Применение RNN для языковых моделей и распознавания именованных сущностей

курс «Математические методы анализа текстов»

Попов Артём Сергеевич

19 сентября 2018 г.

Задача языкового моделирования

Хотим оценивать вероятность появления последовательности слов $(w_1, \ldots, w_n)$ в тексте.

Цепное правило (chain rule):

$$p(w_1, \ldots, w_n) = p(w_n | w_{n-1}, \ldots, w_1) \ldots p(w_2 | w_1) p(w_1)$$

Другая постановка: для любого слова w оценить вероятность появления слова после последовательности слов $(w_1, \ldots w_{n-1})$.

Задача моделирования последовательности

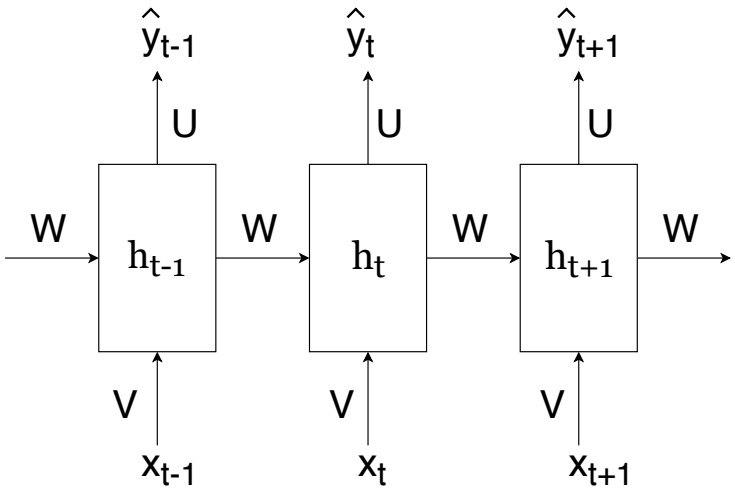
Дано: $\{x_1, \ldots, x_n\}$ — последовательность входных векторов
$\{y_1, \ldots, y_n\}$ — последовательность выходных векторов

Хотим: для любой последовательности входов предсказывать последовательность выходов

Как можно работать с последовательностями?

- Обучение отдельного классификатора на признаках, зависящих от позиции элемента в последовательности
- Графические модели (HMM/CRF)
- Рекуррентные нейронные сети (RNN, LSTM, GRU)
- Комбинация подходов

Модель рекуррентной нейронной сети (RNN)



h_t — скрытое состояние в момент t

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

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

Обучение сети — минимизация суммарных потерь:

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

Сеть обучается с помощью алгоритма Bakpropagation[1]

[1] Часто, вариацию алгоритма Bakpropagation для обучения RNN называют Bakpropagation through time

Детали обучения RNN: производные по U и W

Градиент по U зависит только от величин в момент t:

$$\frac{d\mathcal{L}_t}{dU} =$$

Детали обучения RNN: производные по U и W

Градиент по U зависит только от величин в момент t:

$$\frac{d\mathcal{L}_t}{dU} = \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial U}$$

Градиент по W зависит от всех предыдущих величин:

$$\frac{d\mathcal{L}_t}{dW} =$$

Детали обучения RNN: производные по U и W

Градиент по U зависит только от величин в момент t:

$$\frac{d\mathcal{L}_t}{dU} = \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial U}$$

Градиент по W зависит от всех предыдущих величин:

$$\frac{d\mathcal{L}_t}{dW} = \frac{\partial \mathcal{L}_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\frac{dh_t}{dW}$$

$$\frac{dh_t}{dW} = \frac{\partial h_t}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}}\frac{dh_{t-1}}{dW} =$$

$$= \frac{\partial h_t}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{dh_{t-2}}{dW} =$$

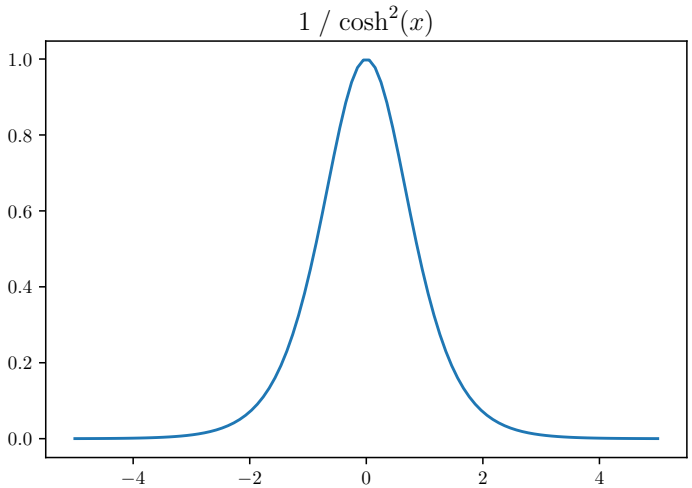
$$= \ \ldots \ = \sum_{k=1}^{t}\left(\prod_{i=k+1}^{t}\frac{\partial h_i}{\partial h_{i-1}}\right)\frac{\partial h_k}{\partial W}$$

Градиент по V считается аналогично градиенту по W

Детали обучения RNN: взрыв и затухание градиентов

Взрыв градиента:

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



Затухание градиента:

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

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

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

если $f = \tanh$

Популярные способы борьбы с взрывом/затуханием:

- Gradient clipping (против взрыва)
- Модели LSTM и GRU (против затухания)

Gradient clipping

Ограничение нормы градиентов:

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

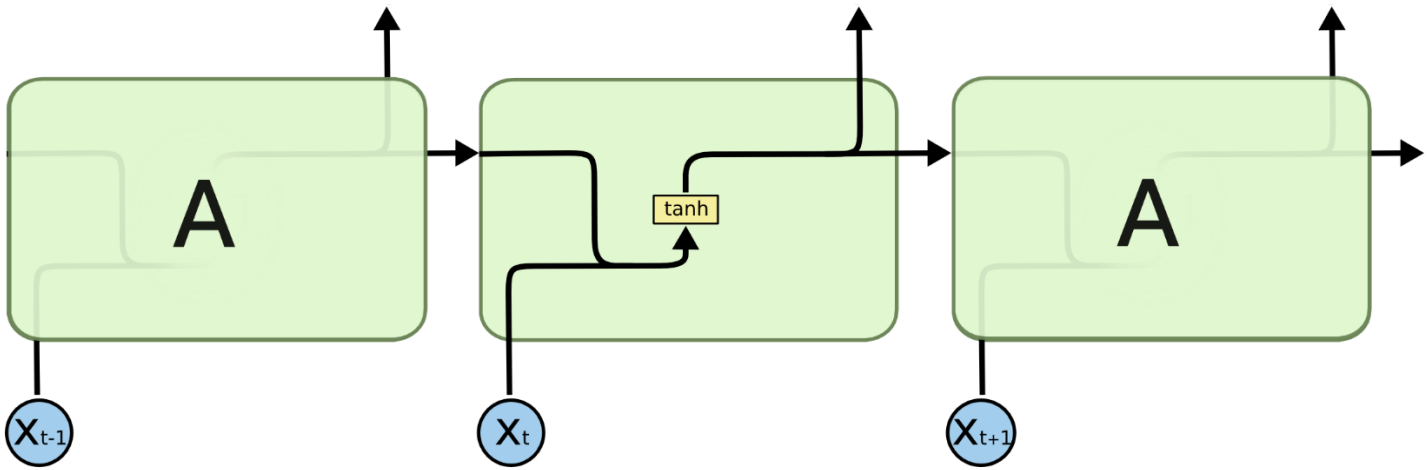
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

Как выбрать порог?
Например, брать среднюю норму градиента для весов по запускам без gradient clipping

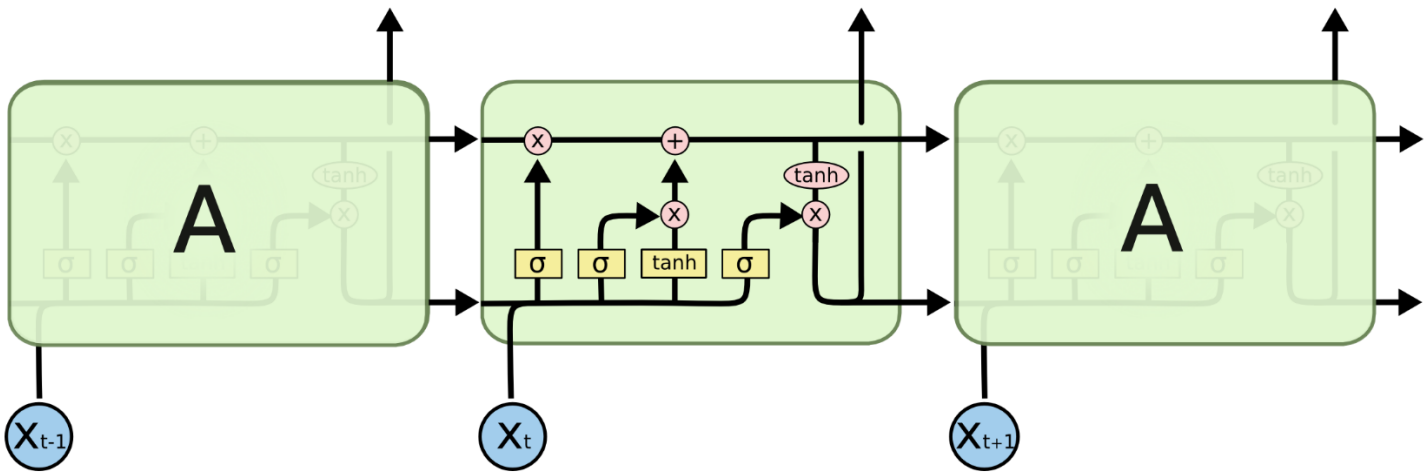
LSTM сеть

Используем более сложную структуру ячейки:

RNN

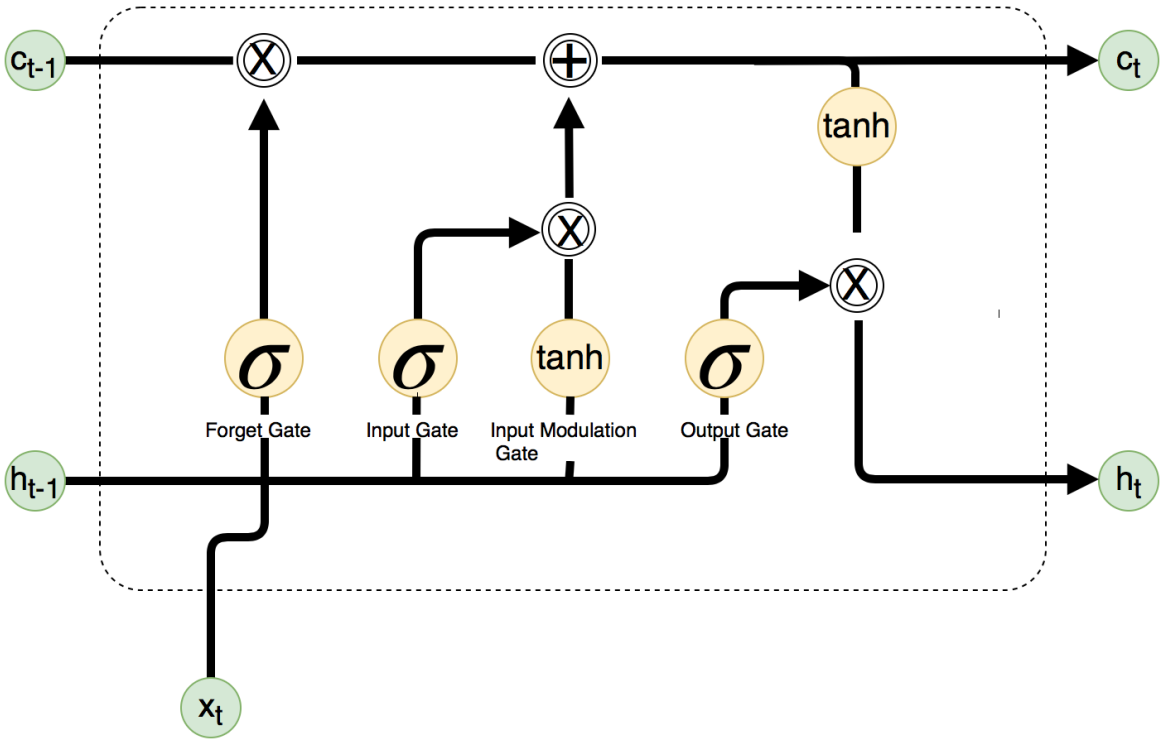


LSTM



LSTM ячейка

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

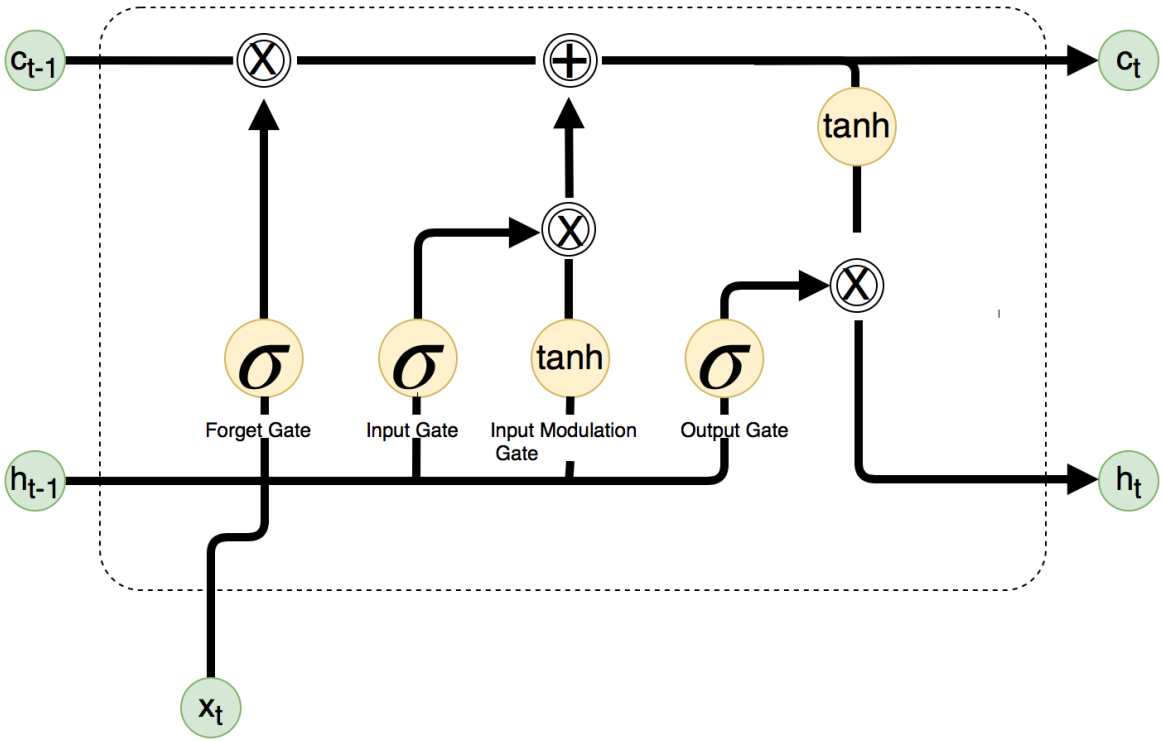


Обучается с помощью алгоритма Backpropagation

Почему решает проблему затухающих градиентов?

LSTM ячейка

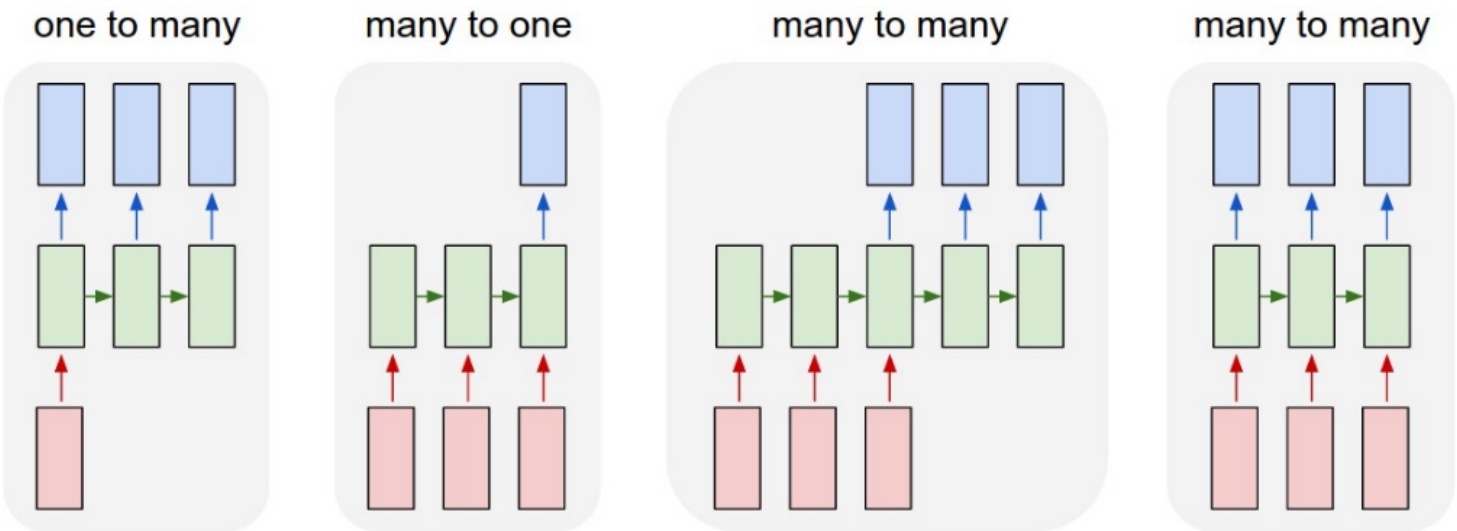
$$z_t = [h_{t-1}, x_t]$$
$$f_t = \sigma(W_f \cdot z_t + b_f)$$
$$i_t = \sigma(W_i \cdot z_t + b_i)$$
$$\hat{C}_t = \text{th}(W_c \cdot z_t + b_c)$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{C}_t$$
$$o_t = \sigma(W_o \cdot z_t + b_o)$$
$$h_t = o_t \cdot \tanh(C_t)$$



Обучается с помощью алгоритма Backpropagation

Почему решает проблему затухающих градиентов? Частично потому, что C_t зависит от C_{t-1} линейно, т.е $\frac{\partial C_t}{\partial C_{t-1}} = f_t$

Разные архитектуры рекуррентных сетей



Примеры задач:

one to many Генерация описания изображения

many to one Классификация предложений

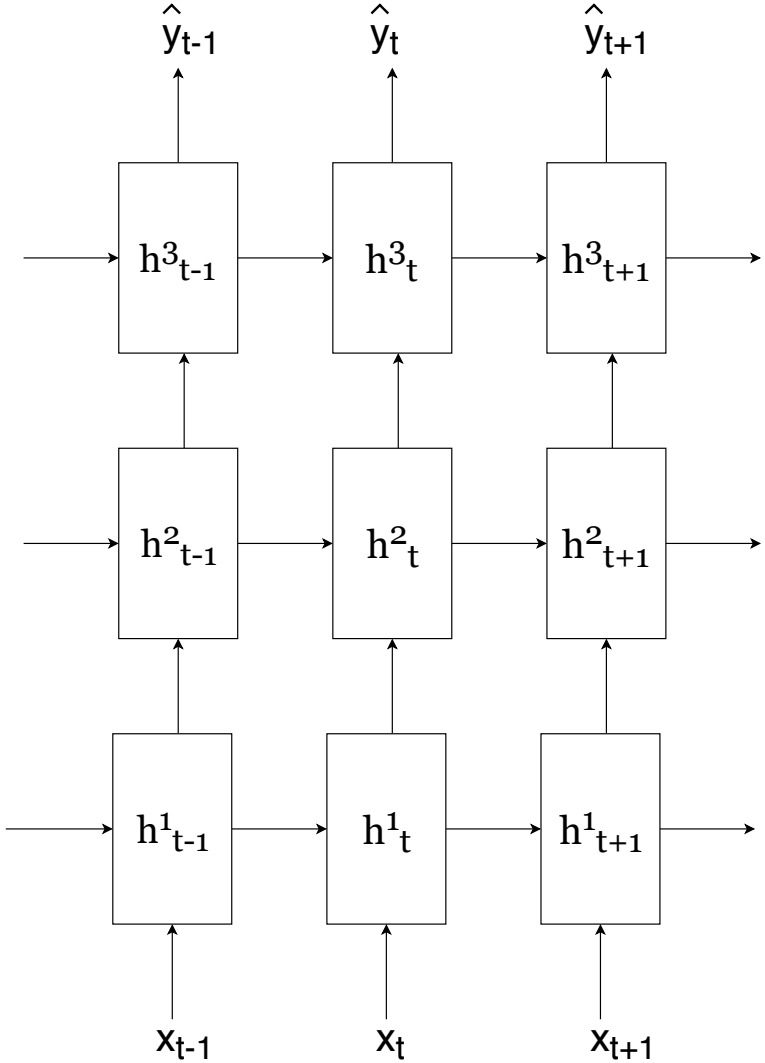
many to many(1) Перевод с одного языка на другой

many to many(2) Определение частей речи

Глубокие рекуррентные сети (deep RNN, layers stacking)

Выходы одной рекуррентной сети подаются на вход другой:

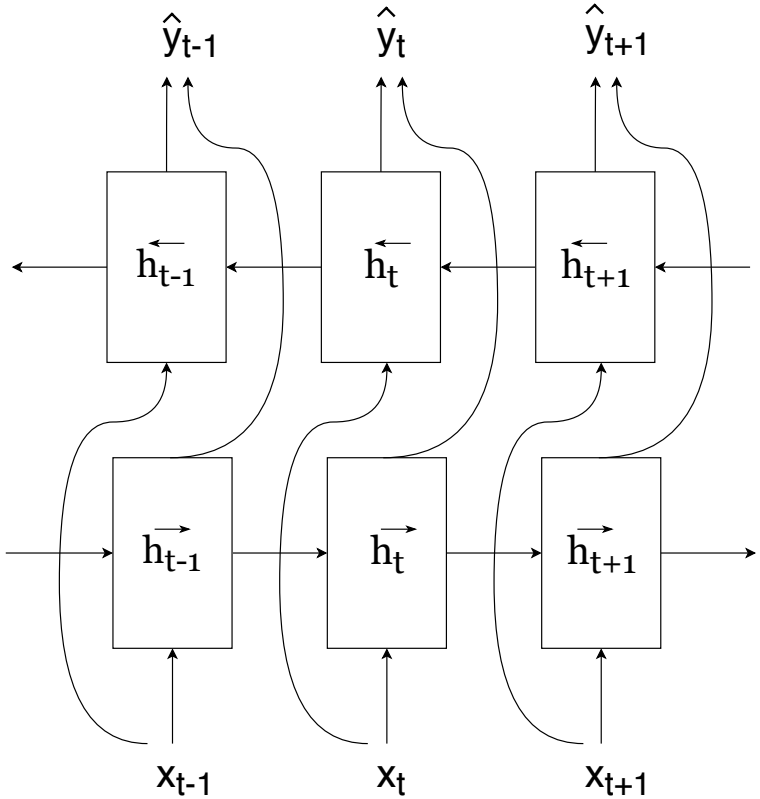
$$h_t^1, C_t^1 = LSTM(h_{t-1}^1, C_{t-1}^1, x_t)$$
$$h_t^2, C_t^2 = LSTM(h_{t-1}^2, C_{t-1}^2, h_t^1)$$
$$h_t^3, C_t^3 = LSTM(h_{t-1}^1, C_{t-1}^1, h_t^2)$$
$$y_t = g(Uh_t^2 + \hat{b})$$



Двунаправленные сети (biderectional)

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

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



На практике часто работают лучше чем однонаправленные!

Резюме по RNN

- RNN — Нейросетевая архитектура для работы с последовательностями
- Обучается с помощью алгоритма Backpropagation
- В исходном виде RNN сложно обучается, необходимо использовать LSTM (или GRU, или другие модификации) и gradient clipping
- С помощью разных архитектур сети можно решать разные задачи

Задача языкового моделирования (language modeling)

Хотим уметь оценивать вероятность $p(w|w_n, \ldots, w_1)$

Предположение марковости (Markov assumption):

$$p(w_n|w_{n-1}, \ldots, w_1) \approx p(w_n|w_{n-1}, \ldots w_{n-k})$$

Идея: моделировать $p(w|w_{n-1}, \ldots w_{n-k})$ с помощью RNN

Почему не моделируем $p(w|w_{n-1}, \ldots, w_1)$?

Задача языкового моделирования (language modeling)

Хотим уметь оценивать вероятность $p(w|w_n, \ldots, w_1)$

Предположение марковости (Markov assumption):

$$p(w_n|w_{n-1}, \ldots, w_1) \approx p(w_n|w_{n-1}, \ldots w_{n-k})$$

Идея: моделировать $p(w|w_{n-1}, \ldots w_{n-k})$ с помощью RNN

Почему не моделируем $p(w|w_{n-1}, \ldots, w_1)$?
Из-за проблемы взрывающихся/затухающих градиентов не можем обрабатывать слишком длинные последовательности + проще работать с последовательностями одинаковой длины

Обозначения

W — множество всех слов, $|W|$ — мощность множества
Слово w_i — вектор $[0, \dots, 0, \underbrace{1}_{i}, 0, \dots, 0]$ длины $|W|$

Применение линейного слоя к one-hot вектору:

$$Vw_i = V_i, \quad V_i \text{ — эмбединг слова } w_i$$

Операция $\mathrm{softmax}$ (мягкий максимум):

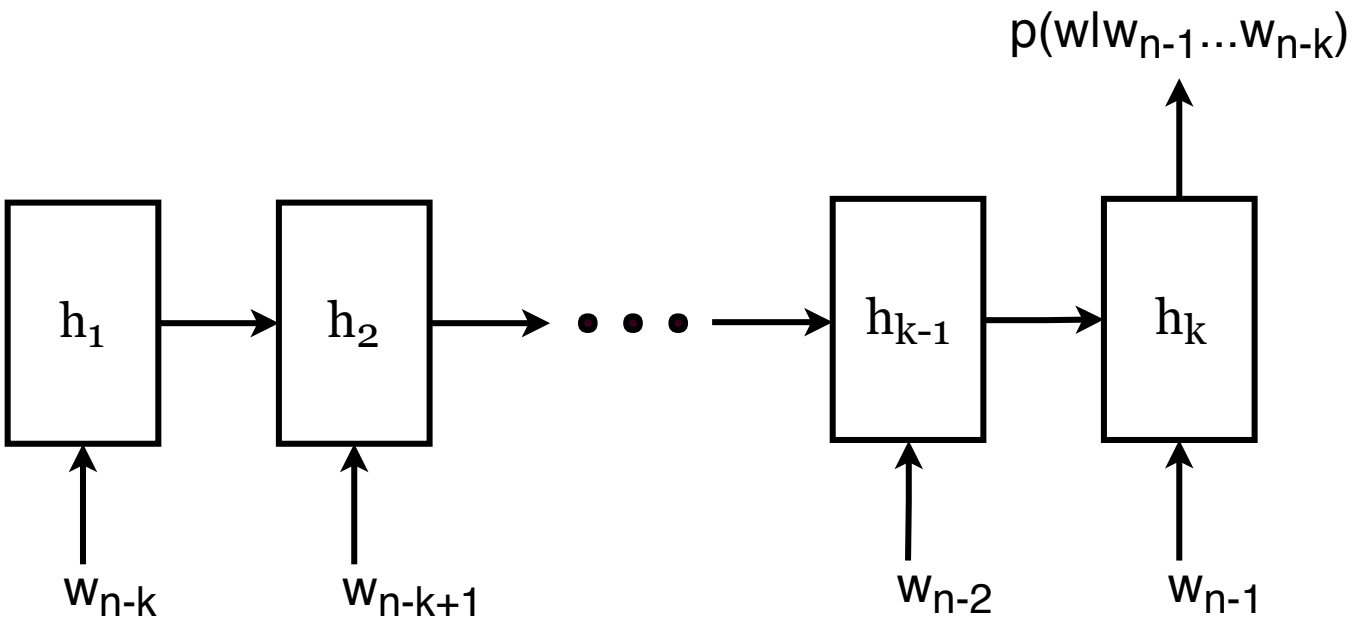
$$\mathrm{softmax}\, x = \left\{ \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)} \right\}_{i=1}^{d} \qquad x \in \mathbb{R}^d$$

$\mathrm{softmax}$ преобразует вектор в дискретное распределение:

$$\hat{y}_t = p(w|w_{n-1}, \dots, w_{n-t}) = \mathrm{softmax}(Uh_t + \hat{b})$$

$$h_t, C_t = LSTM(h_{t-1}, C_{t-1}, w_t)$$

RNN для LM с одним выходом

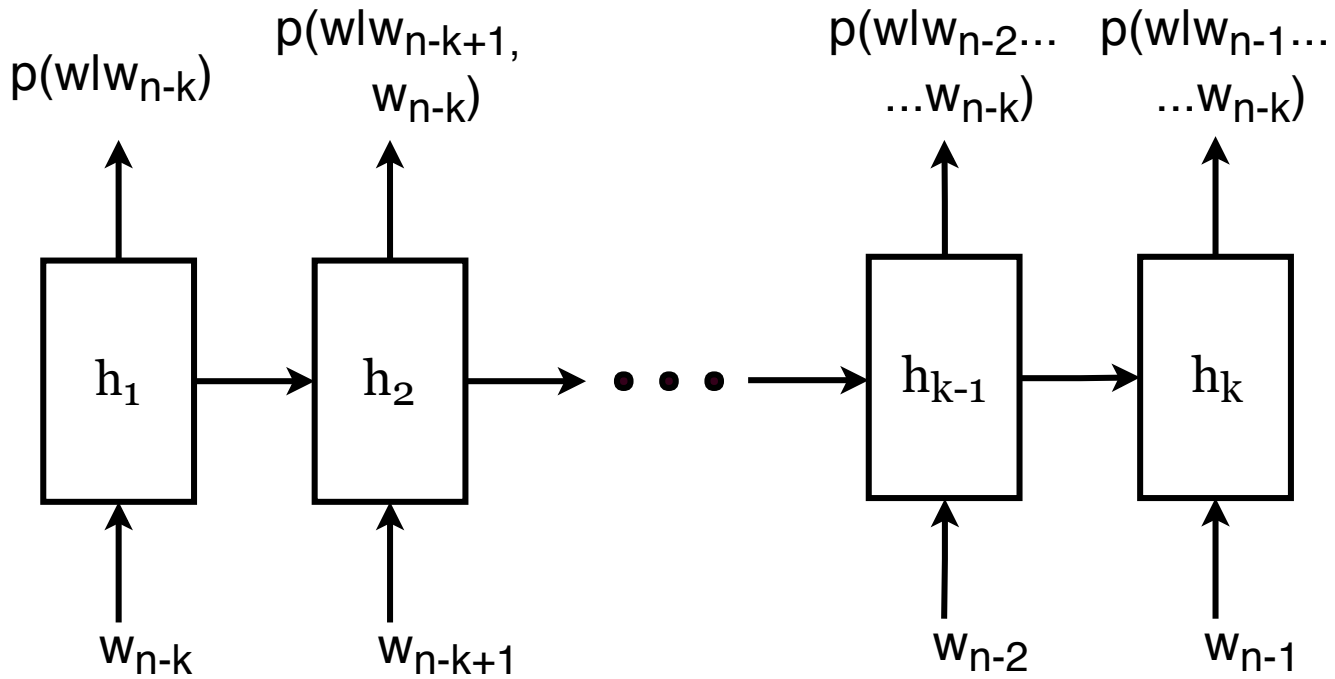


Для каждой последовательности используется функция потерь:

$$\mathcal{L} = \mathcal{L}_k = -\sum_{w \in W} [w = w_n] \log p(w = w_n | w_{n-1}, \ldots, w_{n-k})$$

Можно ли как-то лучше?

RNN для LM с k выходами



Для каждой последовательности используется функция потерь:

$$\mathcal{L} = \sum_{t=1}^{k} \mathcal{L}_t$$

$$\mathcal{L}_t = -\sum_{w \in W} [w = w_t] \log p(w = w_t | w_{n-t}, \ldots, w_{n-k})$$

Слова, не представленные в словаре (out of vocabulary)

Добавление в словарь <UNK> токена

- Заменить часть редких слов на <UNK> токен при обучении
- На каждой итерации обучения с малой вероятностью заменять одно из слов на <UNK>

Использовать посимвольную RNN (charRNN)

- Вероятность встретить новый символ крайне мала...
- Во многих задачах charRNN работает не хуже wordRNN

Использовать посимвольную RNN для новых слов

- Если встречаем незнакомое слово, используем charRNN для его кодирования
- На каждой итерации обучения с малой вероятностью считаем одно из слов новым

Сравнение RNN LM и Kneser-Ney Smoothing[1][2]

Table 2: *Comparison of various configurations of RNN LMs and combinations with backoff models while using 6.4M words in training data (WSJ DEV).*

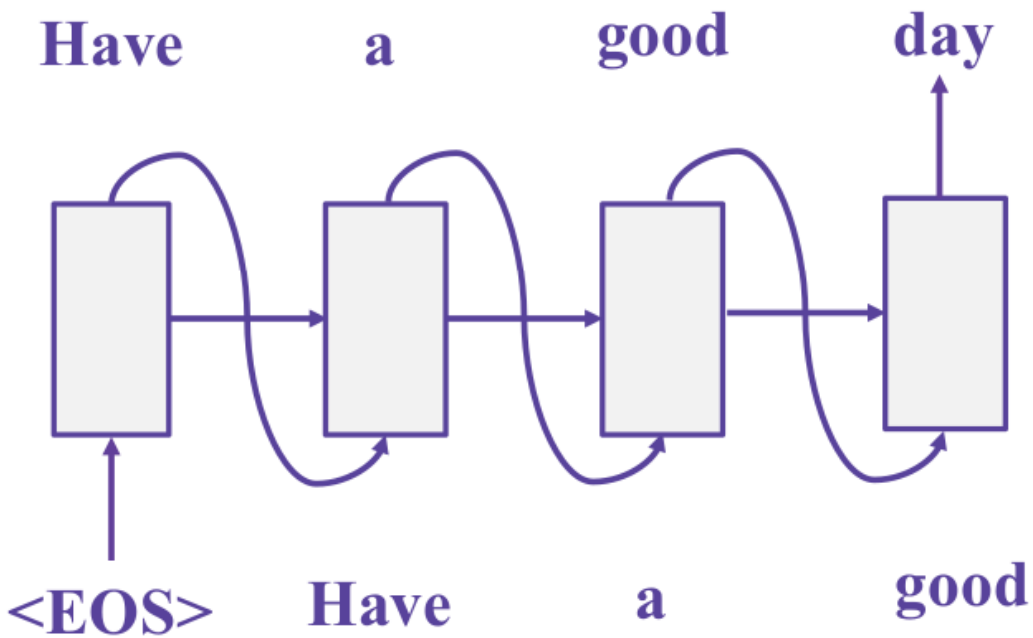
	PPL		WER	
Model	RNN	RNN+KN	RNN	RNN+KN
KN5 - baseline	-	221	-	13.5
RNN 60/20	229	186	13.2	12.6
RNN 90/10	202	173	12.8	12.2
RNN 250/5	173	155	12.3	11.7
RNN 250/2	176	156	12.0	11.9
RNN 400/10	171	152	12.5	12.1
3xRNN static	151	143	11.6	11.3
3xRNN dynamic	128	121	11.3	11.1

[1] Mikolov, Karafiát, Burget, Cernocký, and Khudanpur. Recurrent neural network based language model. INTERSPEECH 2010.

[1] Ноутбук Голдберга с сравнением (ссылка)

Как генерировать текст с помощью обученной RNN?

1. Сгенерировать/выбрать слово w_1
2. Применить RNN к w_1
3. Получить слово w_2, взяв arg max от последнего выхода
4. Применить RNN к w_2
5. ...



Детали реализации генерации

Что можно использовать кроме arg max?

- Сэмплировать слово из полученного распределения.
- Использовать beam search.

Как генерировать конечные последовательности?

- Добавить специальный токен <EOS> в конец каждой обучающей последовательности. При генерации <EOS> прекращать процесс.

Как генерировать первое слово?

- Добавить специальный токен <SOS> в начало каждой последовательности. Всегда начинать новую последовательность с <SOS>.
  Как это работает, если использовать arg max?

Детали реализации генерации

Что можно использовать кроме arg max?

- Сэмплировать слово из полученного распределения.
- Использовать beam search.

Как генерировать конечные последовательности?

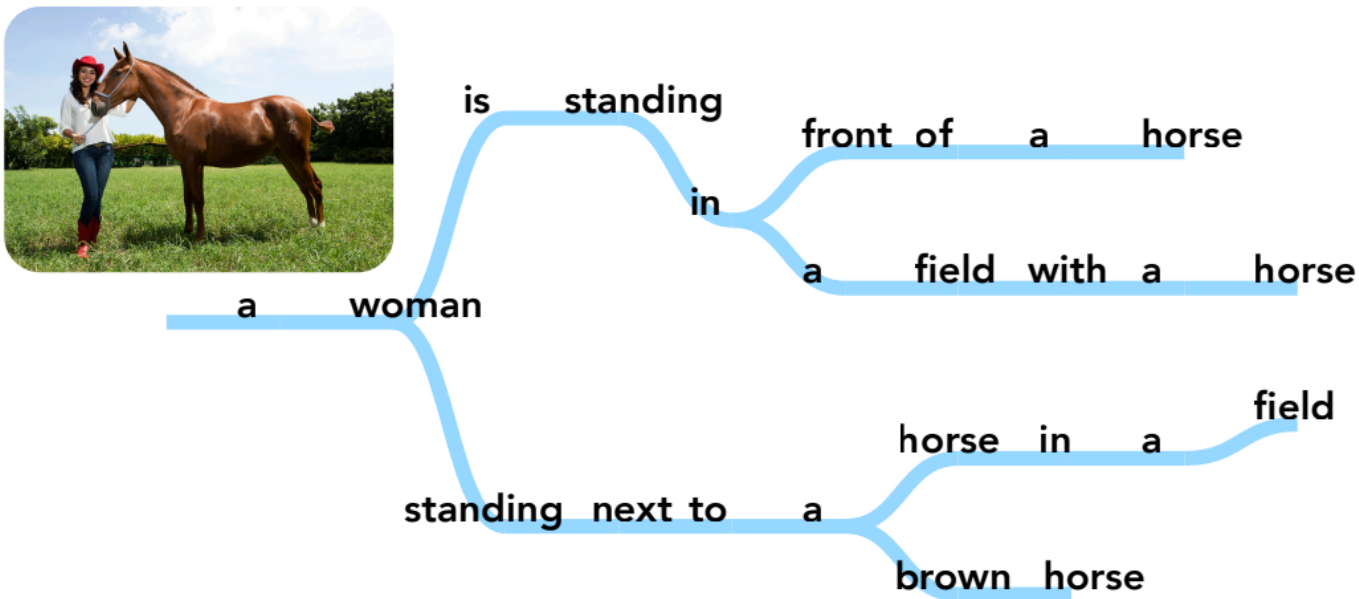
- Добавить специальный токен <EOS> в конец каждой обучающей последовательности. При генерации <EOS> прекращать процесс.

Как генерировать первое слово?

- Добавить специальный токен <SOS> в начало каждой последовательности. Всегда начинать новую последовательность с <SOS>.
  Как это работает, если использовать arg max? Генерирует одно и то же, если $h_0 = 0$.

Beam search (лучевой поиск)

- Применить RNN к w_1
- Выбрать m самых вероятных слов w_2
- К каждой новой последовательности применить RNN
- В каждой последовательности выбрать m самых вероятных слов w_3
- Оставить только m самых вероятных последовательностей
- ...



Отличия в сети при генерации и обучении

Есть существенные отличия в входных данных для сети:

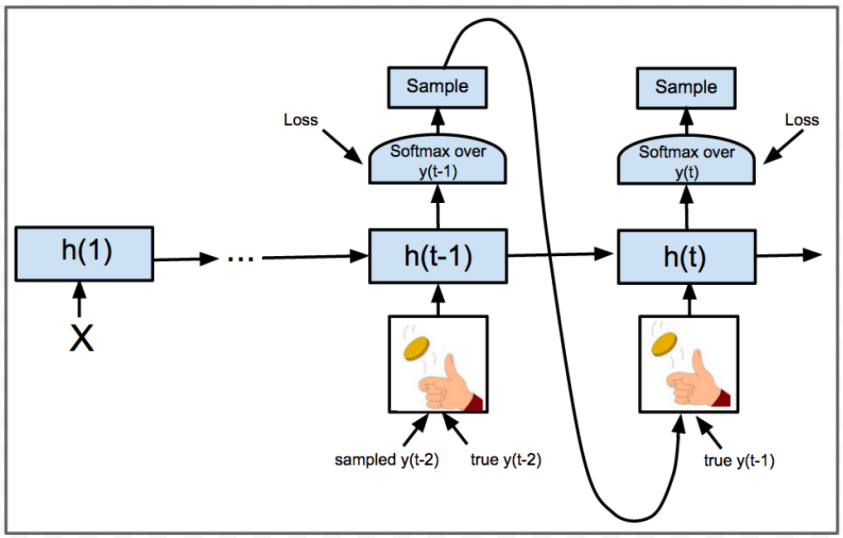
Этап	На входе ячейки
Обучение	Истинное w_i
Тест	Предсказанное $\hat{w}_i = \arg\max p(w \mid w_{i-1}, \ldots, w_{i-k-1})$

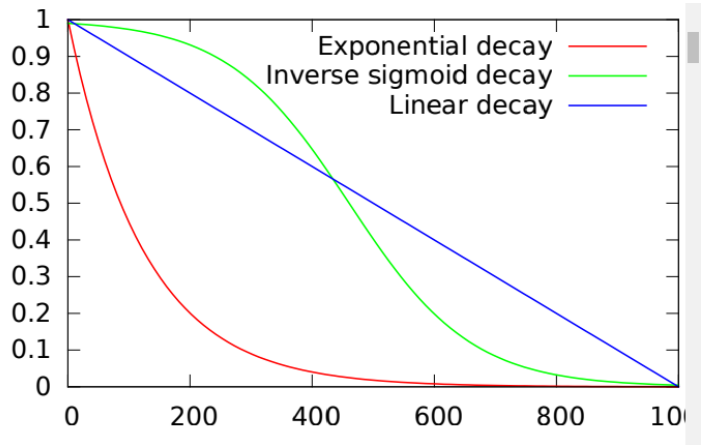
+ Модель быстро обучается обычно с хорошим качеством
– Модель плохо генерирует следующее слово для плохо сгенерированного предложения
(таких случаев нет в обучении)

Beam serach частично решает эту проблему!

Scheduled Sampling[1]

Выбираем с вероятностью ϵ_i истинное слово, иначе сгенерированное:





ϵ_i убывает с течением итераций по одному из трёх законов:

$$\epsilon_i = \max(\epsilon, k - c_i) \quad \epsilon_i = k^i \quad \epsilon_i = k/(k + \exp(i/k))$$

[1] S. Bengio, O. Vinyals, N. Jaitly, N. Shazeer. Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks. 2015

Результаты Scheduled Sampling

Задача описания изображения (image captioning):

Approach vs Metric	BLEU-4	METEOR	CIDER
Baseline	28.8	24.2	89.5
Baseline with Dropout	28.1	23.9	87.0
Always Sampling	11.2	15.7	49.7
Scheduled Sampling	**30.6**	**24.3**	**92.1**
Uniform Scheduled Sampling	29.2	24.2	90.9
Baseline ensemble of 10	30.7	25.1	95.7
Scheduled Sampling ensemble of 5	**32.3**	**25.4**	**98.7**

Uniform Scheduled Sampling — сэмплируем не из модели, а из равномерного распределения

Scheduled Sampling улучшает качество модели и даже качество ансамбля моделей

Резюме по языковым моделям

- RNN хорошо подходит для построения языковых моделей
- Можно использовать как и посимвольные модели, так и пословные
- При генерации текста можно использовать beam search для улучшения результата
- При обучении текста можно использовать scheduled sampling, чтобы расширить обучающую выборку без сильного проигрыша во времени

Задача распознавания именованных сущностей (named entity recognition)

Для каждого слова $(w_1, w_2, \ldots w_n)$, $w_i \in W$ в тексте определить, является ли оно частью некоторой именованной сущности

Варианты типов ответа:

- Входит в именованную сущность/не входит
- BIO-notation: начало сущности (B)/внутри сущности(I)/не сущность(O)
- Тип сущности (персона, место, организация и т.д.)

Задача сводится к классификации каждого слова в последовательности.

Часто, все последовательности приводят к одинаковой длине k, дополняя последовательности специальным <PAD> токеном.

Пример входа

```
Facebook нашел нового финансового директора Финансовым
директором социальной сети Facebook назначен 39-летний
Дэвид Эберсман (David Ebersman), сообщает The Wall
Street Journal.

T1 ORG 0 8 Facebook
T2 ORG 83 91 Facebook
T3 PER 111 142 Дэвид Эберсман (David Ebersman)
T4 ORG 153 176 The Wall Street Journal
```

Обозначения

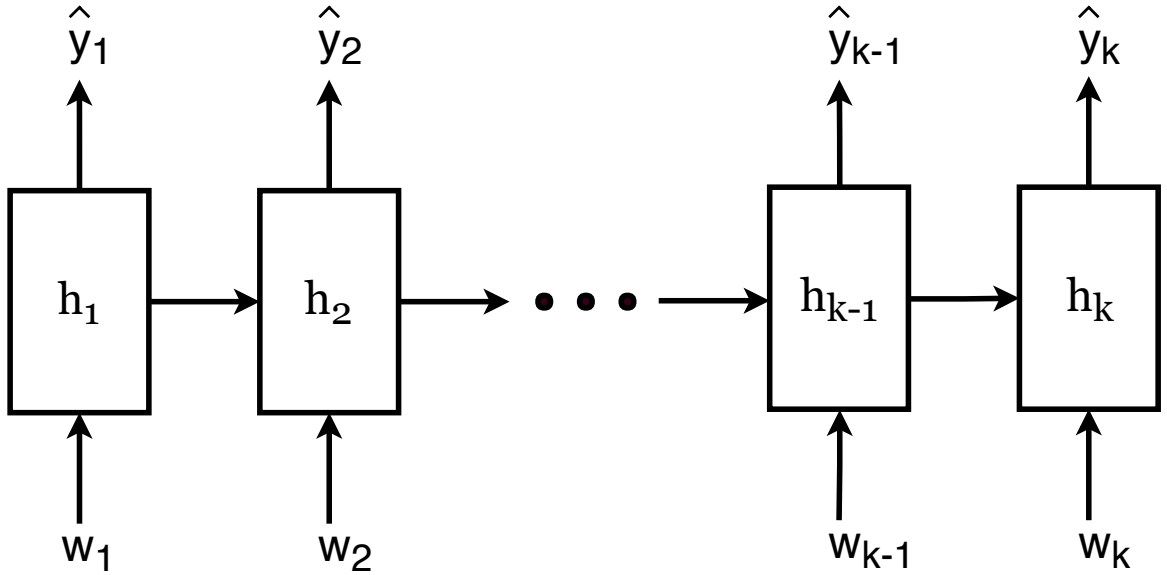
w_t — как и раннее one-hot вектор слова
y_t — метка класса
$\hat{y}_t$ — распределение ответов модели для t-го слова

Модель не меняется, меняется только смысл переменных:

$$\hat{y}_t = p(y|w_t, \dots, w_1) = \mathrm{softmax}(Uh_t + \hat{b})$$

$$h_t, C_t = LSTM(h_{t-1}, C_{t-1}, w_t)$$

LSTM для задачи NER



Используем кросс-энтропийную функцию потерь:

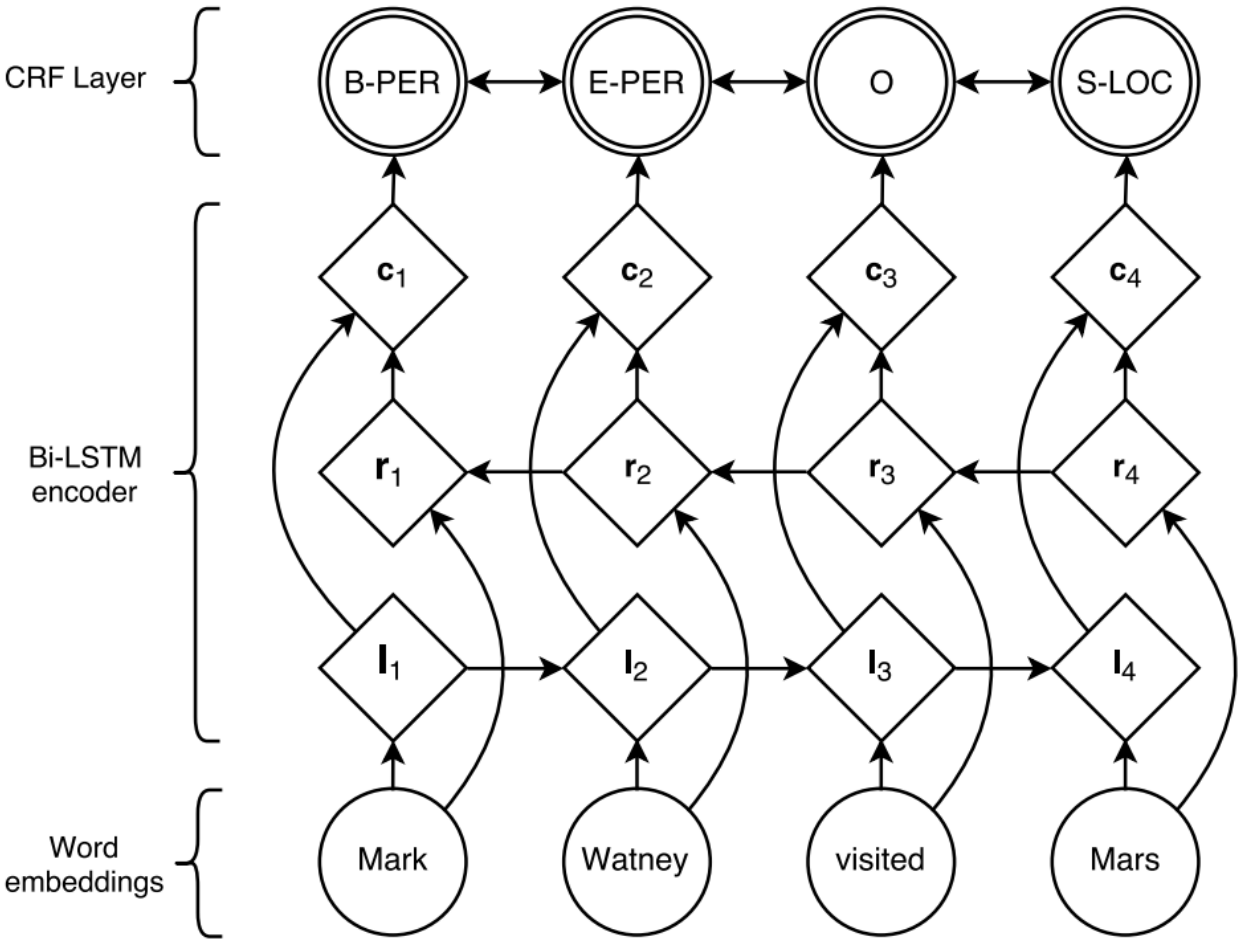
$$\mathcal{L} = \sum_{t=1}^{k} \mathcal{L}_t$$

$$\mathcal{L}_t = -\sum_{y \in Y} [y = y_t] \log p(y = y_t | w_t, \ldots, w_1)$$

Улучшение качества

- Использовать вместо LSTM biLSTM
- Использовать комбинацию LSTM и CRF
  (на следующей лекции подробнее)
- Использовать комбинацию LSTM для NER и LM

biLSTM + CRF[1]



[1]Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer. Neural Architectures for Named Entity Recognition. NAACL-2016.

biLSTM + CRF: детали обучения

Выход biLSTM — вход для CRF

Т.к. CRF обучается через градиентные методы, можем пробрасывать градиенты CRF в backpropgation алгоритме.

Algorithm 1 Bidirectional LSTM CRF model training procedure

```
for each epoch do
    for each batch do
        1) bidirectional LSTM-CRF model forward pass:
            forward pass for forward state LSTM
            forward pass for backward state LSTM
        2) CRF layer forward and backward pass
        3) bidirectional LSTM-CRF model backward pass:

            backward pass for forward state LSTM
            backward pass for backward state LSTM
        4) update parameters
    end for
end for
```

biLSTM + CRF: результаты[1]

Результаты biLSTM + CRF превосходят остальные подходы

Table 2: Comparison of tagging performance on POS, chunking and NER tasks for various models.

		POS	CoNLL2000	CoNLL2003
Random	Conv-CRF (Collobert et al., 2011)	96.37	90.33	81.47
	LSTM	97.10	92.88	79.82
	BI-LSTM	97.30	93.64	81.11
	CRF	97.30	93.69	83.02
	LSTM-CRF	**97.45**	93.80	84.10
	BI-LSTM-CRF	97.43	**94.13**	**84.26**
Senna	Conv-CRF (Collobert et al., 2011)	97.29	94.32	88.67 (89.59)
	LSTM	97.29	92.99	83.74
	BI-LSTM	97.40	93.92	85.17
	CRF	97.45	93.83	86.13
	LSTM-CRF	97.54	94.27	88.36
	BI-LSTM-CRF	**97.55**	**94.46**	**88.83 (90.10)**

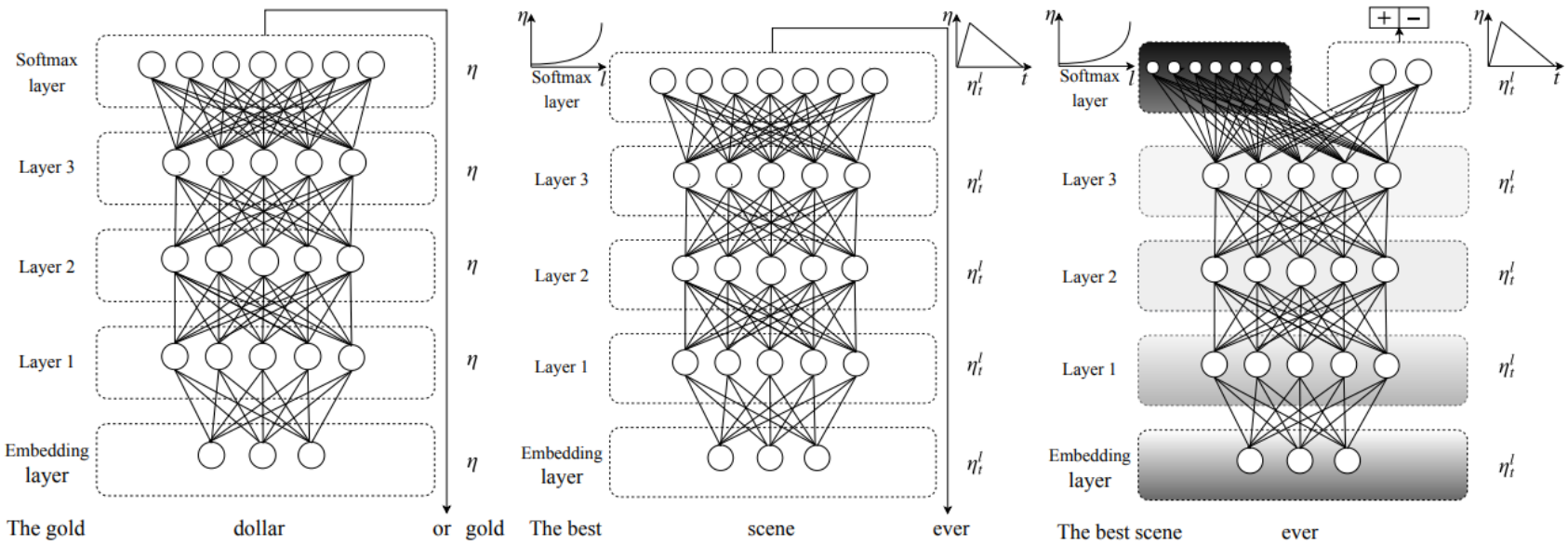
[1] Zhiheng Huang, Wei Xu, Kai Yu, Bidirectional LSTM-CRF Models for Sequence Tagging. 2015

Резюме по NER

- biLSTM + CRF — один из лучших подходов для NER на сегодняшний день.
- Все state-of-the-art для разных языков — вариации этой модели.

Использование LM (language model) для transfer learning[1]

- Обучить LM на большом корпусе (например, википедии), используя достаточно глубокую архитектуру
- Дообучить LM на корпусе, который используется в задаче
- Добавить линейный слой, решающий конечную задачу (например, NER)



(a) LM pre-training (b) LM fine-tuning (c) Classifier fine-tuning

[1] Jeremy Howard, Sebastian Ruder. Universal Language Model Fine-tuning for Text Classification. ACL-2018

Использование LM для transfer learning

Использование LM даёт выигрыш в качестве:

LM fine-tuning	IMDb	TREC-6	AG
No LM fine-tuning	6.99	6.38	6.09
Full	5.86	6.54	5.61
Full + discr	5.55	6.36	5.47
Full + discr + stlr	**5.00**	**5.69**	**5.38**

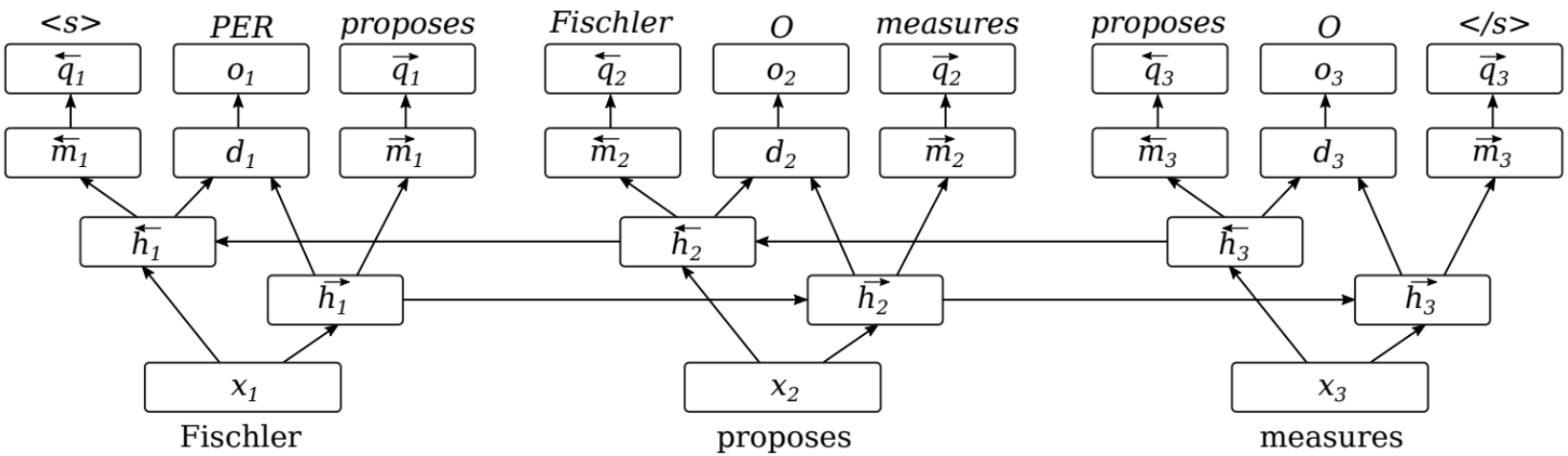
Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

discr — свой learning rate для каждого слоя
stlr — специальный способ изменения learning rate

LM в multitask learning[1]

Используем три функции потерь:

- Кросс-энтропия для NER
- Кросс-энтропия для LM при прямом проходе
- Кросс-энтропия для LM при обратном проходе



[1] Marek Rei. Semi-supervised Multitask Learning for Sequence Labeling. ACL-2017.

LM в multitask learning

Использование дополнительных функций потерь даёт выигрыш в качестве исходной задачи:

	FCE DEV	FCE TEST			CoNLL-14 TEST1			CoNLL-14 TEST2		
	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Baseline	48.78	55.38	25.34	44.56	15.65	16.80	15.80	25.22	19.25	23.62
+ dropout	48.68	54.11	23.33	42.65	14.29	17.13	14.71	22.79	19.42	21.91
+ LMcost	**53.17**	**58.88**	**28.92**	**48.48**	**17.68**	**19.07**	**17.86**	**27.62**	**21.18**	**25.88**

Резюме по приложениям LM

- LM интересны не только сами по себе, их можно использовать для генерации текста