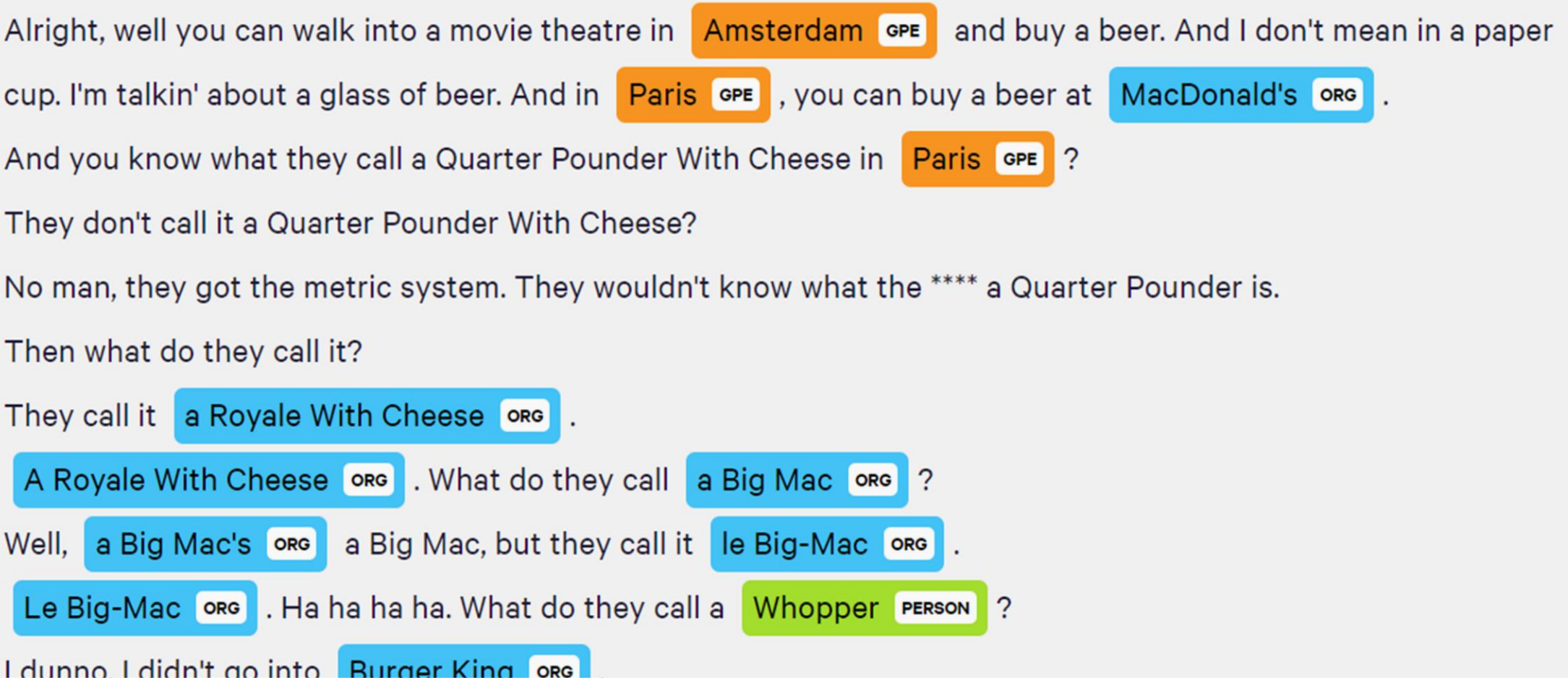


Задача разметки последовательности. Рекуррентные нейронные сети.

Попов Артём, OzonMasters, осень 2022

Natural Language Processing

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

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)

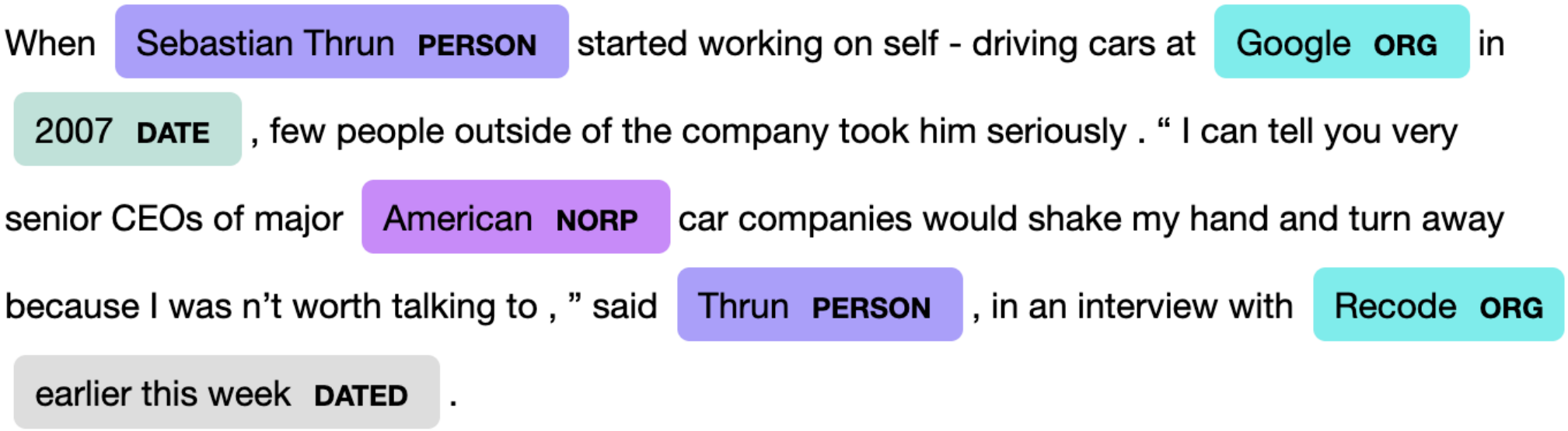
Для каждого слова в предложении определить, является ли оно частью именованной сущности.

Сложная задача, но частично решается при помощи словарей.

Зачем нужно распознавание именованных сущностей?

- Деперсонализация данных
- Проставление тегов к новостям
- Диалоговые системы (“хочу записаться в барбер-шоп Борода”)
- Поиск

Пример работы NER



В зависимости от приложения, категориями в NER могут быть:

- персона, локация, организация, дата-время
- email адрес, телефонный номер, url адрес
- заболевание, симптом, лекарство, вещество, метод лечения

Разметка семантических ролей (semantic role labeling)

Выделить семантические роли у именных групп в предложении (отношения участников к ситуации, обозначаемых глаголами).

Сложная задача, плохо решается rule-based методами.

Примеры семантических ролей:

- агенс – одушевлённый инициатор действия, контролирующий его
- пациенс – участник, на которого направлено действие
- реципиент – участник, чьи интересы затронуты в процессе ситуации
- инструмент –посредством которого осуществляется действие

Мальчик ударил собаку палкой. Я отдал сестре книгу.

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

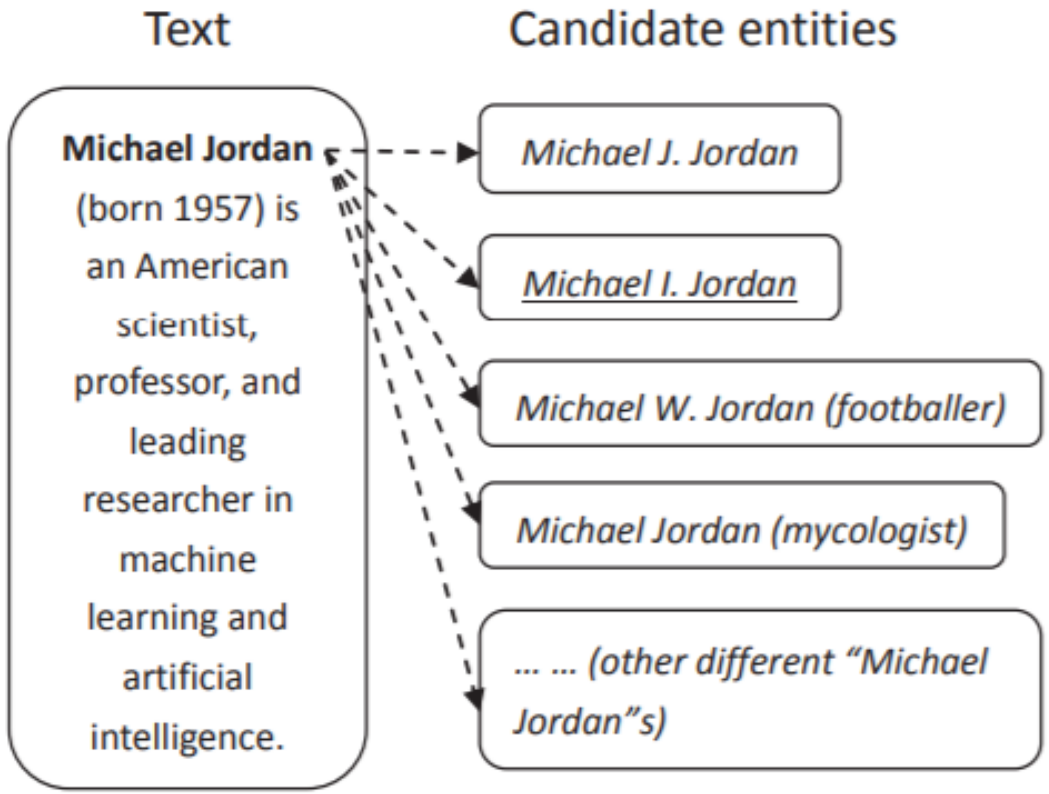
Микро-метрики можно считать по каждому типу сущностей.

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

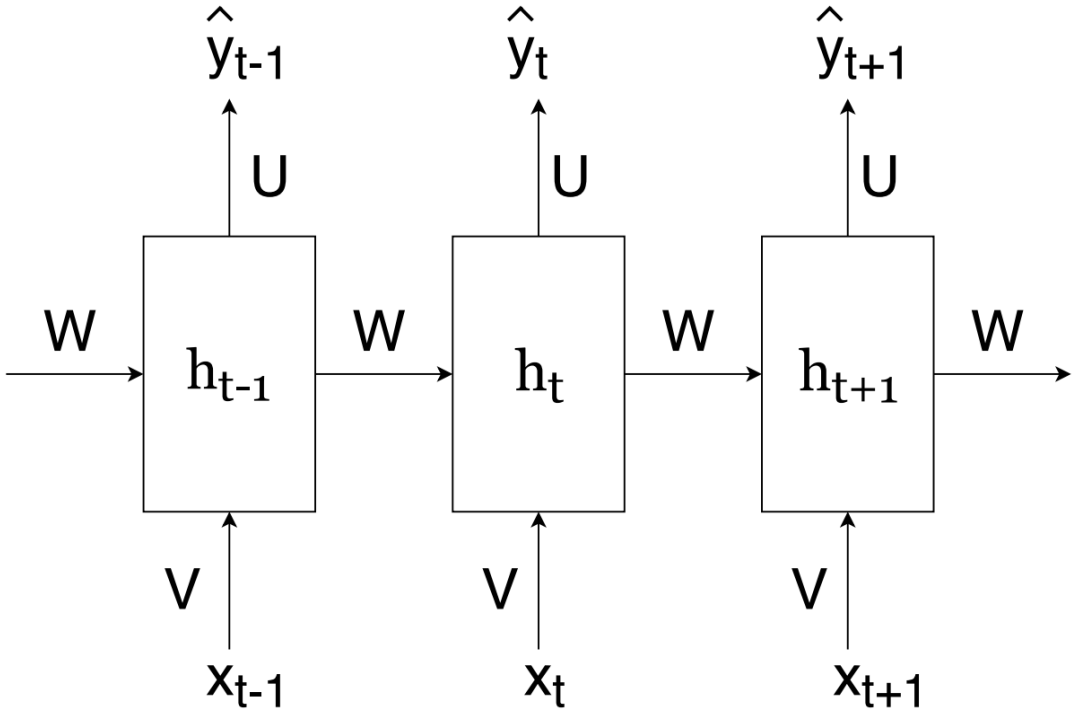
Как собрать выборку для обучения / тестирования?

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:

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

$$\frac{\partial \mathcal{L}_t}{\partial W} =$$

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:

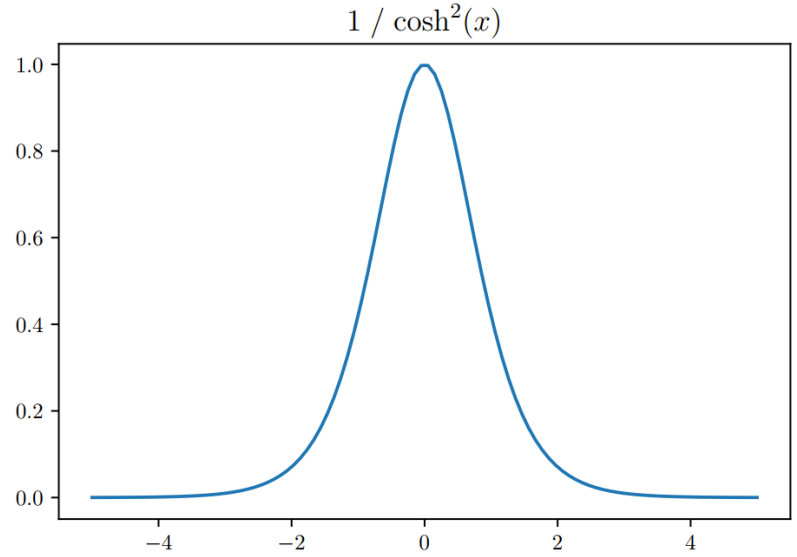
$$\frac{\partial \mathcal{L}_t}{\partial W} = \frac{\partial \mathcal{L}}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{dh_t}{dW}$$

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

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

- Регуляризация $\frac{\partial h_i}{\partial h_{i-1}} \to 1$ (не популярно)
- Truncated Backpropagation Through Time

Gradient clipping

Ограничение нормы градиентов:

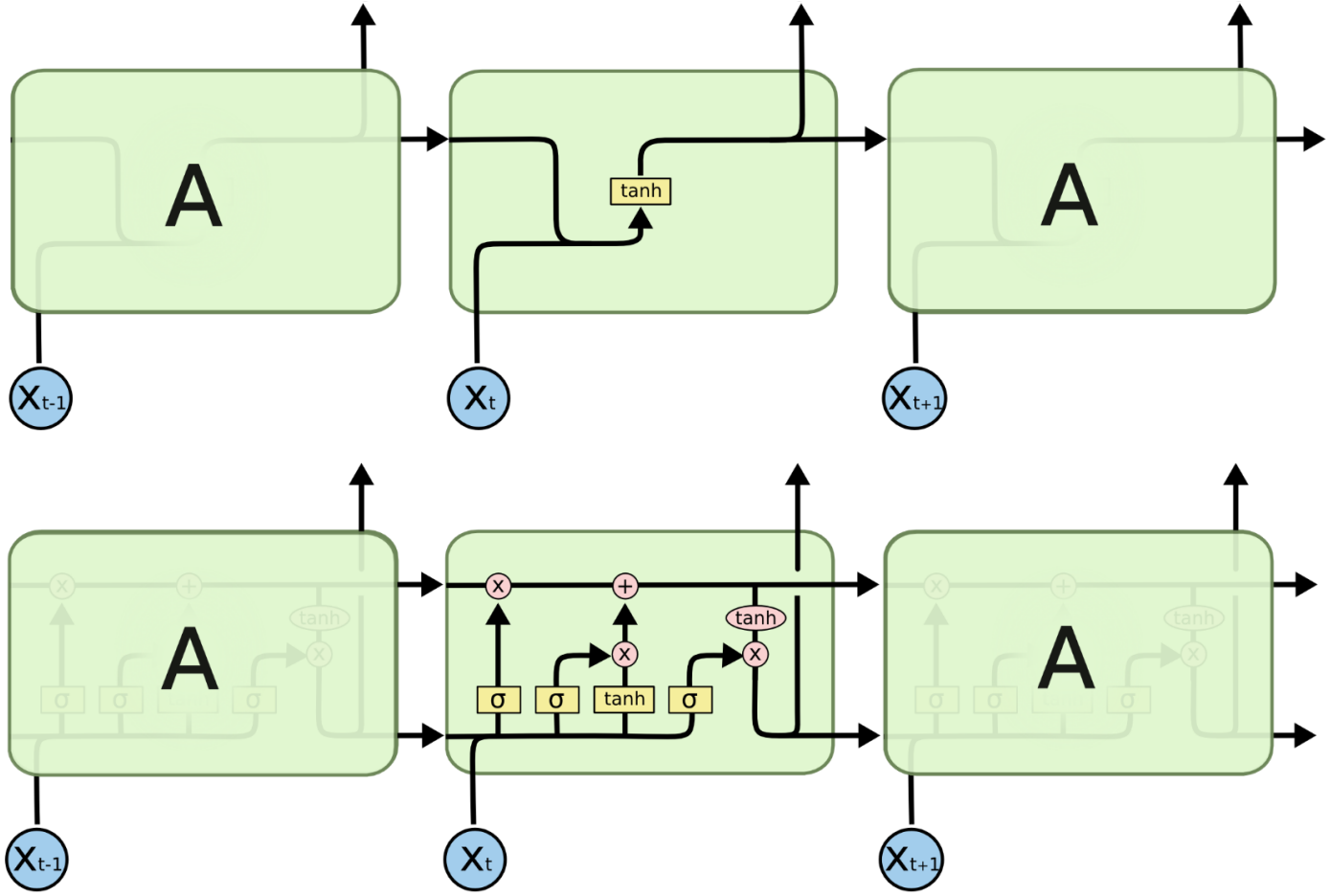
Algorithm 1 Pseudo-code for norm clipping the gradients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
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.

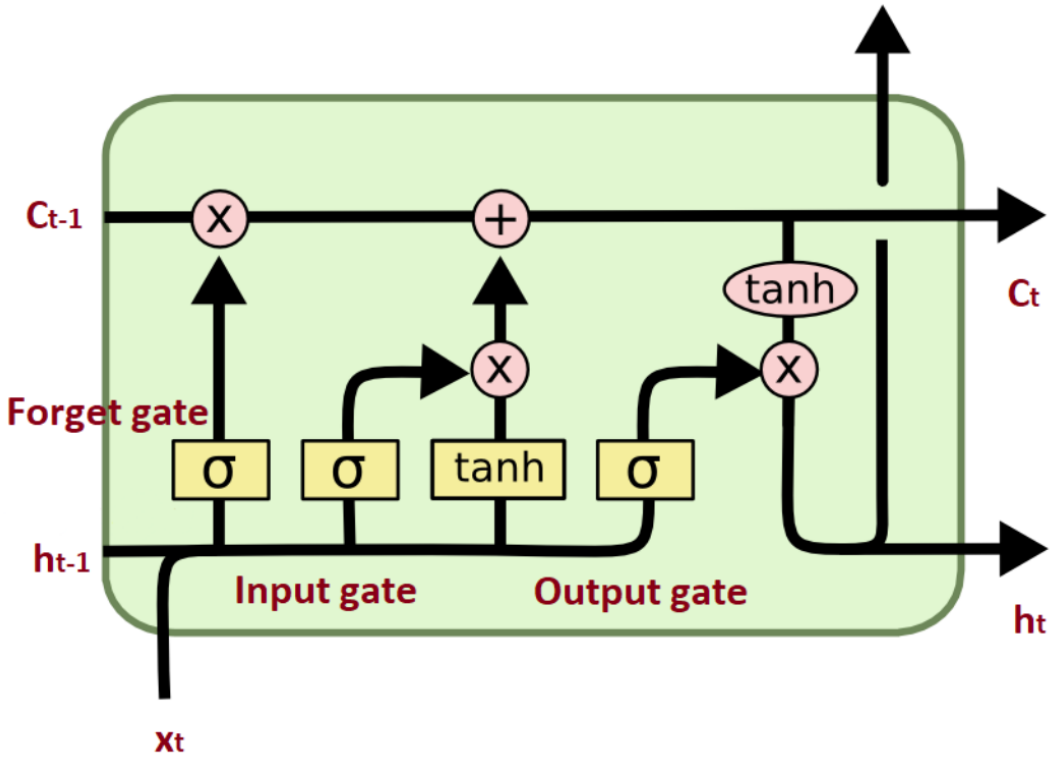
LSTM сеть

Идея. Хотим сделать более сложную структуру ячейки.



Hochreiter and Schmidhuber (1997); Long Short-Term Memory

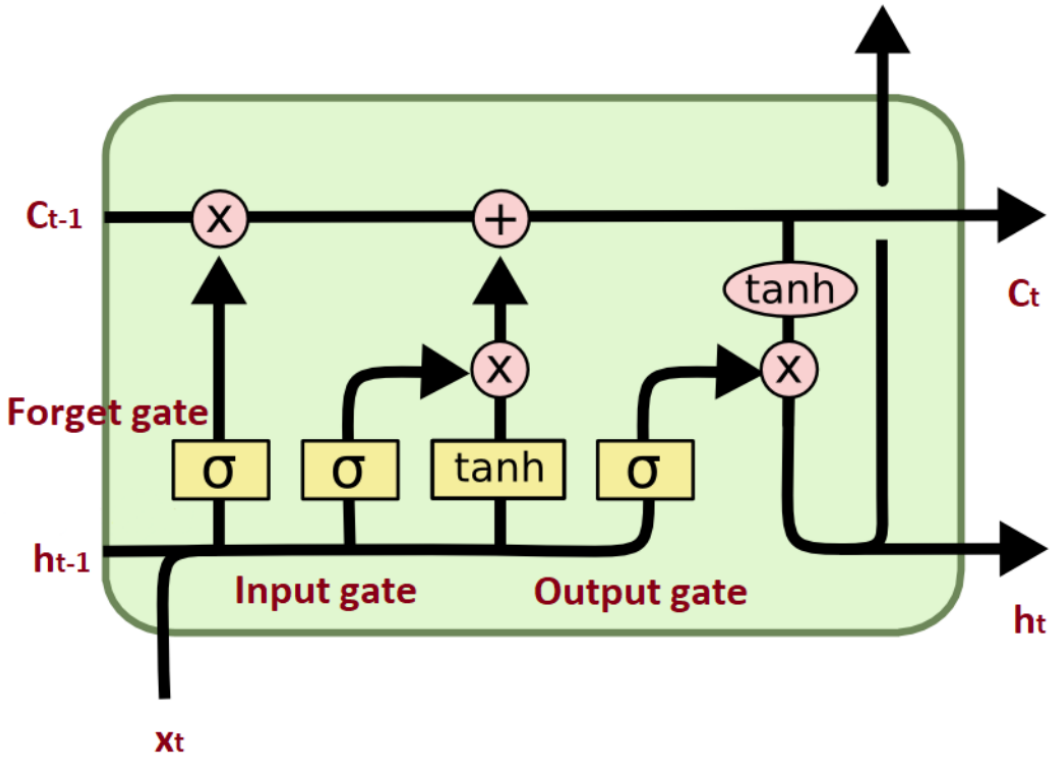
LSTM ячейка



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

За счёт чего решается проблема затухания градиента?

LSTM ячейка

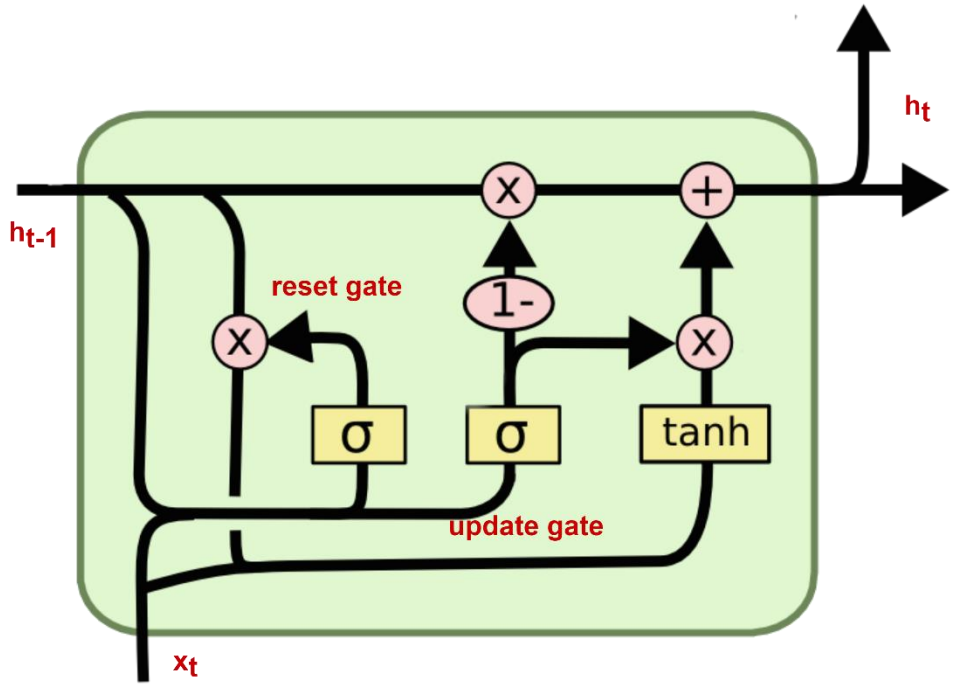


$$z_t = [h_{t-1}, x_t]$$
$$f_t = \sigma(W_f \cdot z_t + b_f)$$
$$i_t = \sigma(W_i \cdot z_t + b_i)$$
$$\hat{C}_t = \text{th}(W_c \cdot z_t + b_c)$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t$$
$$o_t = \sigma(W_o \cdot z_t + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$

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

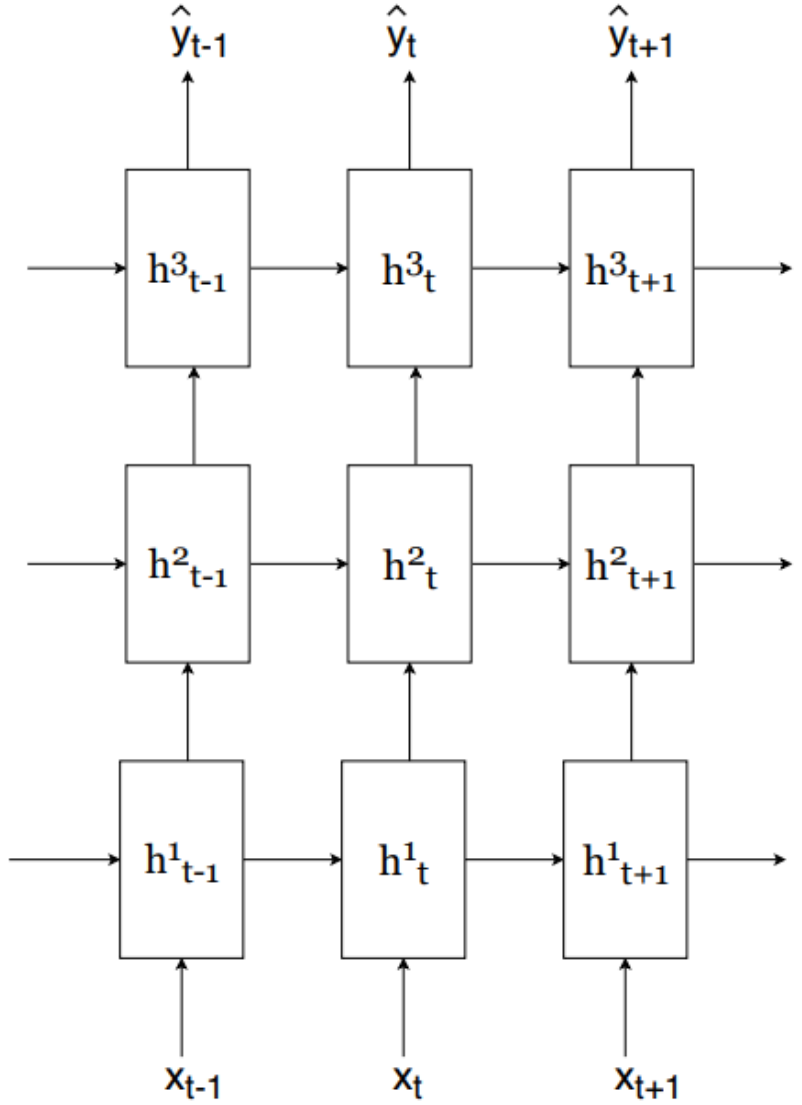
\+ Качество на уровне LSTM

Cho et al (EMNLP 2014); Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation

Глубокие рекуррентные сети (deep RNN)

Подача выходов одной рекуррентной сети на вход другой.

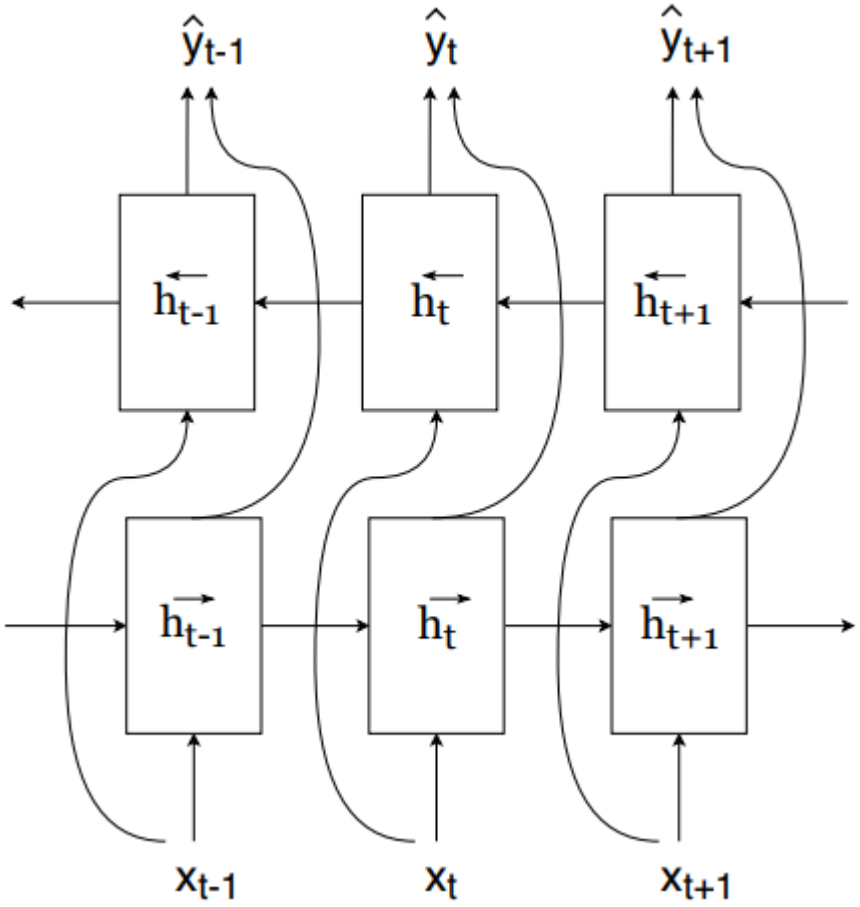
$$h_t^1 = GRU\left(h_{t-1}^1, x_t\right)$$
$$h_t^2 = GRU\left(h_{t-1}^2, h_t^1\right)$$
$$h_t^3 = GRU\left(h_{t-1}^3, h_t^2\right)$$
$$\hat{y}_t = g\left(Uh_t^3 + \hat{b}\right)$$



Двунаправленные сети (bidirectional)

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

$$\overrightarrow{h_t} = \overrightarrow{GRU}(\overrightarrow{h_{t-1}}, x_t)$$
$$\overleftarrow{h_t} = \overleftarrow{GRU}(\overleftarrow{h_{t+1}}, x_t)$$
$$h_t = biGRU_t\left(x, \vec{h}, \overleftarrow{h}\right) = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]$$
$$\hat{y}_t = g(Uh_t + \hat{b})$$



Двунаправленные сети (bidirectional)

При решении классификации, необходимо применять линейный слой к конкатенации первого и последнего выхода:

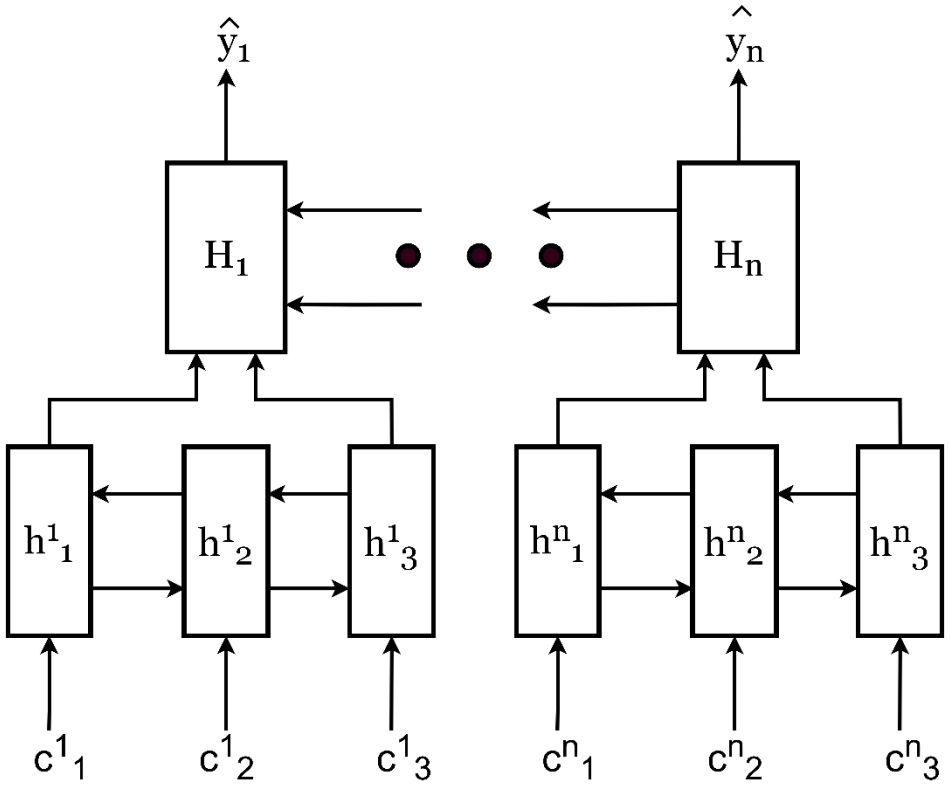
$$biGRU_{emb}(h, x) = \left[\overrightarrow{h_n}, \overleftarrow{h_1}\right]$$

$$y(x) = g\left(U \times biGRU_{emb}(h, x) + \hat{b}\right)$$

В задачах разметки и классификации предпочтительнее использовать двунаправленные сети.

Есть приложения, где двунаправленные сети использовать нельзя (языковое моделирование).

Иерархические сети (hierarchical rnn)



Вход: последовательность из последовательностей
$x_i = [c_1^i, \dots, c_k^i]$

1. Построение эмбеддинга по вложенной последовательности x_i

$$X_i = biGRU_{emb}(h_i, x_i)$$

2. Обработка основной последовательности

$$H_t = biGRU_t\left(X, \vec{H}, \overleftarrow{H}\right)$$
$$\hat{y}_t = g(UH_t + \hat{b})$$

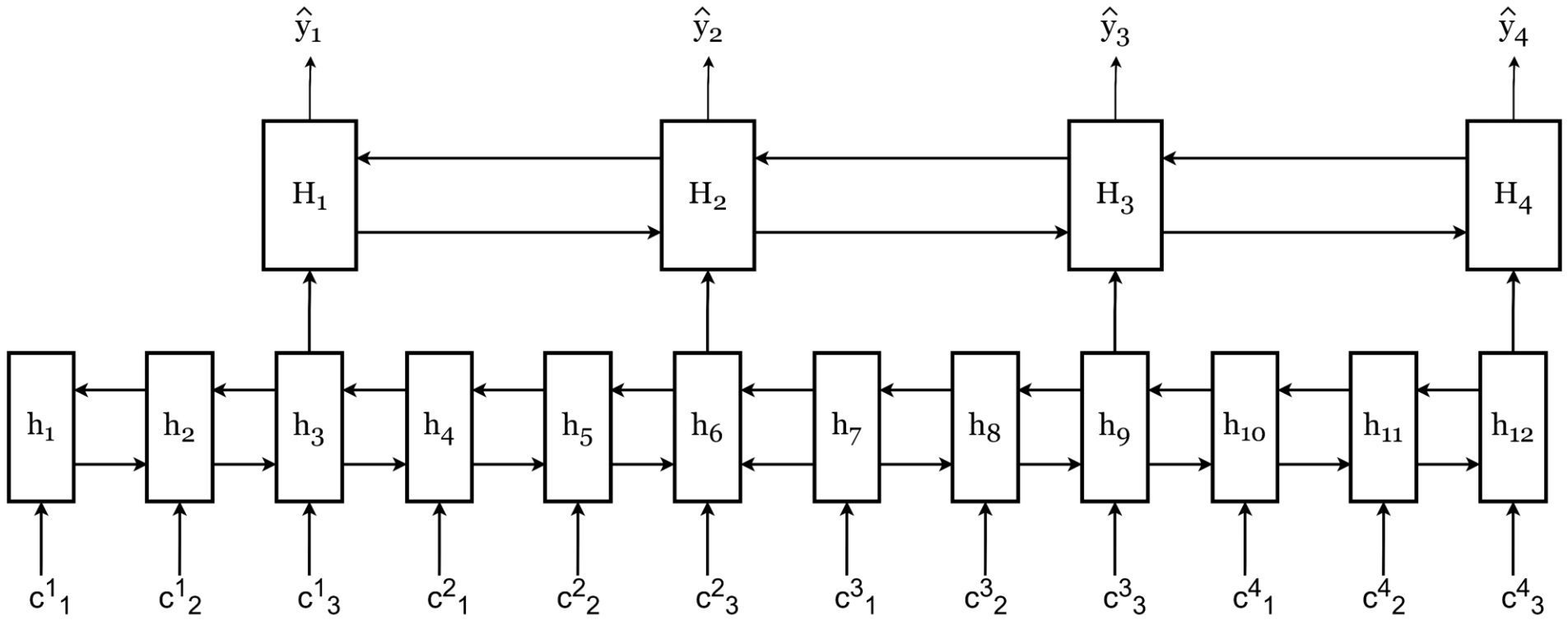
Иерархические сети (hierarchical rnn)

- Учится end-to-end
- Часто используется для работы с OOV словами: вложенная последовательность – символы слов, основная – слова.
- Можно конкатенировать выход вложенной рекуррентной сети с табличным представлением (предобученным или обучаемым)
- На разных уровнях могут быть сети разной направленности и разной глубины
- На разных уровнях могут быть сети разной архитектуры
- Уровней может быть больше двух (но на практике редко)
- Иногда под иерархическими сетями понимают другое…

Иерархические сети (посимвольные)

Вместо того, чтобы обрабатывать вложенные последовательности по отдельности, можно обрабатывать их конкатенацию:

$$c = \left[c_1^1, \ldots, c_k^1, \ldots, c_1^n, \ldots, c_k^n\right], \qquad h_j = biGRU_j\left(c, \vec{h}, \overleftarrow{h}\right), \qquad X_t = \left[\overrightarrow{h_{kt}}, \overleftarrow{h_{kt}}\right]$$



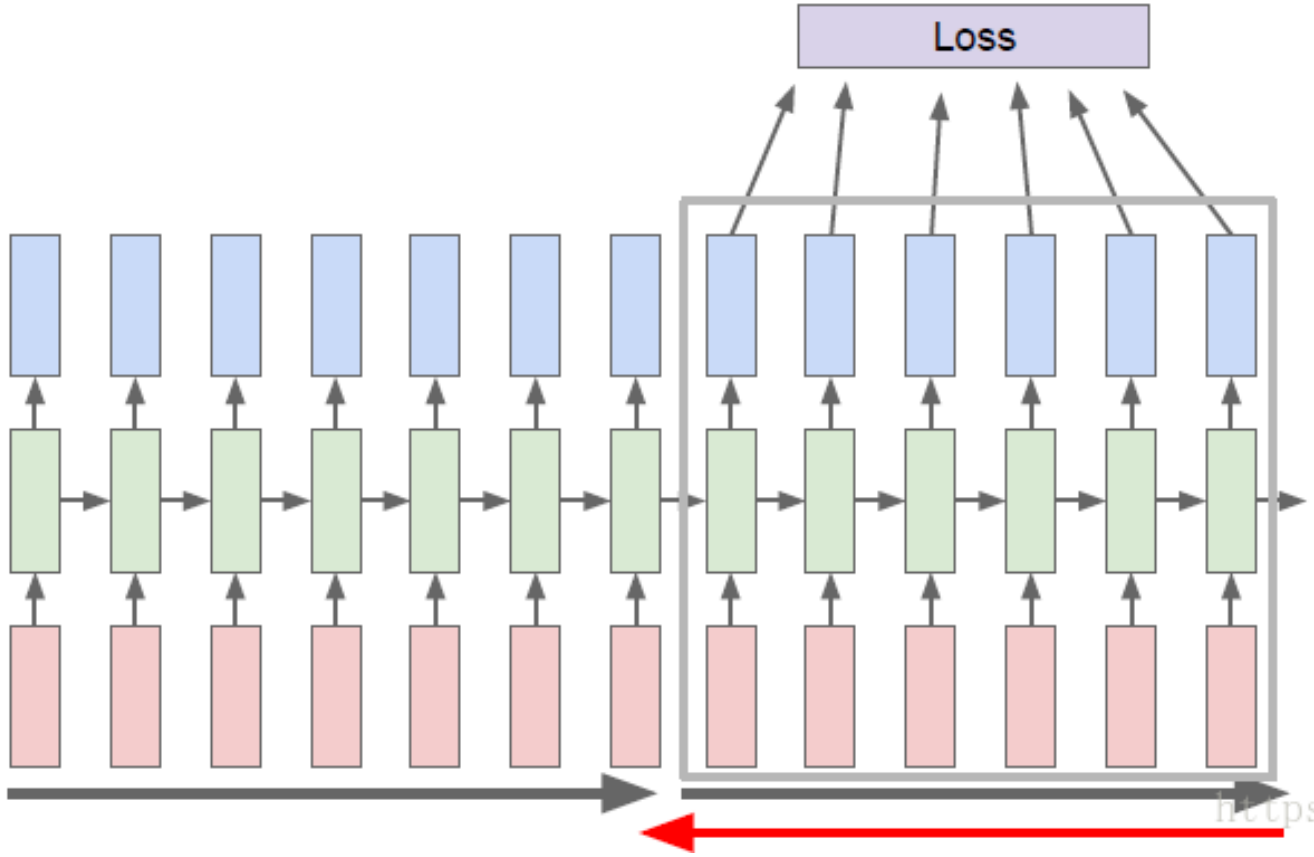
Akbik et al (COLING 2018); Contextual String Embeddings for Sequence Labeling

Truncated backpropagation through time

TBTT – способ избежать взрывов и затуханий градиентов на длинных последовательностях.

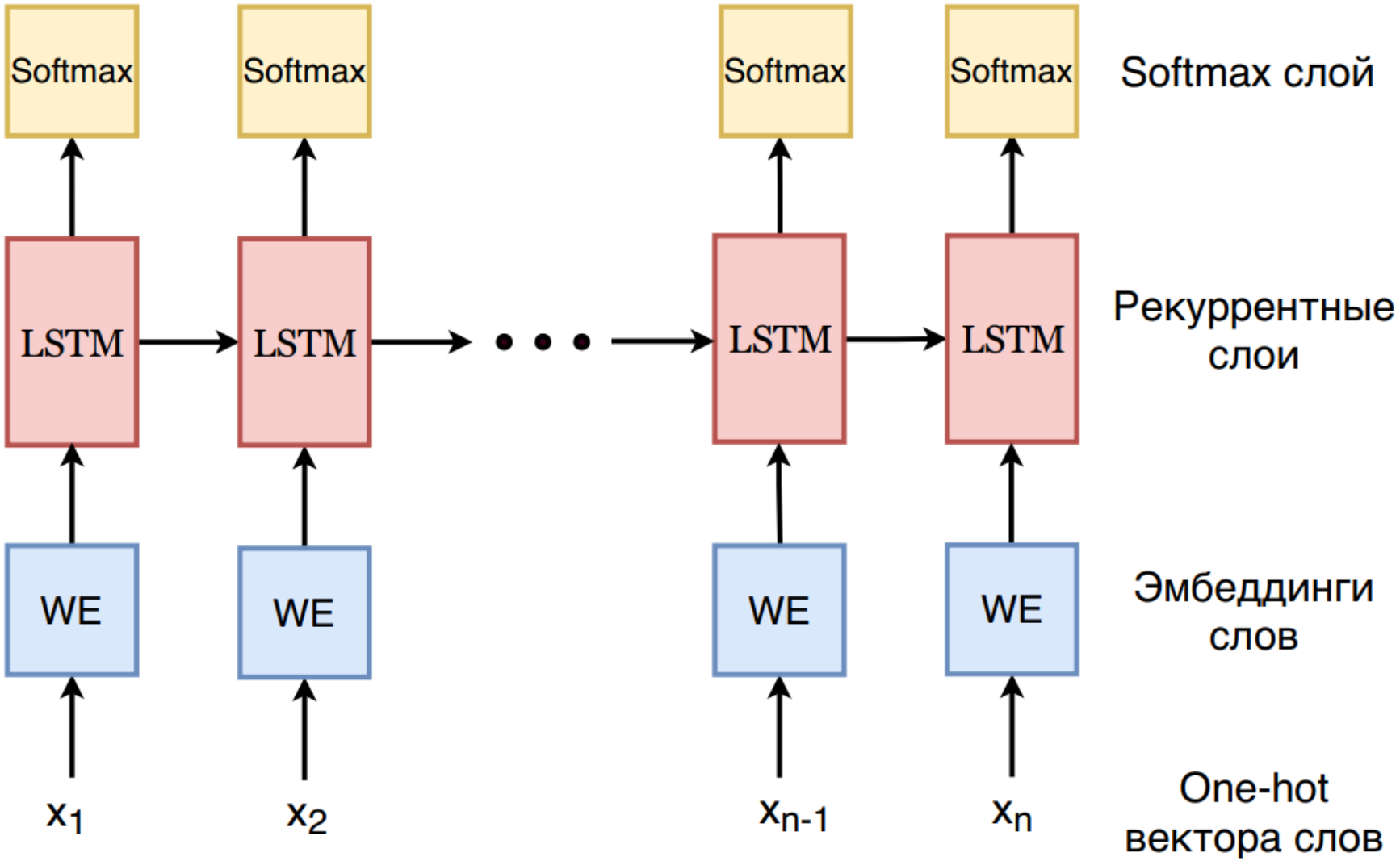
1. Входная последовательность разбивается на части
$$X_1 = [x_1, \dots, x_k], \dots, X_N = [x_{n-k}, \dots, x_n]$$
2. После обработки i-ой части применяем backpropagation на всех элементах i-ой части, делаем шаг оптимизации
$$\sum_{t=k(i-1)+1}^{ki} \mathcal{L}(y_t, \hat{y_t}) \rightarrow \min$$
3. Забываем градиенты, но оставляем значения внутренних состояний h_{ki} для обработки следующей части

Truncated backpropagation through time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Теггер на основе RNN



Теггер на основе RNN

Этап 1. Вычисление вероятностей меток

$$h_t = GRU(h_{t-1}, x_t)$$
$$\hat{y}_t = softmax(Uh_t + \hat{b})$$

Этап 2. Вычисление меток из множества классов Y:

$$y_t = \arg\max_{Y} \hat{y}_t$$

Есть ли в этой схеме какие проблемы?

Теггер на основе RNN

Этап 1. Вычисление вероятностей меток

$$h_t = GRU(h_{t-1}, x_t)$$
$$\hat{y}_t = softmax(Uh_t + \hat{b})$$

Этап 2. Вычисление меток из множества классов Y:

$$y_t = \arg\max_{Y} \hat{y}_t$$

Есть ли в этой схеме какие проблемы?

Нет никакой связи между предсказаниями соседних элементов.

Одно из решений этой проблемы обсудим на следующей лекции.

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

Скорость (распараллеливание): CNN > трансформер > RNN

Качество: трансформер > RNN > CNN

Память: RNN (однонаправленная) > CNN > трансформер

Итоги занятия

- Задача разметки – предсказание тега для каждого элемента входной последовательности
- Основные примеры задачи – задачи POS и NER
- Стандартный бейзлайн – rule-based подход
- Одна из возможных архитектур – рекуррентные нейронные сети (LSTM или GRU)
- LSTM в разметке: двунаправленность, 1-3 слоя, clipping
- Если ваша задача разметки популярна, используйте готовое решение или хотя бы готовую архитектуру