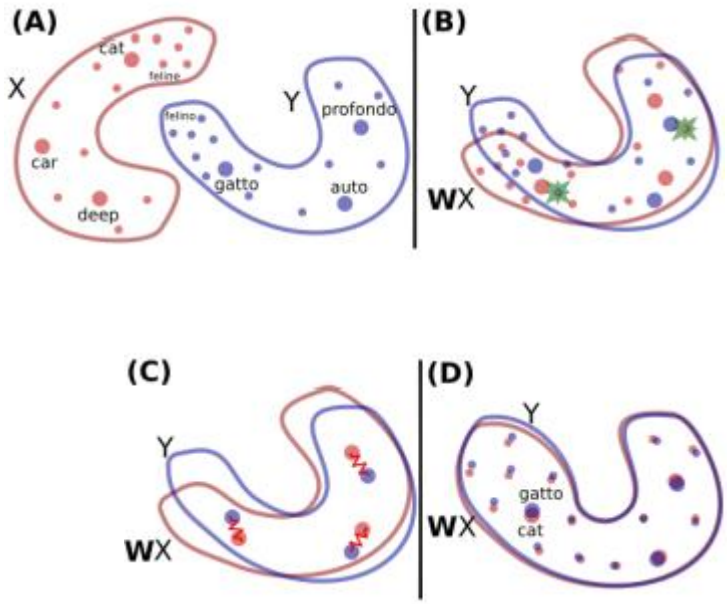
- Обычно, мы работаем с subword-токенизацией (токенизация по буквенным n-граммам)
- Кодировщик и декодировщик не обязаны быть одного размера.
- При обучении можно использовать Adam, обычно используют warm-up расписание для темпа обучения
- Сложность трансормера квадратичная как по длине последовательности, так и по размеру внутреннего слоя.
- Позиционные эмбеддинги могут быть фиксированными или обучаемыми. Также, их можно сделать относительными.

Что ещё следует помнить про трансформеры?

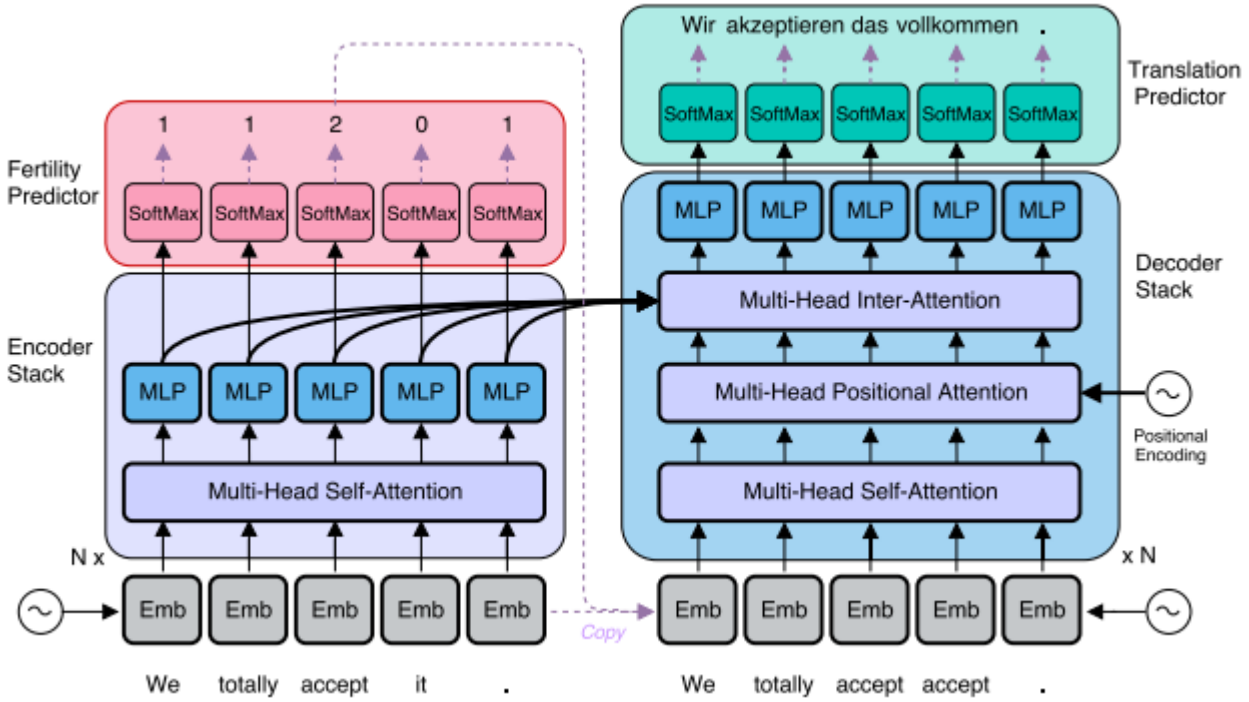
- Трансформеры можно применять во всех задачах, которые мы обсуждали до этого: классификация и разметка
- Трансформеры обычно не требуют детального подбора параметров обучения и архитектуры (в отличии от RNN)
- Трансформеры можно использовать вместе с CRF
- Трансформеры могут содержать очень много attention блоков (BERT – 12 блоков, T5 – 14 блоков, GTP3 – 96 блоков,)

А что ещё можно узнать про машинный перевод?

Машинный перевод без учителя



Неавторегрессионный перевод



[Gu et al (ICLR 2018); Non-Autoregressive Neural Machine Translation]

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