

# Рекуррентные нейронные сети и статья Visualizing and Understanding Recurrent Networks

Каюмов Эмиль

ММП ВМК МГУ

Спецсеминар

«Алгебра над алгоритмами и эвристический поиск закономерностей»

27.10.2016

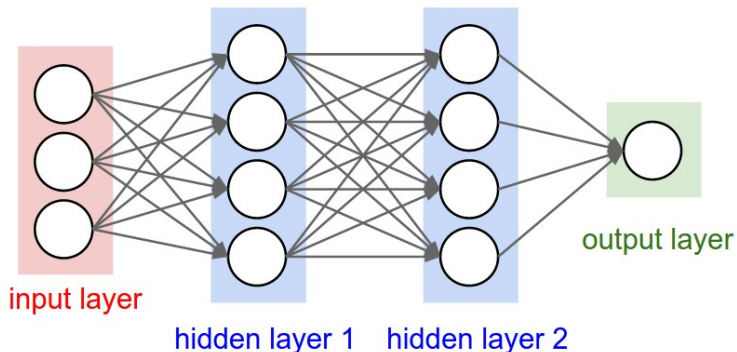
# План

- 1 Обзор рекуррентных нейронных сетей
- 2 Статья [Visualizing and Understanding Recurrent Networks](#)

# Содержание

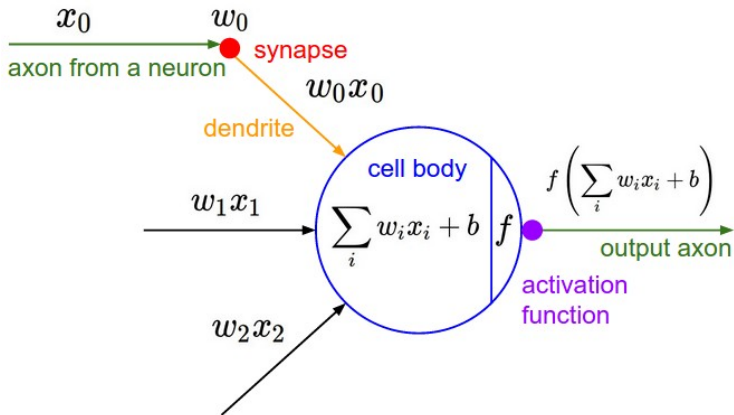
- 1 **Обзор рекуррентных нейронных сетей**
- 2 **Статья Visualizing and Understanding Recurrent Networks**

# Сети прямого распространения



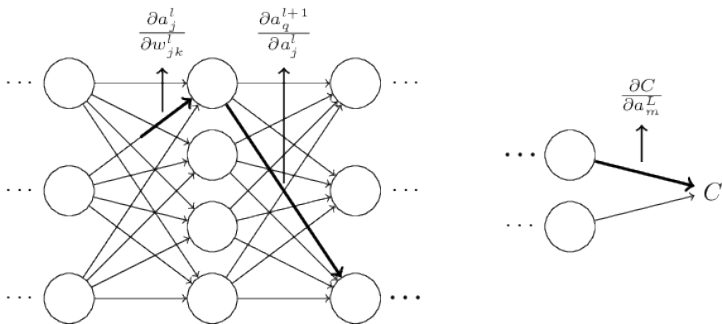
- Все связи направлены строго от входных нейронов к выходным.

# Модель нейрона (с биологической аналогией)



- Каждый нейрон – всего лишь нелинейное преобразование над взвешенной суммой своих входов со смещением.

## Обучение нейронной сети



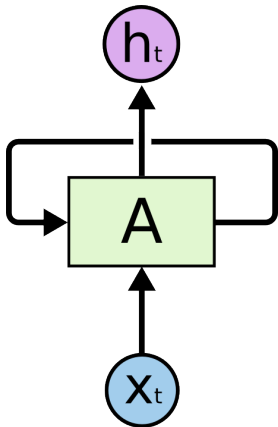
- Для обучения необходимо решать оптимизационную задачу (для «нормального» функционала можно взять производную по любому из весов сети).

# Мотивация

Сети прямого распространения (включая свёрточные сети) имеют некоторые ограничения:

- Фиксированный размер входных данных.
- Фиксированный размер выходных данных.
- Зависимость только от текущего входа и весов.
- Замороженные веса сети.

# Рекуррентные нейронные сети (RNN)



Появляется связь между нейронами одного уровня сети.

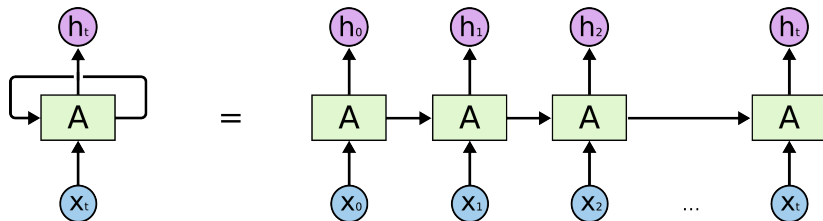
- Получаем аналог памяти (внутреннее состояние ячейки).
- Теперь выход сети зависит не только от текущего входа и весов, но и от всего увиденного раньше.

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

$$y_t = f(W_{hy}h_t)$$



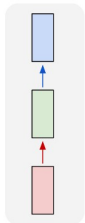
## Разворачивание RNN



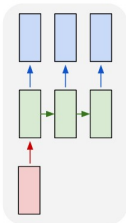
- То есть на самом деле можно обучать сеть аналогично сети прямого распространения.

# Виды RNN

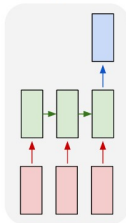
one to one



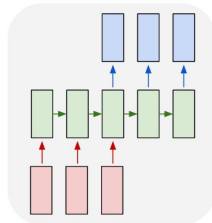
one to many



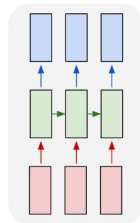
many to one



many to many



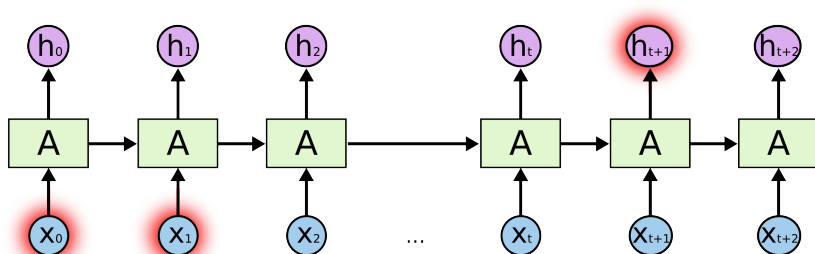
many to many



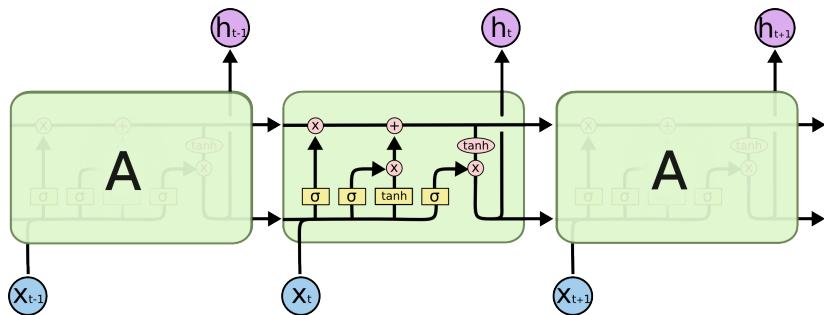
- Можно задавать длину входа/выхода (оперируем последовательностями, а не конкретными объектами).

# Проблемы RNN

Кроме exploding/vanishing gradients при обучении есть проблема с долговременной памятью:

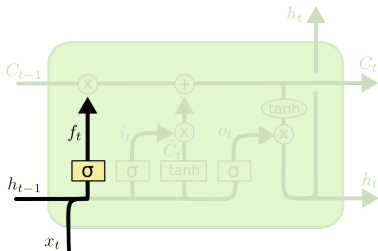


## LSTM



- Long Short Term Memory networks разработана, чтобы избежать проблем с долговременной памятью.

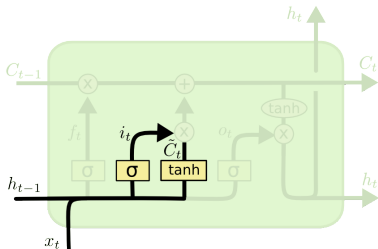
# LSTM: forget memory



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- На основе нового входа и внутреннего состояния решаем, сколько информации из памяти нужно забыть.

## LSTM: new memory

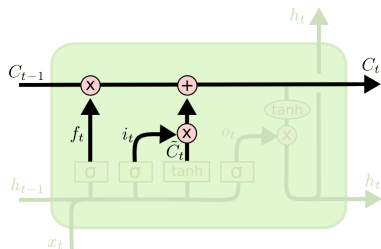


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

- На основе нового входа и внутреннего состояния решаем, какую информацию и сколько её нужно запомнить в памяти.

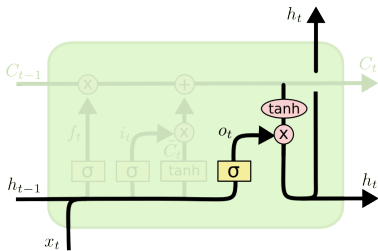
## LSTM: update memory



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Обновляем память по forget и input gate.

## LSTM: output



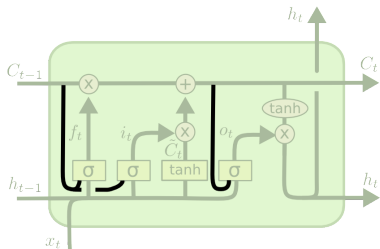
$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

- На основе входа, внутреннего состояния и новой памяти генерируем выход и новое внутреннее состояние.



## LSTM: варианты



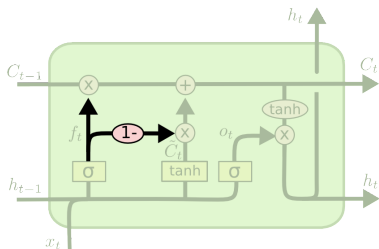
$$f_t = \sigma(W_f \cdot [C_{t-1}, h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \cdot [C_{t-1}, h_{t-1}, x_t] + b_i)$$

$$o_t = \sigma(W_o \cdot [C_t, h_{t-1}, x_t] + b_o)$$

- Используем для всех вычислений ещё и память.

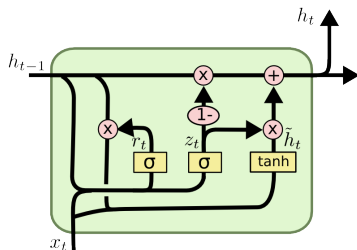
## LSTM: варианты



$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

- Соответствие между forget и input gate.

## GRU (Gated Recurrent Units)



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- Получается более простая модель.

# Применение RNN

- Машинный перевод.
- Генерация текстов.
- Описание изображений.
- Распознавание речи.
- Разметка видео, предложений и прочего.

# Содержание

- 1 Обзор рекуррентных нейронных сетей
- 2 **Статья Visualizing and Understanding Recurrent Networks**

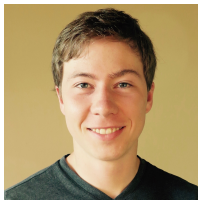
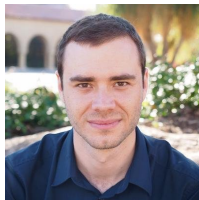
# Visualizing and Understanding Recurrent Networks

## VISUALIZING AND UNDERSTANDING RECURRENT NETWORKS

Andrej Karpathy\*    Justin Johnson\*    Li Fei-Fei  
Department of Computer Science, Stanford University  
{karpathy, jjohns, feifeili}@cs.stanford.edu

### ABSTRACT

Recurrent Neural Networks (RNNs), and specifically a variant with Long Short-Term Memory (LSTM), are enjoying renewed interest as a result of successful applications in a wide range of machine learning problems that involve sequential data. However, while LSTMs provide exceptional results in practice, the source of their performance and their limitations remain rather poorly understood. Using character-level language models as an interpretable testbed, we aim to bridge this gap by providing an analysis of their representations, predictions and error types. In particular, our experiments reveal the existence of interpretable cells that keep track of long-range dependencies such as line lengths, quotes and brackets. Moreover, our comparative analysis with finite horizon  $n$ -gram models traces the source of the LSTM improvements to long-range structural dependencies. Finally, we provide analysis of the remaining errors and suggests areas for further study.



# Мотивация

- RNN (в частности LSTM) показывают хорошее качество в различных задачах.
- Непонятно, что у них внутри.
- Непонятно, какие у этих сетей ограничения.

Поэтому:

- Возьмём charRNN (RNN, LSTM, GRU) и исследуем.

## Что использовали?

### Сети:

- Vanilla RNN
- LSTM
- GRU

### Модель:

- Подавая на вход один символ, предсказываем вероятность следующего символа.
- Используем one-hot кодирование для входящих символов.
- Разворачиваем сеть на 50 шагов для обучения.

### Датасеты:

- Война и мир (3 258 246 символов)
- Код ядра Linux (6 206 996 символов)



## Сравнение (1)

Layers	LSTM			RNN			GRU		
	1	2	3	1	2	3	1	2	3
Size	War and Peace Dataset								
64	1.449	1.442	1.540	1.446	1.401	1.396	1.398	<b>1.373</b>	1.472
128	1.277	1.227	1.279	1.417	1.286	1.277	1.230	<b>1.226</b>	1.253
256	1.189	<b>1.137</b>	1.141	1.342	1.256	1.239	1.198	1.164	1.138
512	1.161	1.092	1.082	-	-	-	1.170	1.201	<b>1.077</b>
	Linux Kernel Dataset								
64	1.355	<b>1.331</b>	1.366	1.407	1.371	1.383	1.335	1.298	1.357
128	1.149	1.128	1.177	1.241	<b>1.120</b>	1.220	1.154	1.125	1.150
256	1.026	<b>0.972</b>	0.998	1.171	1.116	1.116	1.039	0.991	1.026
512	0.952	0.840	0.846	-	-	-	0.943	0.861	<b>0.829</b>

Figure: Cross-entropy loss

## Сравнение (2)

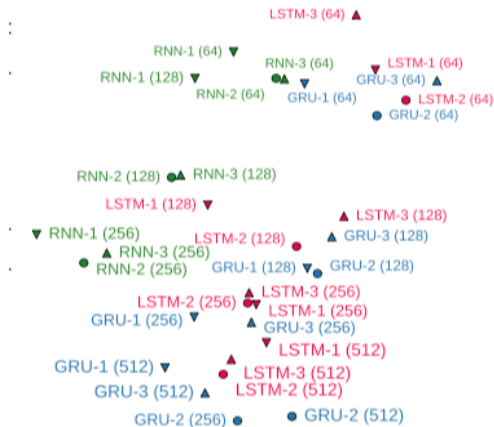


Figure: t-SNE предсказаний на тестовой выборке

# Состояние памяти LSTM (1)

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."

## Состояние памяти LSTM (2)

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                     struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                   (void **)&df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM '%s\' is invalid\n",
                df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

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;
}
```

# Состояние памяти LSTM (3)

Cell that robustly activates inside if statements:

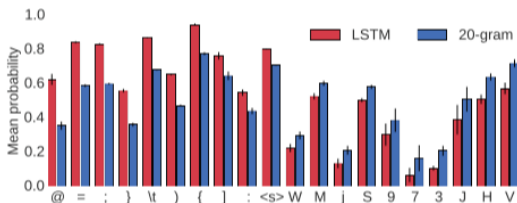
```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
                           siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

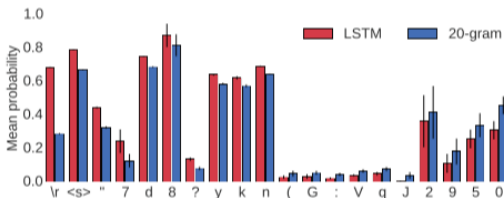
```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
     */
}
```

# Гипотеза долгой памяти (1)

- Linux Kernel

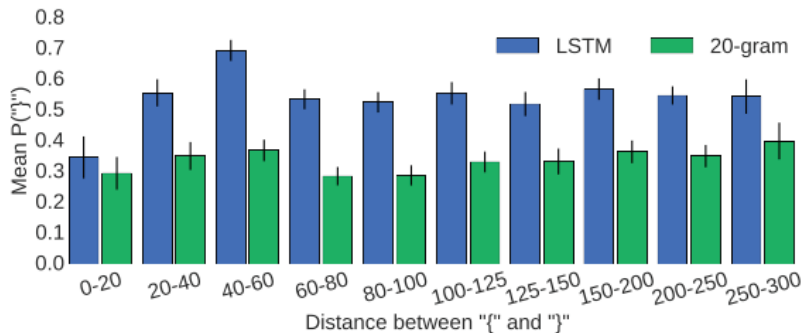


- War and Peace



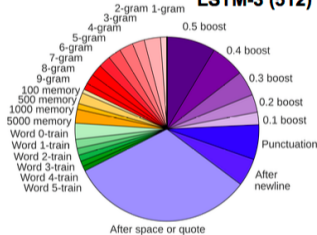
(20-gram – n-gram модель с Kneser-Neu сглаживанием)

## Гипотеза долгой памяти (2)

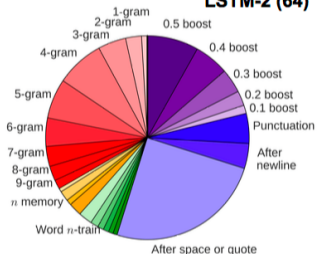


## Ошибки

LSTM-3 (512)








LSTM-2 (64)



- n-gram предсказание
- Динамическая память длины n
- Редкие слова
- Начало слов
- Пунктуационные ошибки
- Оставшиеся ошибки



## Ссылки

-  [Visualizing and Understanding Recurrent Networks \(article\)](#)
-  [The Unreasonable Effectiveness of Recurrent Neural Networks \(Andrej Karpathy blog\)](#)
-  [charRNN on Torch \(Andrej Karpathy github\)](#)
-  [Understanding LSTM Networks \(colah's blog\)](#)
-  [On the difficulty of training recurrent neural networks \(article\)](#)