Языковое моделирование

курс «Практикум на ЭВМ», весна 2019

Попов Артём Сергеевич

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

19 сентября 2018 г.

Задача языкового моделирования (language modeling)

Дано:

- $D = \{d_1, \ldots, d_N\}$ — коллекция документов
- документ d — последовательность слов $\{w_1, \ldots, w_{n_d}\}$ n_d — длина документа d
- слово $w_i \in W$, W — множество всех слов

Задача языкового моделирования:
оценить вероятность появления любой последовательности слов $(w_1, \ldots, w_n)$ в «реальном» тексте.

Языковая модель (language model, LM) — модель, позволяющая вычислить вероятность $p(w_1, \ldots, w_n)$ для любых $w_1, \ldots, w_n \in W$.

Стандартные обозначения

Цепное правило (chain rule):

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

Предположение Маркова:

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

Ещё одна постановка задачи:
оценить вероятность появления слова w после последовательности слов $(w_1, \ldots, w_n)$.

Посимвольный вариант задачи:
оценить вероятность появления символа c после последовательности символов $(c_1, \ldots, c_n)$.

Оценка качества модели

- Правдоподобие

$$\mathcal{L}(D_{test}) = \prod_{d \in D_{test}} \prod_{i=1}^{n_d} p(w_i | w_1^{i-1})$$

- Перплексия

$$\mathcal{P}(D_{test}) = \mathcal{L}(D_{test})^{-1/N_{test}}$$

- Итоговое качество приложения

Языковая модель почти никогда не представляет ценности сама по себе!

Приложения LM

- Исправление опечаток
- Машинный перевод
- Распознавание рукописного текста
- Распознавание речи
- ...

Приложения LM: постобработка результата MT

Результат модели машинного перевода:

- исходное предложение: *дом, милый дом.*
- варианты перевода: *home sweet home, house sweet house*
- выбираем ответ с помощью языковой модели:

$$p(home, sweet, home) \gg p(house, sweet, house)$$

В каких ситуациях такая постобработка может существенно улучшать качество?

Приложения LM: постобработка результата MT

Результат модели машинного перевода:

- исходное предложение: *дом, милый дом.*
- варианты перевода: *home sweet home, house sweet house*
- выбираем ответ с помощью языковой модели:

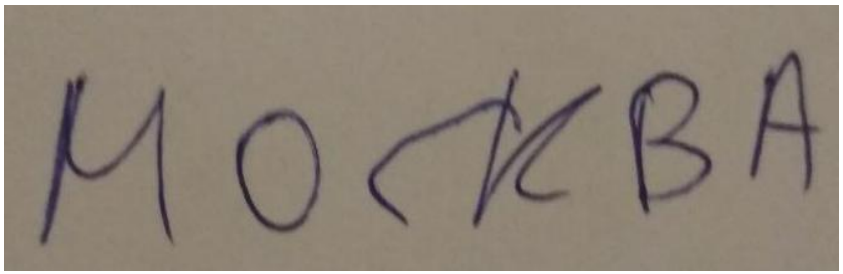
$$p(home, sweet, home) \gg p(house, sweet, house)$$

В каких ситуациях такая постобработка может существенно улучшать качество?

Если размер выборки из предложений их переводов небольшой, но есть большое количество неразмеченных предложений на языке перевода.

Приложения LM: постобработка результата OCR

Результат модели распознавания текста:

- 

- варианты распознавания: *москва*, *можва*
- выбираем ответ с помощью языковой модели:

$$p(\text{м, о, с, к, в, а}) \gg p(\text{м, о, ж, в, а})$$

Может помочь, если есть априорное знание о распознаваемом тексте (например, это поле в документе).

Приложения LM: исправление опечаток

Спеллчекер Питера Норвига + языковое моделирование[1][2]

1. входное предложение $w_1, \ldots, w_n$
2. для каждого слова w_i генерируем кандидатов:
   - удаляем один символ
   - вставляем лишний символ
   - меняем один символ на другой
   - меняем два соседних символа местами
3. удаляем кандидатов, не входящих в W
4. с помощью языковой модели выбираем лучшего из кандидатов

[1] http://norvig.com/spell-correct.html
[2] https://habr.com/ru/post/346618/

N-граммные модели

Используем предположение Маркова с параметром k.

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

Используем численную оценку вероятности:

$$p(w|w_1, \ldots, w_k) = \frac{C(w_1, \ldots, w_k, w)}{C(w_1, \ldots, w_k)},$$

где $C(w_1, \ldots, w_k)$ — число появлений последовательности $(w_1, \ldots w_k)$ в обучающей выборке.

Обучение модели — запоминание статистик появления последовательностей.

Проблемы N-граммных моделей

Какие проблемы есть у N-граммных модели?

Проблемы N-граммных моделей

Какие проблемы есть у N-граммных модели?

1. Проблема оценки первого слова в предложении
2. Проблема Out-of-vocabulary слов.
   Если слово w не встречалось в словаре, любая условная вероятность, содержащая в посылке w, будет равна 0
3. Чем больше N тем лучше учитываем контекст, но тем больше нулевых вероятностей

Биграммная модель (N=2) на примере

Предложение, для которого хотим оценить вероятность:

$$d = (\text{i}, \text{will}, \text{be}, \text{back})$$

Биграммная модель (N=2) на примере

Предложение, для которого хотим оценить вероятность:

$$d = (\text{i}, \text{will}, \text{be}, \text{back})$$

Можно так:

$$p(d) = p(\text{back}|\text{be})p(\text{be}|\text{will})p(\text{will}|\text{i})p(\text{i})$$

Но лучше так:

$$p(d) = p(\text{back}|\text{be})p(\text{be}|\text{will})p(\text{will}|\text{i})p(\text{i}|\text{<start>})$$

Добавляем в начало каждого предложения токен <start>, чтобы лучше моделировать вероятности первых слов.

Сглаживание Лапласа (Add-one smoothing)

Если слова нет в словаре, любая вероятность, содержащая это слово, будет равна нулю.

$$p(w|w_1, \ldots, w_k) = \frac{C(w_1, \ldots, w_k, w) + \alpha}{C(w_1, \ldots, w_k) + \alpha|W|},$$

Чем плох такой подход?

Откат (Katz backoff)

Основная идея: если не встречали n-грамму, но встречали $(n-k)$-грамму, то можем произвести «откат»

$$\hat{p}(w|w_1^k) = \begin{cases} \beta(w_1^k, w)p(w|w_1^k), \text{ если } C(w_1^k, w) > 0 \\ \alpha(w_1^k)p(w|w_2^k), \text{ иначе} \end{cases}$$

где $\alpha(w_1^k)$ и $\beta(w_1^k, w)$ выбираются из условия:

$$\sum_{w \in W} \hat{p}(w|w_1^k) = 1$$

Интерполяция (Interpolation smoothing, Jelinek-Mercer smoothing)

Смесь из нескольких моделей:

$$\hat{p}(w|w_1^k) = \sum_{i=1}^{k} \lambda_i p(w|w_i^k)$$

$$\sum_{i=1}^{k} \lambda_i = 1$$

Другие виды сглаживания

- Good-Turing
- Witten-Bell
- Kneser-Ney
- Absolute discounting

Задача моделирования последовательности

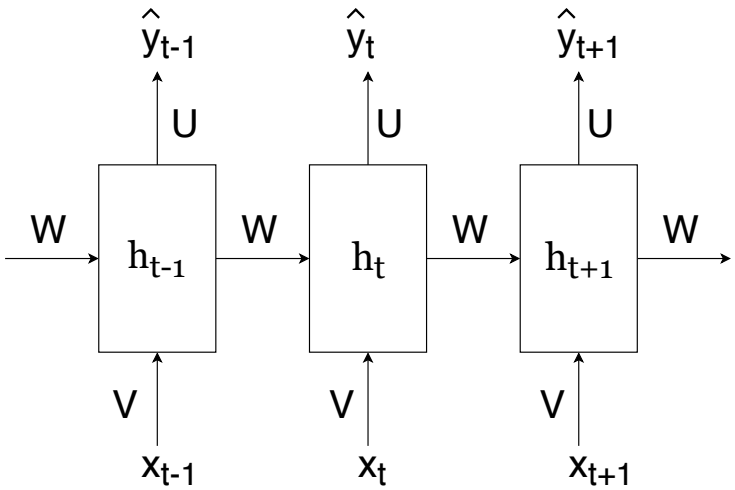
Дано: $\{x_1, \ldots, x_n\}$ — последовательность входов
$\{y_1, \ldots, y_n\}$ — последовательность выходов
$x_i \in \mathbb{R}^d, y_i \in \mathbb{R}^D$

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 Backpropagation[3]

[3] Часто, вариацию алгоритма Backpropagation для обучения RNN называют Backpropagation 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:

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

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

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

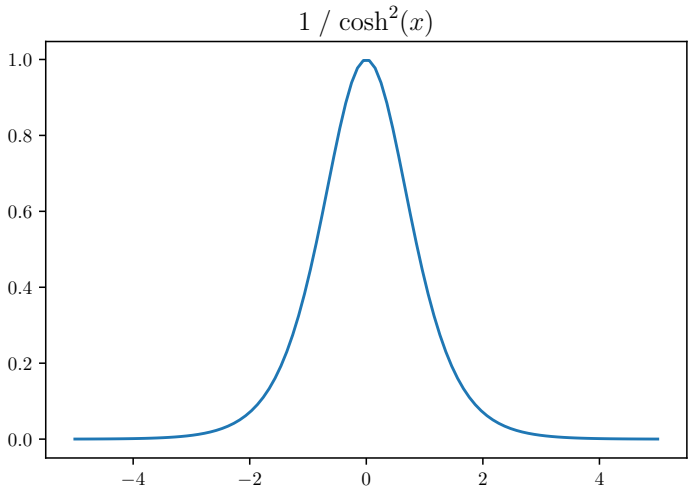
$$= \ \ldots \ = \sum_{k=1}^{t}\left(\prod_{i=k+1}^{t}\frac{\partial h_i}{\partial h_{i-1}}\right)\frac{\partial h_k}{\partial W}$$

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:

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

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

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

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

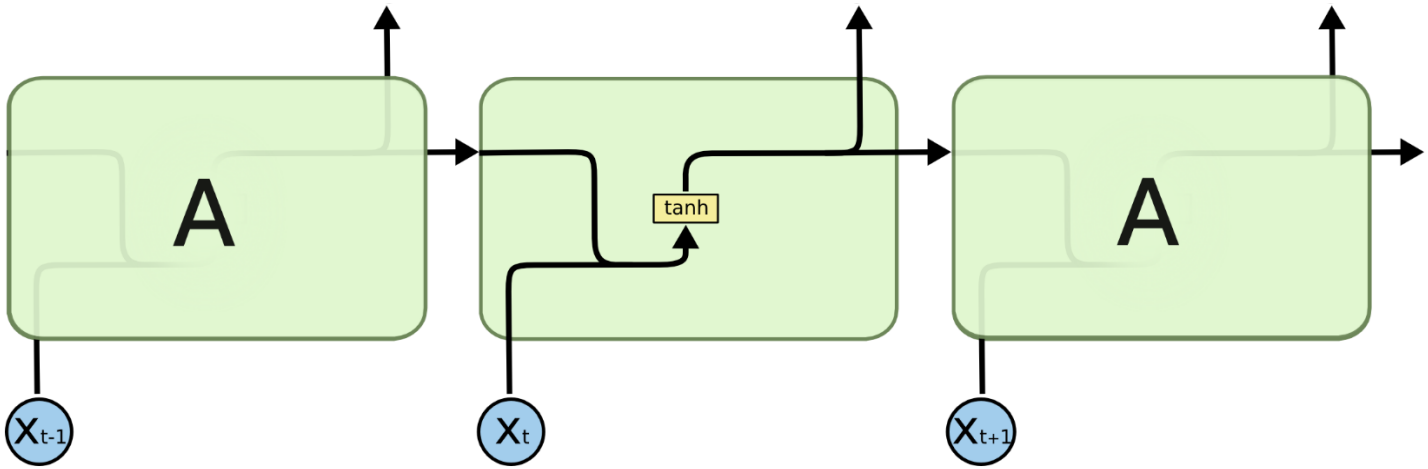
$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

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

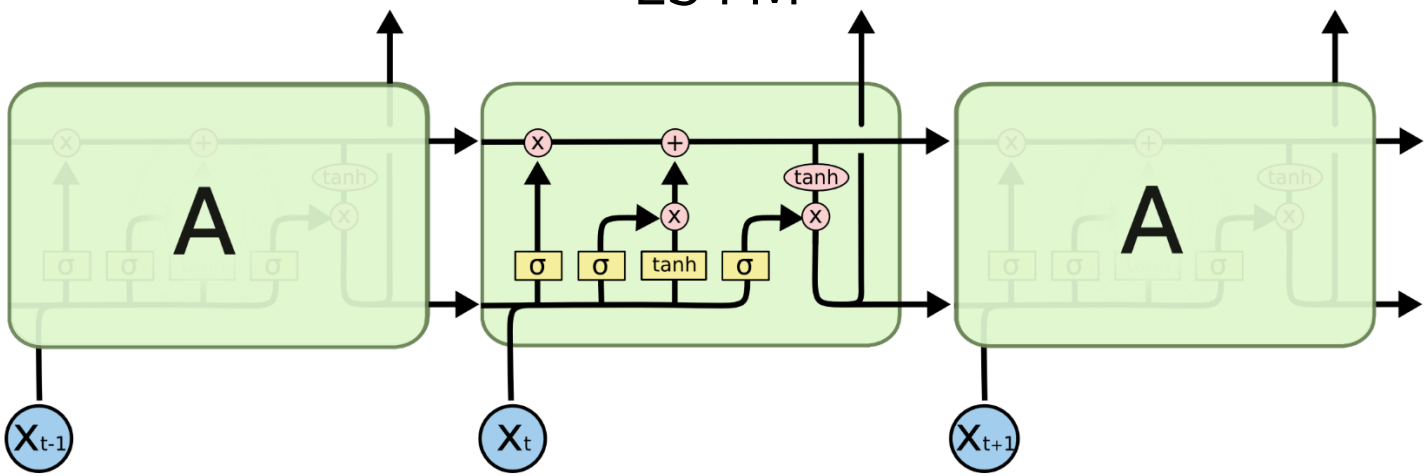
LSTM сеть

Используем более сложную структуру ячейки:

RNN

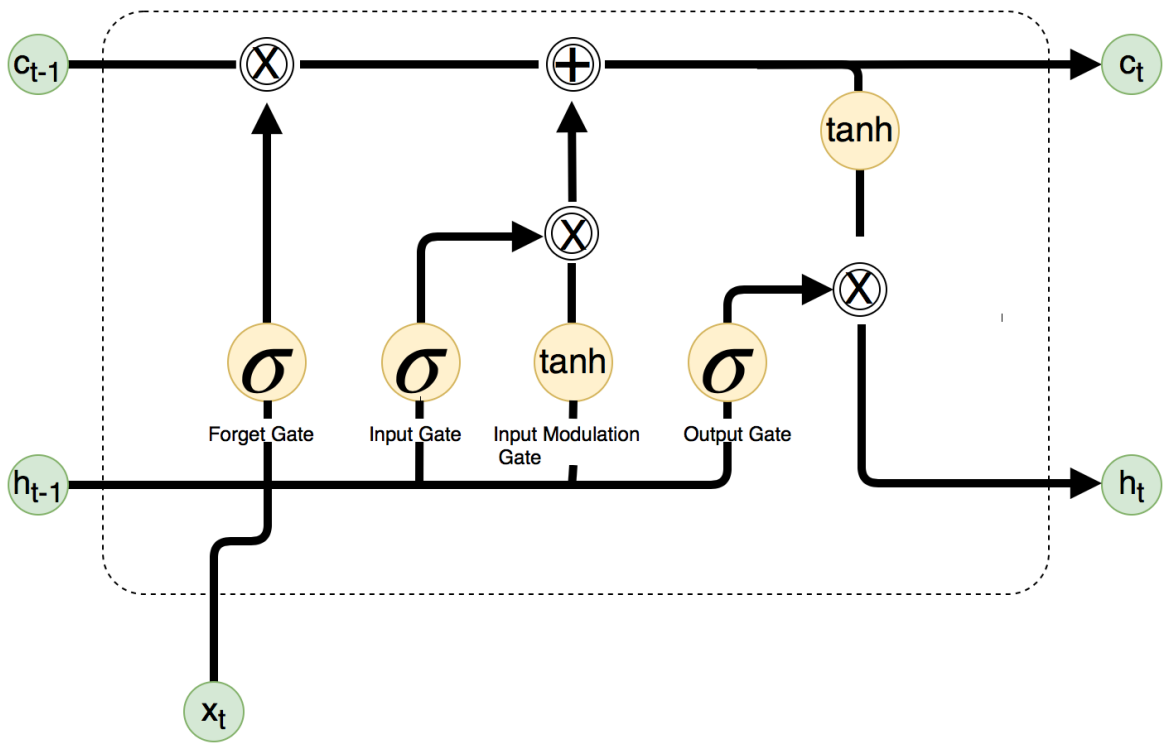


LSTM



LSTM ячейка

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

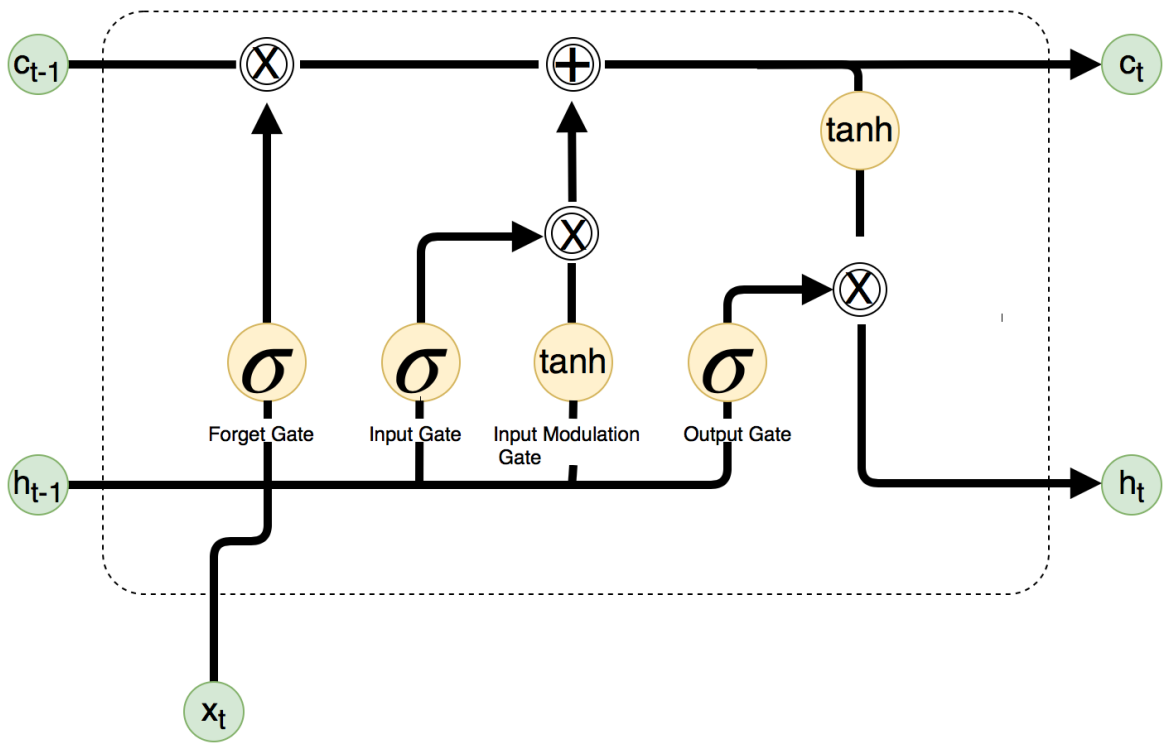


Обучается с помощью алгоритма Backpropagation

Почему решает проблему затухающих градиентов?

LSTM ячейка

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

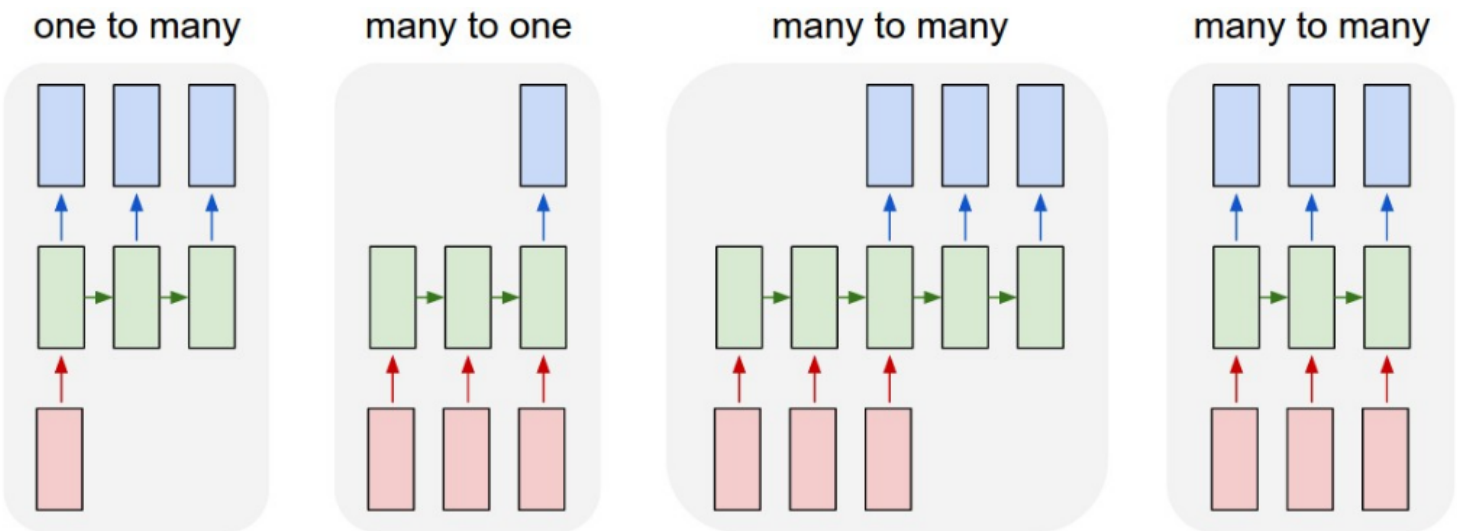


Обучается с помощью алгоритма Backpropagation

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

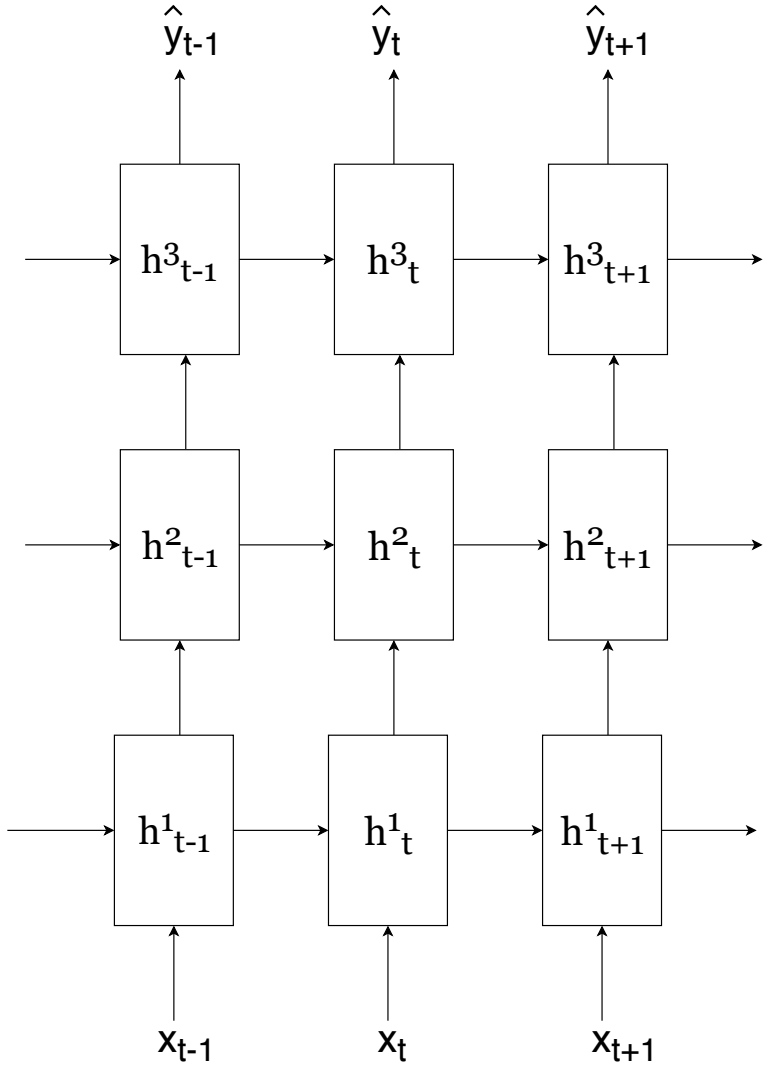
many to many(1) Перевод с одного языка на другой

many to many(2) Определение частей речи

Глубокие рекуррентные сети (deep RNN, layers stacking)

Выходы одной рекуррентной сети подаются на вход другой:

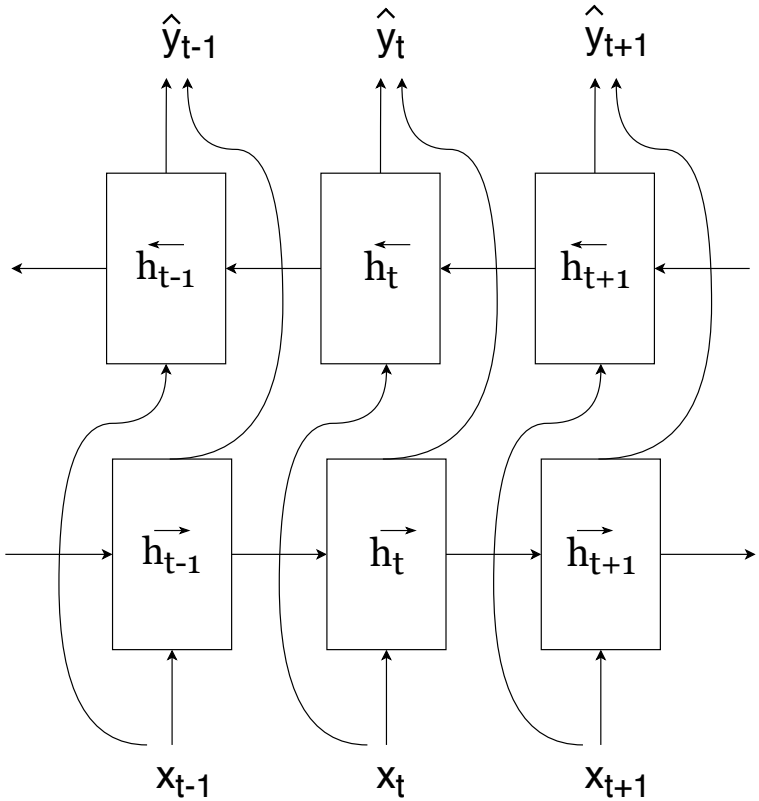
$$h_t^1, C_t^1 = LSTM(h_{t-1}^1, C_{t-1}^1, x_t)$$
$$h_t^2, C_t^2 = LSTM(h_{t-1}^2, C_{t-1}^2, h_t^1)$$
$$h_t^3, C_t^3 = LSTM(h_{t-1}^1, C_{t-1}^1, h_t^2)$$
$$y_t = g(Uh_t^2 + \hat{b})$$



Двунаправленные сети (bidirectional)

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

1. Из-за проблемы взрывающихся/затухающих градиентов не можем обрабатывать длинные последовательности
2. Технически 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):

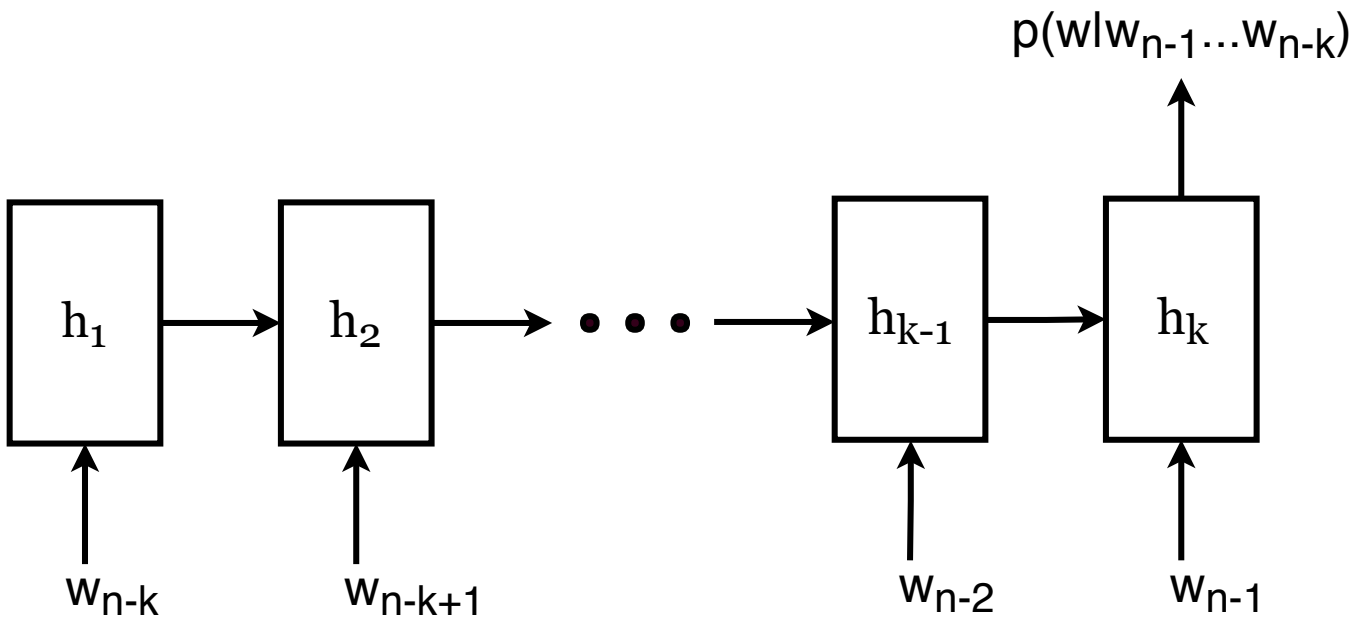
$$\mathrm{softmax}\, x = \left\{ \frac{\exp(x_i)}{\sum_{j=1}^{d} \exp(x_j)} \right\}_{i=1}^{d} \qquad x \in \mathbb{R}^d$$

$\mathrm{softmax}$ 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

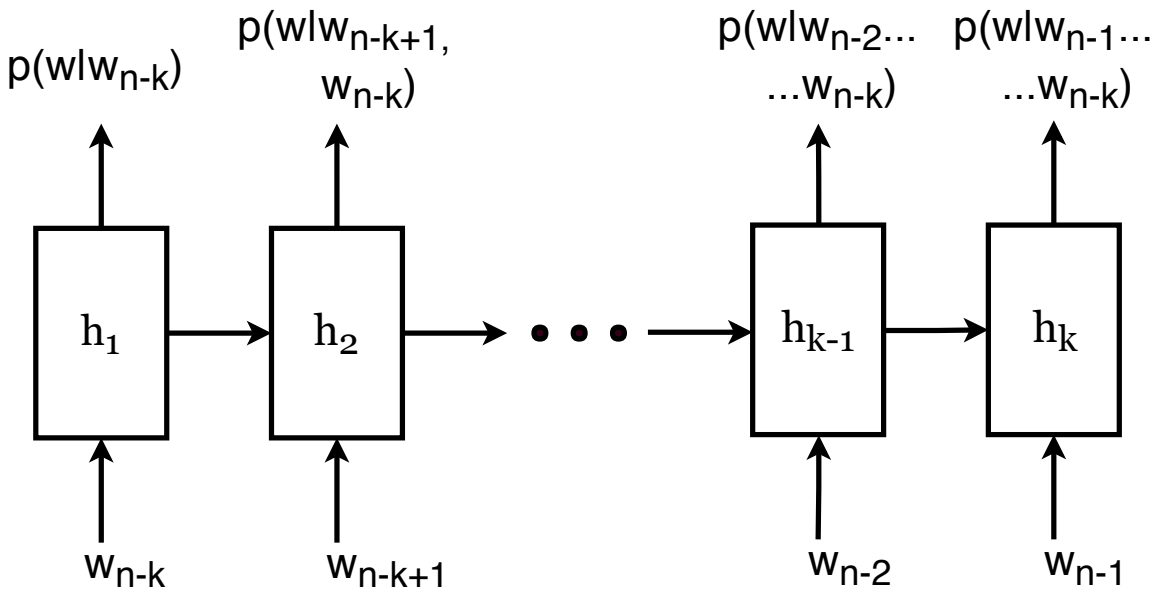


Для каждой последовательности используется функция потерь:

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

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 редкие слова на <UNK> токены 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).*

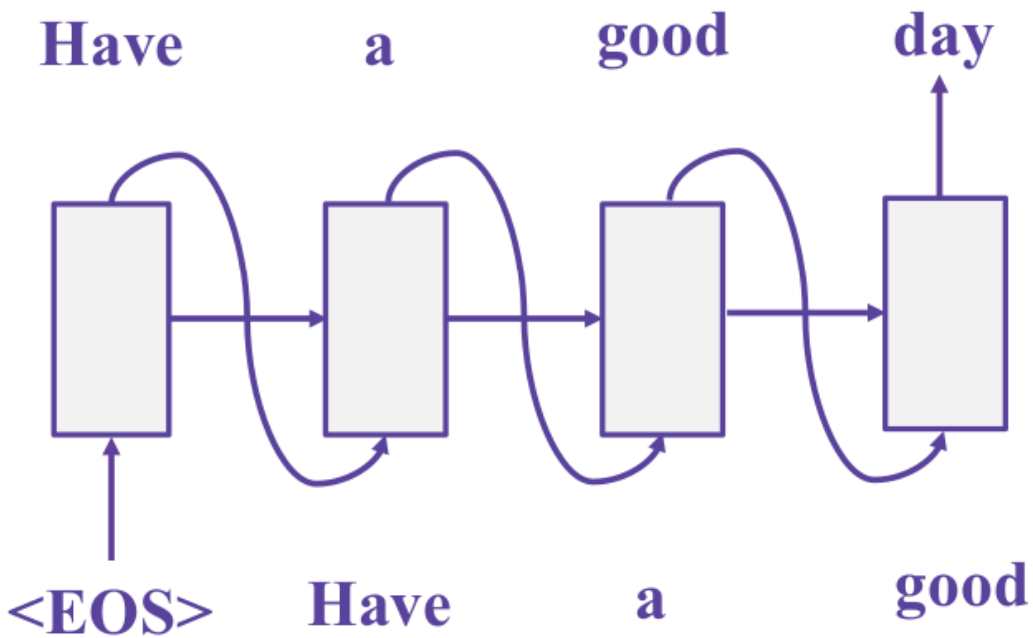
	PPL		WER	
Model	RNN	RNN+KN	RNN	RNN+KN
KN5 - baseline	-	221	-	13.5
RNN 60/20	229	186	13.2	12.6
RNN 90/10	202	173	12.8	12.2
RNN 250/5	173	155	12.3	11.7
RNN 250/2	176	156	12.0	11.9
RNN 400/10	171	152	12.5	12.1
3xRNN static	151	143	11.6	11.3
3xRNN dynamic	128	121	11.3	11.1

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

[3]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?

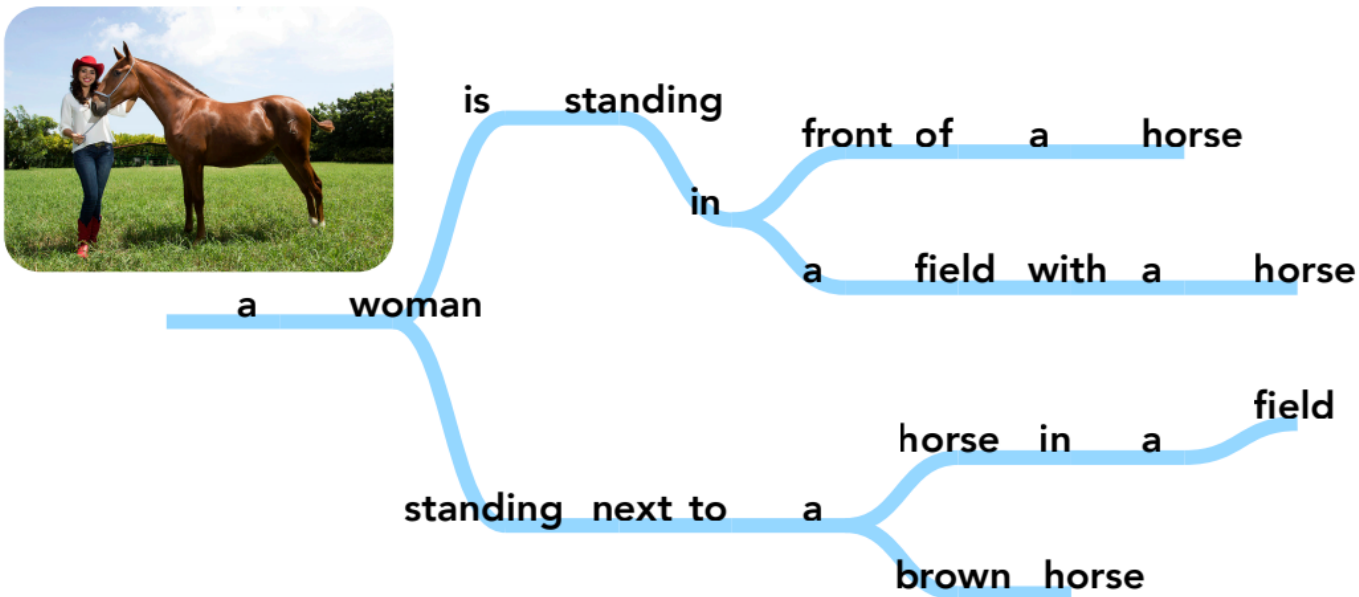
- Добавить специальный токен <EOS> в конец каждой обучающей последовательности. При генерации <EOS> прекращать процесс.

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 m самых вероятных последовательностей
- ...



Отличия в сети при генерации и обучении

Есть существенные отличия в входных данных для сети:

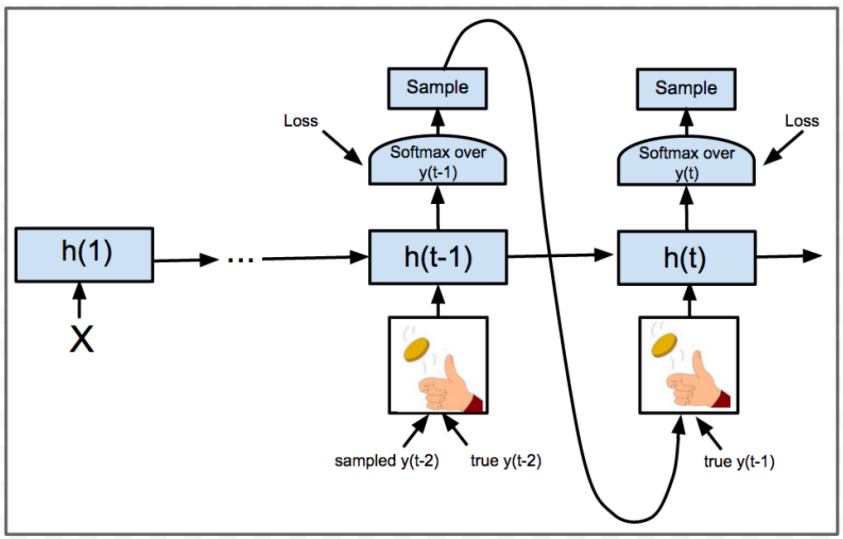
Этап	На входе ячейки
Обучение	Истинное w_i
Тест	Предсказанное $\hat{w}_i = \arg\max p(w \mid w_{i-1}, \ldots, w_{i-k-1})$

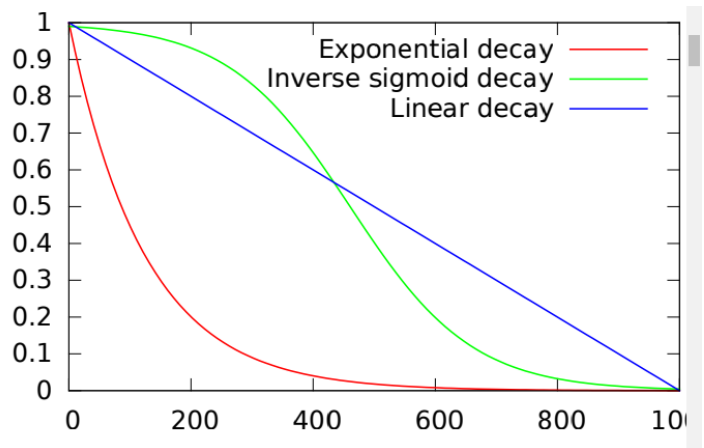
+ Модель быстро обучается обычно с хорошим качеством
– Модель плохо генерирует следующее слово для плохо сгенерированного предложения (таких случаев нет в обучении)

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:

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

[3]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

Дополнительные приложения LM

- Trasfer learning (перенос обучения)
- Multitask learning
- Регуляризация сети
- ...

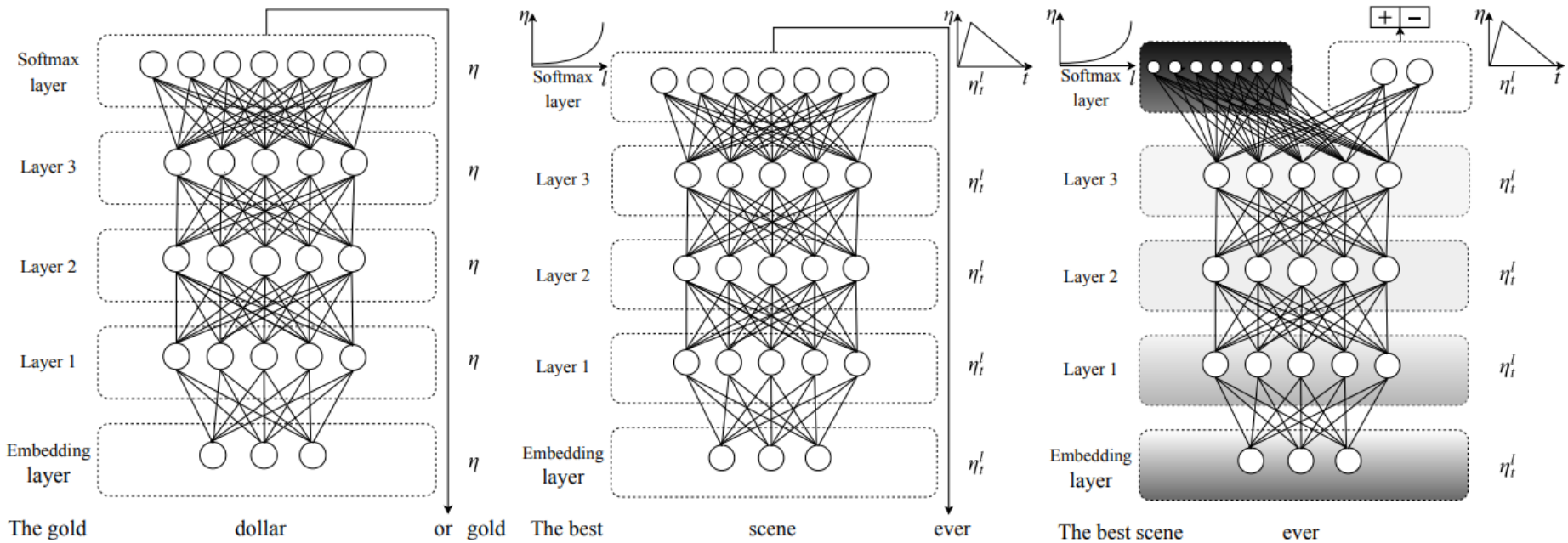
Использование LM (language model) для transfer learning[1]

- 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)

Примеры моделей:

- ELMO
- ULMfit
- GPT
- BERT

ULMfit



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

[3] 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**

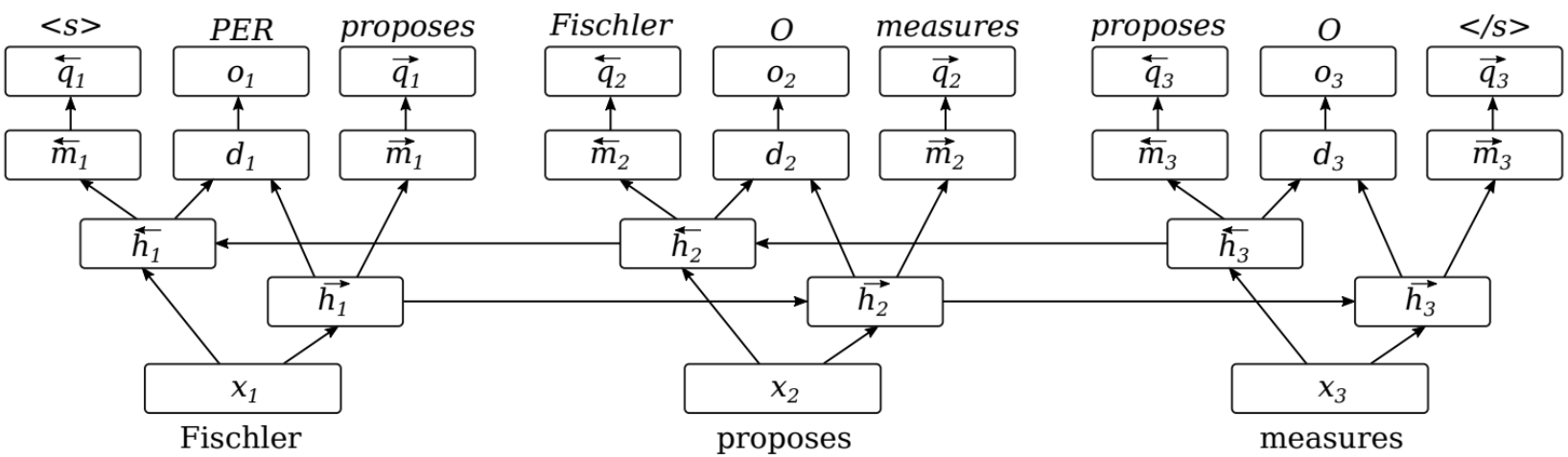
Table 6: Validation error rates for ULMFiT with different variations of LM fine-tuning.

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



[3]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 не только для генерации текста