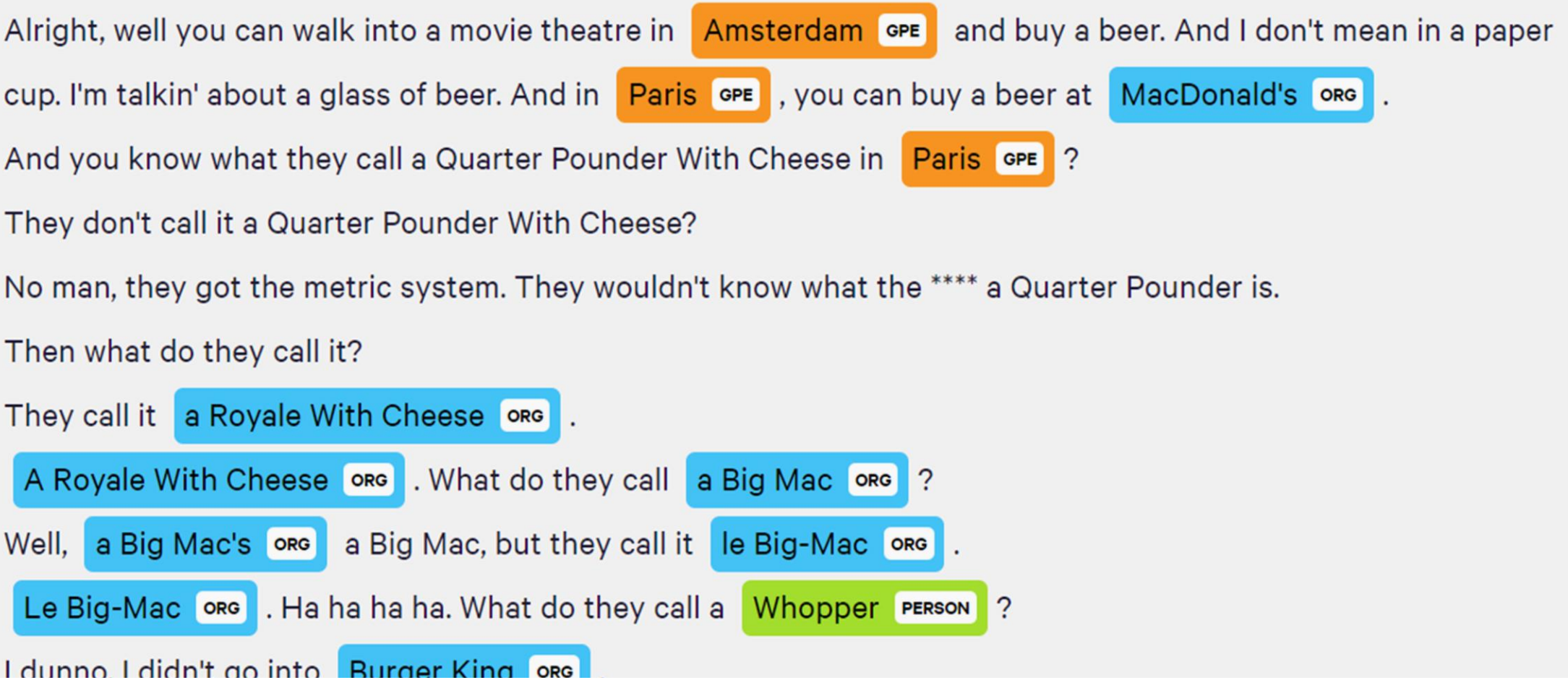


Задача разметки последовательности. Нейросетевые подходы к её решению.

Попов Артём, кафедра ММП ВМК МГУ

Математические методы анализа текстов, осень 2021

Пример: разметка библиографии

Вы разрабатываете систему отслеживания научных публикаций. По каждой библиографической ссылке в статье необходимо получить:

1. Имена авторов
2. Название статьи
3. Название журнала / конференции
4. Год публикации

Какой бейзлайн вы можете придумать для этой задачи?

Пример: разметка библиографии

Вы разрабатываете систему отслеживания научных публикаций. По каждой библиографической ссылке в статье необходимо получить:

1. Имена авторов
2. Название статьи
3. Название журнала / конференции
4. Год публикации

Какой бейзлайн вы можете придумать для этой задачи?

Простейшее rule-based решение – разобрать ссылку при помощи регулярных выражений.

Сложность задачи: разнообразие ссылок

David Blei, Andrew Ng, Michael Jordan. Latent Dirichlet allocation. JMLR, 2003.

D.Blei, A.Ng, M.Jordan. Latent Dirichlet allocation // Journal of Machine Learning Research. 2003. V.3. Pp.9931022.

[Blei et al. 2003] David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. The Journal of Machine Learning Research, V.3. 993 - 1022

Blei, D., Ng, A. & Jordan, M. J. Mach. Learn. Res. 3 (January 2003), 993 --1022.

Blei, David, Ng, Andrew, and Jordan, Michael. Latent Dirichlet allocation. Journal of Machine Learning Research, 3: 993-1022, 2003.

Постановка задачи разметки последовательности

Дано множество размеченных последовательностей (x, y):

- $x = (x_1, \ldots, x_n)$ – входная последовательность (слова)
- $y = (y_1, \ldots, y_n)$ – выходная последовательность (метки, теги)

Необходимо по входной последовательности предсказать элементы выходной последовательности.

1. Метка y_i соответствует слову x_i. Длины x, y из одной пары совпадают, но могут различаться с длинами других пар.
2. Две последовательности можно привести к одной длине дополнив короткую специальным <PAD> токеном.

Другие названия: sequence tagging, sequence labeling

Примеры задач

- Распознавание частей речи (Part of speech tagging, POS)
- Распознавание именованных сущностей
  (Named Entity Recognition, NER)
- Разметка семантических ролей
  (Semantic Role Labeling, SRL)
- Выделение текстовых полей в данных (Slot filing)
- Разметка библиографической информации
- Сегментация текста
  (например, по смыслу на background, methods, results)
- Фильтрация текстового нежелательного контента

Составные сущности. BIO-нотация.

Именованная сущность может состоять из нескольких токенов. В этом случае обычно используют BIO-нотацию:

- B (Begin) – первое слово сущности
- I (Inside) – второе слово сущности
- O (Outside) – слово не входит ни в какую сущность

Betty	**came**	**to**	**Los**	**Angeles**	**to**	**become**	**an**	**actress**
B-PER	O	O	B-LOC	I-LOC	O	O	O	O

Подходы к задаче разметке

- Rule-based подход
- Классификатор на каждой позиции, использующий признаки контекста позиции
- Графические модели (HMM / MEMM / CRF)
- Нейронные сети (рекуррентные, трансфомеры, свёрточные)
- Комбинация нейронных сетей и графических моделей

Оценивание качества разметки

Макро-метрики: агрегация по предложениям.

Микро-метрики: агрегация по сущностям.

- Микро-precision – доля правильно распознанных сущностей среди всех распознанных сущностей
- Микро-recall – доля правильно распознанных сущностей среди всех истинных сущностей
- Микро-f1 –среднее гармоническое precision и recall

Микро-метрики можно считать по каждому типу сущностей.

Определение частей речи (POS)

- Для каждого слова в предложении определить его часть речи
- Простая задача – часто можно определить часть речи слова даже без знания его контекста

Зачем нужна разметка частей речи?

Определение частей речи (POS)

- Для каждого слова в предложении определить его часть речи
- Простая задача – часто можно определить часть речи слова даже без знания его контекста

Зачем нужна разметка частей речи?

- Снятие омонимии (мыло_NOUN, мыло_VERB)
- Дополнительный признак / дополнительное представление
- Построение сложных правил
- Выделение стоп-слов (союзы, предлоги обычно стоп-слова)
- Группировка слов по важности (при определении темы текста существительные важнее глаголов)

Распознавание именованных сущностей (NER)

- Для каждого слова в предложении определить, является ли оно частью именованной сущности.
- Сложнее чем POS, но может быть частично решена при помощи словарей

Зачем нужно распознавание именованных сущностей?

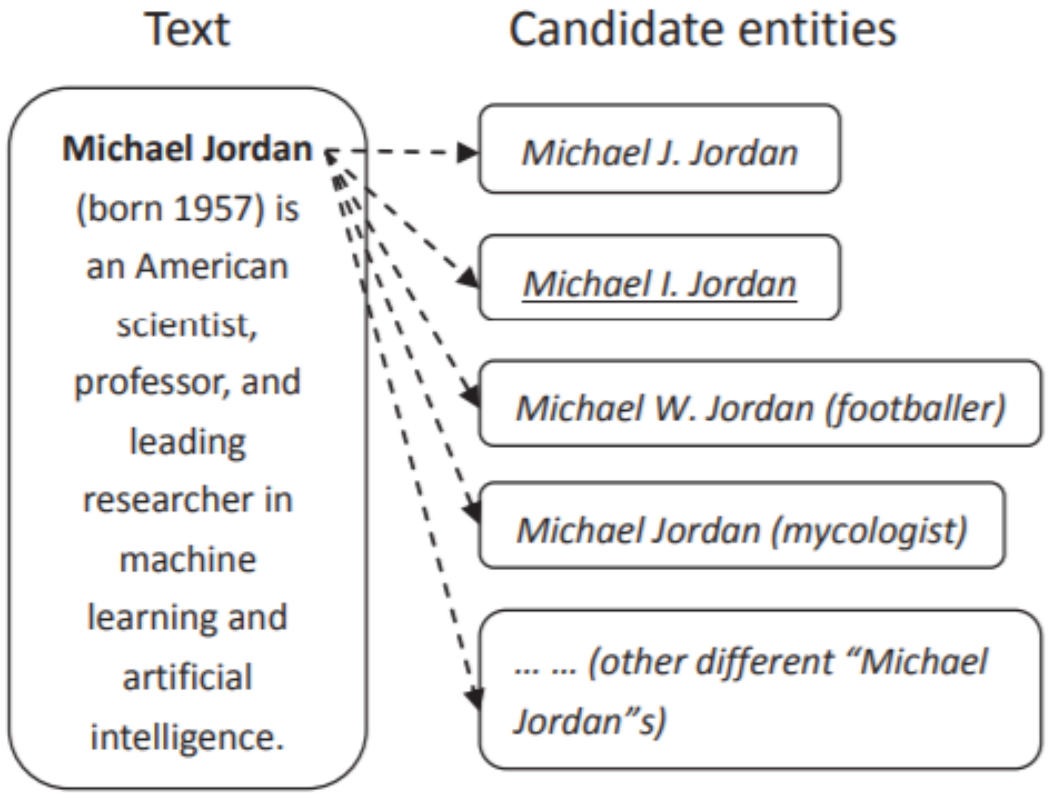
- Диалоговые системы
- Поиск
- Деперсонализация данных
- Проставление тегов к новостям

От Recognition к Linking

Иногда после нахождения именованной сущности, требуется сопоставить её с сущностью из базы знаний (named entity linking).

Триплеты для обучения:

- Предложение
- Упоминаемая сущность
- Ссылка на сущность в базе



Wang et al. Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions

NEL на примере Википедии

Хотим сопоставлять найденные сущности со статьями Википедии.

Общий принцип решения:

1. Ищем статьи с упоминанием найденной сущности в заголовке.
2. Получаем эмбеддинг для каждой статьи (можно посчитать заранее) и найденной сущности с учётом контекста
3. Находим статью с максимальной близостью по эмбеддингам

Как собрать выборку для обучения / тестирования?

NEL на примере Википедии

Хотим сопоставлять найденные сущности со статьями Википедии.

Общий принцип решения:

1. Ищем статьи с упоминанием найденной сущности в заголовке.
2. Получаем эмбеддинг для каждой статьи (можно посчитать заранее) и найденной сущности с учётом контекста
3. Находим статью с максимальной близостью по эмбеддингам

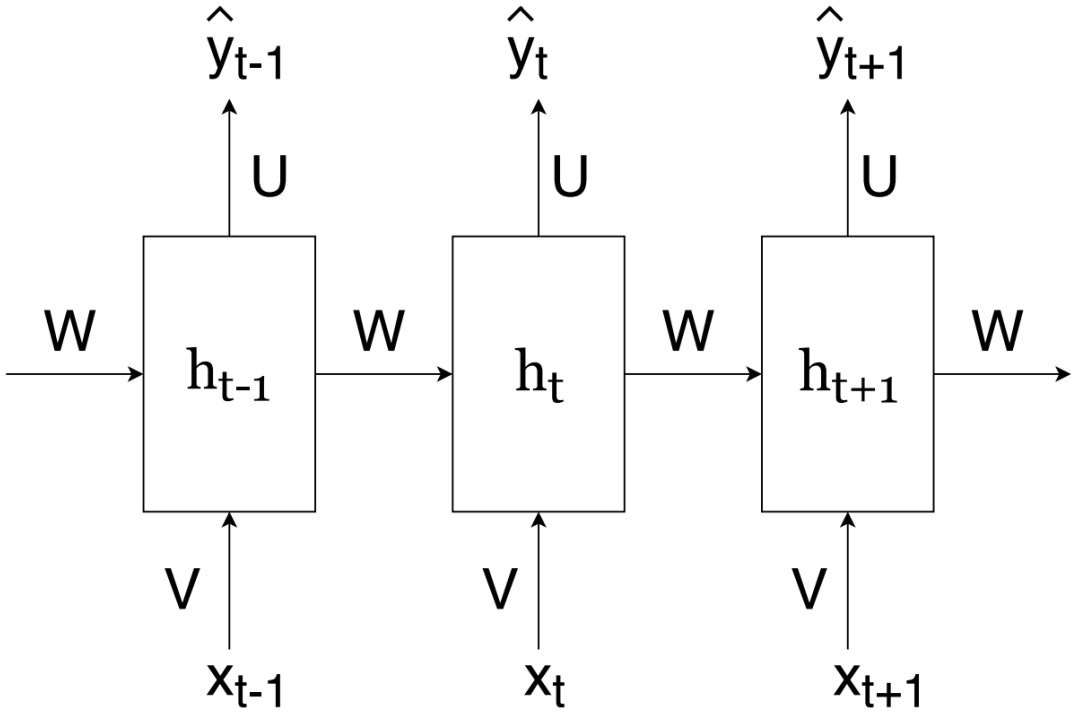
Как собрать выборку для обучения / тестирования?

Можем собрать триплеты (предложение, сущность, статья). В Википедии есть ссылки на статью о сущности при первом её упоминании.

Рекуррентные сети

- Определение рекуррентной нейронной сети
- Борьба со взрывом и затуханием. LSTM и GRU
- Особенности применения рекуррентных нейронных сетей

Модель рекуррентной нейронной сети (RNN)



h_t – скрытое состояние сети в момент времени t

Принцип работы сети:

$$h_t = f(Vx_t + Wh_{t-1} + b)$$

$$\hat{y}_t = g\left(Uh_t + \hat{b}\right)$$

Обучение сети (backpropagation through time):

$$\sum_{t=1}^{n} \mathcal{L}(y_t, \hat{y}_t) \rightarrow \min_{V,W,U,b,\hat{b}}$$

[Rumelhart et al. Learning internal representations by error propagation (1985)]

Детали обучения RNN: производные по U и W

Градиент по U зависит только от величин в момент t:

$$\frac{\partial \mathcal{L}_t}{\partial U} = \frac{\partial \mathcal{L}}{\partial y_t}\frac{\partial y_t}{\partial U}$$

Градиент по W зависит от всех предыдущих величин:

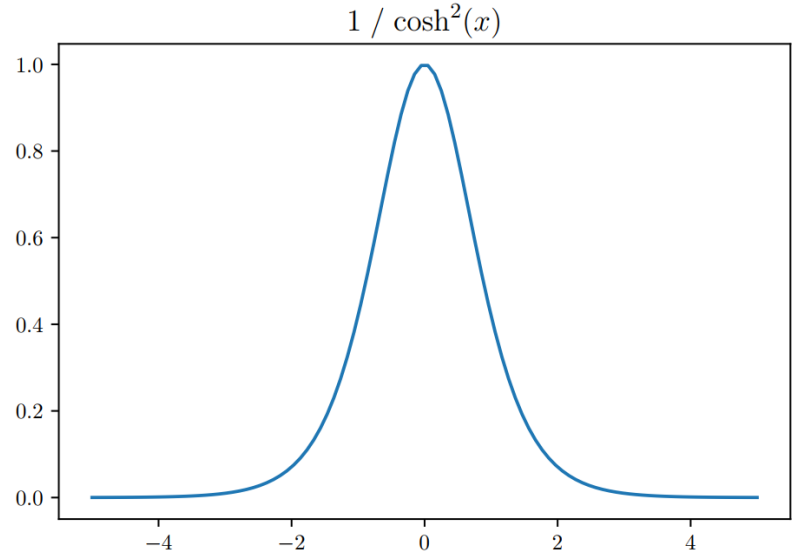
$$\frac{\partial \mathcal{L}_t}{\partial W} = \frac{\partial \mathcal{L}}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{dh_t}{dW}$$

$$\frac{dh_t}{dW} = \frac{\partial h_t}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}}\frac{dh_{t-1}}{dW} = \cdots = \sum_{k=1}^{t}\left(\prod_{i=k+1}^{t}\frac{\partial h_i}{\partial h_{i-1}}\right)\frac{\partial h_k}{\partial W}$$

Взрыв и затухание градиента

Взрыв градиента:

$$\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \to \infty$$



Затухание градиента:

$$\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \to 0$$

$$\frac{\partial h_i}{\partial h_{i-1}} = diag\left(\frac{1}{ch^2(z_i)}\right) W$$

$$z_i = Vx_i + Wh_{i-1} + b$$

$$f = \tanh$$

Как бороться с взрывом и затуханием градиентов?

Способы борьбы с взрывом и затуханием

Взрыв

- Gradient clipping (подрезка градиентов)

Затухание

- Усложнение архитектуры: LSTM / GRU

Взрыв + затухание (не популярно)

- Регуляризация $\frac{\partial h_i}{\partial h_{i-1}} \to 1$

Gradient clipping

Ограничение нормы градиентов:

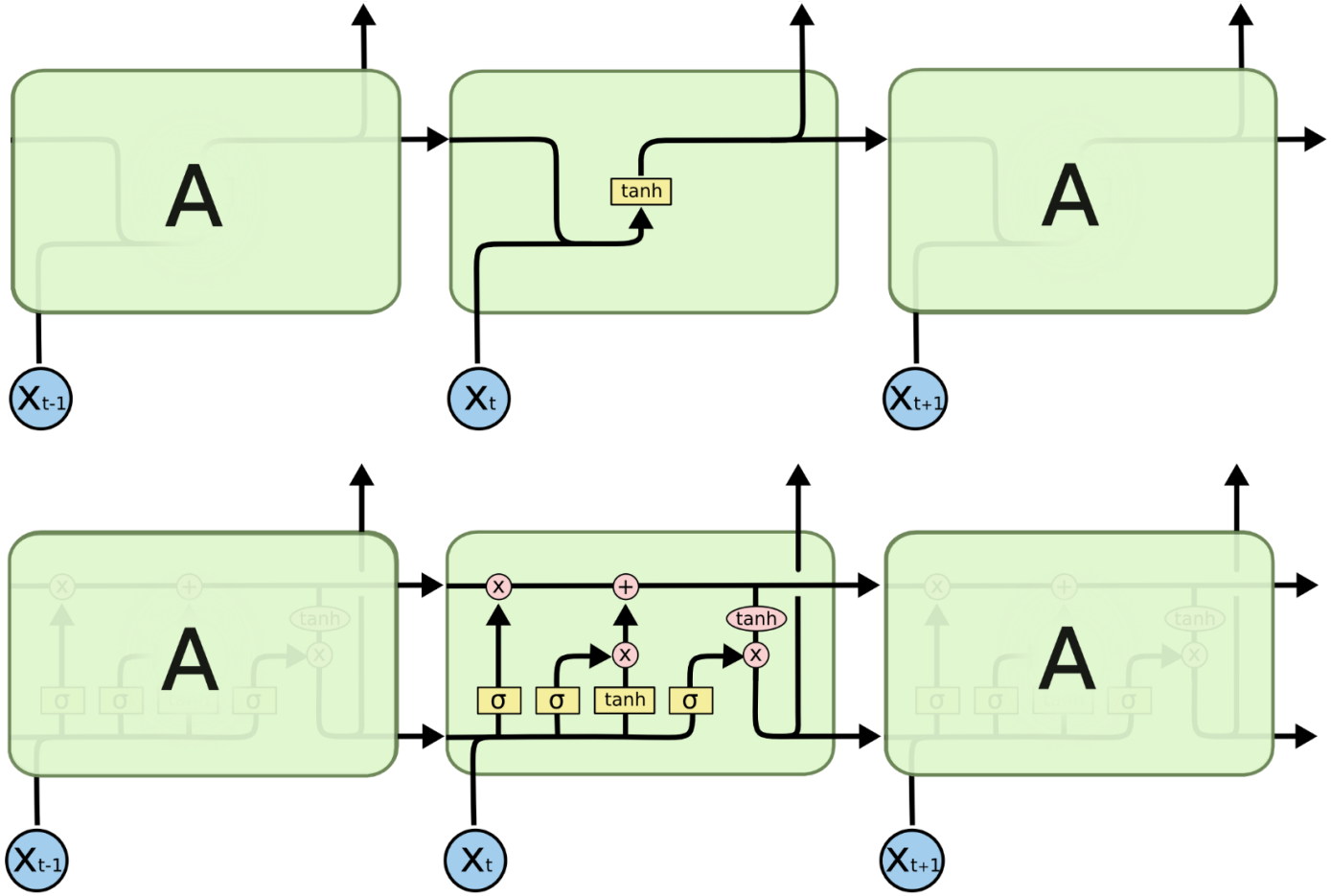
Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

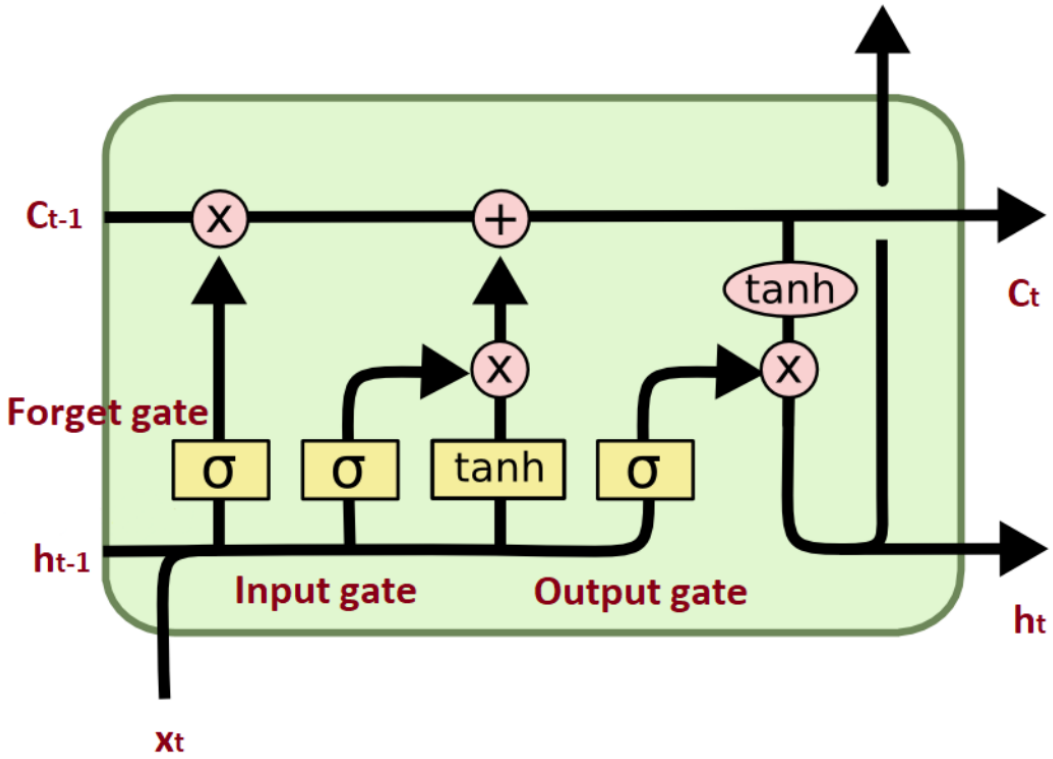
В качестве порога обычно используют небольшую константу.
Можно брать среднюю норму градиента для весов по запускам без gradient clipping.

LSTM сеть

Идея. Хотим сделать более сложную структуру ячейки.



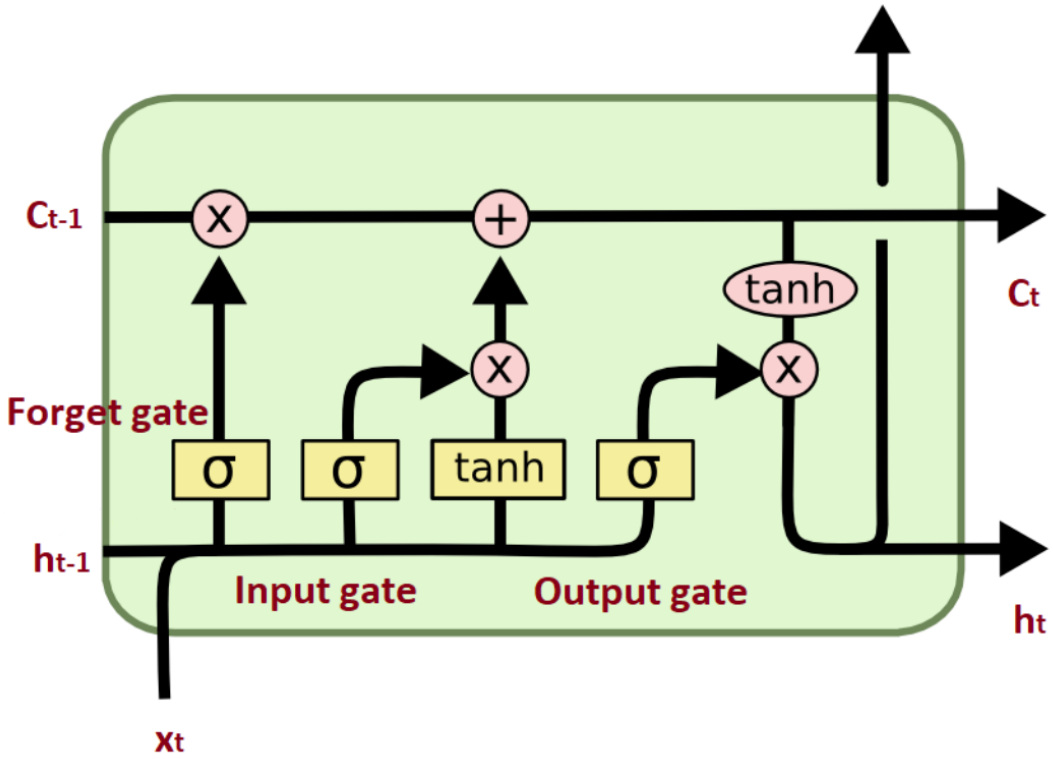
LSTM ячейка



$$z_t = [h_{t-1}, x_t]$$
$$f_t = \sigma(W_f \cdot z_t + b_f)$$
$$i_t = \sigma(W_i \cdot z_t + b_i)$$
$$\hat{C}_t = \text{th}(W_c \cdot z_t + b_c)$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t$$
$$o_t = \sigma(W_o \cdot z_t + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

За счёт чего решается проблема затухания градиента?

LSTM ячейка

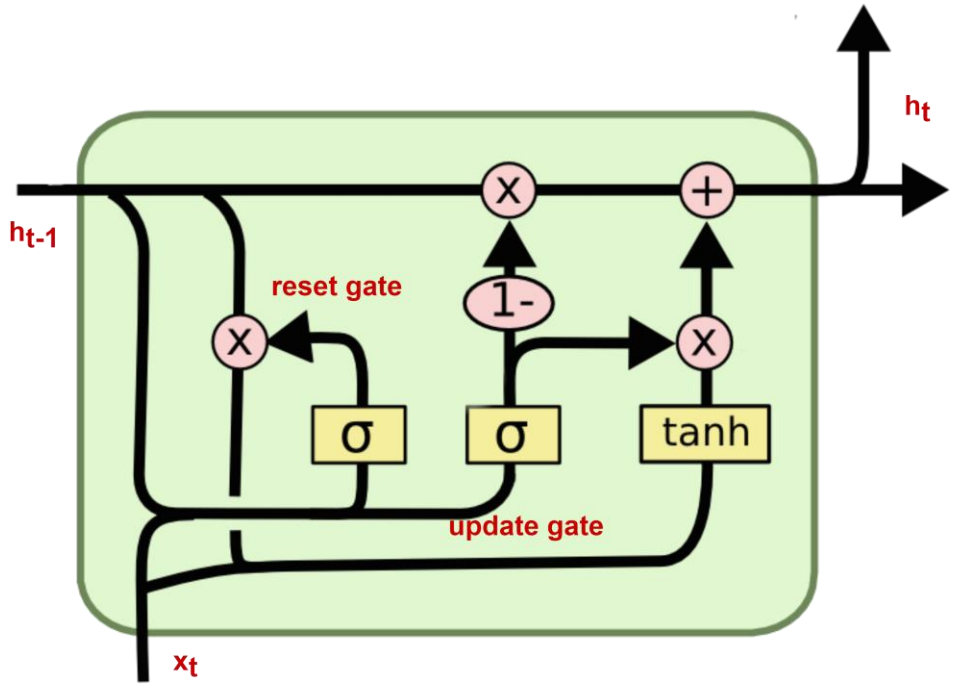


$$z_t = [h_{t-1}, x_t]$$
$$f_t = \sigma(W_f \cdot z_t + b_f)$$
$$i_t = \sigma(W_i \cdot z_t + b_i)$$
$$\hat{C}_t = \text{th}(W_c \cdot z_t + b_c)$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t$$
$$o_t = \sigma(W_o \cdot z_t + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

За счёт чего решается проблема затухания градиента?

Инициализируем b_f большим значением, чтобы значение производной было близко к 1.

GRU ячейка



Два гейта берут на себя функции трёх из LSTM:

$$z_t = [h_{t-1}, x_t]$$

$$u_t = \sigma(W_u \cdot z_t + b_u)$$

$$r_t = \sigma(W_r \cdot z_t + b_r)$$

$$\hat{h}_t = \tanh(W_h \cdot [r_t h_{t-1}, x_t] + b_h)$$

$$h_t = (1 - u_t) \cdot h_{t-1} + u_t \cdot \hat{h}_t$$

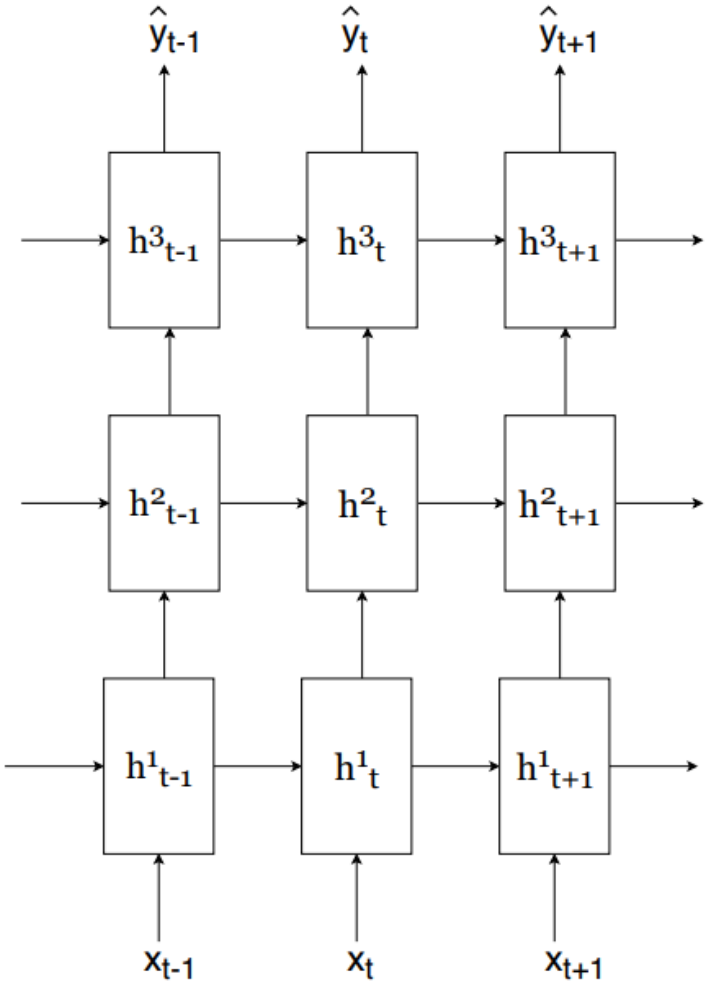
\+ Быстрее учится чем LSTM

\+ Качество на уровне LSTM

Глубокие рекуррентные сети (deep RNN)

Подача выходов одной рекуррентной сети на вход другой.

$$h_t^1, C_t^1 = LSTM(h_{t-1}^1, C_{t-1}^1, x_t)$$
$$h_t^2, C_t^2 = LSTM(h_{t-1}^2, C_{t-1}^2, x_t)$$
$$h_t^3, C_t^3 = LSTM(h_{t-1}^3, C_{t-1}^3, x_t)$$
$$y_t = g\left(Uh_t^3 + \hat{b}\right)$$

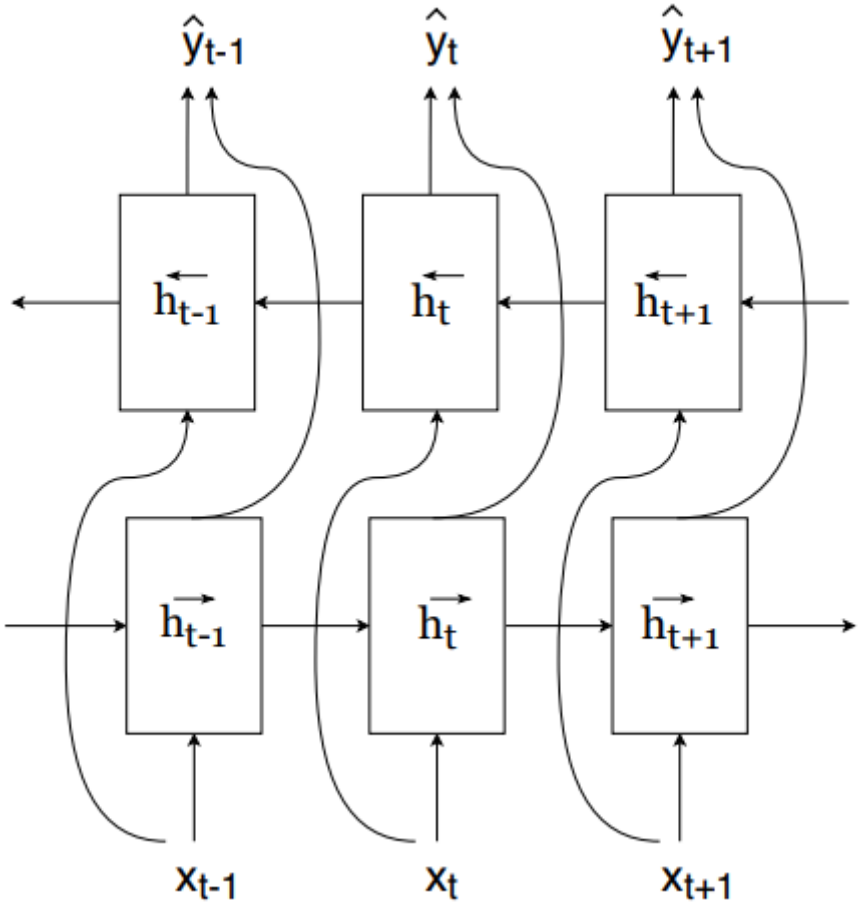


Двунаправленные сети (bidirectional)

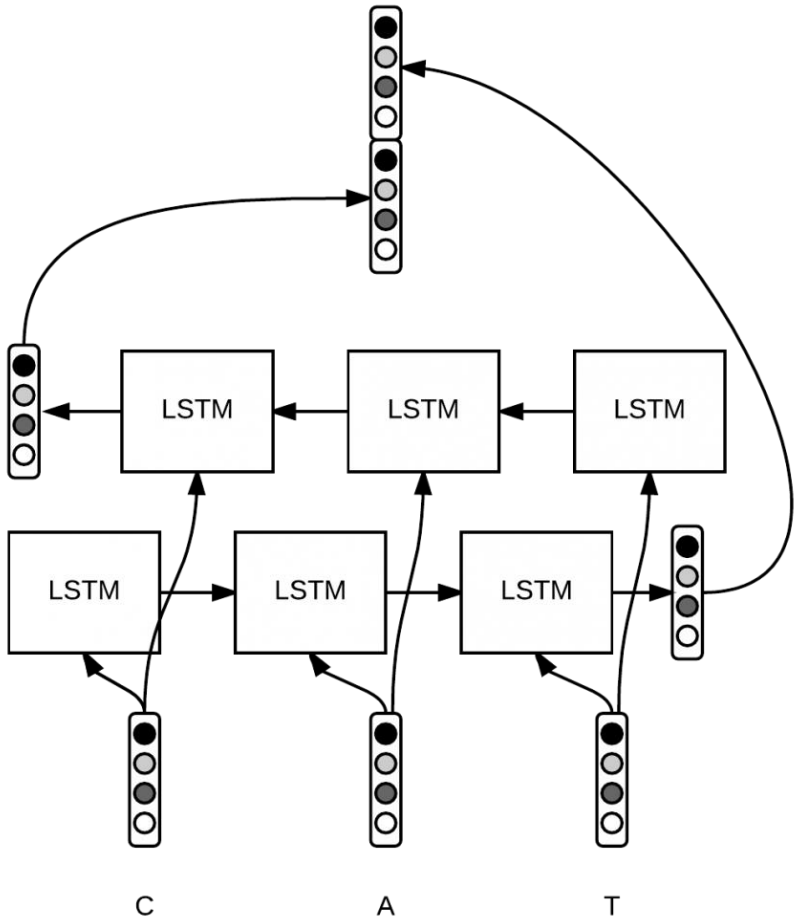
Конкатенация выходов двух рекуррентных сетей, одна идёт слева направо, другая справа налево.

$$\overrightarrow{h_t}, \overrightarrow{C_t} = \overrightarrow{LSTM}(\overrightarrow{h_{t-1}}, \overrightarrow{C_{t-1}}, x_t)$$
$$\overleftarrow{h_t}, \overleftarrow{C_t} = \overleftarrow{LSTM}(\overleftarrow{h_{t+1}}, \overleftarrow{C_{t+1}}, x_t)$$
$$y_t = g(U[\overrightarrow{h_t}, \overleftarrow{h_t}] + \hat{b})$$

Мощнее однонаправленных, но не везде их можно применять!



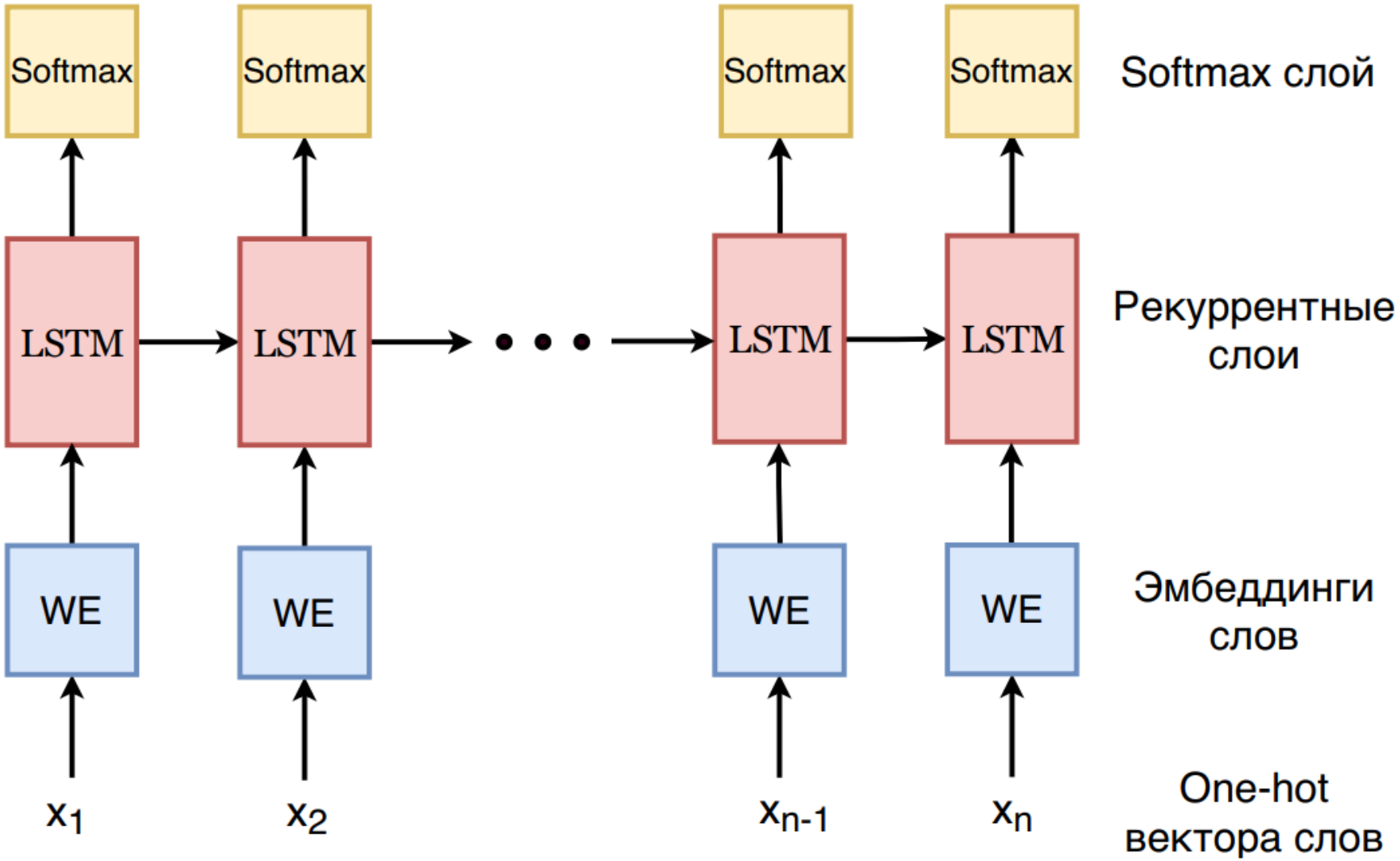
Иерархические сети (hierarchical rnn)



Обычно используются для решения проблемы OOV слов.

1. Каждое слово посимвольно прогоняем через отдельную LSTM
2. Эмбеддинг слова – последнее состояние “вложенной” LSTM
3. Поверх эмбеддингов слов работает стандартная LSTM
4. Всё учится end-to-end

Теггер на основе RNN



LSTM в задаче разметки

- При предобработке слова обычно не приводятся к нижнему регистру
- Лучше использовать bidirectional сеть
- Может быть несколько слоёв (но редко > 2)
- Эмбеддинги слов могут быть:
  - инициализированы предобученной моделью, заморожены во время обучения
  - инициализированы предобученной моделью, обучаются во время обучения
  - случайно инициализированы, обучаются во время обучения
- Dropout помогает при обучении (иногда лучше использовать специальный Dropout для RNN)

Преимущества и недостатки реккурентных сетей

- Насколько верно предположение, что вся информация о последовательности может быть закодирована одним вектором состояния?

- Невозможно хорошо распараллелить вычисления

+ Возможно обработать последовательность любой длины

+ Количество используемой памяти не зависит от длины последовательности

Трансформеры в задаче разметки

- Механизм self-attention (самовнимание)
- Позиционные представления (positional encoding)
- Энкодер трансформера

Модель трансформер (transformer)

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

Составляющие трансформера:

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

На этой лекции рассмотрим как устроен энкодер трансформера, остальное на следующих лекциях.

Vaswani et al. Attention Is All You Need (2017)

Механизм self-attention (SA)

Вход:

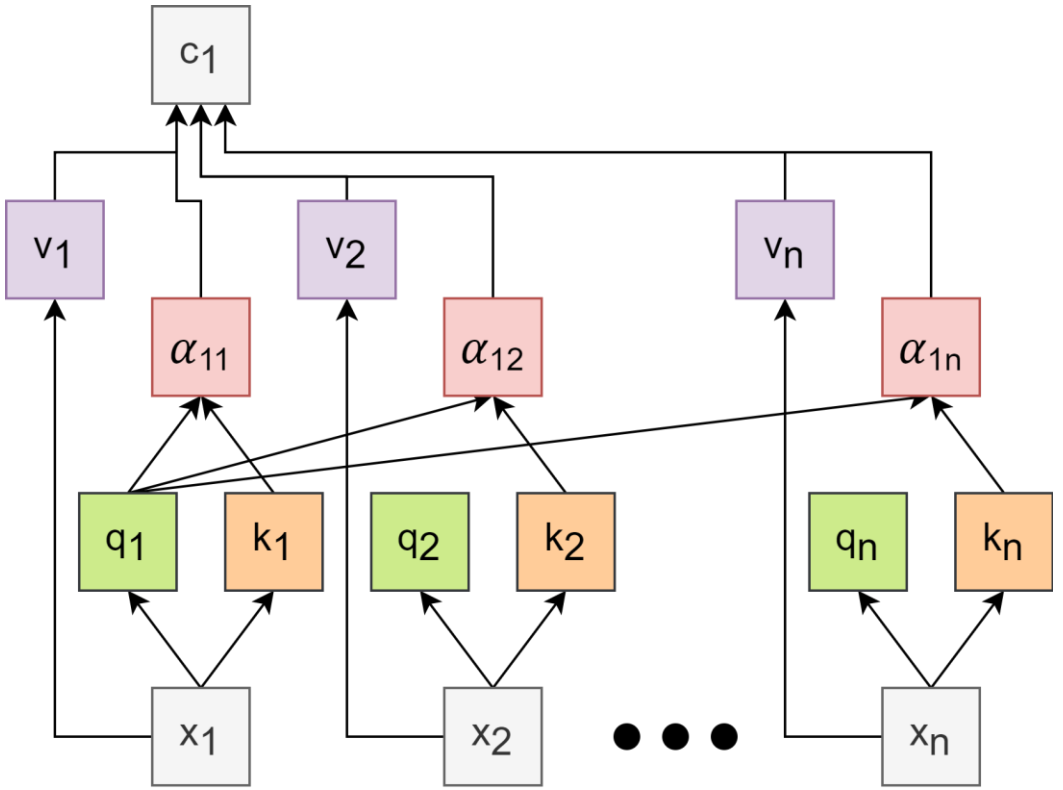
последовательность $(x_1, \dots, x_n)$, $x_i \in \mathbb{R}^d$

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”

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

3. Вычисляем выпуклую комбинацию значений v

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

SA – частный случай общего принципа внимания

Идея: хотим уметь пересчитывать вектор запроса на основе релевантного контекста.

Вход:

- $q \in \mathbb{R}^{d_1}$ - вектор запроса
- $h = (h_1, \ldots, h_n)$, $h_i \in \mathbb{R}^{d_2}$ - “контекст” запроса

Выход: $c \in \mathbb{R}^d$ – пересчитанный вектор запроса

$$c(q,h) = \sum_{j=1}^{n} sim\left(Query(q), Key\left(h_j\right)\right) Value(v_j)$$

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 ($\boldsymbol{\theta}$): 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
   $$MHSA(x;\theta)_i = y_i = \left[c_i^1, \dots, c_i^N\right] W$$

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?

$\mathrm{O}(\mathrm{n}^2\mathrm{d} + \mathrm{nd}^2)$

- 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

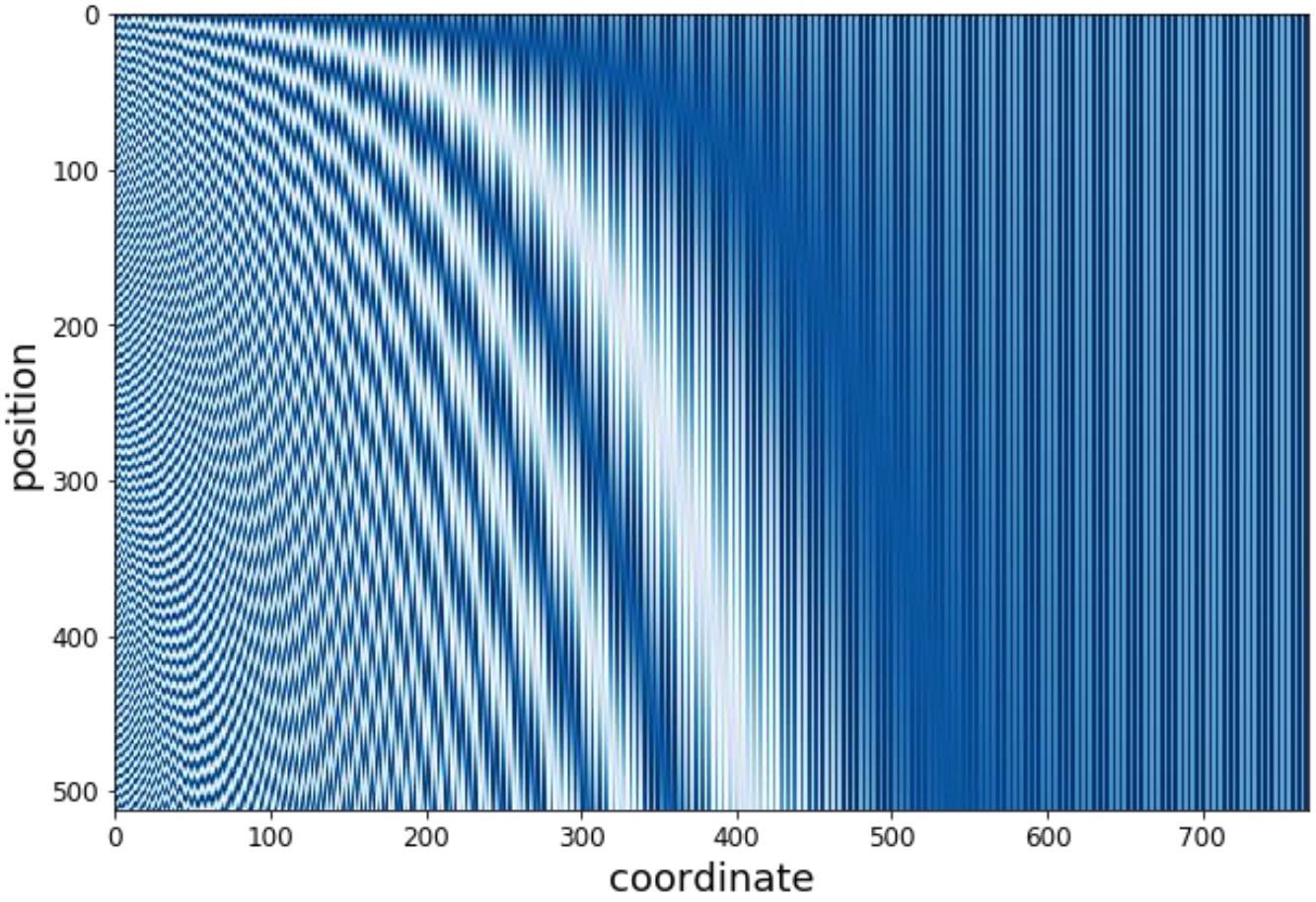
- фиксированные позиционные эмбеддинги
- обучающиеся позиционные эмбеддинги
- относительные позицонные эмбеддинги (relative encoding)

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

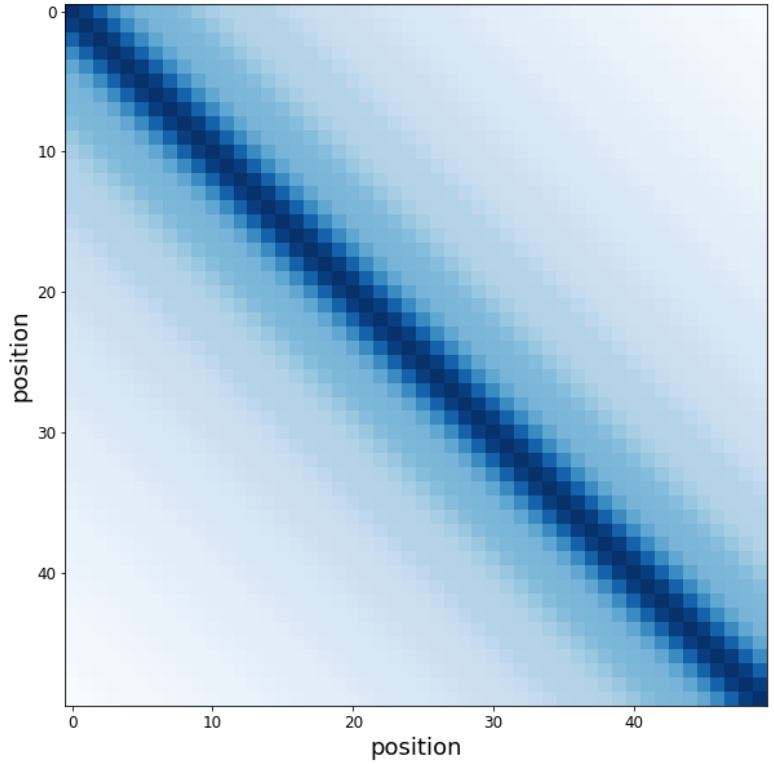
Теорема. Для любых векторов p_i и p_{i+k} верно $E_k p_i = p_{i+k}$.

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

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

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

Матрица Грама для позиционных эмбеддингов



Обучающиеся позиционные представления

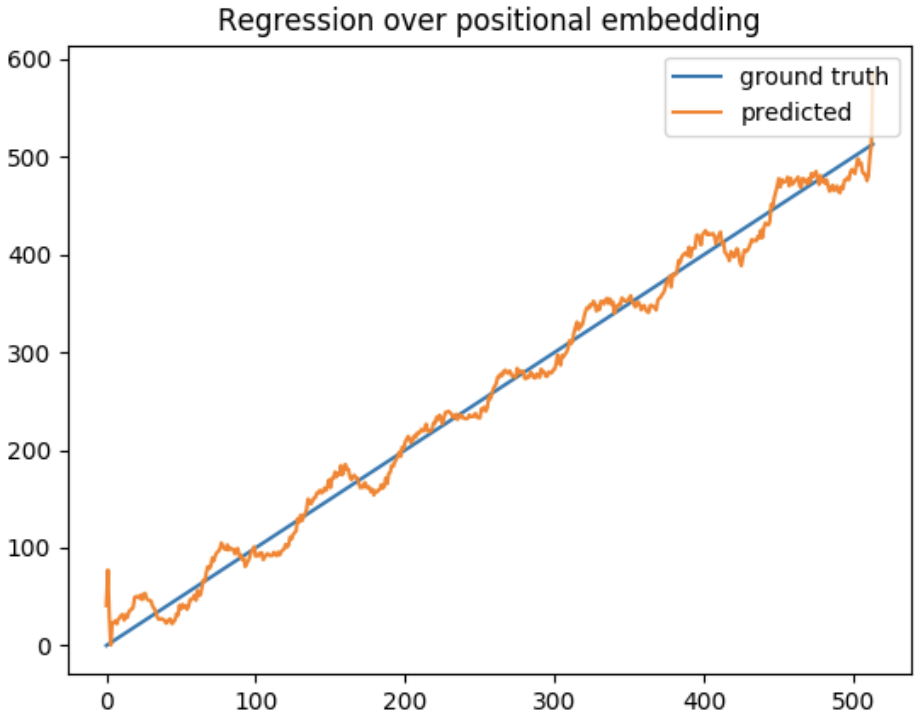
Идея. Вместо использования фиксированного вектора, будем учить представление для каждой позиции.

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:

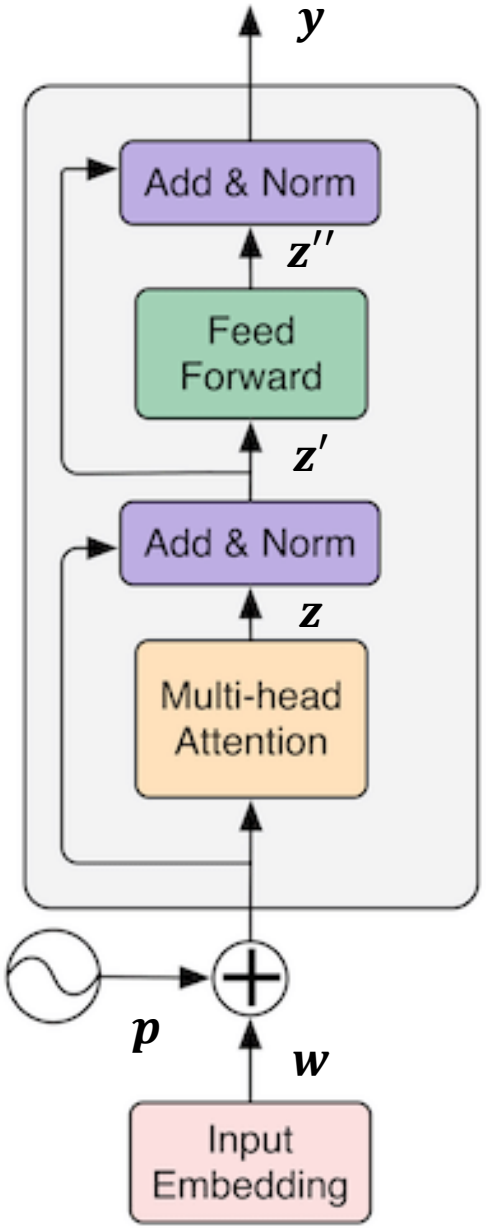
– Требует большего количества операций

\+ 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)

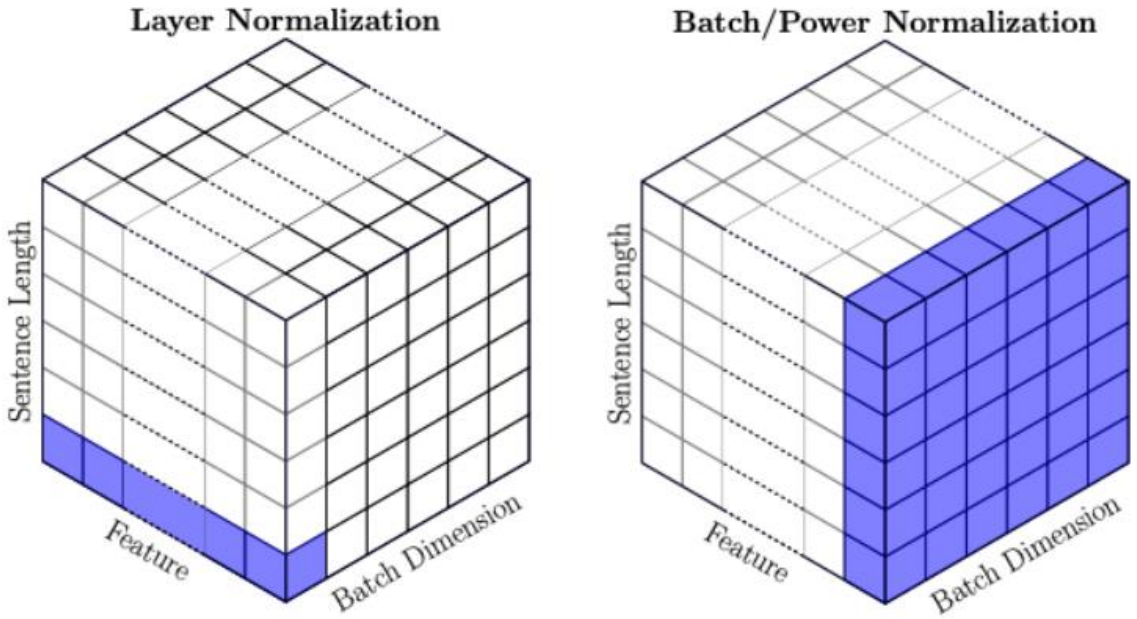
$$LN(x;\mu,\sigma) = \left\{\sigma_j \frac{x^j - \mu_x}{\sigma_x} + \mu_j\right\}_{j=\overline{1..d}}$$

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

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

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

Почему LN, а не BN?



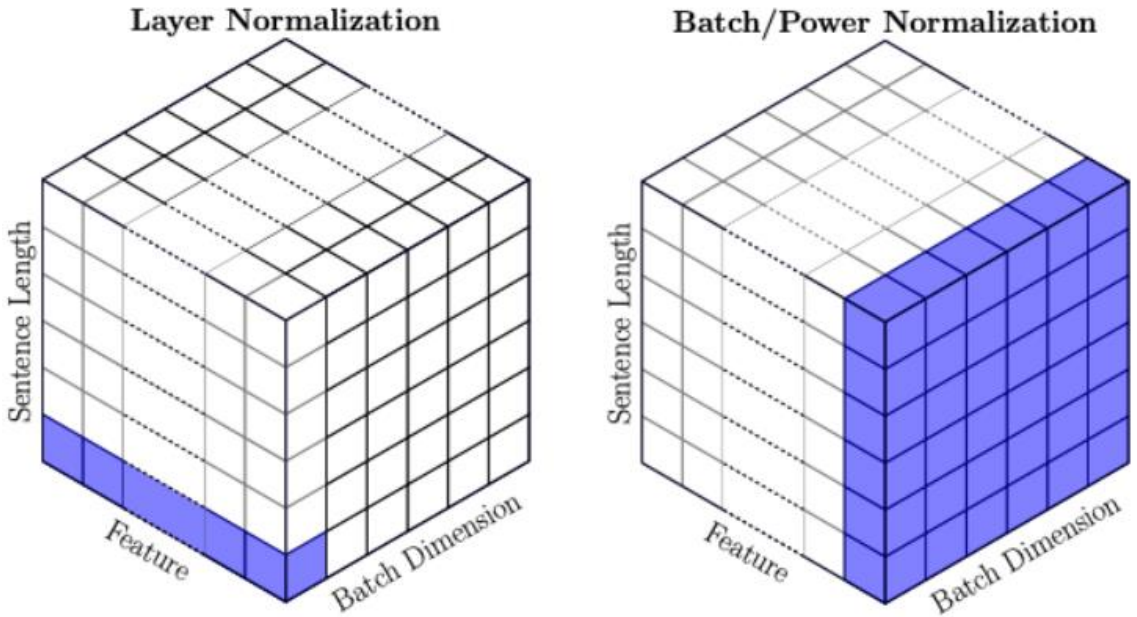
Layer normalization (нормализация слоя)

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

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

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

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



Почему LN, а не BN?

Для распараллеливания по элементам последовательности.

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).

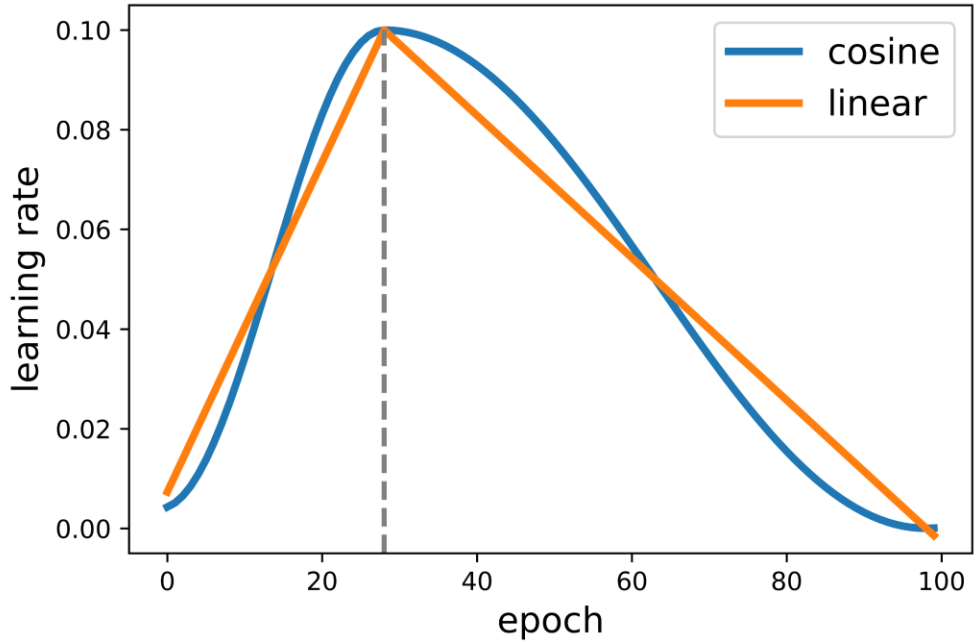
Warmup sheduler: основные стратегии

Два основных типа:

- Cosine annealing
- Linear annealing

Важные параметры:

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



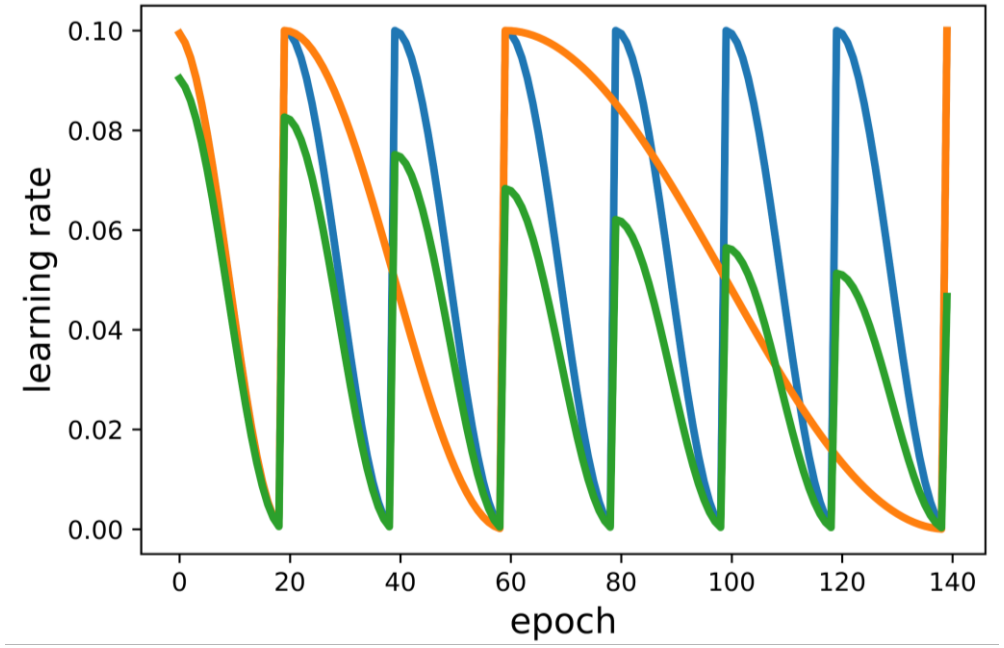
Pytorch: OneCycleLR

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?

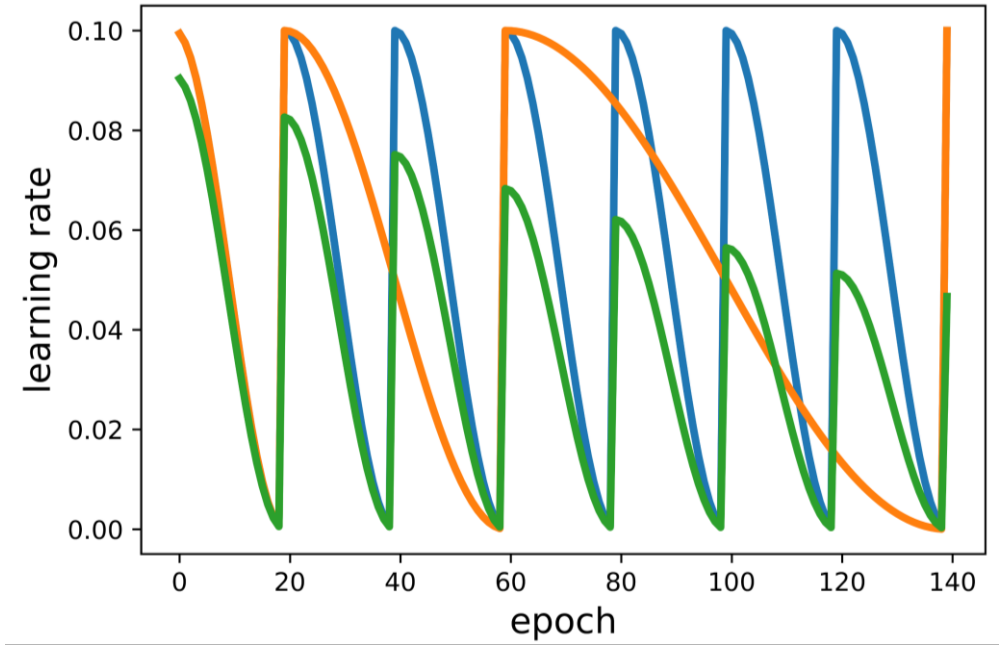
Smith. Cyclical Learning Rates for Training Neural Networks

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

Преимущества и недостатки трансформеров

+ Очень хороши во всех задачах обработки последовательностей
+ Можно строить глубокие архитектуры (20 и более слоёв)
+ Хорошо параллелятся, поэтому достаточно быстро учатся
– Применимы только на небольших последовательностях

Полезные ссылки и ресурсы

Библиотеки с моделями NER/POS:

- Spacy – пайплайны для разных задач (Tok2Vec, CNN-based), есть частичная поддержка русского языка
- UDPipe – пайплайны для разных задач (LSTM-CRF), есть частичная поддержка русского языка
- pymorphy2 – неконтекстный rule-based для русского языка
- nltk – rule-based и n-граммные модели для POS
- rnnmorph – POS для русского языка (biLSTM-CRF)
- deepavlov – NER для русского языка (transformer, BERT-based)
- natasha – NER для русского языка (CNN-CRF)

Резюме по лекции

- Задача разметки – предсказание тега для каждого элемента входной последовательности
- Основные примеры задачи – задачи POS и NER
- Стандартный бейзлайн – rule-based подход
- Основные архитектуры – рекуррентные нейронные сети (LSTM) и энкодер трансформера
- LSTM в разметке: двунаправленность, 1-3 слоя, clipping
- Transformer в разметке: любое число слоёв, warmup
- Если ваша задача разметки популярна, используйте готовое решение или хотя бы готовую архитектуру