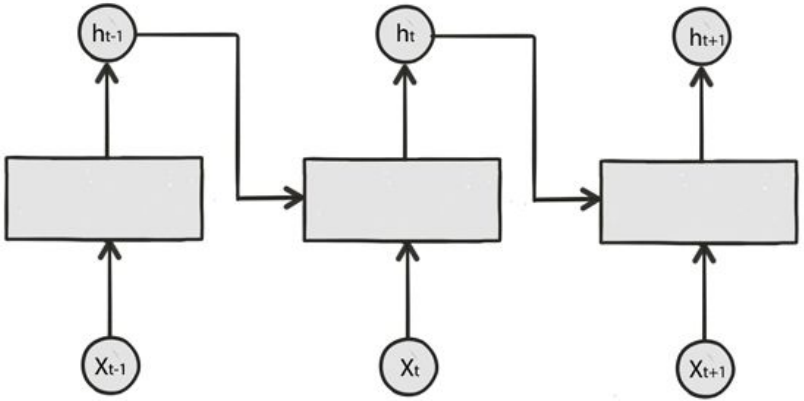
Вывод алгоритмических закономерностей с помощью Stack RNN

Харатян Андрей

Рекуррентные нейронные сети (RNN)
Basics



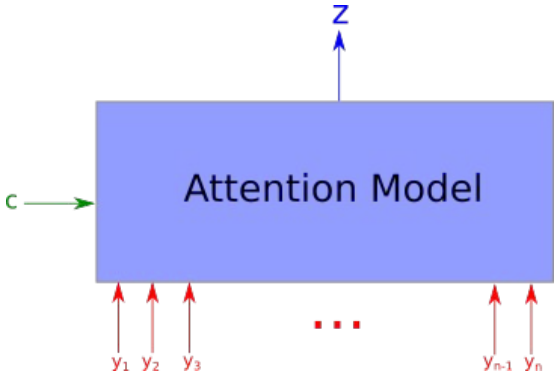
Elman model

$$h_t = \sigma(U x_t + R_h h_{t-1})$$

$$y_t = softmax(V h_t)$$

HUNGRY FOR McDonalds AND Attention

Рекуррентные нейронные сети (RNN)
Attention

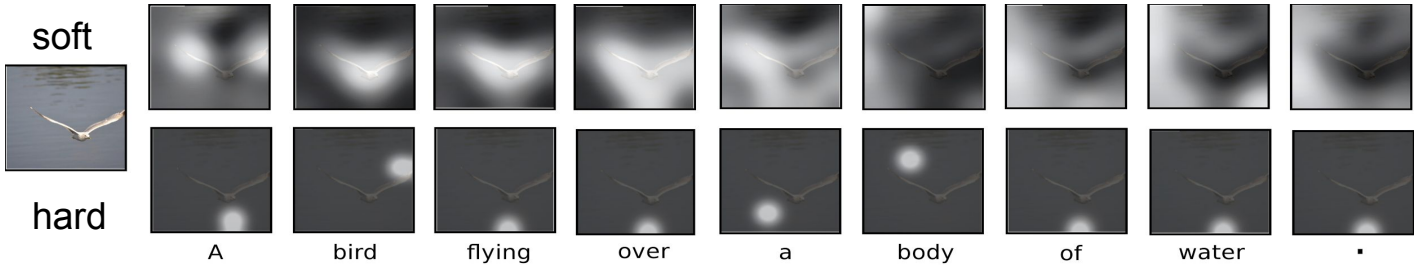


$$\text{next word} = f(image, \text{last word})$$

$$h_t = f(x, h_{t-1})$$

$$\text{next word} = g(h_t)$$

$$h_t = f(attention(x, h_{t-1}), h_{t-1})$$

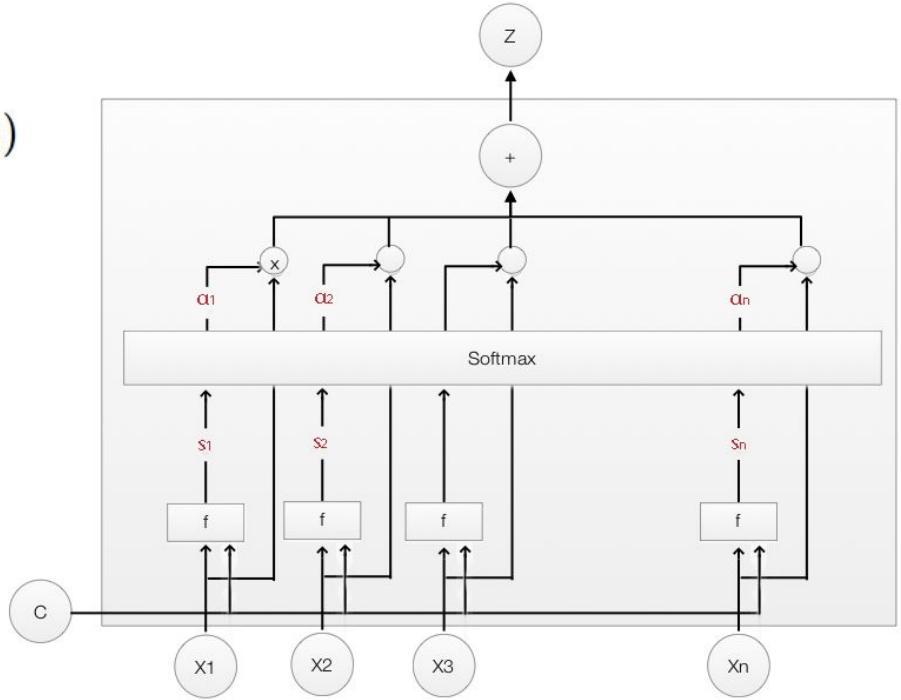


Рекуррентные нейронные сети (RNN)
Soft Attention

$$s_i = \tanh(W_c C + W_x X_i) = \tanh(W_c h_{t-1} + W_x x_i)$$

$$\alpha_i = softmax(s_1, s_2, \ldots, s_i, \ldots)$$

$$Z = \sum_i \alpha_i x_i$$

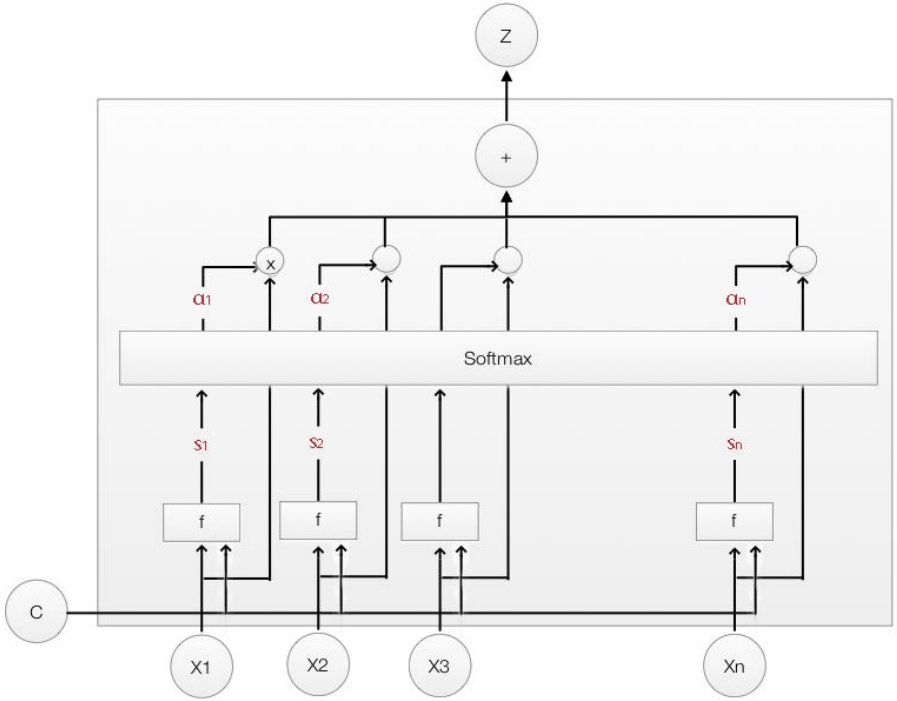


Рекуррентные нейронные сети (RNN)
Soft Attention

Attention as differentiable layer

No sampling

Training with backprop



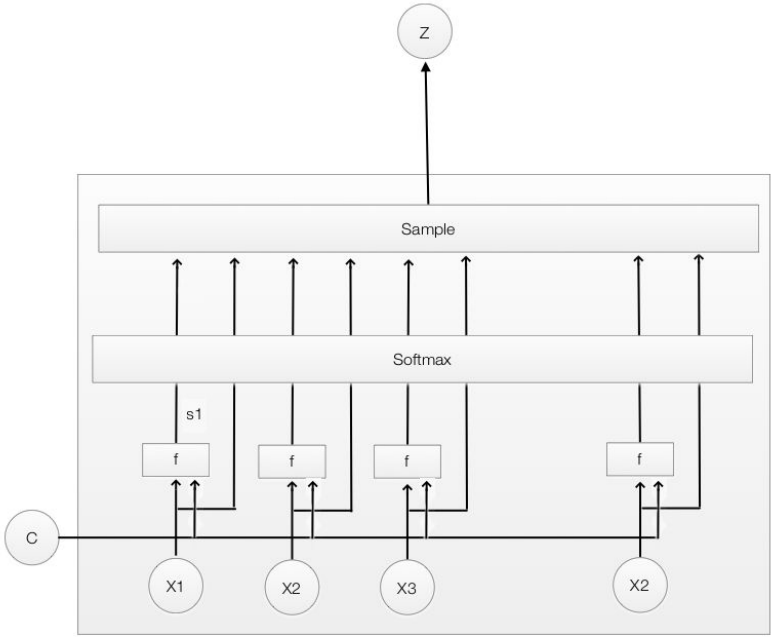
Рекуррентные нейронные сети (RNN)
Hard Attention

Attention as stochastic process; sampling

Supports discrete decisions (number of steps)

Training with REINFORCE

$$Z \sim x_i, \alpha_i$$



Рекуррентные нейронные сети (RNN)
Attention

Hard attention

Better results (sometimes?)

More principled

Hard to train

Soft attention

Not always applicable

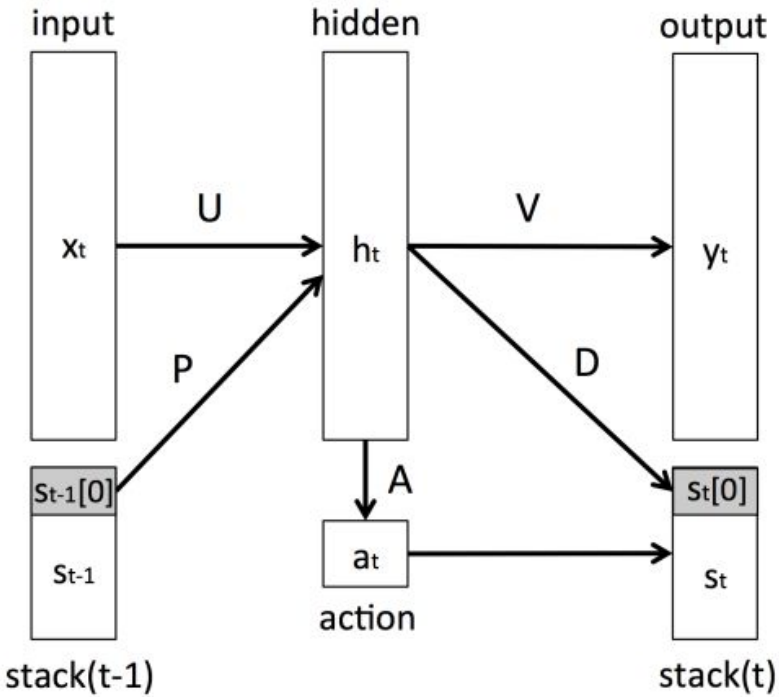
Easy to train

	BLEU				
Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
Log Bilinear°	70.8	48.9	34.4	24.3	20.03
Soft-Attention	70.7	49.2	34.4	24.3	**23.90**
Hard-Attention	**71.8**	**50.4**	**35.7**	**25.0**	23.04

Задача

Sequence generator	Example
$\{a^n b^n \mid n > 0\}$	aab**ba**aab**bba**b**a**aaaab**bbbb**
$\{a^n b^n c^n \mid n > 0\}$	aaab**bbccca**b**ca**aaaab**bbbbccccc**
$\{a^n b^n c^n d^n \mid n > 0\}$	aab**bccdda**aab**bbcccdddab**c**d**
$\{a^n b^{2n} \mid n > 0\}$	aab**bbba**aab**bbbbba**b**b**
$\{a^n b^m c^{n+m} \mid n, m > 0\}$	aabc**cca**aabbc**ccccca**bc**c**
$n \in [1, k],\ X \to nXn,\ X \to =$	$(k = 2)$ 12=**21**2122=**2212**11121=**12111**

Stack RNN: Концепция



Добавим стек, в котором будем сохранять какие-то сведения о данных и через который будем обновлять скрытые состояния

3 Операции: PUSH, POP и NO-OP

PUSH - добавляет новый элемент на верх стека

POP - удаляет верхний элемент стека

NO-OP - ничего не делает

Stack RNN: Формулы

$$s_t[0] = a_t[\texttt{PUSH}]\sigma(Dh_t) + a_t[\texttt{POP}]s_{t-1}[1]$$

$$s_t[i] = a_t[\texttt{PUSH}]s_{t-1}[i-1] + a_t[\texttt{POP}]s_{t-1}[i+1]$$

$$s_t[0] = a_t[\texttt{PUSH}]\sigma(Dh_t) + a_t[\texttt{POP}]s_{t-1}[1] + a_t[\texttt{NO-OP}]s_{t-1}[0]$$

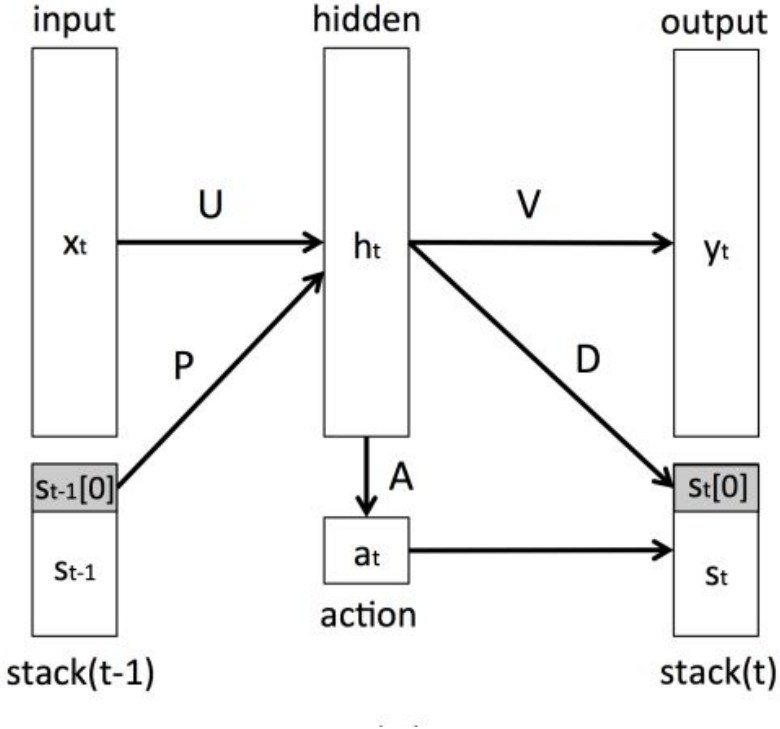
Stack RNN: Формулы

$$a_t = softmax(Ah_t)$$

$$h_t = \sigma\left(Ux_t + Rh_{t-1} + Ps^k_{t-1}\right)$$

$$s_t[0] = a_t[\texttt{PUSH}]\sigma(Dh_t) + a_t[\texttt{POP}]s_{t-1}[1]$$

$$y_t = softmax(Vh_t)$$



n-Stack RNN

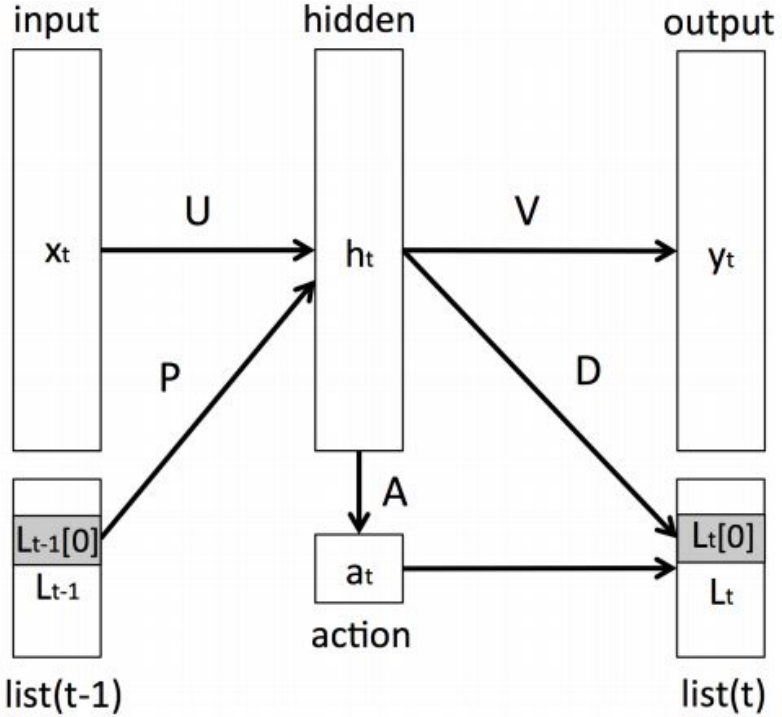
Один стек это не очень круто так как можем делать только одно действие в один момент времени.

Сделаем несколько

??????

PROFIT

List RNN: Концепция



Все то же самое, только теперь вместо стека двусвязный список

3 Операции: LEFT, RIGHT и INSERT и пишущая головка (HEAD)

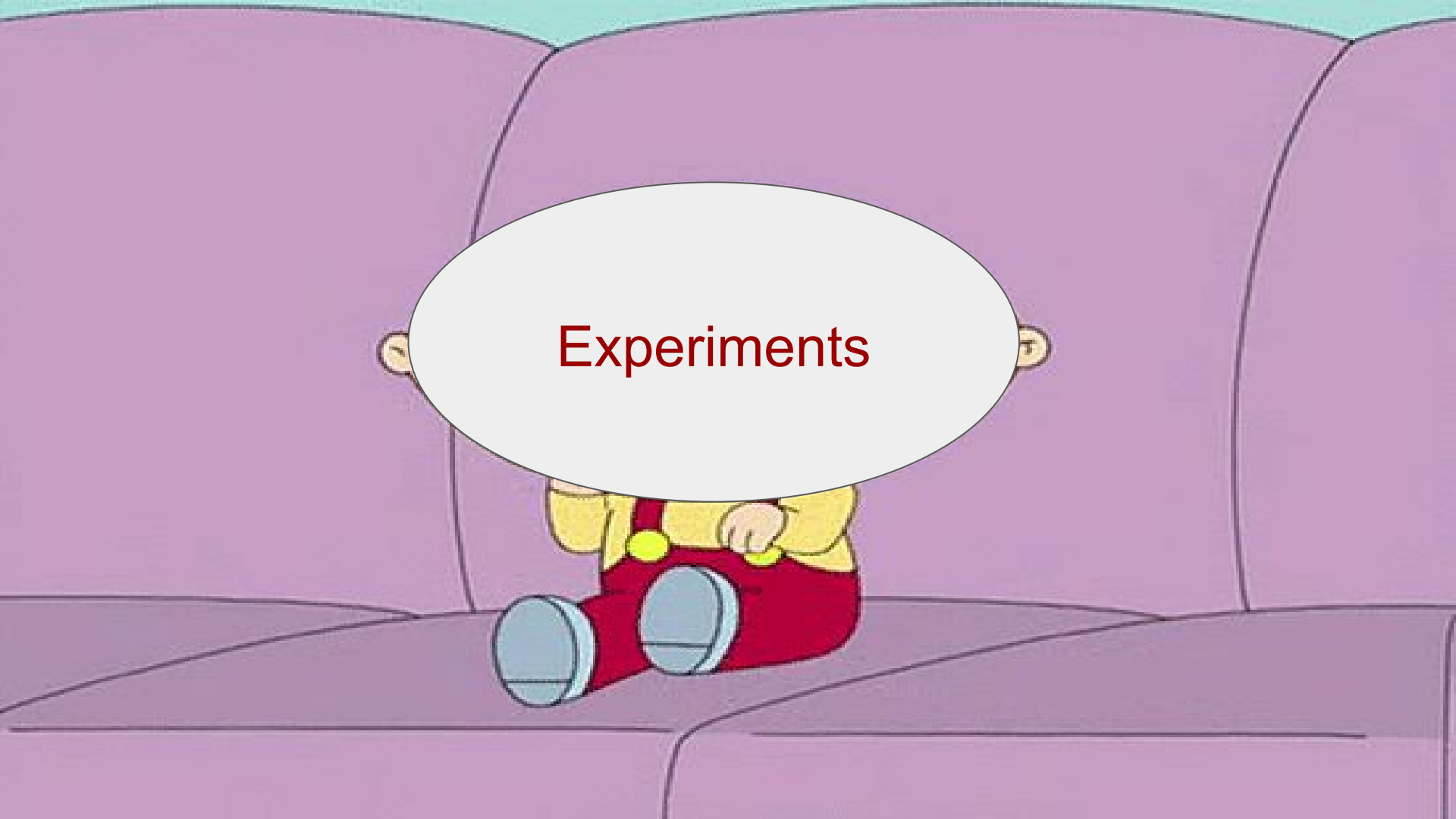
LEFT - сдвинуть HEAD влево

RIGHT - сдвинуть HEAD вправо

INSERT - вставить новый элемент на место HEAD

List RNN: Формулы

$$L_t[i] = \begin{cases} a_t[\texttt{RIGHT}]L_{t-1}[i+1] + a_t[\texttt{LEFT}]L_{t-1}[i-1] + a_t[\texttt{INSERT}]\sigma(Dh_t) & \text{if } i = \texttt{HEAD}, \\ a_t[\texttt{RIGHT}]L_{t-1}[i+1] + a_t[\texttt{LEFT}]L_{t-1}[i-1] + a_t[\texttt{INSERT}]L_{t-1}[i+1] & \text{if } i < \texttt{HEAD}, \\ a_t[\texttt{RIGHT}]L_{t-1}[i+1] + a_t[\texttt{LEFT}]L_{t-1}[i-1] + a_t[\texttt{INSERT}]L_{t-1}[i] & \text{if } i > \texttt{HEAD}. \end{cases}$$


Experiments

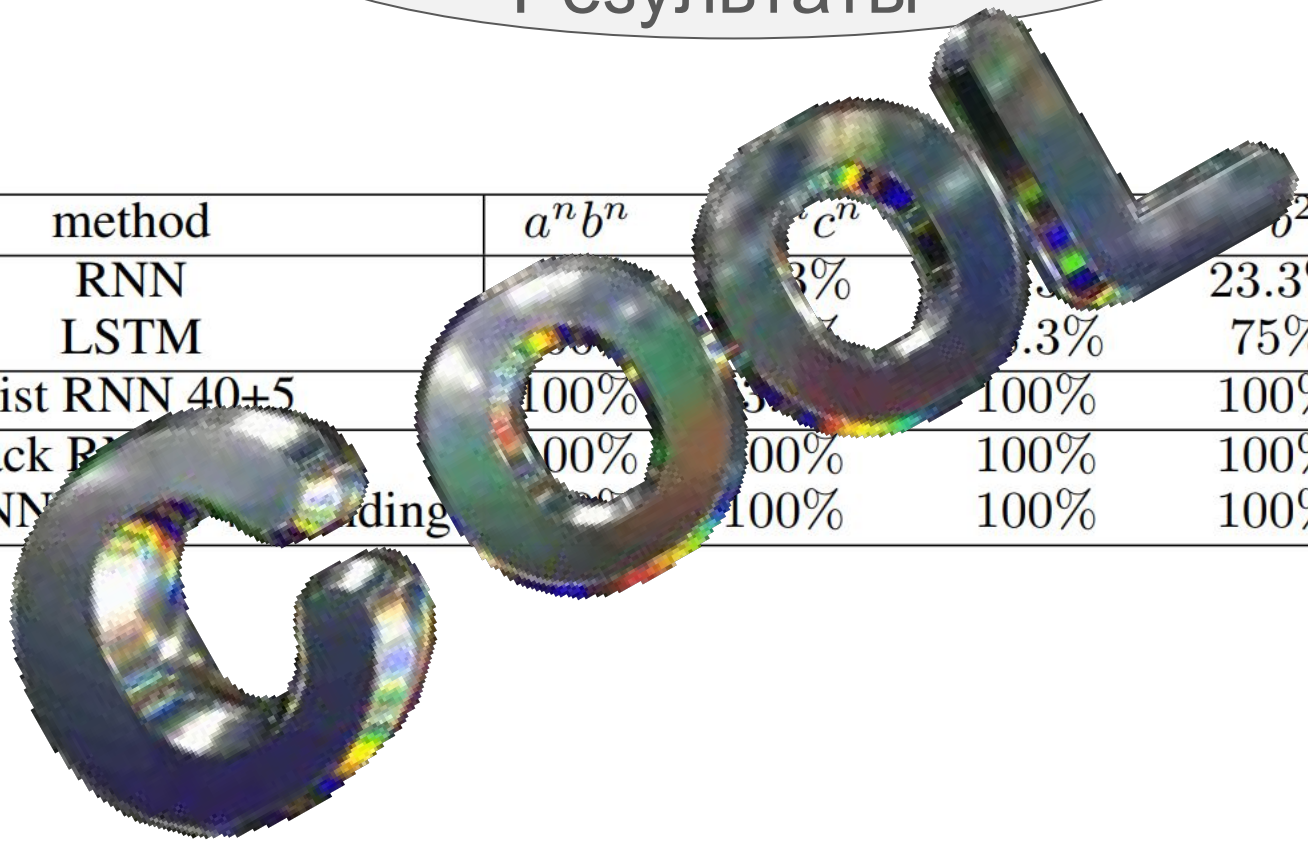
Sequence generators: Пример

current	next	prediction	proba(next)	action		stack1[top]	stack2[to
b	a	a	0.99	POP	POP	-1	0.53
a	a	a	0.99	PUSH	POP	0.01	0.97
a	a	a	0.95	PUSH	PUSH	0.18	0.99
a	a	a	0.93	PUSH	PUSH	0.32	0.98
a	a	a	0.91	PUSH	PUSH	0.40	0.97
a	a	a	0.90	PUSH	PUSH	0.46	0.97
a	b	a	0.10	PUSH	PUSH	0.52	0.97
b	**b**	**b**	0.99	PUSH	PUSH	0.57	0.97
b	**b**	**b**	1.00	POP	PUSH	0.52	0.56
b	**b**	**b**	1.00	POP	PUSH	0.46	0.01
b	**b**	**b**	1.00	POP	PUSH	0.40	0.00
b	**b**	**b**	1.00	POP	PUSH	0.32	0.00
b	**b**	**b**	1.00	POP	PUSH	0.18	0.00
b	**b**	**b**	0.99	POP	PUSH	0.01	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.00
b	**b**	**b**	0.99	POP	POP	-1	0.01
b	**a**	**a**	0.99	POP	POP	-1	0.56

Sequence generators:
Результаты

method	$a^n b^n$	$a^n b^n c^n$	$a^n b^n c^n d^n$	$a^n b^{2n}$	$a^n b^m c^{n+m}$
RNN	25%	23.3%	13.3%	23.3%	33.3%
LSTM	100%	100%	68.3%	75%	100%
List RNN 40+5	100%	33.3%	100%	100%	100%
Stack RNN 40+10	100%	100%	100%	100%	43.3%
Stack RNN 40+10 + rounding	100%	100%	100%	100%	100%

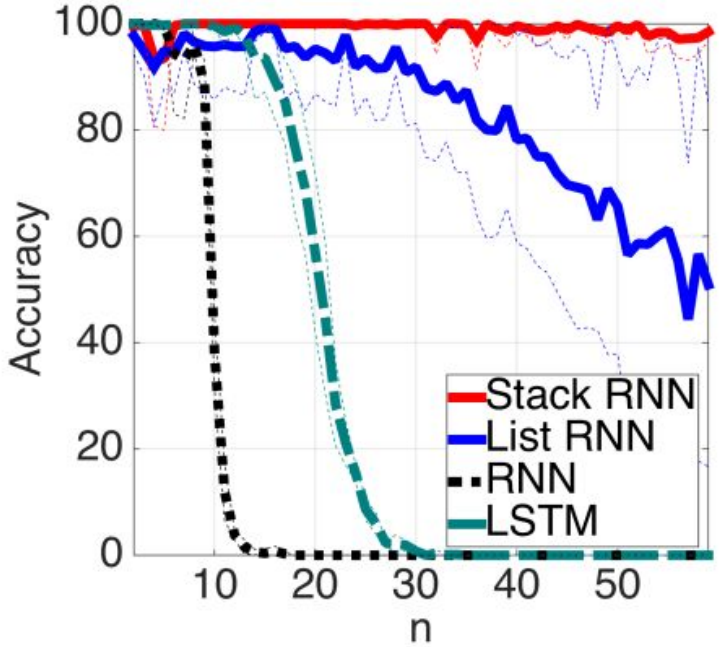
Sequence generators:
Результаты



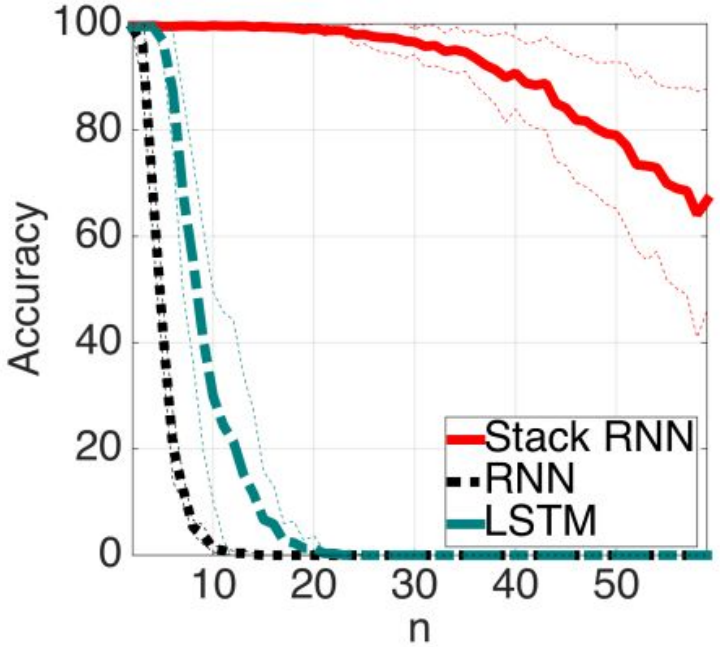
method	$a^n b^n$	$a^n b^n c^n$	b^{2n}	$a^n b^m c^{n+m}$
RNN		3%	23.3%	33.3%
LSTM		.3%	75%	100%
List RNN 40+5	100%	100%	100%	100%
Stack R	00%	00%	100%	43.3%
Stack RNN	ding	100%	100%	100%

Больше Графиков

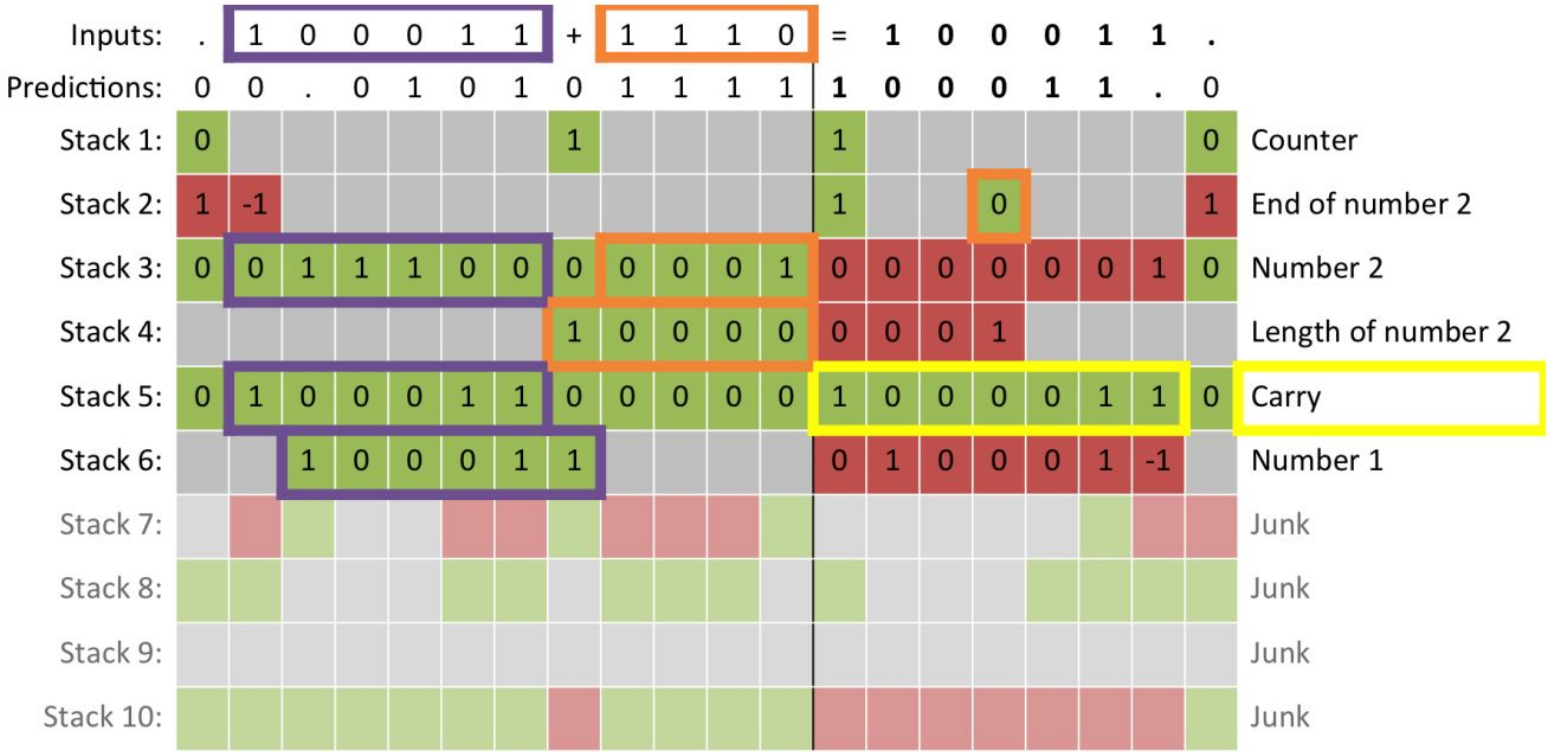
Memorization

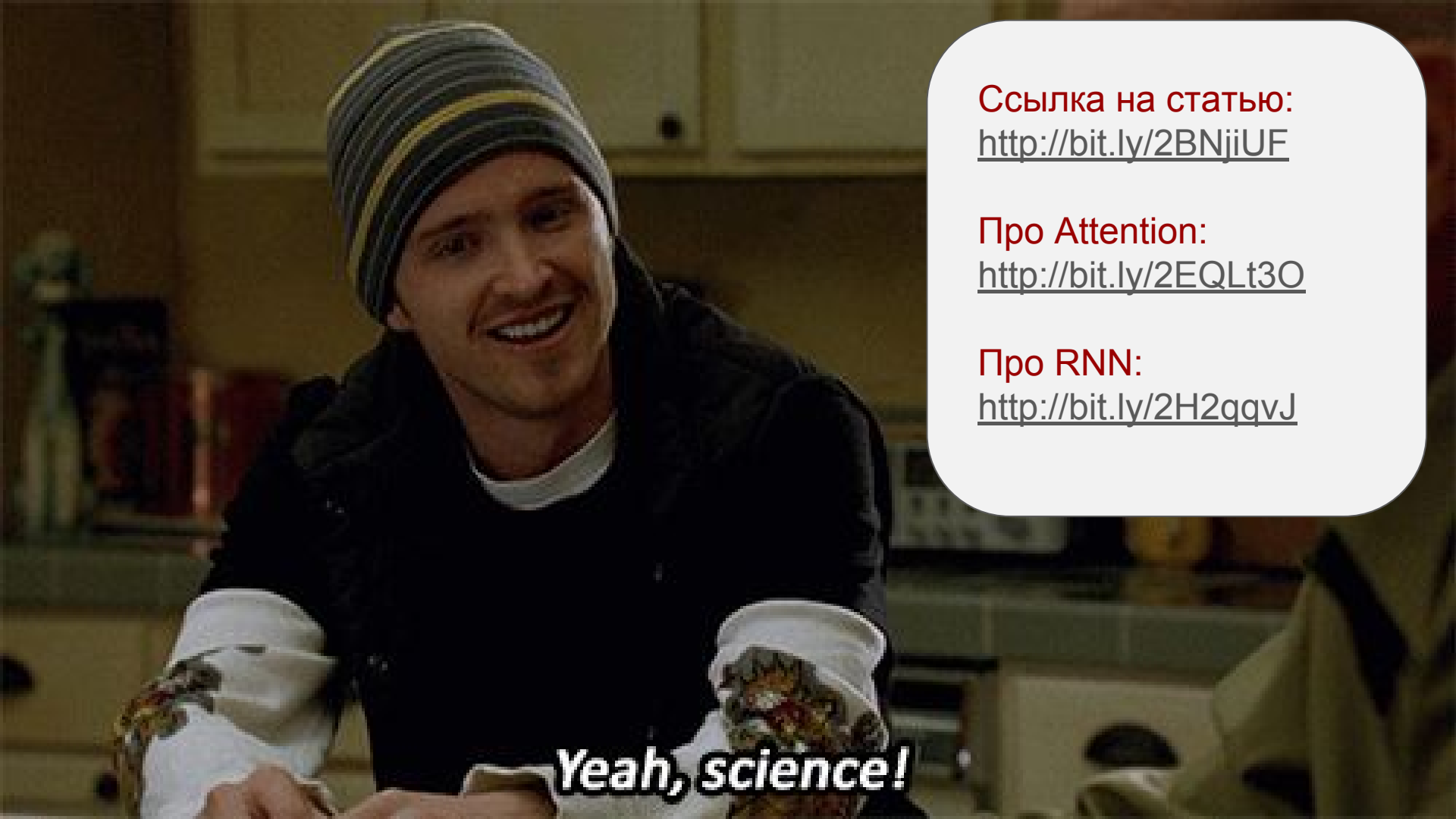


Binary addition



Binary addition:
Пример





Ссылка на статью:
http://bit.ly/2BNjiUF

Про Attention:
http://bit.ly/2EQLt3O

Про RNN:
http://bit.ly/2H2qqvJ