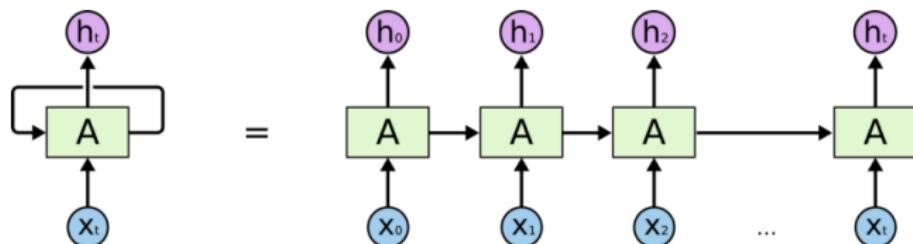


# Рекуррентные нейронные сети

Виктор Китов  
[victorkitov.github.io](http://victorkitov.github.io)



# Содержание

## 1 Основные понятия и приложения

- Введение
- Аналитическая запись
- Настройка RNN
- Архитектуры

## 2 Расширения RNN

## 3 Проблемы с градиентами

## 4 Рекуррентные сети с вентилями

## 5 Генеративные модели

## 1 Основные понятия и приложения

- Введение
- Аналитическая запись
- Настройка RNN
- Архитектуры

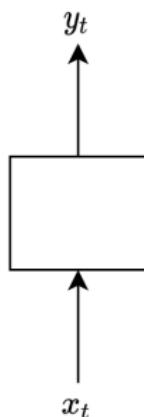
# Работа с последовательностями

- Данные в виде последовательности
  - динамика цен на акции
  - динамика действий посетителя веб-сайта
  - динамика погоды
  - текст - последовательность слов
  - речь - последовательность звуков
  - видео - последовательность кадров
- Можно использовать свёрточные нейросети над последовательностями
  - но свертка имеет ограниченную область видимости
- Рекуррентные нейросети (Recurrent neural net, RNN) помнят (в теории) всю историю.

# Классическая и рекуррентная модель (RNN)

Классическая модель: объекты обрабатываются независимо.

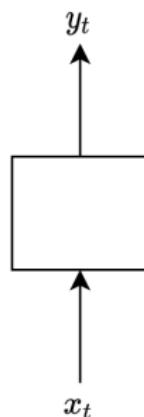
classical  
i.i.d.  
model



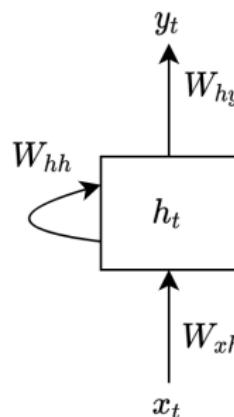
# Классическая и рекуррентная модель (RNN)

Рекуррентная сеть (RNN): объекты обрабатываются последовательно и зависимо (через внутреннее скрытое состояние  $h_t$ ).

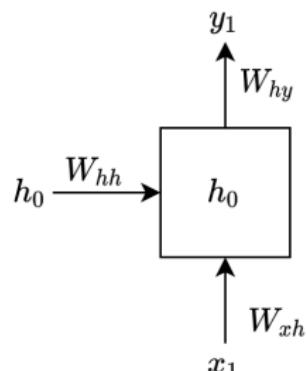
classical  
i.i.d.  
model



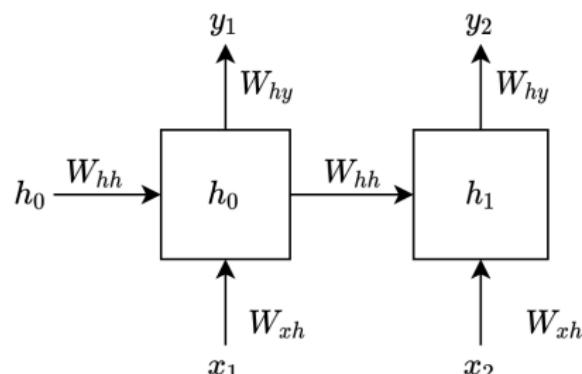
recurrent  
neural  
net



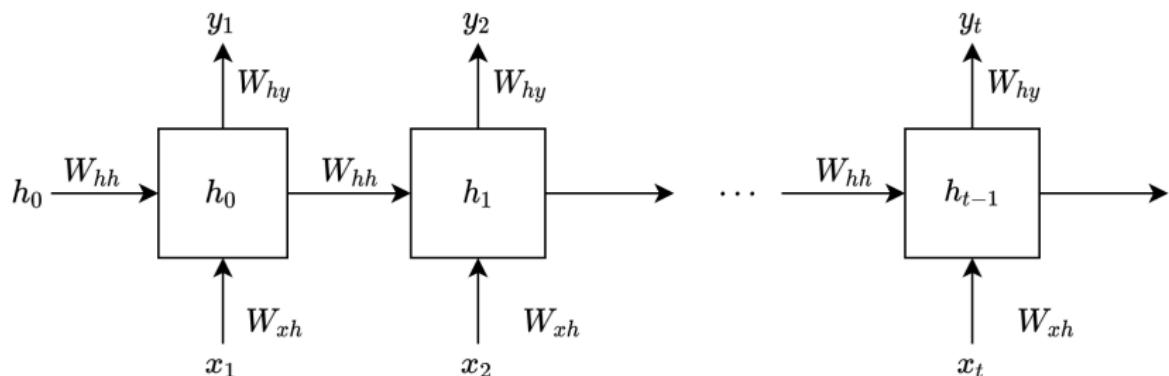
# Разворачивание RNN (unfolding)



# Разворачивание RNN (unfolding)



# Разворачивание RNN (unfolding)



## 1 Основные понятия и приложения

- Введение
- Аналитическая запись
- Настройка RNN
- Архитектуры

# Простейшие RNN

- Сеть Элмана

$$h_t = g(W_{xh}x_t + W_{hh}\textcolor{red}{h}_{t-1} + b_h)$$
$$\hat{y}_t = f(W_{hy}h_t + b_y)$$

- Сеть Джордана

$$h_t = g(W_{xh}x_t + W_{yh}\hat{y}_{t-1} + b_h)$$
$$\hat{y}_t = f(W_{hy}h_t + b_y)$$

- Если  $y$  - итоговый выход, то сеть Элмана более выразительная, т.к.  $\dim(h) \gg \dim(y)$ , но  $y$  тоже может быть многомерный признаковых входом для внешнего MLP.

# Разворачивание сети Элмана

$f, g$  - некоторые нелинейности, напр.  $\sigma(\cdot)$ .

$h_0 :=$  инициализация

$h_1 := g(W_{hh}h_0 + W_{xh}x_1 + b_h)$

$\hat{y}_1 := f(W_{hy}h_1 + b_y)$

# Разворачивание сети Элмана

$f, g$  - некоторые нелинейности, напр.  $\sigma(\cdot)$ .

$h_0 :=$  инициализация

$h_1 := g(W_{hh}h_0 + W_{xh}x_1 + b_h)$

$\hat{y}_1 := f(W_{hy}h_1 + b_y)$

$h_2 := g(W_{hh}h_1 + W_{xh}x_2 + b_h)$

$\hat{y}_2 := f(W_{hy}h_2 + b_y)$

# Разворачивание сети Элмана

$f, g$  - некоторые нелинейности, напр.  $\sigma(\cdot)$ .

$h_0 :=$  инициализация

$h_1 := g(W_{hh}h_0 + W_{xh}x_1 + b_h)$

$\hat{y}_1 := f(W_{hy}h_1 + b_y)$

$h_2 := g(W_{hh}h_1 + W_{xh}x_2 + b_h)$

$\hat{y}_2 := f(W_{hy}h_2 + b_y)$

...

$h_t := g(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$

$\hat{y}_t := f(W_{hy}h_t + b_y)$

# Общий случай

В более общем случае  $f, g$  не нелинейности, а более сложные ф-ции:

$$\begin{aligned} h_t &:= g_{W_{hh}, W_{xh}}(x_t, h_{t-1}) \\ \hat{y}_t &:= f_{W_{hy}}(h_t) \end{aligned}$$

Разворачивание (unfolding):

$$\begin{aligned} \hat{y}_t &= f(h_t) = f(g(x_t, h_{t-1})) = f(g(x_t, g(x_{t-1}, h_{t-2}))) = \\ &= f(g(x_t, g(x_{t-1}, g(x_{t-2}, h_{t-3})))) = \dots = F(x_t, x_{t-1}, \dots, x_1, h_0) \end{aligned}$$

# Векторные представления (embeddings)

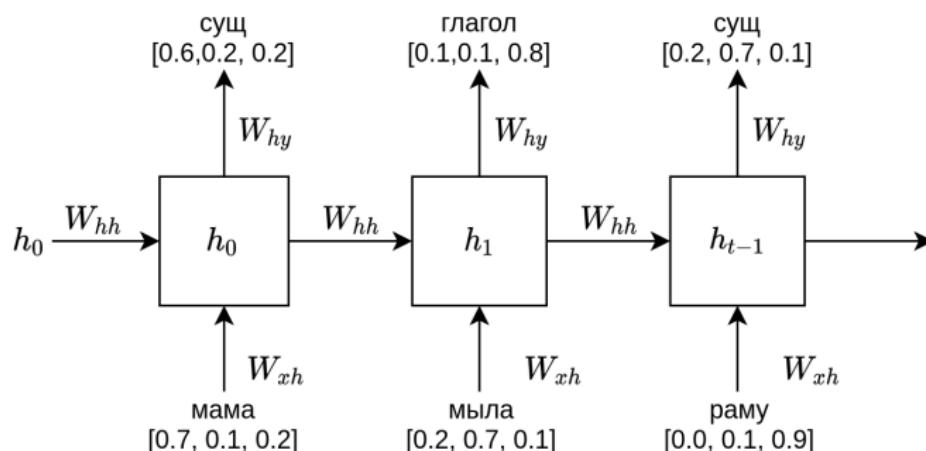
- Вещественный вход:  $x_t = \text{in}$  или  $x'_t = [in_t, in_{t-1}, \dots in_{t-k}]$ .
- Вещественный выход:  $\hat{y}_t = out_t$  либо  $\hat{y}_t = MLP(out_t)$ .
- Дискретный вход: используем *embedding*( $in_t$ )
  - one-hot, если категорий мало (символы)
  - для слов представления обычно выучиваются отдельно
    - из языковой модели (self-supervised), много данных

$$p(y_1, y_2, \dots, y_N) = p(y_1)p(y_1|y_2)\dots p(y_N|y_{N-1}, y_{N-2}, \dots, y_1)$$

- Дискретный выход:  $out_t = [p(y=1|x_{1:t}), \dots, p(y=C|x_{1:t})]$ ,

$$\hat{y}_t = \arg \max \{out_t\}$$

# Векторные представления (embeddings)



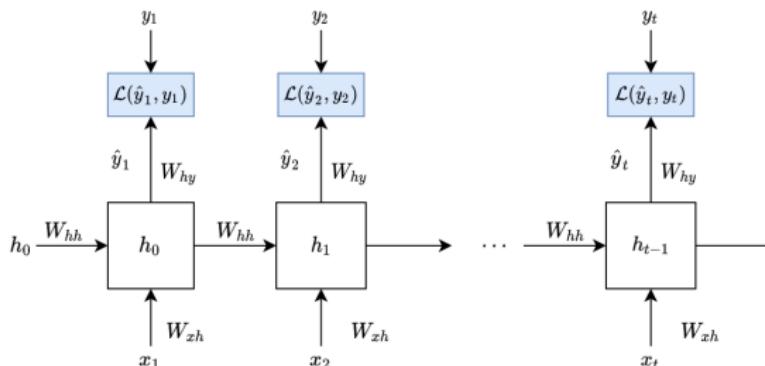
## 1 Основные понятия и приложения

- Введение
- Аналитическая запись
- Настройка RNN
- Архитектуры

# Настройка RNN

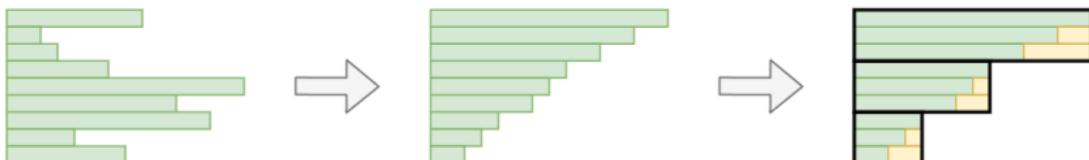
Обучение - пропускаем  $\nabla \left\{ \sum_{t=1}^T \mathcal{L}_t (\hat{y}_t, y_t) \right\}$  через развернутую сеть ( $\# \text{слоев} = \text{длине последовательности}$ ).

- англ. back propagation through time (BPTT)
- если последовательность слишком длинная  $\Rightarrow$  обрезка по порогу (truncated BPTT)



# Настройка RNN

- Из-за обрезки по  $t$  реально выученная RNN не будет учитывать всю историю
  - нужен отдельный учёт слишком далеких, но важных событий
  - пример:  $ABCD \rightarrow AABC$  - работает, а  $ABCD \rightarrow DCBA$  нет для длинных последовательностей.
- Обучение - минибатчами: упорядочиваем последовательности по  $\downarrow$  длины, разбиваем на минибатчи, в каждом минибатче выравниваем длины, дополняя спец. символом <EMPTY>.



# Обсуждение

- Переводим последовательность в последовательность:

$$x_1, x_2, \dots x_t \rightarrow \hat{y}_1, \hat{y}_2, \dots \hat{y}_t$$

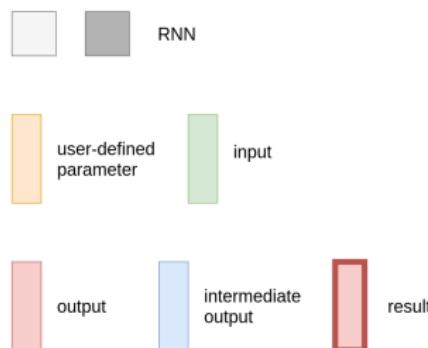
$$\hat{y}_t = f(x_t, h_t) = f(x_t, x_{t-1}, \dots x_1, h_0)$$

- Можем обрабатывать тексты (последовательность слов или символов)
  - каждый элемент - вектор, напр. one-hot-encoding
- Общие параметры:  $W_{xh}$ ,  $W_{hh}$ ,  $W_{hy}$  (parameter sharing, как в свёртках)
  - т.е. в любой момент времени работает одна модель
    - с изменяемым внутренним состоянием  $h_t$  (hidden state)

## 1 Основные понятия и приложения

- Введение
- Аналитическая запись
- Настройка RNN
- Архитектуры

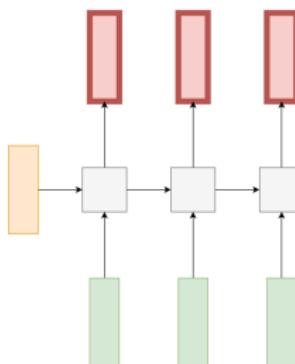
# Архитектуры RNN: обозначения



## Типы применений:

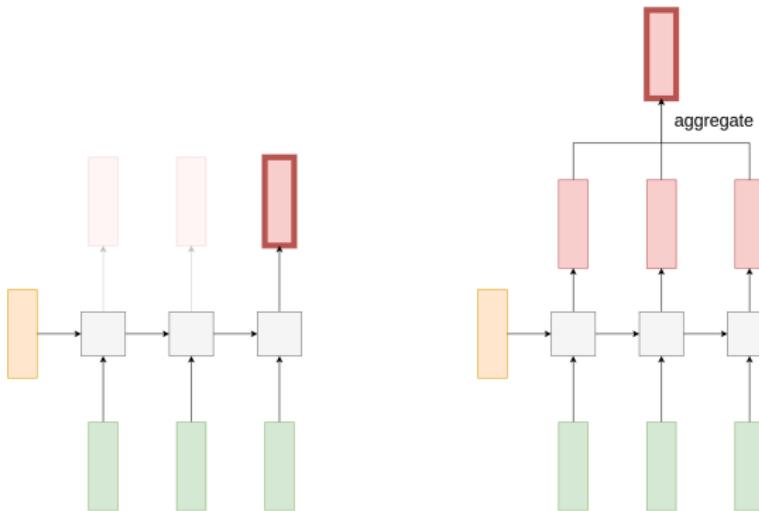
- synced many-to-many
- many-to-one
- one-to-many
- many-to-many

# Архитектуры RNN: synced many-to-many



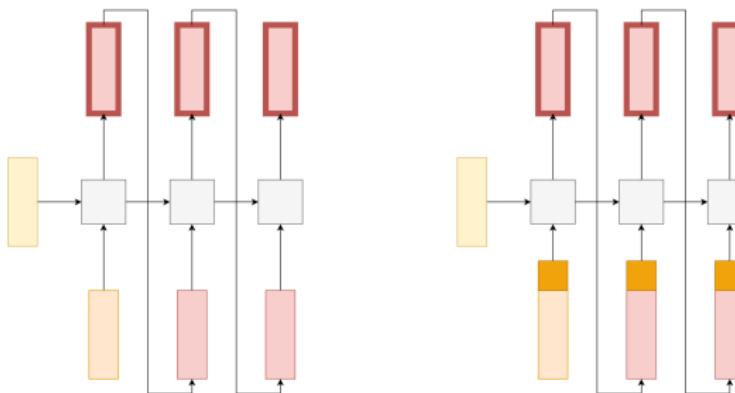
- Применения: разметка частей речи, выделение именованных сущностей (name entity recognition).
- Выходы: вероятности классов.

# Архитектуры RNN: many-to-one



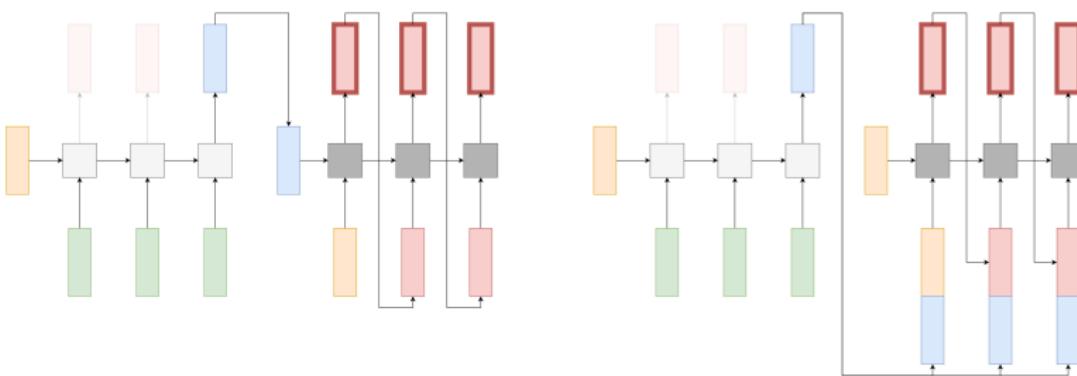
- Агрегация: усреднение, взвешенное усреднение, поэл. максимум.
- Применения: анализ тональности (sentiment analysis), классификация текстов по темам, видео по событиям на них.
- Выход: embedding последовательности для MLP.

# Архитектуры RNN: one-to-many



- Применения: описание изображений (image captioning), ответы на вопросы по картинке (visual question answering), генерация текста по теме, генерация музыки определённого жанра.
- Выходы: вероятности слов, нот.
- Входы - эмбеддинги выбранных (напр. макс. вероятных) токенов.
- Генерация, пока не сгенерируется  $\langle \text{EOS} \rangle$ .
  - альтернативно: доп. выход  $p(\text{end of sequence})$ .

## Архитектуры RNN: many-to-many (seq2seq<sup>1</sup>)



- Применения: перевод с одного языка на другой (machine translation), суммаризация длинного текста, перевод музыки в другой жанр, handwritten text generation.
- Выходы: вероятности слов, нот.
- Входы генератора - эмбеддинги выбранных (напр. макс. вероятных) токенов.
- Генерация до  $\langle \text{EOS} \rangle$  либо  $p(\text{end of sequence}) < t$

<sup>1</sup><https://arxiv.org/pdf/1409.3215.pdf>

# Содержание

1 Основные понятия и приложения

2 Расширения RNN

3 Проблемы с градиентами

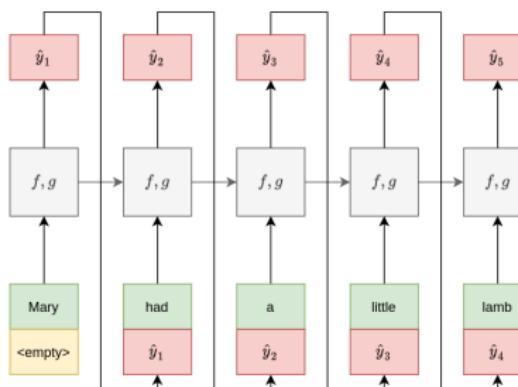
4 Рекуррентные сети с вентилями

5 Генеративные модели

# Прогноз, учитывающий предыдущие прогнозы

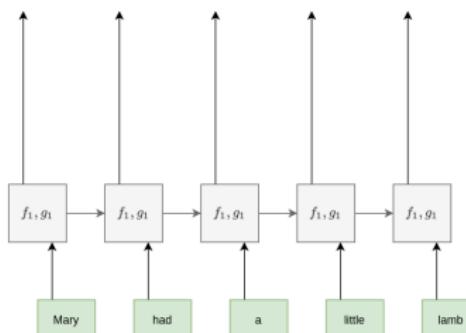
Чтобы прогноз учитывал предыдущие прогнозы, нужно добавить его в след. вход

- можно добавить конкатенацию нескольких предыдущих прогнозов
- либо выход RNN(предыдущие прогнозы) - помним "всё".
- важно, напр., при разметке частей речи - они зависят от соседних частей речи.



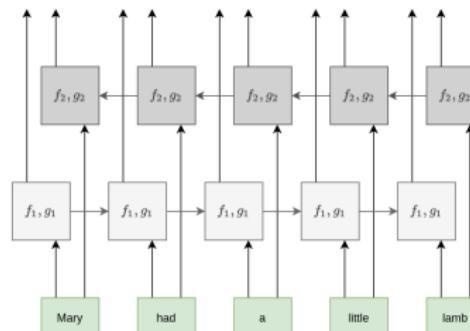
# Двунаправленная RNN

- Двунаправленная (bidirectional) RNN учитывает контекст и слева, и справа
  - пример: разметка частей речи, классификация событий на видео
- $f_1(\cdot), g_1(\cdot)$  - RNN, идущая слева-направо:



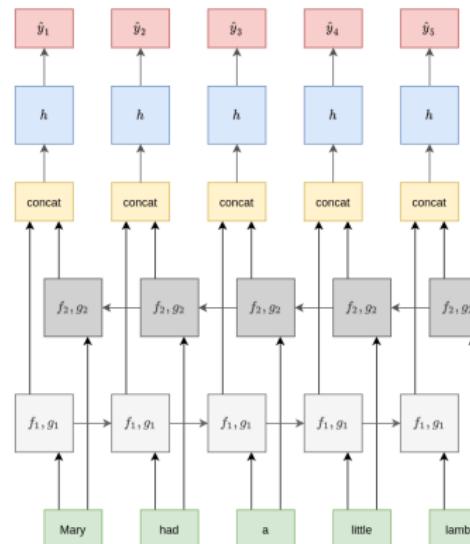
# Двунаправленная RNN

- Двунаправленная (bidirectional) RNN учитывает контекст и слева, и справа
  - пример: разметка частей речи, классификация событий на видео
- $f_2(\cdot), g_2(\cdot)$  - RNN, идущая справа-налево:



# Двунаправленная RNN

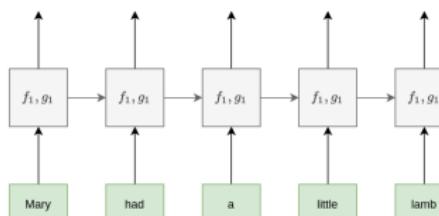
- Двунаправленная (bidirectional) RNN учитывает контекст и слева, и справа
  - пример: разметка частей речи, классификация событий на видео
- $h(\cdot)$  - MLP от конкатенации выходов двух RNN:



# Многослойная RNN

Многослойная (stacked) RNN позволяет извлекать более высокоранговые признаки.

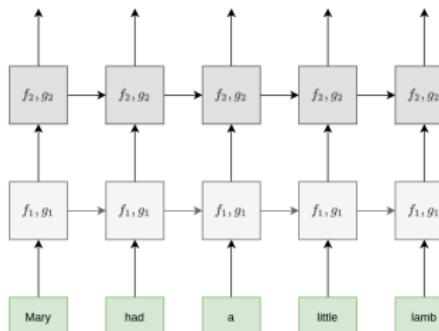
- можно комбинировать stacked+bidirectional RNN, прорасывать связи



# Многослойная RNN

Многослойная (stacked) RNN позволяет извлекать более высокоранговые признаки.

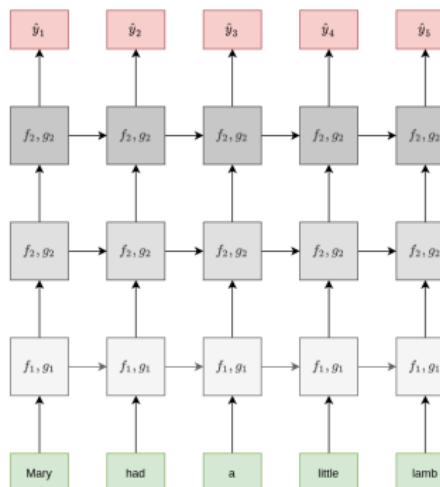
- можно комбинировать stacked+bidirectional RNN, прорасывать связи



# Многослойная RNN

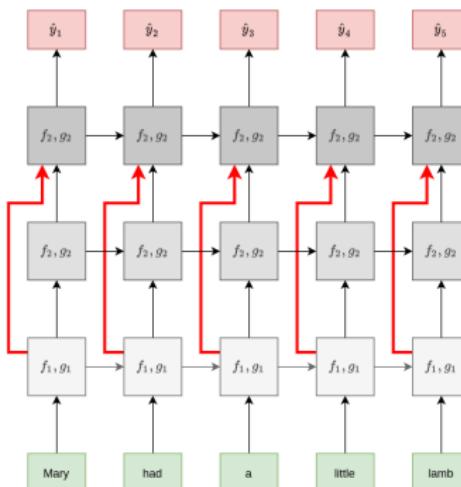
Многослойная (stacked) RNN позволяет извлекать более высокоранговые признаки.

- можно комбинировать stacked+bidirectional RNN, прорасывать связи



# Многослойная RNN с пробросом связей

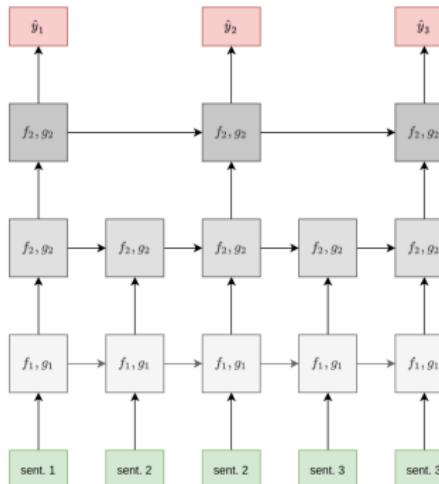
Вертикальный проброс связей, как в ResNet:



# Clockwork RNN

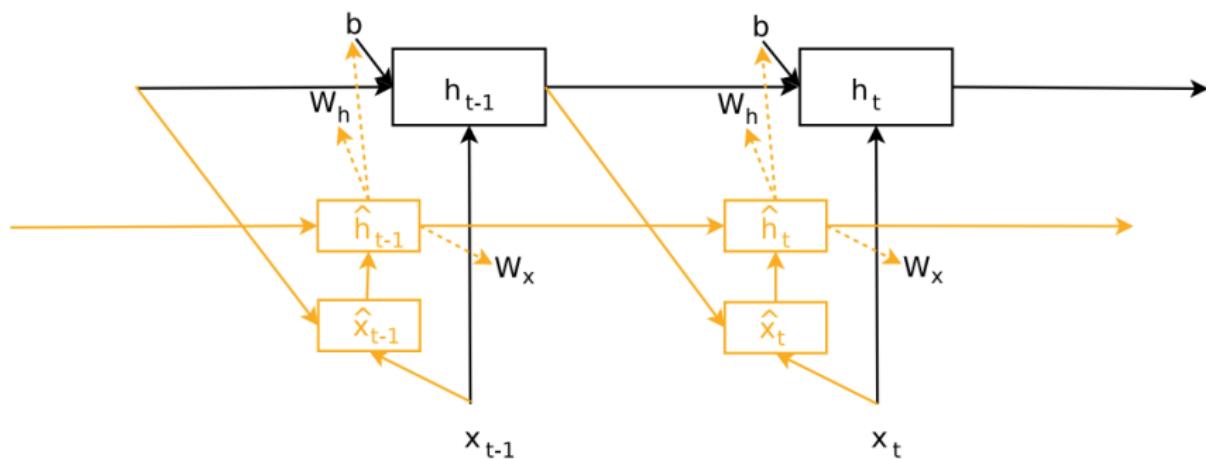
Clockwork RNN:

- нужний уровень идёт по словам
- верхний уровень идёт по предложениям



## Динамическая RNN<sup>2</sup>

В динамической RNN параметры меняются внешней гиперсетью в зависимости от входа и внутр. состояния сети:



<sup>2</sup><https://arxiv.org/pdf/1609.09106.pdf>

# Содержание

1 Основные понятия и приложения

2 Расширения RNN

3 Проблемы с градиентами

4 Рекуррентные сети с вентилями

5 Генеративные модели

# Производные в RNN

$$h_t = g \left( \underbrace{W_{xh}x_t + W_{hh}h_{t-1} + b_h}_{u_t} \right)$$

$$\hat{y}_t = f \left( \underbrace{W_{hy}h_t + b_y}_{v_t} \right)$$

Производные  $\mathcal{L}(\hat{y}_t, y_t)$  по  $b_y$  и  $W_{hy}$  считаются стандартно<sup>3</sup>:

$$\frac{\partial \mathcal{L}}{\partial b_y} = \frac{\partial \mathcal{L}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial v_t} \frac{\partial v_t}{\partial b_y} = \text{diag}(f'(v_t)) \frac{\partial \mathcal{L}}{\partial \hat{y}_t}$$

$$\frac{\partial \mathcal{L}}{\partial W_{hy}} = \frac{\partial \mathcal{L}}{\partial v_t} \frac{\partial v_t}{\partial W_{hy}} = \frac{\partial \mathcal{L}}{\partial v_t} h_t^T = \frac{\partial \mathcal{L}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial v_t} h_t^T = \text{diag}(f'(v_t)) \frac{\partial \mathcal{L}}{\partial \hat{y}_t} h_t^T$$

<sup>3</sup>Расчет векторных производных.

# Производные в RNN<sup>4</sup>

Производные по  $b_h$ ,  $W_{hh}$ ,  $W_{xh}$  из-за рекуррентности становятся

$$\frac{\partial \mathcal{L}}{\partial W_{hh}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{dh_t}{dW_{hh}} = \\ \frac{\partial \mathcal{L}}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \left( \frac{\partial h_t}{\partial W_{hh}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W_{hh}} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \frac{\partial h_{t-2}}{\partial W_{hh}} + \dots \right)$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial h_t}{\partial u_t} \frac{\partial u_t}{\partial h_{t-1}} = \text{diag}(g'(u_t)) W_{hh}$$

$$\frac{\partial h_t}{\partial h_{t-2}} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} = \text{diag}(g'(u_t)) W_{hh} \text{diag}(g'(u_{t-1})) W_{hh}$$

...

$$\frac{\partial h_t}{\partial h_k} = \underbrace{\text{diag}(g'(u_t)) W_{hh} \times \dots \times \text{diag}(g'(u_{k+1})) W_{hh}}_{t-k \text{ слагаемых}}$$

---

<sup>4</sup>Детальный вывод.

## Взрывающийся и затухающий градиент

$$\frac{\partial h_t}{\partial h_k} = \underbrace{\text{diag}(g'(u_t)) W_{hh} \times \dots \times \text{diag}(g'(u_{k+1})) W_{hh}}_{t-k \text{ слагаемых}}$$

- В частном случае  $h_t = Wh_{t-1}$ :

$$\frac{\partial h_t}{\partial h_k} = (W_{hh})^{t-k}$$

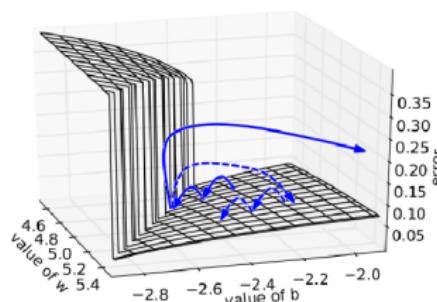
спектральное разложение (только для  $W_{hh} = W_{hh}^T$ )

$$W_{hh} = B : B^T \Lambda B \longrightarrow (W_{hh})^n = B^T \Lambda^n B$$

- $|\lambda_i| < 1$ : затухающий градиент (vanishing gradient)
  - RNN быстро забывает историю
- $|\lambda_i| > 1$ : взрывающийся градиент (exploding gradient)
  - нестабильное обучение

## Взрывающийся градиент

- Демонстрация проблемы взрывающегося градиента:



- В общем случае СЗ случаины => столкнемся с одной из проблем.
  - в обычных сетях проблема не так актуальна, т.к. там матрицы разные

## Решение проблемы взрывающегося градиента

Решение проблемы взрывающегося градиента:

- использовать truncated backpropagation through time
  - но RNN будет хуже учитывать историю
- добавить регуляризацию ( $\mathcal{L}$  будет более выпуклой)
- обрезать норму градиента по порогу  $t$  (gradient clipping):

$$\text{если } \|\nabla_w L(\hat{y}_i, y_i)\| \leq t : \quad w \rightarrow w - \varepsilon \nabla_w L(\hat{y}_i, y_i)$$

$$\text{если } \|\nabla_w L(\hat{y}_i, y_i)\| > t : \quad w \rightarrow w - \varepsilon \frac{t}{\|\nabla_\theta L(\hat{y}_i, y_i)\|} \nabla_\theta L(\hat{y}_i, y_i)$$

- можно обрезать по относительной величине  $\|\nabla_w L\| / \|w\|$
- использовать teacher forcing (не даёт RNN сильно уходить в сторону)

## Проблема затухающего градиента

- Нелинейности:  $ReLU$ ,  $LeakyReLU$
- Инициализировать  $W_{hh} = I$ .
- Добавить регуляризацию на несильное отклонение от ортогональности

$$\left\| W_{hh}^T W_{hh} - I \right\|_F^2 \text{ либо } \left\| W_{hh}^T W_{hh} - I \right\|_F^2 + \left\| W_{hh} W_h^T - I \right\|_F^2$$

т.к. у ортогональных матриц все  $|\lambda_i| = 1^5$ .

- Разделить состояние на быстро и медленно меняющиеся компоненты:  $h_t = [h_t^{slow}; h_t^{fast}]$ 
  - $h_t^{slow} = \alpha h_{t-1}^{slow} + (1 - \alpha) W_{slow} x_t$ ,  $\alpha \approx 1$
  - $h_t^{fast} = \sigma(V_{xh} x_t + V_{fast} h_{t-1}^{fast} + V_{slow} h_t^{slow})$
- Какие состояния быстро, а какие медленно менять? Можно использовать вентили, решающие это автоматически.

---

<sup>5</sup>Докажите.

# Содержание

1 Основные понятия и приложения

2 Расширения RNN

3 Проблемы с градиентами

4 Рекуррентные сети с вентилями

5 Генеративные модели

## Проблема обычной RNN

- Проблема обычной RNN: вся память перезаписывается на каждой итерации.
  - поэтому RNN быстро забывает прошлое
- Важно помнить прошлое глубоко в истории:
  - автоматические ответы на вопросы
  - машинный перевод
  - суммаризация текстов

## Проблема обычной RNN

- Проблема обычной RNN: вся память перезаписывается на каждой итерации.
  - поэтому RNN быстро забывает прошлое
- Важно помнить прошлое глубоко в истории:
  - автоматические ответы на вопросы
  - машинный перевод
  - суммаризация текстов
- Решение: использование вентилей



## Вентили

- Пусть  $s$ -старое состояние,  $x$ -новый вход,  $s'$ -новое состояние,  $s, s' \in \mathbb{R}^n$ .
- Вентиль  $g \in \{0, 1\} \in \mathbb{R}^n$  контролирует позиции, которые нужно обновить:
- Пример ( $\odot$  - поэлементное умножение):

$$\begin{array}{c}
 \begin{bmatrix} 8 \\ 11 \\ 3 \\ 7 \\ 5 \\ 15 \end{bmatrix} \leftarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \odot \begin{bmatrix} 10 \\ 11 \\ 12 \\ 13 \\ 14 \\ 15 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \odot \begin{bmatrix} 8 \\ 9 \\ 3 \\ 7 \\ 5 \\ 8 \end{bmatrix} \\
 s' \qquad g \qquad x \qquad (1-g) \qquad s
 \end{array}$$

- Какие вентили перекрывать? нужно их настраивать.
- Кусочно-постоянный вентиль не сможем оптимизировать.
  - поэтому используем гладкий вентиль  $g = \sigma(h_\theta(x, s))$
  - $h$ : дифференцируемая функция
  - $\theta$ : настраиваемые параметры

## Популярные RNN с вентилями

Популярные RNN с вентилями:

- Long short-term memory (LSTM)
  - "сеть долгой кратковременной памяти"
- Gated recurrent unit (GRU)

RNN с вентилями работают лучше на больших данных чем простые RNN.

# Long short-term memory (LSTM)<sup>6</sup>

$$\mathbf{f}_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad \text{forget gate}$$

$$\mathbf{i}_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad \text{input gate}$$

$$\mathbf{o}_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \quad \text{output gate}$$

$$c_t = \mathbf{f}_t \odot c_{t-1} + \mathbf{i}_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) \quad \text{inner state}$$

$$h_t = \mathbf{o}_t \odot \tanh(c_t) \quad \text{observed output}$$

$x_t$ -вход,  $h_t$  -выход.

Параметры:

- матрицы:  $W_f, U_f, W_i, U_i, W_o, U_o, W_c, U_c$
- вектора:  $b_f, b_i, b_o, b_c$
- инициализация:  $c_0, h_0$

---

<sup>6</sup>Long-short term memory (1997).

## Усложнение: peephole LSTM<sup>7</sup>

$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f)$	forget gate
$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1} + b_i)$	input gate
$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_0)$	output gate
$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c)$	inner state
$h_t = o_t \odot \tanh(c_t)$	observed output

$x_t$  - вход,  $h_t$  - выход.

Параметры:

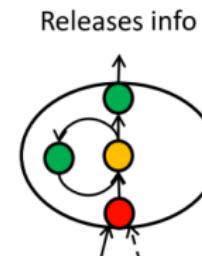
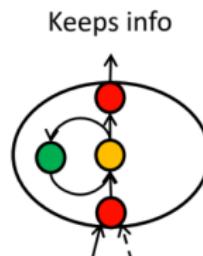
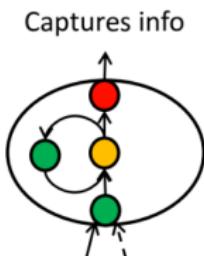
- матрицы:  $W_f, U_f, W_i, U_i, W_o, U_o, W_c, U_c, V_f, V_i, V_o$
- вектора:  $b_f, b_i, b_o, b_c$
- инициализация:  $c_0, h_0$

---

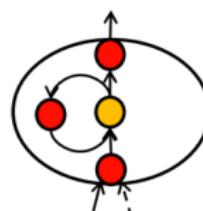
<sup>7</sup>Recurrent nets that time and count (2000).

# Иллюстрация

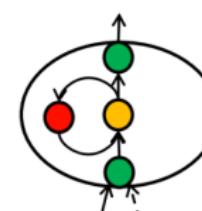
Вход-внизу, память-в середине, выход-наверху.



Erases info



= RNN



● - gate is close

● - gate is open

# Комментарии

$$\begin{aligned}
 f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) && \text{forget gate} \\
 i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) && \text{input gate} \\
 o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) && \text{output gate} \\
 c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_c x_t + U_c h_{t-1} + b_c) && \text{inner state} \\
 h_t &= o_t \odot \tanh(c_t) && \text{observed output}
 \end{aligned}$$

- $c_t = f_t \odot c_{t-1} + \dots$  обеспечивает более долгую память (но все равно не бесконечную - см. memory networks)
- Рекомендуется инициализация  $b_f \geq 1$ 
  - вначале сеть пытается запоминать все
- Эффективное упрощение: coupled forget and input gate<sup>8</sup>:

$$f_t = 1 - i_t$$

<sup>8</sup><https://arxiv.org/pdf/1503.04069.pdf>

# Gated recurrent unit (GRU)<sup>9</sup> - упрощение LSTM

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$

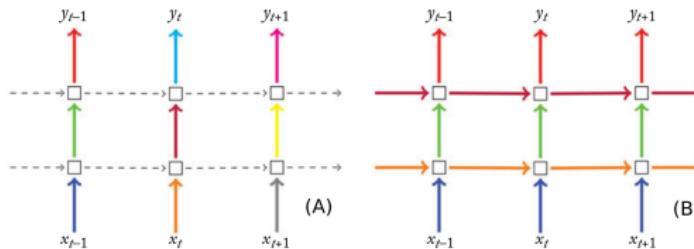
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tanh(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h)$$

- $x_t$  - вход,  $z_t$  - forget gate,  $r_t$  - input gate,  $h_t$  - и состояние, и выход
- Параметры:
  - матрицы:  $W_z, U_z, W_r, U_r, W_h, U_h$
  - вектора:  $b_z, b_r, b_h$
  - инициализация:  $h_0$
- По сравнению с LSTM 2, а не 3 вентиля (coupled forget and input gate), состояние=выход.
  - меньше параметров и переобучения, но проще - нужно сравнивать на данных.

<sup>9</sup><https://arxiv.org/pdf/1406.1078.pdf>

# Dropout в RNN

- Проблема: dropout во времени приводит к полному забыванию истории.
- Решения:
  - 1 применять dropout к нерекуррентным связям<sup>10</sup> (a)
  - 2 Variational dropout<sup>11</sup> (b):
    - для каждой последовательности генерировать и фиксировать маску для верт. и горизонт. связей.
    - маска не меняется во времени, долгосрочные связи остаются.



<sup>10</sup><https://arxiv.org/pdf/1312.4569.pdf>

<sup>11</sup><https://arxiv.org/pdf/1512.05287.pdf>

## Dropout в RNN: ZoneOut<sup>12</sup>

ZoneOut: состояния стохастически сохраняются (а не зануляются как в DropOut) с пред. шага.

$$\text{RNN: } h_t = d_t \odot g(W_{xh}x_t + W_{hh}h_{t-1} + b_h) + (1 - d_t) \odot I$$

LSTM:

$$i_t, f_t, o_t = \sigma(W_x x_t + W_h h_{t-1} + b)$$

$$g_t = \tanh(W_{xg} x_t + W_{hg} h_{t-1} + b_g)$$

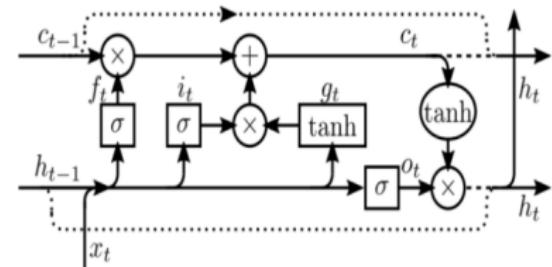
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$

ZoneOut:

$$c_t = d_t^c \odot c_{t-1} + (1 - d_t^c) \odot (f_t \odot c_{t-1} + i_t \odot g_t)$$

$$h_t = d_t^h \odot h_{t-1} + (1 - d_t^h) \odot (o_t \odot \tanh(f_t \odot c_{t-1} + i_t \odot g_t))$$



<sup>12</sup><https://arxiv.org/pdf/1606.01305.pdf>

# Quasi-Recurrent Neural Networks<sup>13</sup>

Quasi-Recurrent Neural Networks - рекуррентная архитектура, заточенная на лёгкую параллелизацию. Это комбинация:

- назад смотрящих  $z_t = conv(x_t, x_{t-1}, \dots x_{t-K})$
- простых рекуррентных преобразований, независимых по каналам, например

$$h_t = f_t \odot h_{t-1} + (1 - f_t) \odot z_t$$

где  $Z = conv(X)$ ,  $F = \sigma(conv(X))$  - везде свёртки, поэтому параллелизация.

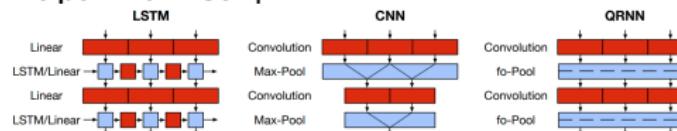


Figure 1: Block diagrams showing the computation structure of the QRNN compared with typical LSTM and CNN architectures. Red signifies convolutions or matrix multiplications; a continuous block means that those computations can proceed in parallel. Blue signifies parameterless functions that operate in parallel along the channel/feature dimension. LSTMs can be factored into (red) linear blocks and (blue) elementwise blocks, but computation at each timestep still depends on the results from the previous timestep.

<sup>13</sup><https://arxiv.org/pdf/1611.01576v2.pdf>

# Содержание

- 1 Основные понятия и приложения
- 2 Расширения RNN
- 3 Проблемы с градиентами
- 4 Рекуррентные сети с вентилями
- 5 Генеративные модели
  - Настройка генеративных моделей
  - Примеры генерации
  - Проблемы генерации и их решения

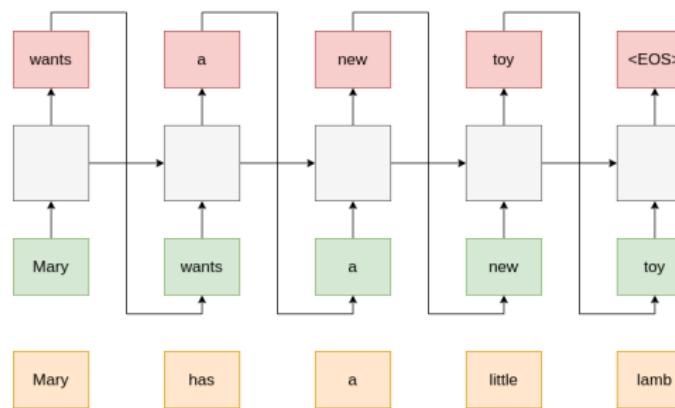
5

## Генеративные модели

- Настройка генеративных моделей
- Примеры генерации
- Проблемы генерации и их решения

# Настройка генеративных моделей: free sampling

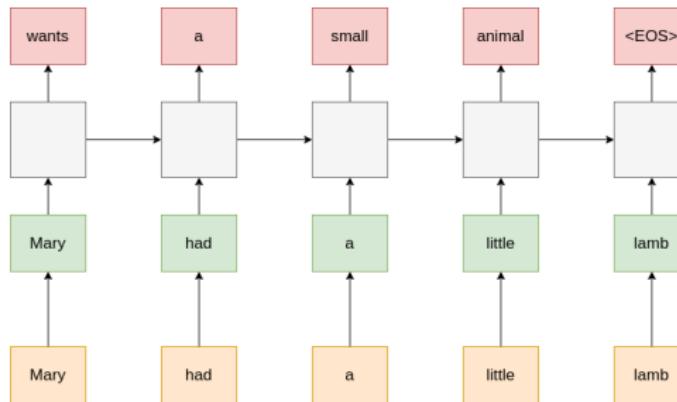
- Генеративные модели: one-to-many, many-to-many.
- Стандартное обучение (free sampling)



Если модель ушла в сторону - ей тяжелее вернуться.

- Недостаток: медленная сходимость
- Достоинство: устойчивость - даже уйдя в сторону, модель учится восстанавливаться

# Настройка генеративных моделей: teacher forcing



- Teacher forcing:
  - обучение: подаём на вход всегда истинный вход
  - применение: подаём выход пред. шага
- Достоинство: быстрее сходимость
- Недостаток: различаются обучение и применение.
  - модель хуже восстанавливается, допустив ошибку, т.к. при обучении видела только правильные траектории.

# Комбинации обычного обучения и teacher forcing

Последовательное обучение:

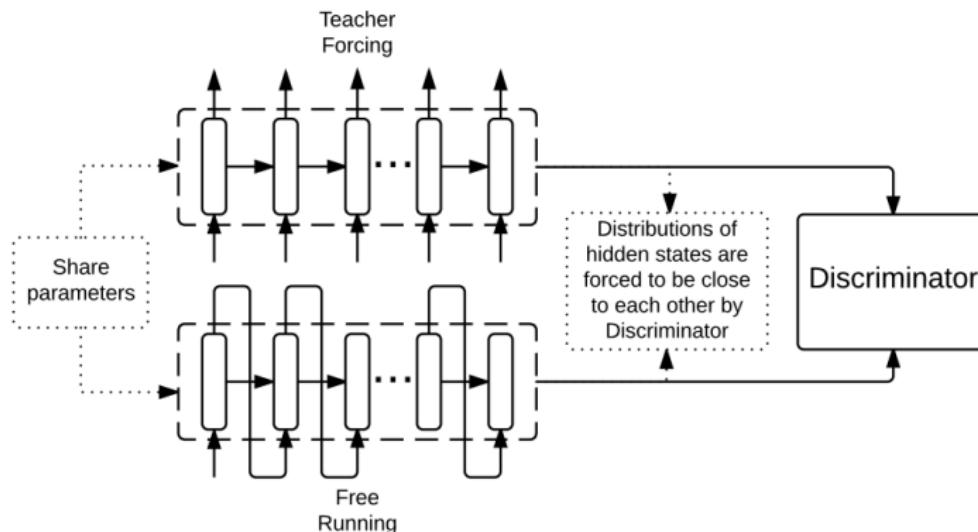
- ① предобучим модель с помощью teacher forcing
- ② дообучим обычным методом

Стохастическое смешивание: на каждом шаге  $t$

$$x_t = \begin{cases} y_{t-1}, & p \\ \hat{y}_{t-1}, & 1 - p \end{cases}$$

# Комбинации обычного обучения и teacher forcing

Professor forcing<sup>14</sup>: обучаем обоими способами (parameter sharing), дополнительно пытаясь обмануть дискриминатор различающий free sampling и teacher forcing обучение.



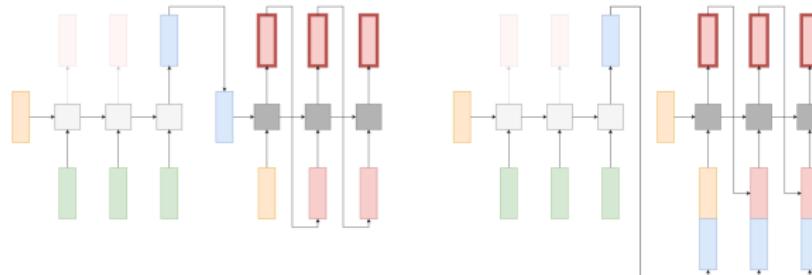
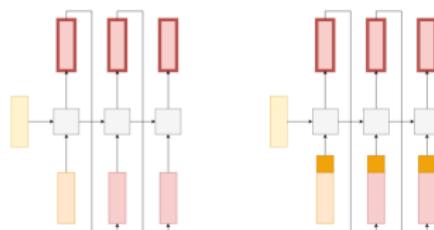
<sup>14</sup><https://arxiv.org/pdf/1610.09038.pdf>

## 5 Генеративные модели

- Настройка генеративных моделей
- Примеры генерации
- Проблемы генерации и их решения

## Генерация слов в one-to-many, many-to-many

Рассмотрим генерацию слов в архитектурах one-to-many, many-to-many:



## Генерация текстов<sup>15</sup>

Генерация текста "по Шекспиру":

PANDARUS:

Alas, I think he shall be come approached and the day  
When little strain would be attain'd into being never fed,  
And who is but a chain and subjects of his death,  
I should not sleep.

Second Senator:

They are away this miseries, produced upon my soul,  
Breaking and strongly should be buried, when I perish  
The earth and thoughts of many states.

DUKE VINCENTIO:

Well, your wit is in the care of side and that.

Second Lord:

They would be ruled after this chamber, and  
my fair nues begun out of the fact, to be conveyed,  
Whose noble souls I'll have the heart of the wars.

<sup>15</sup> <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

# Генерация текстов<sup>16</sup>

## Генерация тестов "википедии":

Naturalism and decision for the majority of Arab countries' capitalide was grounded by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal in the [[Protestant Immineners]], which could be said to be directly in Cantonese Communication, which followed a ceremony and set inspired prison, training. The emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known in western [[Scotland]], near Italy to the conquest of India with the conflict.

```
{ { cite journal | id=Cerling Nonforest Department|format=Newlymeslated|none } }
  'www.e-complete'.

'''See also'''': [[List of ethical consent processing]]

== See also ==
*[[Iender dome of the ED]]
*[[Anti-autism]]
```

<sup>16</sup><http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

## Генеративные модели

## Примеры генерации

Генерация текстов<sup>17</sup>

## Генерация тестов "научной статьи":

*Proof.* Omitted.  $\square$

**Lemma 0.1.** Let  $\mathcal{C}$  be a set of the construction.

Let  $\mathcal{C}$  be a gerber covering. Let  $\mathcal{F}$  be a quasi-coherent sheaves of  $\mathcal{O}$ -modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

*Proof.* This is an algebraic space with the composition of sheaves  $\mathcal{F}$  on  $X_{\text{itale}}$  we have

$$\mathcal{O}_X(\mathcal{F}) = \{\text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where  $\mathcal{G}$  defines an isomorphism  $\mathcal{F} \rightarrow \mathcal{G}$  of  $\mathcal{O}$ -modules.  $\square$

**Lemma 0.2.** This is an integer  $Z$  is injective.

*Proof.* See Spaces, Lemma ??.

**Lemma 0.3.** Let  $S$  be a scheme. Let  $X$  be a scheme and  $X$  is an affine open covering. Let  $\mathcal{U} \subset \mathcal{X}$  be a canonical and locally of finite type. Let  $X$  be a scheme. Let  $X$  be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let  $X$  be a scheme. Let  $X$  be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

be a morphism of algebraic spaces over  $S$  and  $Y$ .

*Proof.* Let  $X$  be a nonzero scheme of  $X$ . Let  $X$  be an algebraic space. Let  $\mathcal{F}$  be a quasi-coherent sheaf of  $\mathcal{O}_X$ -modules. The following are equivalent

- (1)  $\mathcal{F}$  is an algebraic space over  $S$ .
- (2) If  $X$  is an affine open covering.

Consider a common structure on  $X$  and  $X$  the functor  $\mathcal{O}_X(U)$  which is locally of finite type.  $\square$

This since  $\mathcal{F} \in \mathcal{F}$  and  $x \in \mathcal{G}$  the diagram

$$\begin{array}{ccc} S & \longrightarrow & \\ \downarrow & & \\ \xi & \longrightarrow & \mathcal{O}_{X'} \\ & \nearrow \text{gor}_x & \downarrow \\ & = \alpha' & \\ & \downarrow & \\ & = \alpha' & \longrightarrow \\ & & \downarrow & \\ \text{Spec}(K_v) & & \text{Mor}_{\mathcal{S}\text{ets}} & d(\mathcal{O}_{X_{\mathcal{S}\text{ets}}}, \mathcal{G}) \\ & & \downarrow & \\ & & X & \end{array}$$

is a limit. Then  $\mathcal{G}$  is a finite type and assume  $S$  is a flat and  $\mathcal{F}$  and  $\mathcal{G}$  is a finite type  $f_*$ . This is of finite type diagrams, and

- the composition of  $\mathcal{G}$  is a regular sequence,
- $\mathcal{O}_{X'}$  is a sheaf of rings.

*Proof.* We have see that  $X = \text{Spec}(R)$  and  $\mathcal{F}$  is a finite type representable by algebraic space. The property  $\mathcal{F}$  is a finite morphism of algebraic stacks. Then the cohomology of  $X$  is an open neighbourhood of  $U$ .  $\square$

*Proof.* This is clear that  $\mathcal{G}$  is a finite presentation, see Lemmas ??.

A reduced above we conclude that  $U$  is an open covering of  $C$ . The functor  $\mathcal{F}$  is a "field"

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_x \dashv (\mathcal{O}_{X_{\text{red}}}) \rightarrow \mathcal{O}_{X,x}^{-1} \mathcal{O}_{X,x}(\mathcal{O}_{X,x}^0)$$

is an isomorphism of covering of  $\mathcal{O}_{X,x}$ . If  $\mathcal{F}$  is the unique element of  $\mathcal{F}$  such that  $X$  is an isomorphism.

The property  $\mathcal{F}$  is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme  $\mathcal{O}_X$ -algebra with  $\mathcal{F}$  are opens of finite type over  $S$ . If  $\mathcal{F}$  is a scheme theoretic image points.  $\square$

If  $\mathcal{F}$  is a finite direct sum  $\mathcal{O}_X$ , is a closed immersion, see Lemma ??.

This is a sequence of  $\mathcal{F}$  is a similar morphism.

<sup>17</sup> <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

# Генерация кода<sup>18</sup>

Генерация "исходного кода Линукса":

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
```

<sup>18</sup> <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

# Интерпретация отдельных нейронов

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.
```

```
Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

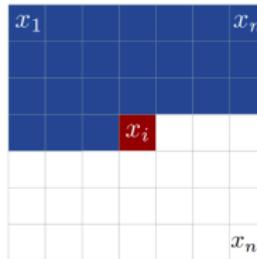
Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

# PixelRNN<sup>19</sup>: генерация изображений

- PixelRNN - последовательно генерируется каждый пиксель изображения  $N \times N$  слева-направо сверху-вниз:

$$p(x) = \prod_{i=1}^{N^2} p(x_i | x_1, \dots, x_{i-1})$$

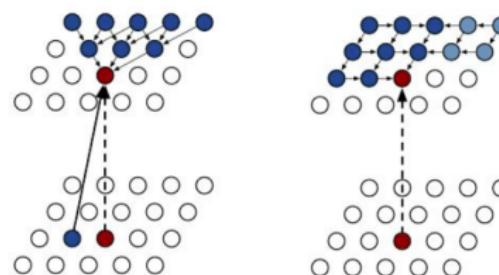


- Для пикселя: последовательная классификация из 255 классов:  $p(R|context)$ ,  $p(G|R, context)$ ,  $p(B|R, G, context)$

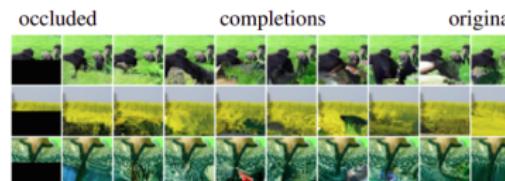
<sup>19</sup><https://arxiv.org/pdf/1601.06759.pdf>

# PixelRNN

- Можем учитывать разную информацию о предыдущих генерациях и состояниях:

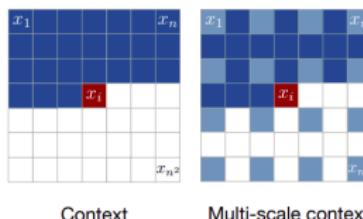


Возможное применение - заполнение пропущенных областей:

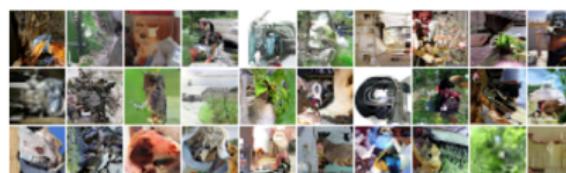


# PixelRNN

- Лучше работает 2x стадийная генерация:
  - генерируем пиксели уменьшенного изображения
  - генерируем пиксели высокоразмерного изображения при условии генерации первого шага (идея-применить к superresolution)



Примеры генерации с нуля:



# Условная генерация: описание изображений<sup>20</sup>

Описание изображений (image captioning): CNN извлекает признаковое описание изображения, которое подается RNN.



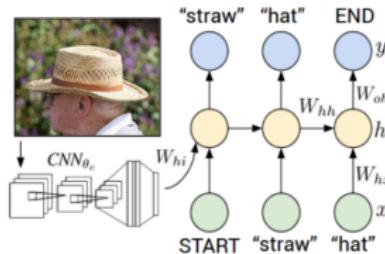
"man in black shirt is playing guitar."



"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."



<sup>20</sup>Больше по теме.

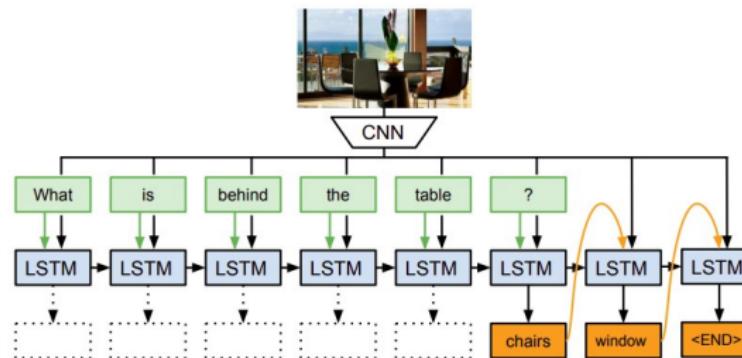
# Ответы на вопросы по изображению<sup>21</sup>

## Ответы на вопросы по изображению (visual question answering)

Is the umbrella upside down?  
yes  
no



How many children are in the bed?  
2  
1



<sup>21</sup>Обзор методов: <https://arxiv.org/pdf/1605.02697.pdf>.

## 5 Генеративные модели

- Настройка генеративных моделей
- Примеры генерации
- Проблемы генерации и их решения

## Проблема новых слов

- В тестовой выборке могут появляться слова, которые RNN не видела в обучении.
- Решение 1 - word dropout: заменяем случайные слова в обучении на тэг `<Unknown>`.
- Модель учится строить прогнозы с присутствием незнакомых слов.
- Улучшение в качестве векторных представлений слов использовать:

[word embedding; word characters embedding]

- тогда даже в случае `<Unknown>` получаем информативное представление новых слов.
- word characters embedding получаем проходом др. RNN по символам слова
  - лучше по би/триграммам символов

# Генерация теста

- Пусть  $s_1, \dots, s_W$  - готовность модели предсказывать слова  $1, 2, \dots, W$ . Как генерировать след. слово?
- Макс. вероятное слово - самый правильный текст, но слишком однообразный

$$\hat{y}_t = \arg \max_i \{s_i\}$$

- Сэмплировать слово - текст часто неправильный, но макс. разнообразный

$$\hat{y}_t \sim \text{Categorical} \left( \left\{ \frac{e^{s_i}}{\sum_k e^{s_k}} \right\}_i \right)$$

- Температура  $\tau$  управляет противоречием правильность-разнообразность:

$$\hat{y}_t \sim \text{Categorical} \left( \left\{ \frac{e^{s_i/\tau}}{\sum_k e^{s_k/\tau}} \right\}_i \right)$$

# Генерация предложений

- Слова генерируются последовательно:
  - $y_1$ , score= $p(y_1|x)$
  - $y_2$ , score= $p(y_1y_2|x) = p(y_1|x)p(y_2|y_1, x)$
  - $y_3$ , score= $p(y_1|x)p(y_2|y_1, x)p(y_3|y_1, y_2, x)$

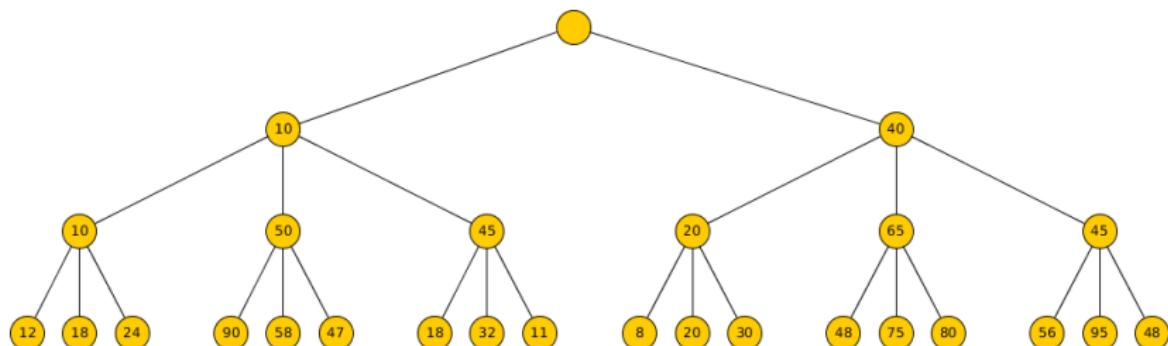
...

- Хотим максимально вероятную последовательность

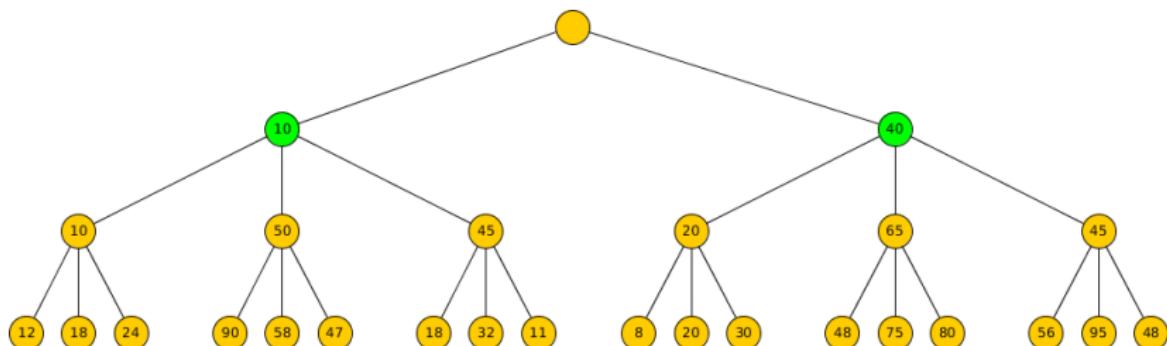
$$p(y_1y_2\dots y_N|x) \rightarrow \max_{y_1, \dots, y_N}$$

- Выбор следующего слова на каждом шаге = выбор пути в дереве выигрышей.
- Жадная стратегия - выбор максимально вероятного  $p(y_i|y_1, \dots, y_{i-1}, x)$  на каждом шаге  $i$ .

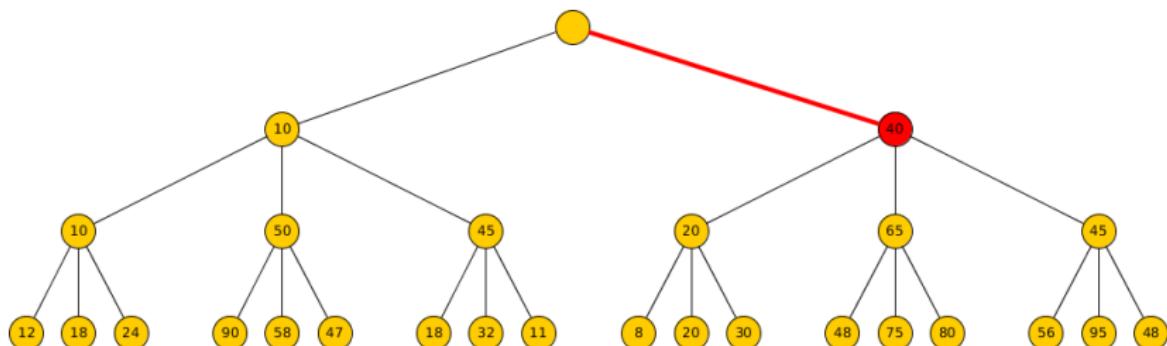
# Жадная стратегия



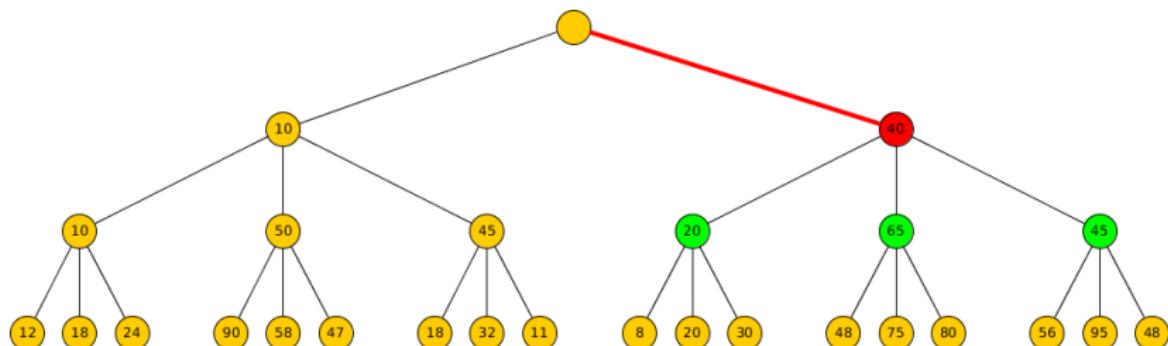
# Жадная стратегия



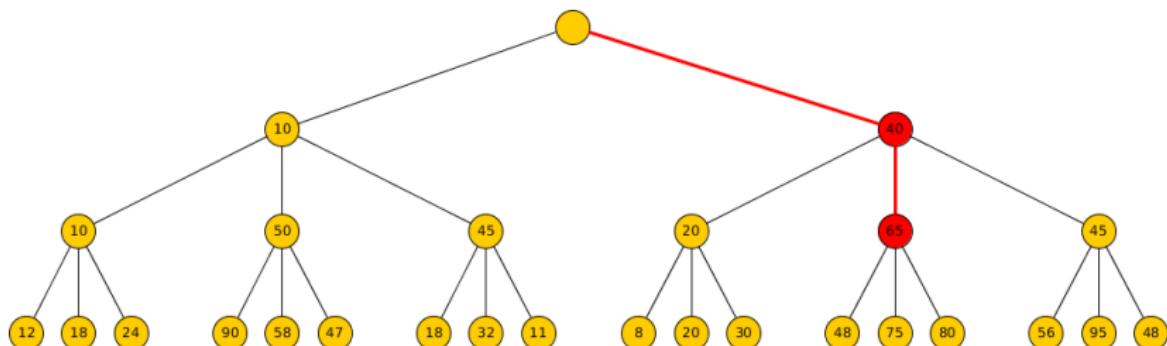
# Жадная стратегия



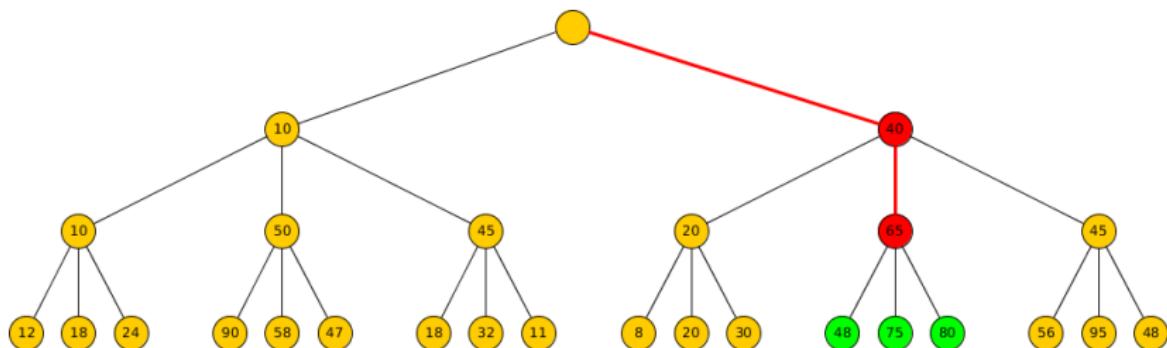
# Жадная стратегия



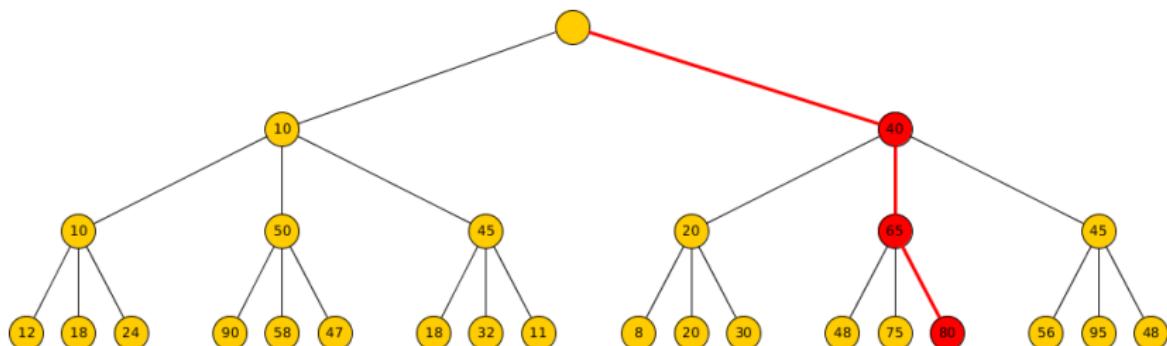
# Жадная стратегия



# Жадная стратегия



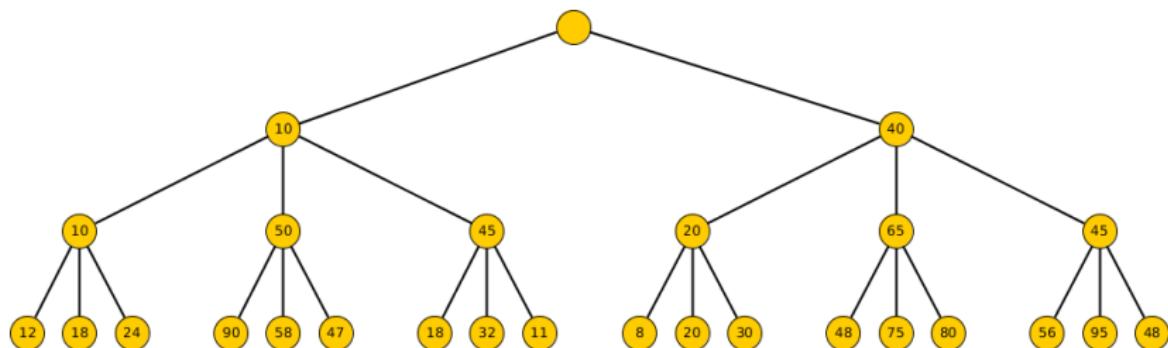
# Жадная стратегия



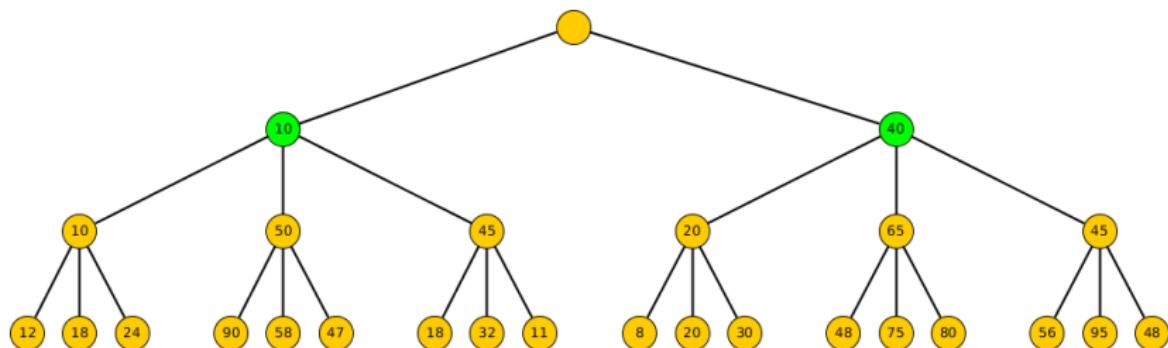
# Лучевой поиск

- Жадная стратегия сложности  $O(WN)$ ,  $W = \#\text{слов}$ ,  $N$ -длина посл-ти.
  - самая эффективная, оптимальность лишь на 1 шаг вперёд
- Лучевой поиск (beam search): расширенный жадный поиск в ширину
  - поддерживаем  $K$  лучших альтернатив, вместо одной.
  - выбираем лучшую из  $K$  в самом конце
  - сложность  $O(KWN)$
- $K \in [1, 2, \dots W^N]$  контролирует выбор: скорость $\leftarrow\rightarrow$ полнота перебора.

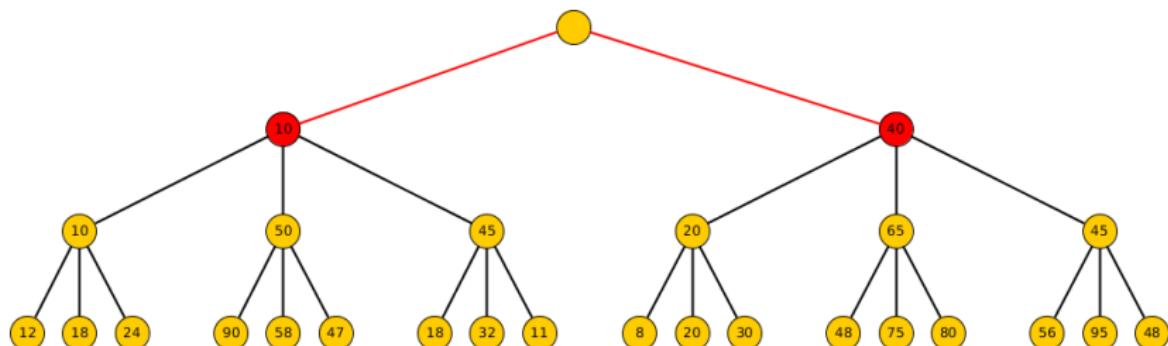
# Лучевой поиск ( $K = 2$ )



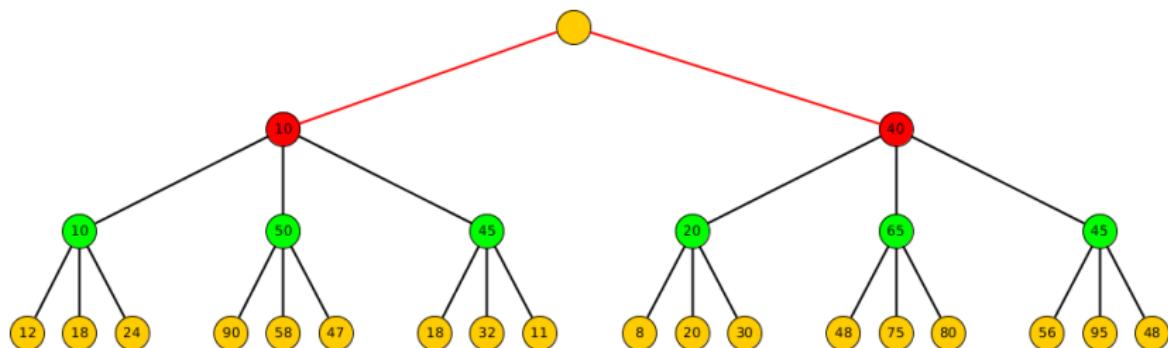
# Лучевой поиск ( $K = 2$ )



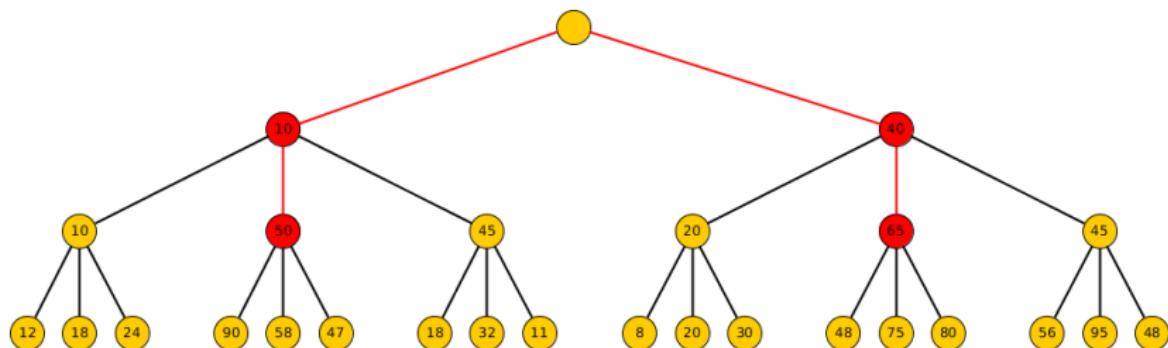
# Лучевой поиск ( $K = 2$ )



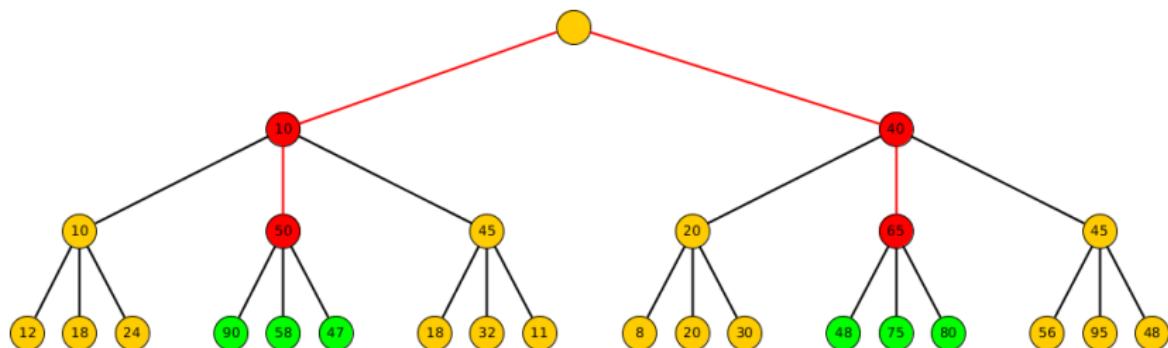
# Лучевой поиск ( $K = 2$ )



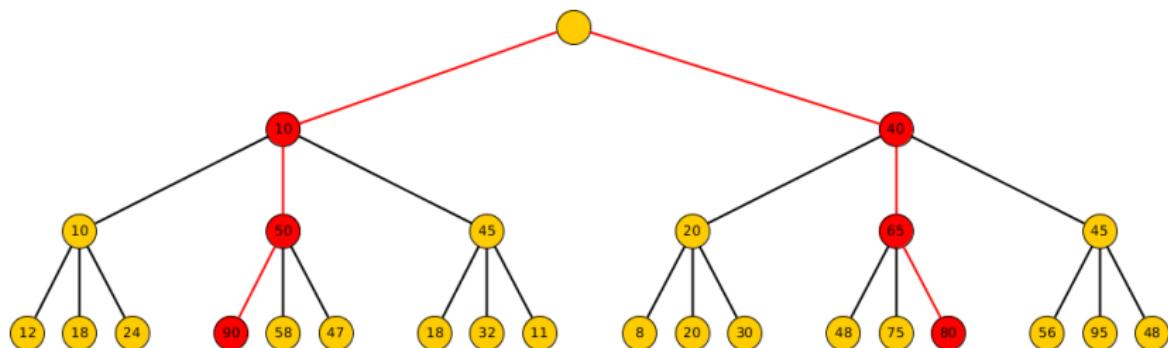
# Лучевой поиск ( $K = 2$ )



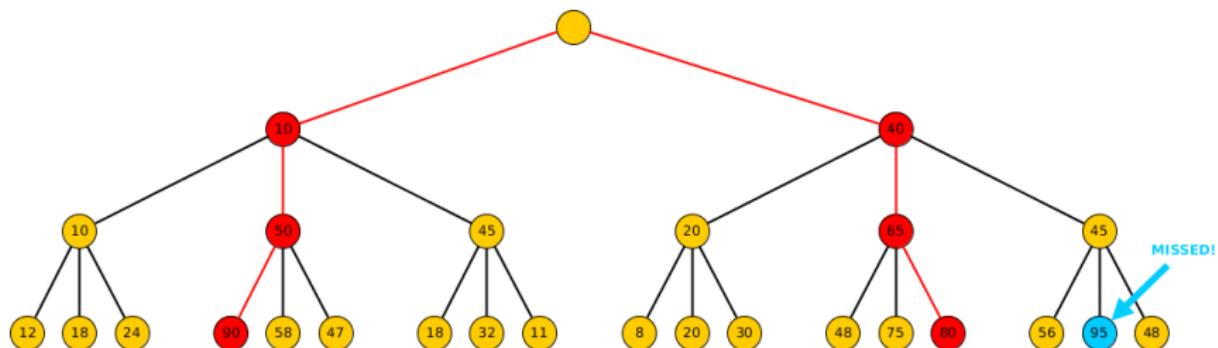
# Лучевой поиск ( $K = 2$ )



# Лучевой поиск ( $K = 2$ )



# Лучевой поиск ( $K = 2$ )



## Заключение

- RNN обрабатывают последовательности произв. длины.
  - используя общие параметры => затухающий, взрывающийся градиент
- Архитектуры: many-to-one, one-to-many, many-to-many.
- Основные архитектуры: LSTM, GRU.
- Обучение: backpropagation through time (на практике trunkated и минибатчами)
- Используется специальный DropOut
- Инициализация: ортогональные матрицы, смещения -  $\uparrow$  запоминание истории.