Обработка естественного языка. Рекуррентные нейронные сети.

МФТИ, Сбертех

Ключевые операции со текстом

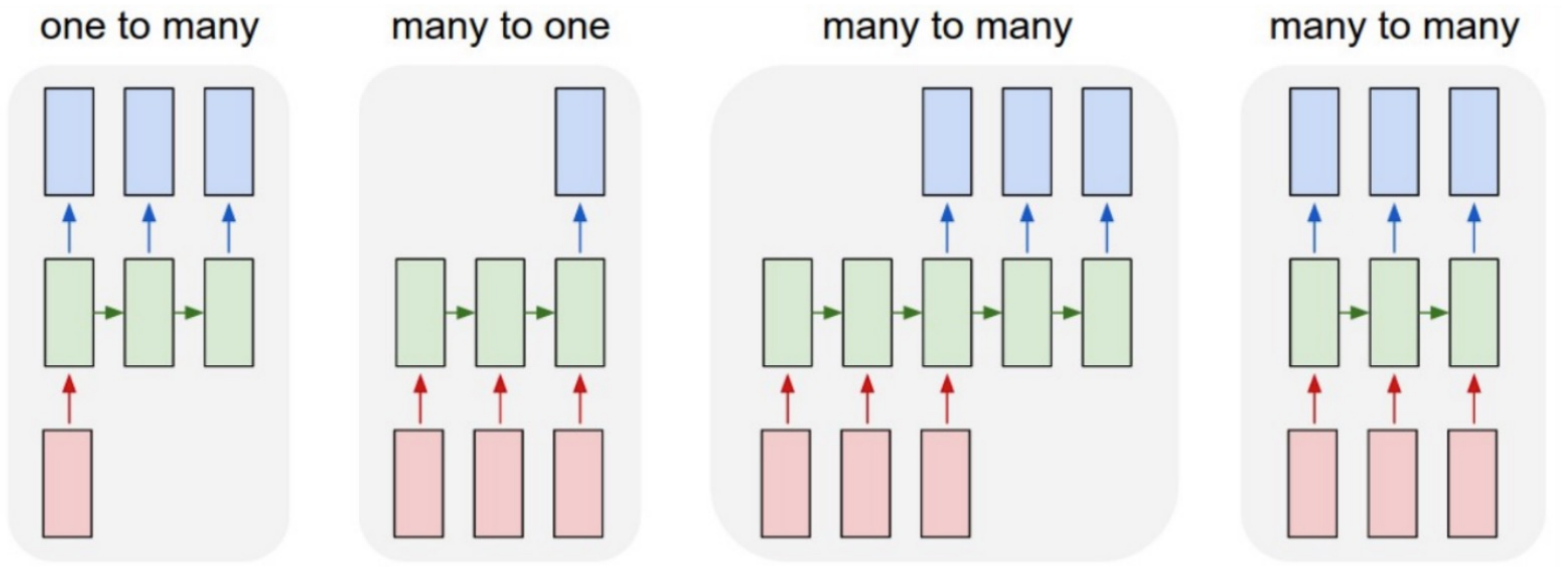
- **Препроцессинг** – набор операций для упрощения работы с текстом – удаление пунктуации, приведение к нижнему регистру, удаление лишних пробелов и тд.

- **Токенизация** – разбиение исходного текста на слова, ngram, символы. Необходимо для формирования текста, как множества лексических единиц.
  - Динозавр – дино, за, вр

- **Нормализация(лемматизация)** – убирает из исходного текста грамматическую составляющую(падежи, числа, глагольные виды и времена, род и тд), оставляя лишь смысловую составляющую.
  - Ивану Смирнову было поручено – Иван Смирнов быть поручить

- **Стемминг** – отсечение от слова окончаний и суффиксов, чтобы оставшаяся часть была одинаковой для всех грамматических форм слова. Актуально для русского и английского языка.
  - Подстрелить – подстрел
  - Animals - animal

Контекстно-зависимые задачи



Генерация текста

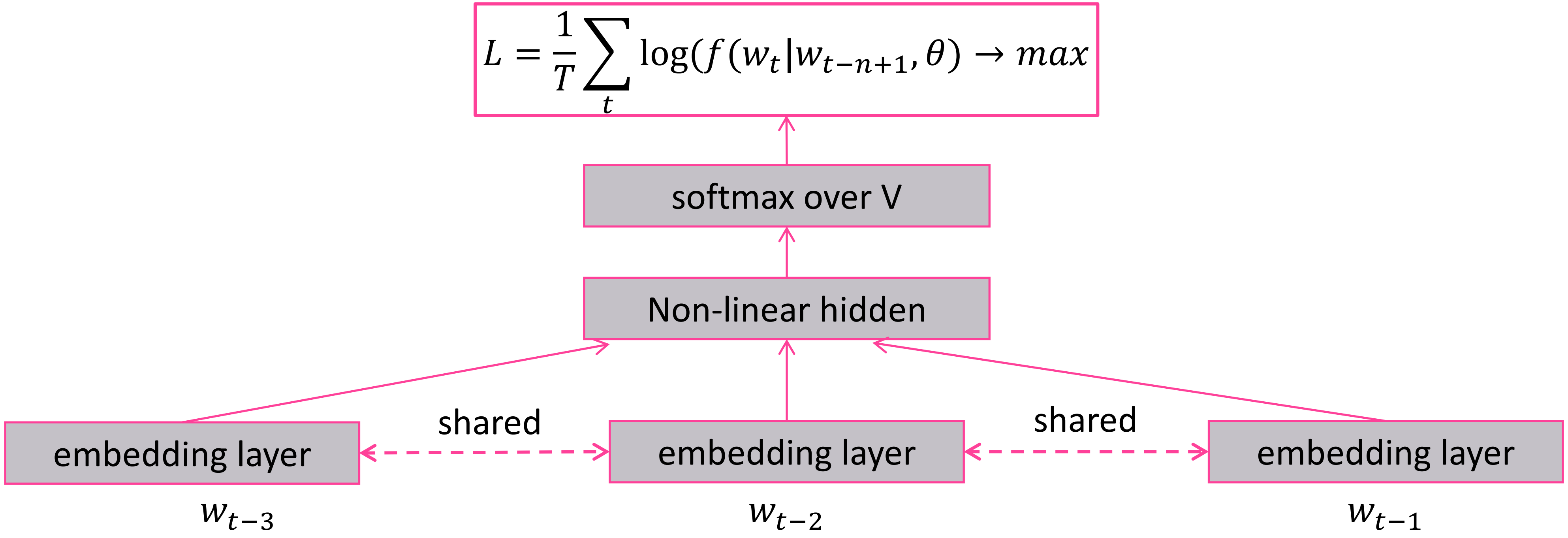
Классификация

Машинный перевод

NER

Типы входной последовательности
- Символы
- Ngrams
- Слова

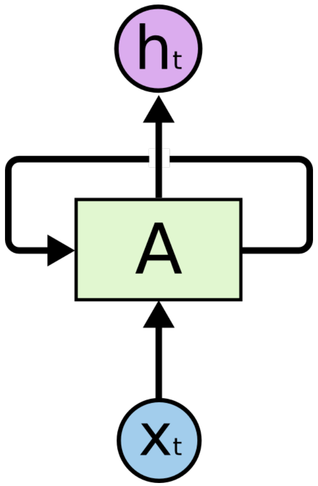
Языковое моделирование

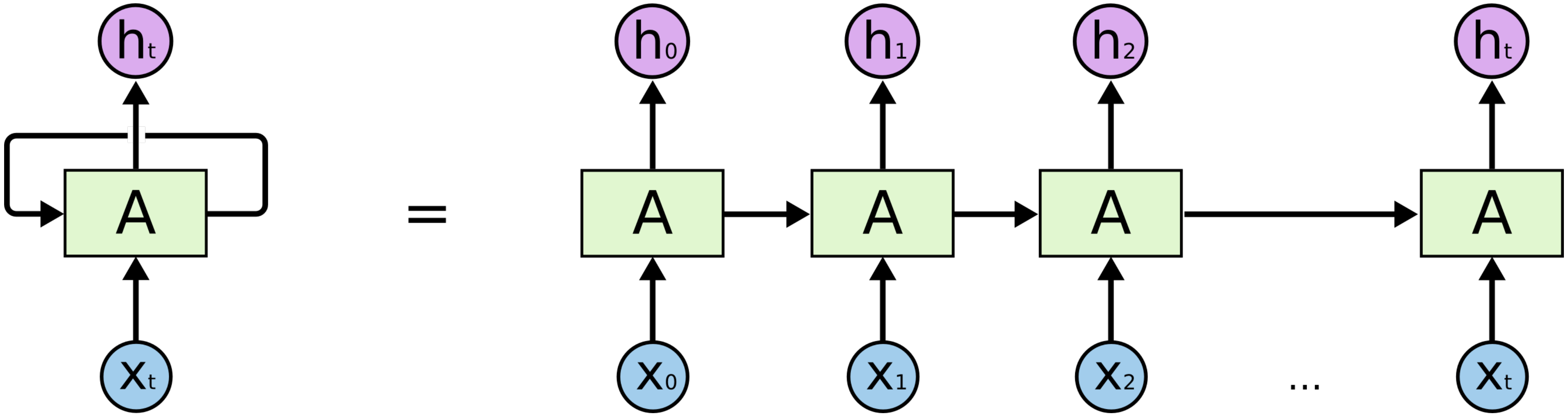


Проблемы

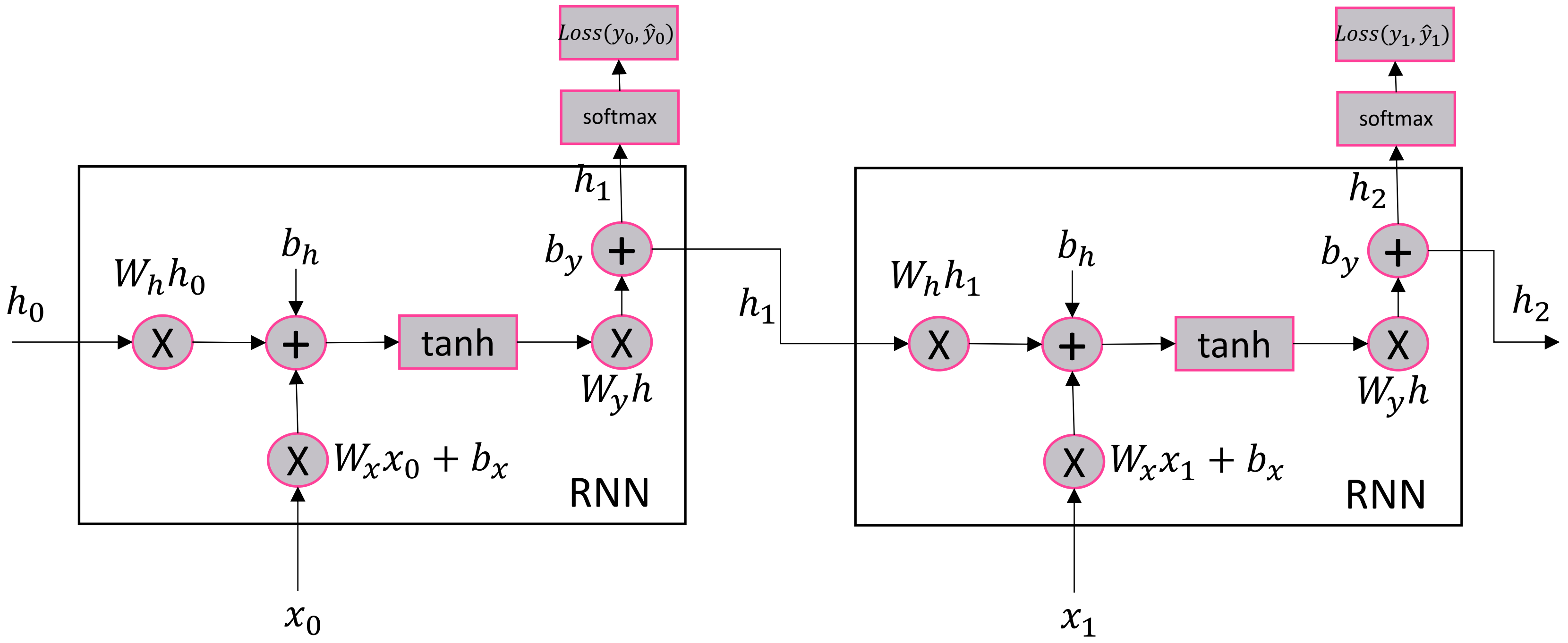
1) Фиксированный размер окна
2) Меняем только вес внутри окна – нет длительного контекста.

Vanilla Recurrent Neural Net

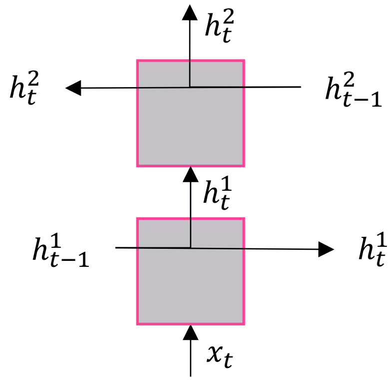




Vanilla Recurrent Neural Net

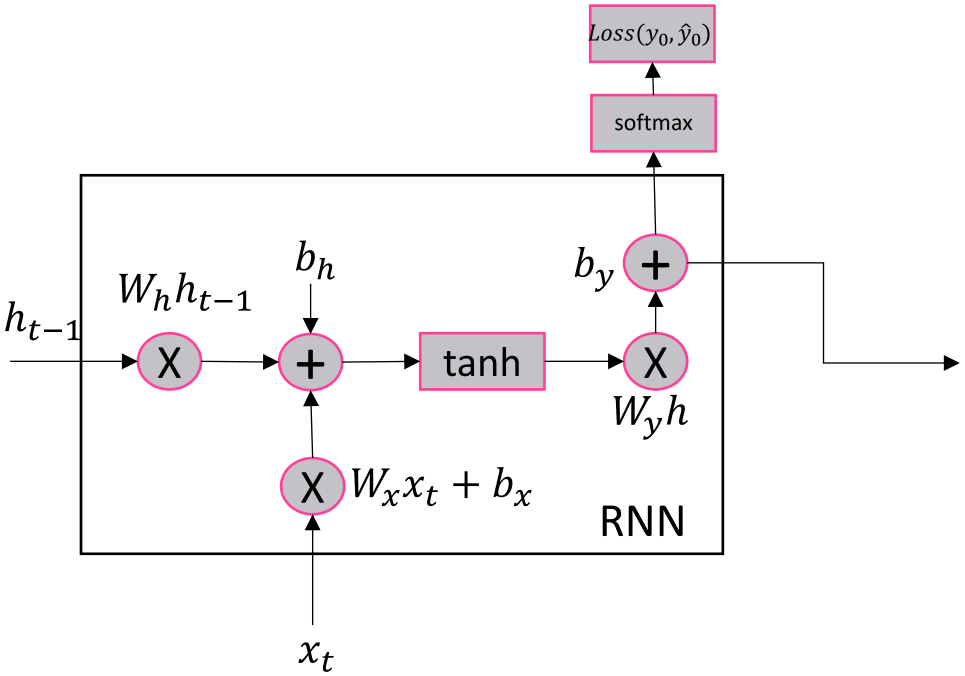


$$h_t = tanh(W_h h_{t-1} + b_h + W_x x_t + b_x)$$
$$z_t = softmax\left(W_y h_t + b_y\right) \in R^{|V|}$$
$$\mathrm{Loss}_i = \ -y_i \log \mathrm{z_t}$$
$$Sample\ Loss = \sum_{i=0} Loss_i$$
$$W_h, b_h\ , W_x, b_x, W_y, b_y\ update$$

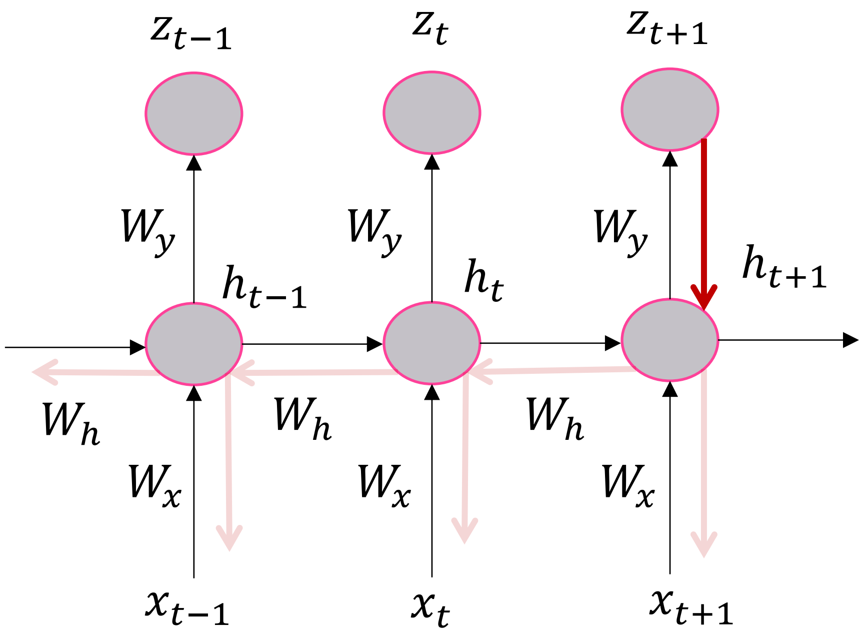


Можем построить еще одну перевернутую сеть над RNN(BiRnn) – для лучшей работы с контекстом.

Backpropagation through time



$$h_t = tanh(W_h h_{t-1} + b_h + W_x x_t + b_x)$$
$$z_t = softmax\left(W_y h_t + b_y\right) \in R^{|V|}$$
$$\mathrm{Loss}_i = \; -y_i \log \mathrm{z_t}$$

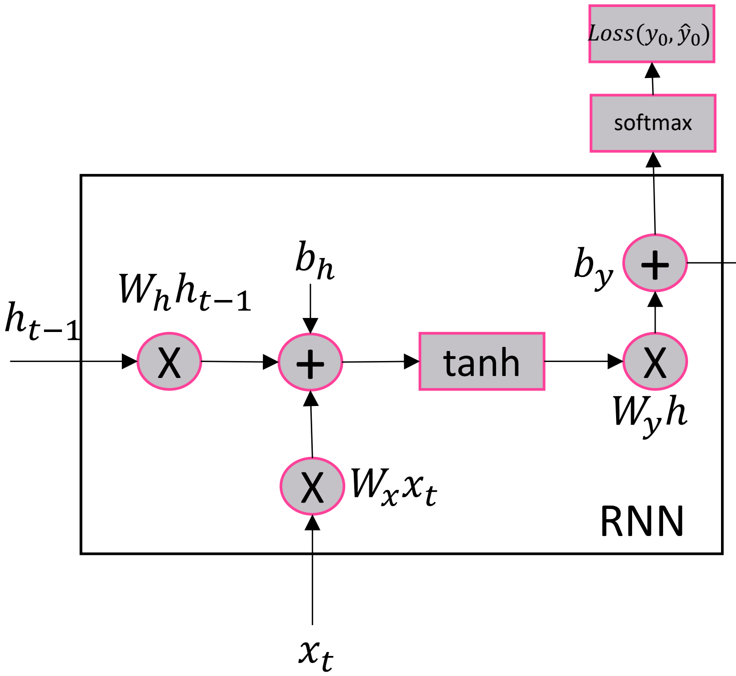


$$\alpha_t = W_y h_t + b_y$$
$$z_t = softmax(\alpha_t)$$
$$\frac{\partial L}{\partial \alpha_t} = z_t - y_t$$

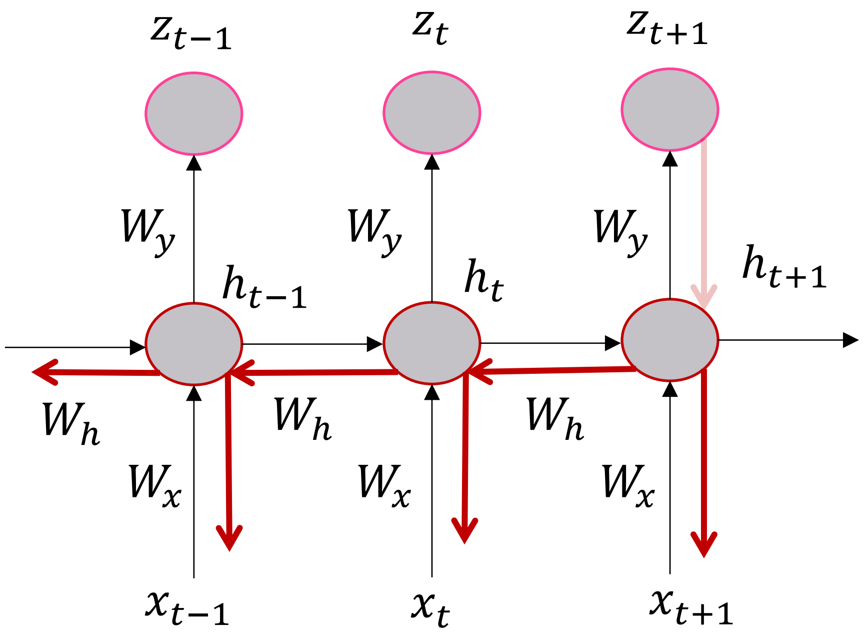
$$\frac{\partial L}{\partial W_y} = \sum_t^T \frac{\partial L}{\partial W_y} = \sum_t^T \frac{\partial L}{\partial z_t} \frac{\partial z_t}{\partial a_t} \frac{\partial a_t}{\partial W_y} = \sum_t^T (z_t - y_t) h_t$$

$$\frac{\partial L}{\partial b_y} = \sum_t^T \frac{\partial L}{\partial W_y} = \sum_t^T \frac{\partial L}{\partial z_t} \frac{\partial z_t}{\partial a_t} \frac{\partial a_t}{\partial b_y} = \sum_t^T (z_t - y_t)$$

Backpropagation through time



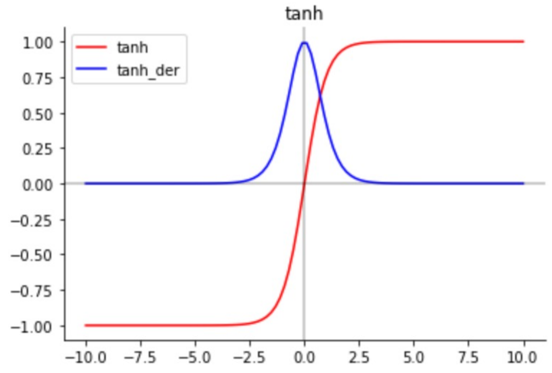
$$h_t = tanh(W_h h_{t-1} + b_h + W_x x_t + b_x)$$
$$z_t = softmax(W_y h_t + b_y) \in R^{|V|}$$
$$\mathrm{Loss}_i = \; -y_i \log z_t$$



Так как текущее состояние зависит от предыдущего и так далее ($\frac{\partial h_t}{\partial h_k} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \dots \frac{\partial h_{t-k}}{\partial h_k}$) для получения полных производных по $\frac{\partial L}{\partial W_x}$ и $\frac{\partial L}{\partial W_h}$, $\boldsymbol{\frac{\partial h_t}{\partial h_k}}$ надо включить в произведение производных.

$$\frac{\partial L}{\partial W_x} = \sum_t^T \frac{\partial L_t}{\partial W_x} = \sum_{k=1}^t \frac{\partial L_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \boldsymbol{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W_x} \qquad \frac{\partial L}{\partial W_h} = \sum_t^T \frac{\partial L_t}{\partial W_h} = \sum_{k=1}^t \frac{\partial L_t}{\partial z_t} \frac{\partial z_t}{\partial h_t} \boldsymbol{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W_h}$$

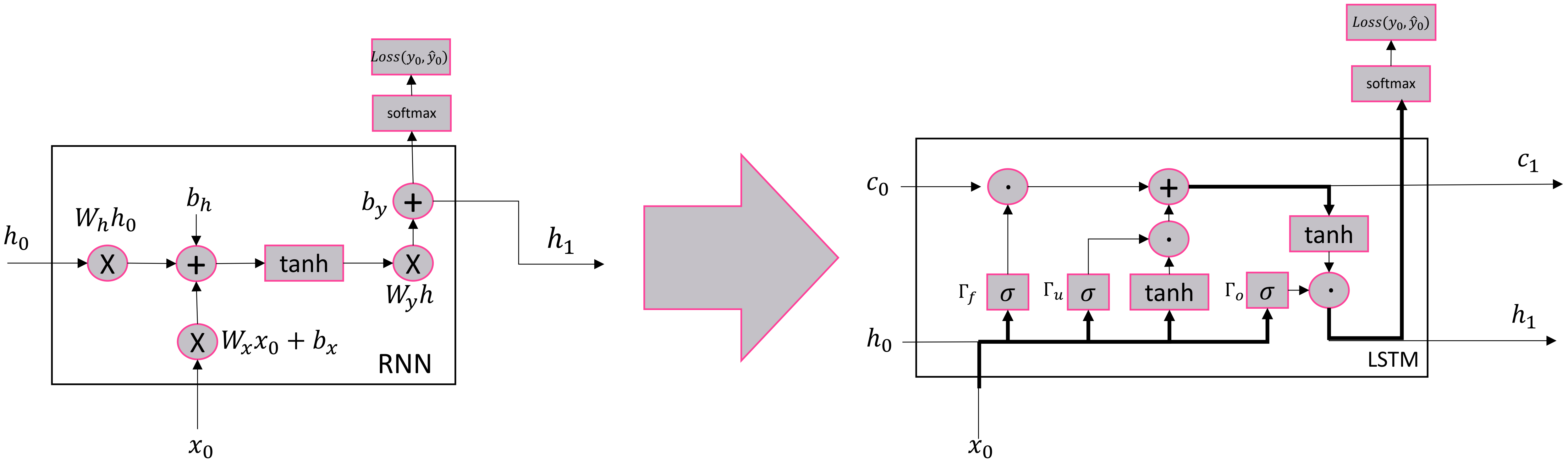
$\boldsymbol{\frac{\partial h_t}{\partial h_k}}$ - якобиан. Перемножение большого количества якобианов ведет к взрыву($\lambda_1 > 1$) или затуханию градиента($\lambda_1 < 1$).



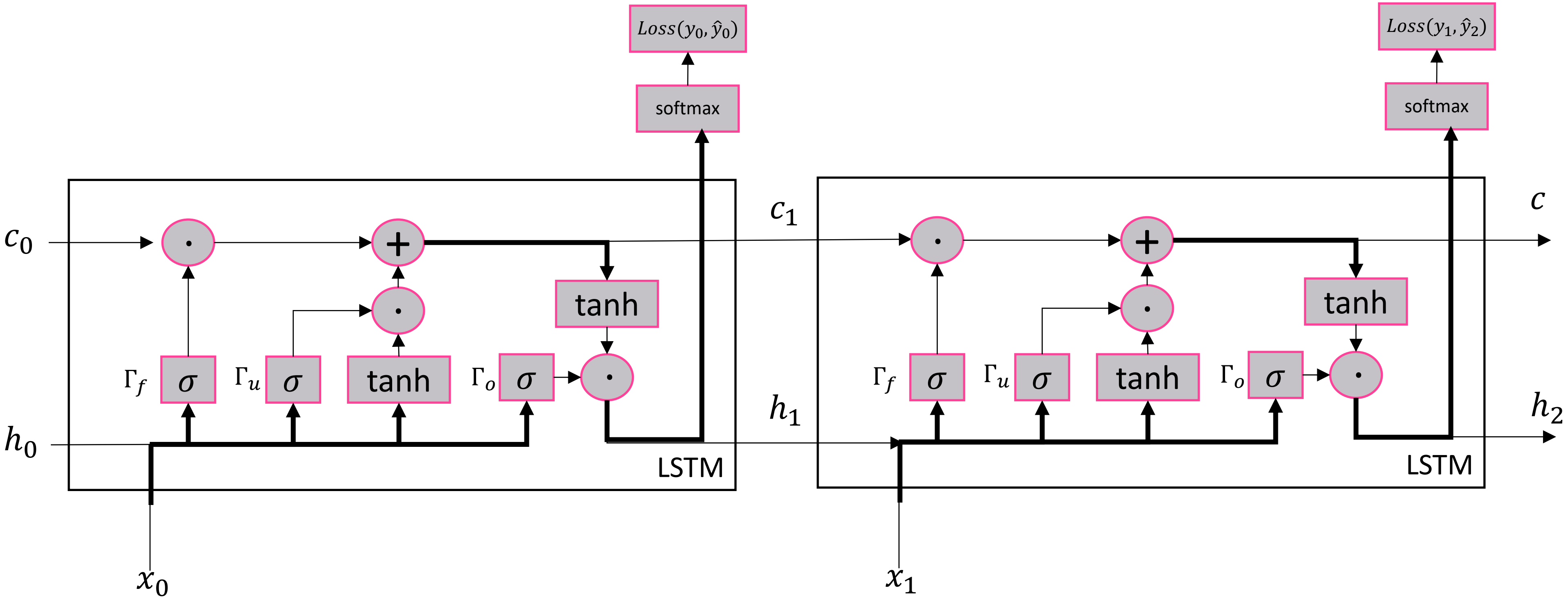
Но tanh больше способствует именно затуханию

[Изучение сложностей обучения RNN(Pascanu, Mikolov, Bengio)]

Long-Short Term memory Cell(LSTM)



Концепции фильтров в LSTM



Γ_f - forget gate

Состоит из W_f, b_f. Определяет какую информацию стоит выбросить из c_t для обновления c_t. Поэлементное применение сигмоиды возвращает каждый элемент вектора h в диапазоне от 0 до 1.

Γ_u - update gate

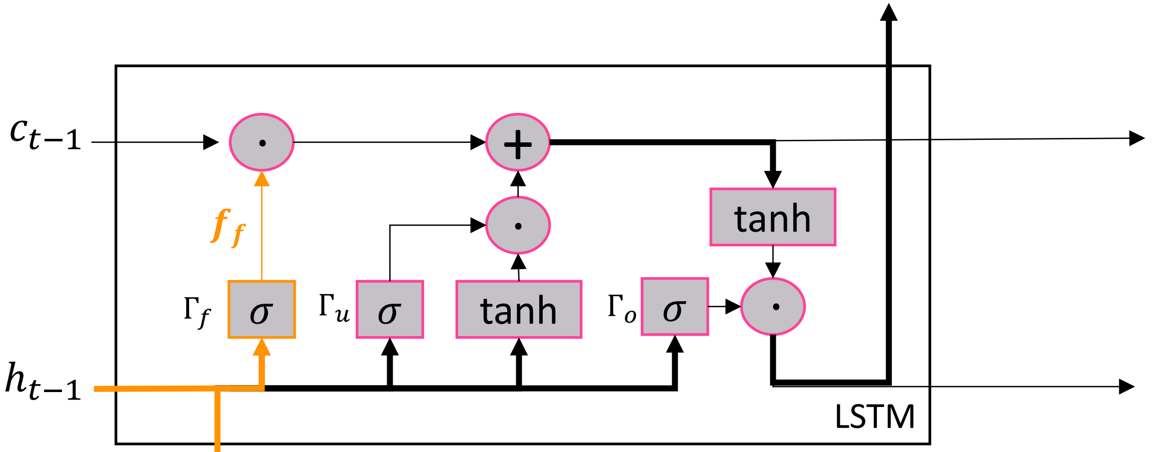
Состоит из W_u, b_u. Определяет, какую информацию добавляем в c_t из преобразованного через $\tanh$ текущих h_t, x_t.

Γ_o - output gate

Состоит из W_o, b_0. Определяет сколько информации мы передаем в h_{t+1}. Объединяет x_t, h_t с преобразованным C_t.

Архитектура LSTM

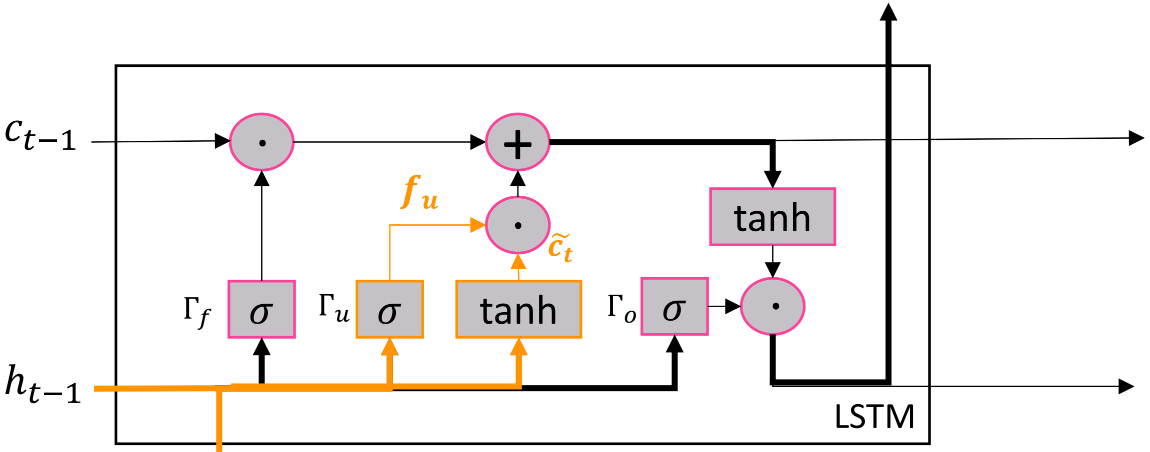
1.



x_t

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

2.

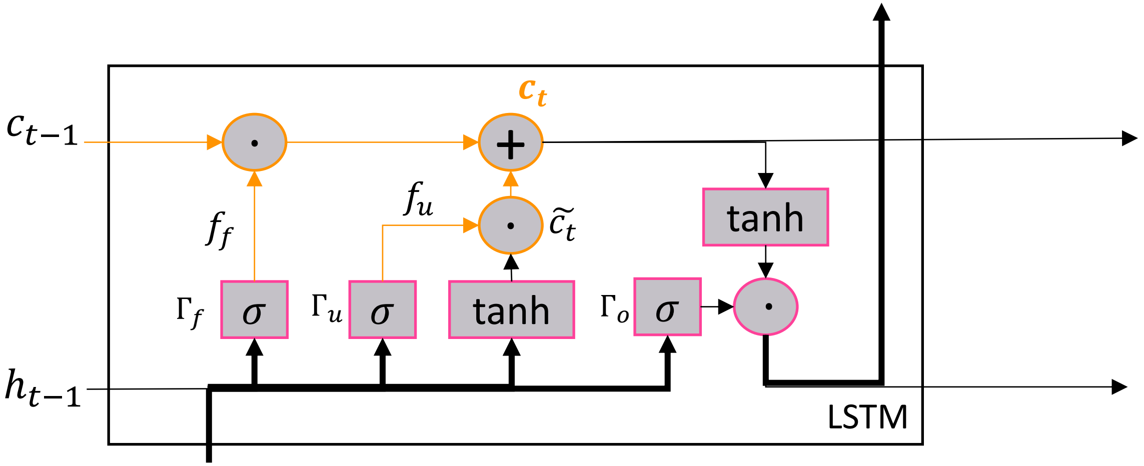


x_0

$$\boldsymbol{f_u = \sigma(W_u[h_{t-1}; x_t] + b_u)}$$

$$\boldsymbol{\tilde{c}_t = \tanh(W_c[h_{t-1}; x_t] + b_c)}$$

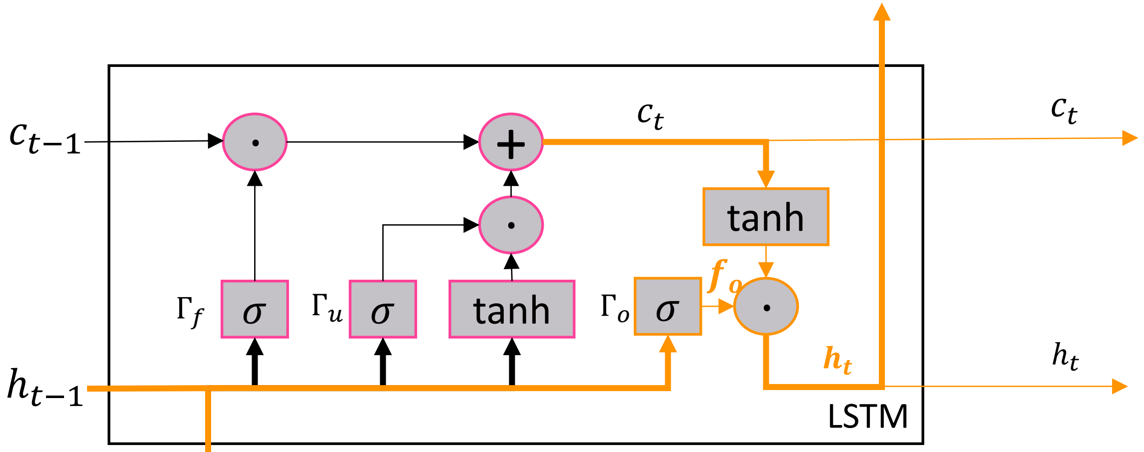
3.



x_0

$$\boldsymbol{c_t = f_f \cdot c_{t-1} + f_u \cdot \tilde{c}_t}$$

4.

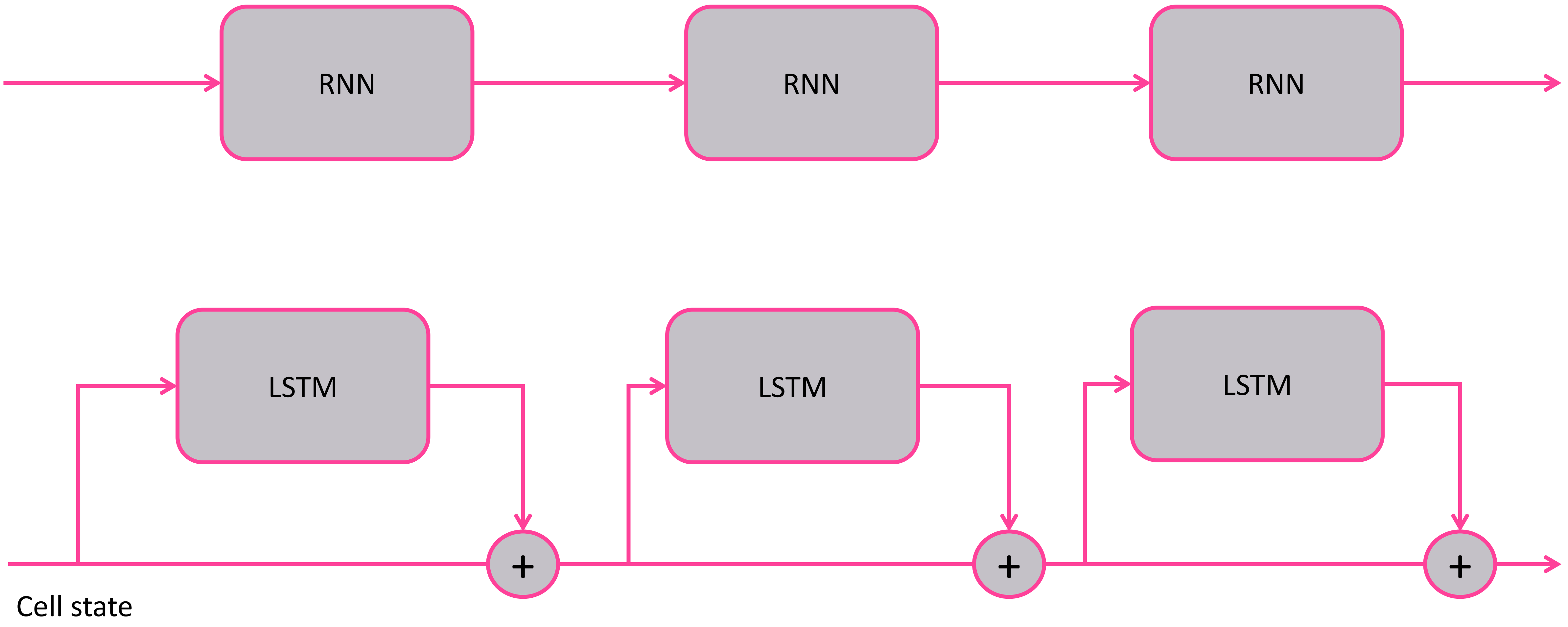


x_0

$$\boldsymbol{f_o = \sigma(W_o[h_t; x_t] + b_o)}$$

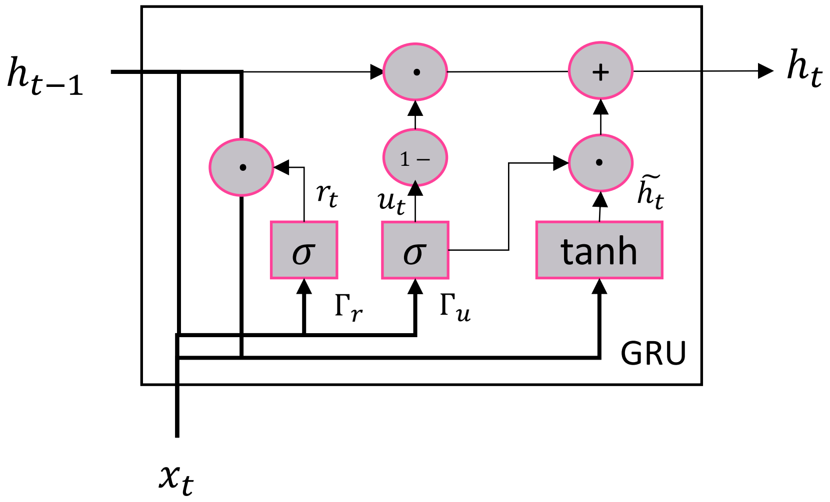
$$\boldsymbol{h_t = f_o \tanh(c_t)}$$

Преимущества LSTM



Без учета forget gate

Фильтры в GRU



$\boldsymbol{\Gamma_r}$ − **Reset gate**

Состоит из W_r, b_r. Определяет, сколько информации надо забыть с h_{t-1}.

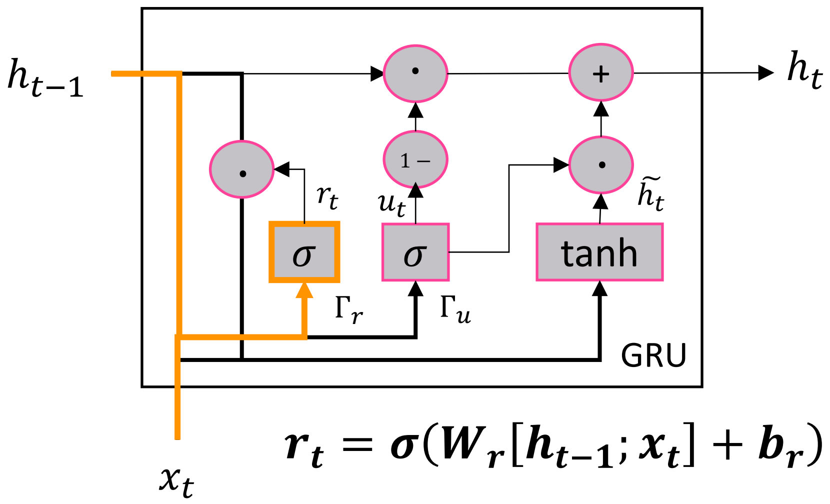
$\boldsymbol{\Gamma_u}$ − **Update gate**

Состоит из W_u, b_u. Определяет сколько информации с h_{t-1} надо передать в h_t.

https://arxiv.org/pdf/1406.1078v3.pdf

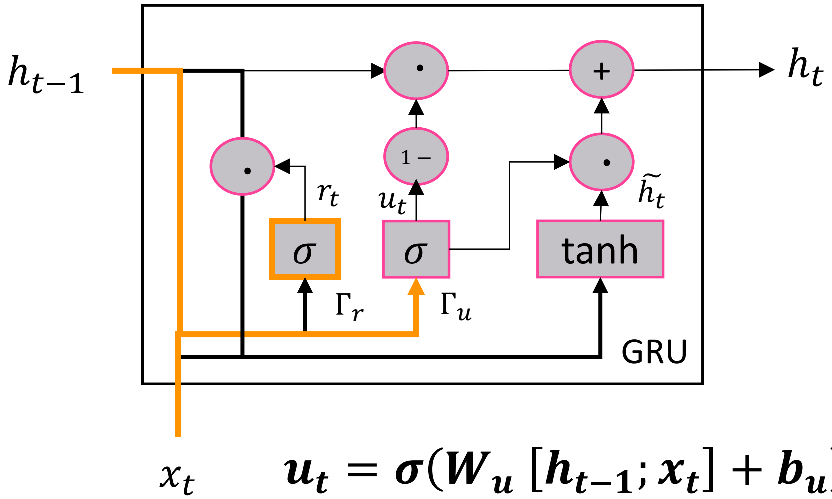
Архитектура GRU

1.



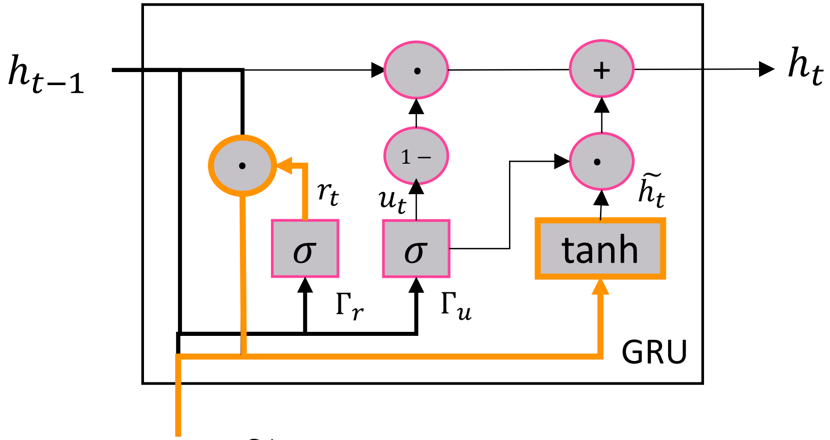
$$\boldsymbol{r_t = \sigma(W_r[h_{t-1}; x_t] + b_r)}$$

2.



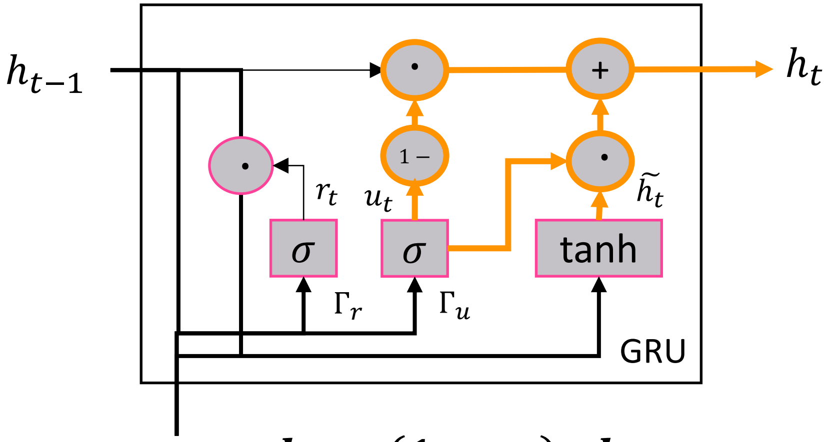
$$\boldsymbol{u_t = \sigma(W_u\,[h_{t-1}; x_t] + b_u)}$$

3.



$$\boldsymbol{\tilde{h}_t = tanh(W[r_t \cdot h_{t-1}; x_t] + b)}$$

4.

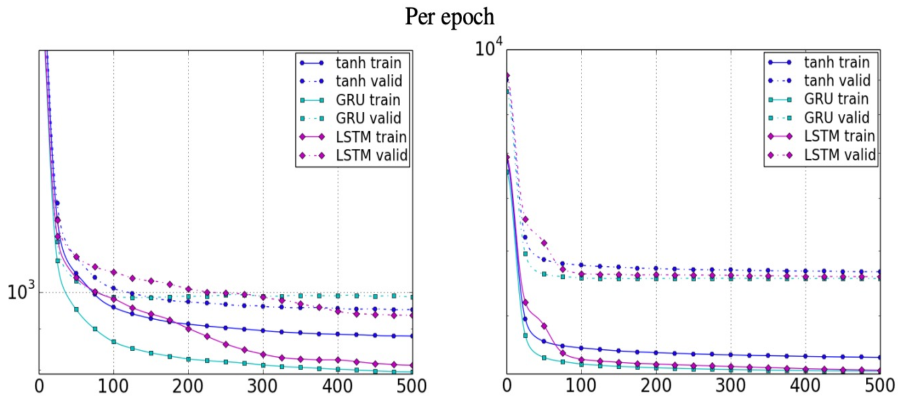


$$\boldsymbol{h_t = (1 - u_t) \cdot h_{t-1} + u_t \cdot \tilde{h}_t}$$

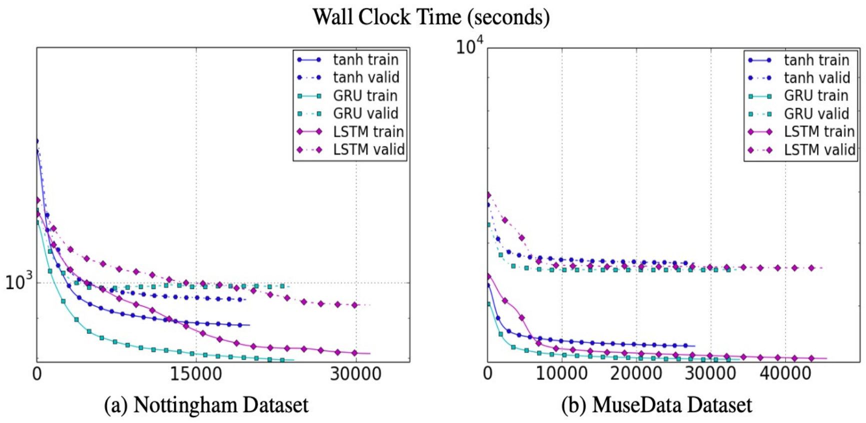
GRU vs LSTM

- **Схожесть**
  - В GRU, как и в LSTM – текущее состояние не перезаписывает предыдущее, как в RNN, а добавляется сверху. Это позволяет избежать исчезновения градиента.
    - Γ_u в GRU по функционалу схоже с Γ_o LSTM – сколько информации из h_{t-1} мы добавляем в h_t.

- **Различия**
  - В GRU есть 2 фильтра – Γ_u(update) и Γ_r(reset), в то время как в LSTM 3 - Γ_f(forget), Γ_u(update), Γ_o(output). То есть в GRU нет контроля над тем, какую информацию увидит следующее состояние – отсутствует Γ_o
  - В LSTM также есть контроль информации с предыдущего состояния Γ_f, в GRU сеть видит полную информацию с предыдущего состояния.
  - Для обучения LSTM обычно нужно больше данных, чем для GRU

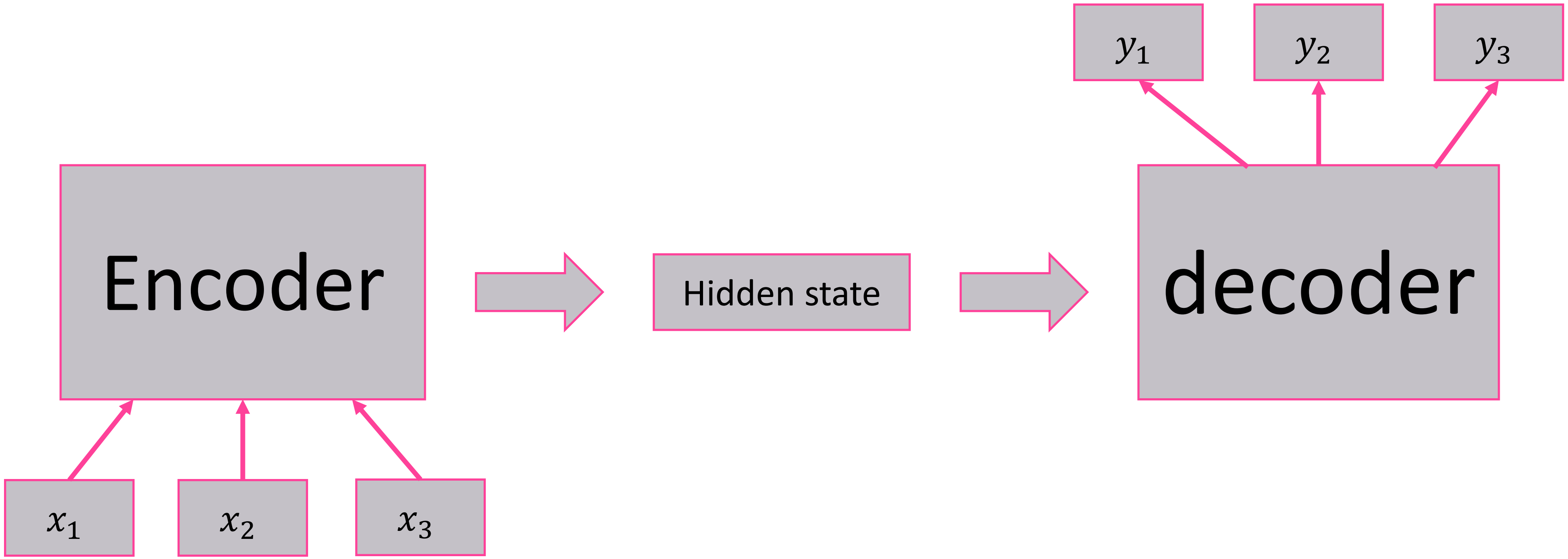


GRU и LSTM сходятся похоже

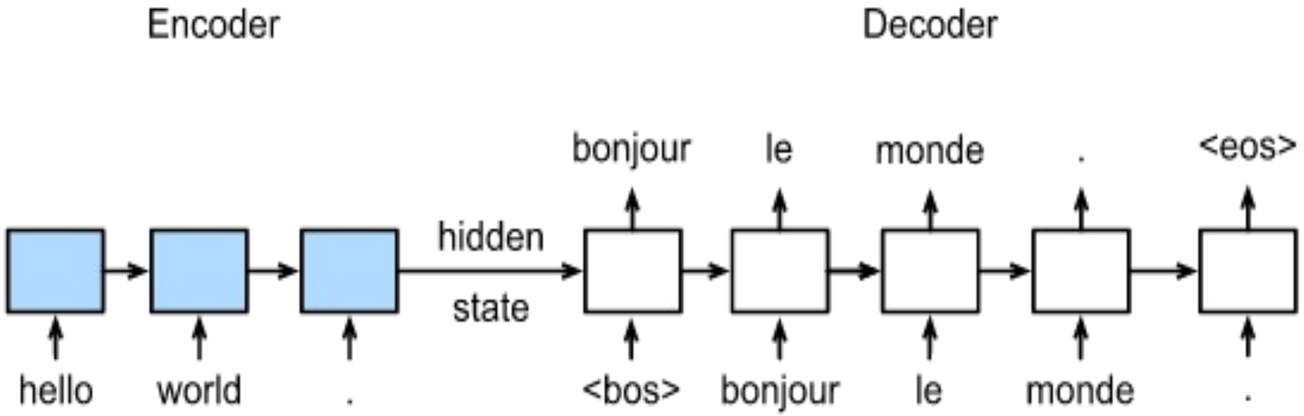


Но GRU сходится быстрее

Seq2seq



Encoder
Hidden state
decoder
x_1
x_2
x_3
y_1
y_2
y_3



Encoder
Decoder
bonjour
le
monde
.
<eos>
hidden
state
hello
world
.
<bos>
bonjour
le
monde
.

Эффективность рекуррентных сетей

For $\bigoplus_{n=1,\dots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on X, U is a closed immersion of S, then $U \to T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps M along the set of points Sch_{fppf} and $U \to U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ??. Hence we obtain a scheme S and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of X. Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S.

Proof. See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example ??. It may replace S by $X_{spaces,étale}$ which gives an open subspace of X and T equal to S_{Zar}, see Descent, Lemma ??. Namely, by Lemma ?? we see that R is geometrically regular over S.

Lemma 0.1. *Assume (3) and (3) by the construction in the description.*

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over U compatible with the complex

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S. Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) *f is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

Proof. This is form all sheaves of sheaves on X. But given a scheme U and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{\mathcal{X},\dots,0}$.

Lemma 0.2. *Let X be a locally Noetherian scheme over S, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

Lemma 0.3. *In Situation ??. Hence we may assume $\mathfrak{q}' = 0$.*

Proof. We will use the property we see that $\mathfrak{p}$ is the mext functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F-algebra where δ_{n+1} is a scheme over S. □

Proof. Omitted. □

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 $X_{étale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

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

Lemma 0.2. *This is an integer $\mathcal{Z}$ 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

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

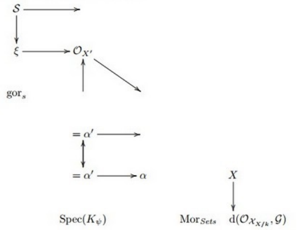
be a morphism of algebraic spaces over S and Y.

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. □

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



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 $X = \mathrm{Spec}(R)$ 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. □

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 $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

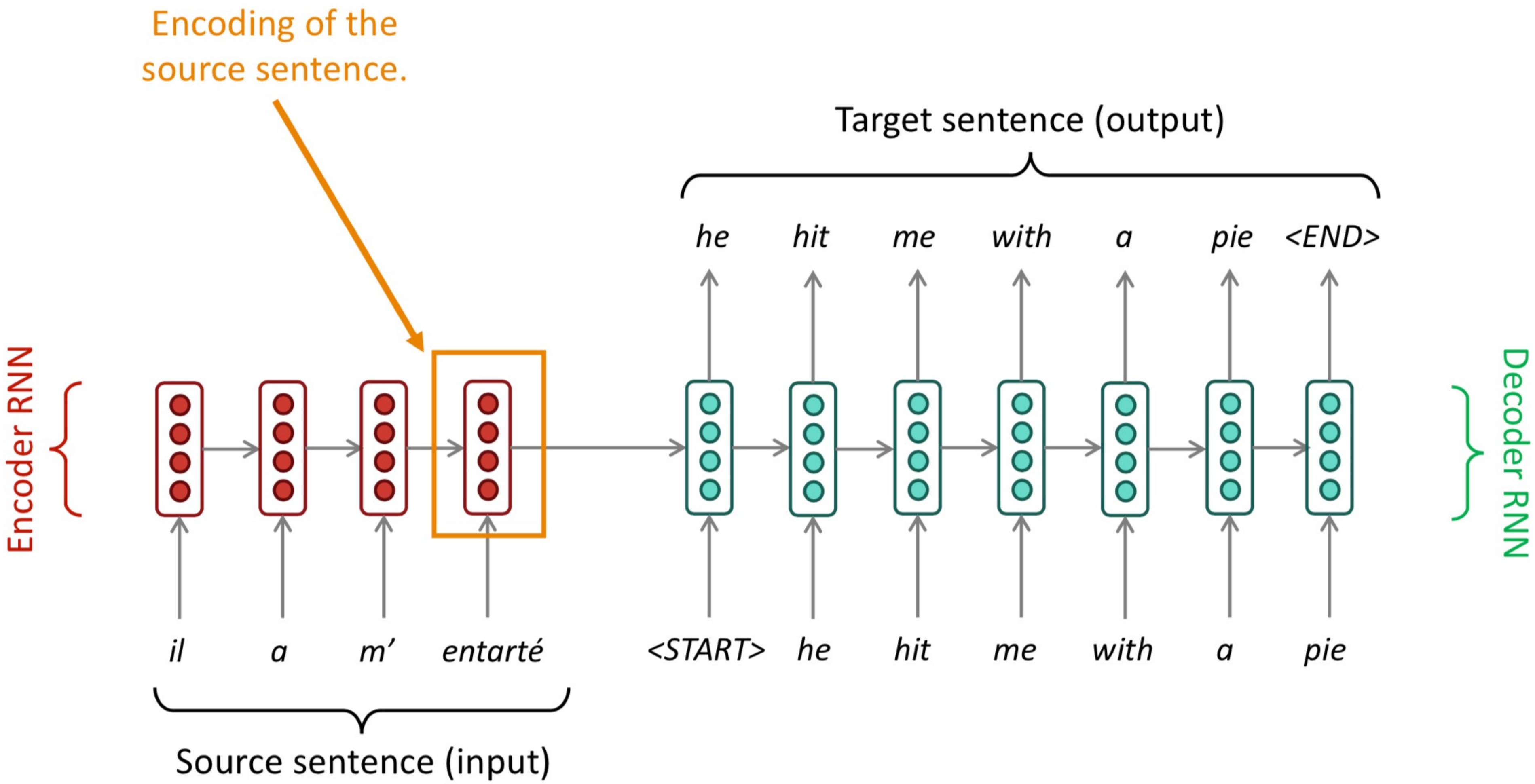
$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \ \text{-1}(\mathcal{O}_{X_{étale}}) \longrightarrow \mathcal{O}^{-1}_{X_\ell}\mathcal{O}_{X_\lambda}(\mathcal{O}^{\overline{v}}_{X_\eta})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. 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. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

```c
/*
 * 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;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
  rw->name = "Getjbbregs";
  bprm_self_clearl(&iv->version);
  regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
  return segtable;
}
```

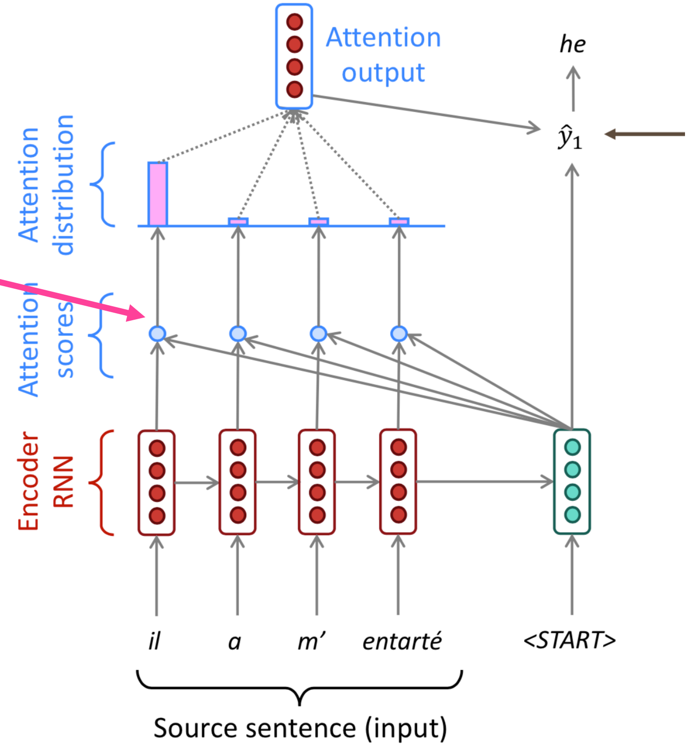
Механизм внимания(attention)



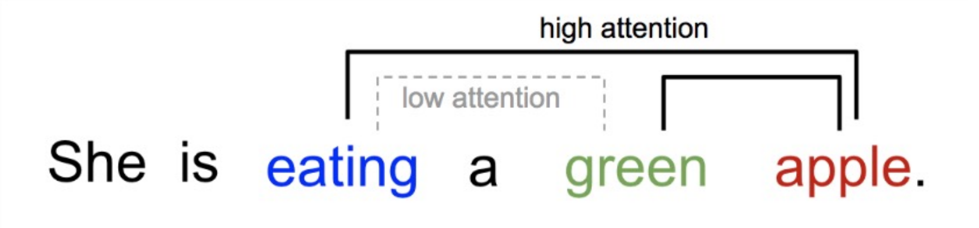
Механизм внимания(attention)

- Attention позволяет решить проблему раскодирования целого предложения из одного вектора
- Помогает определить вклад каждого входного слова в текущее переводимое
- На каждом этапе модель “смотрит” на разные входные элементы, что позволяет учитывать информацию с этапа энкодера.

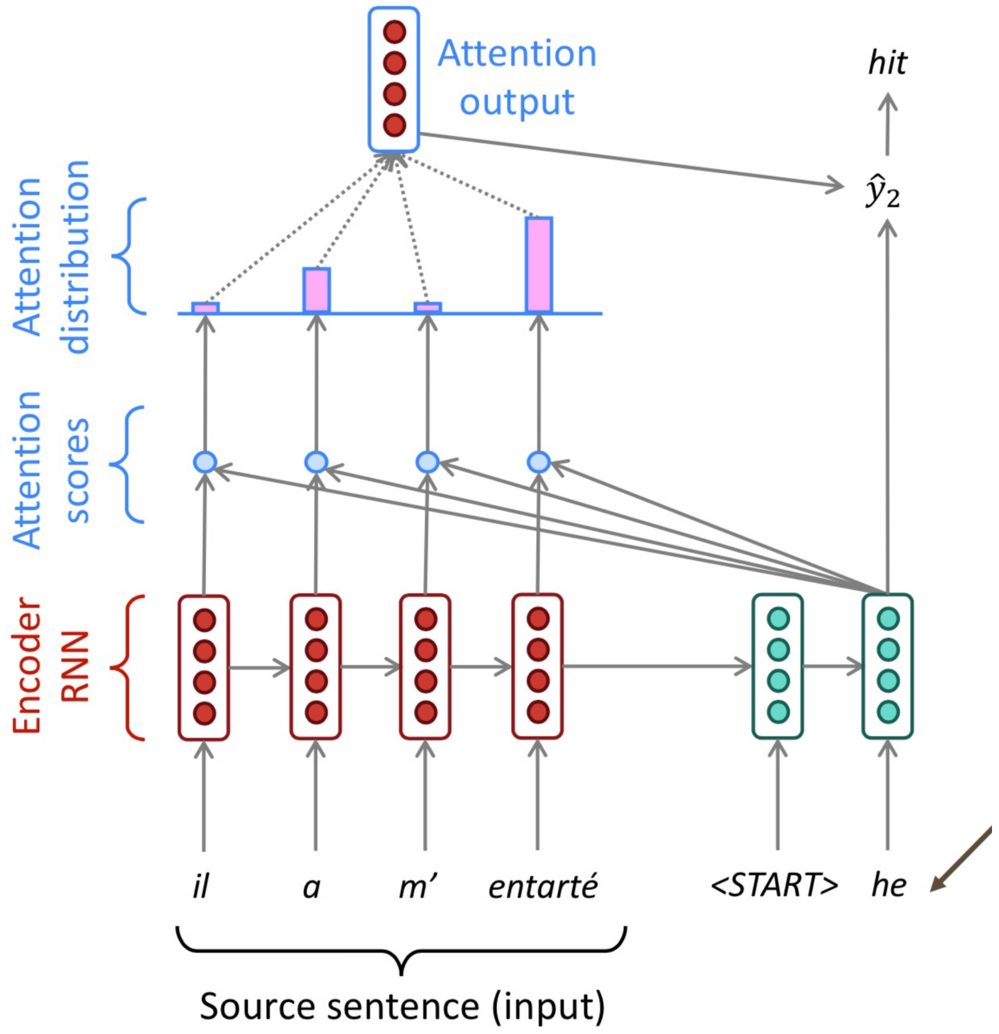
Dot product



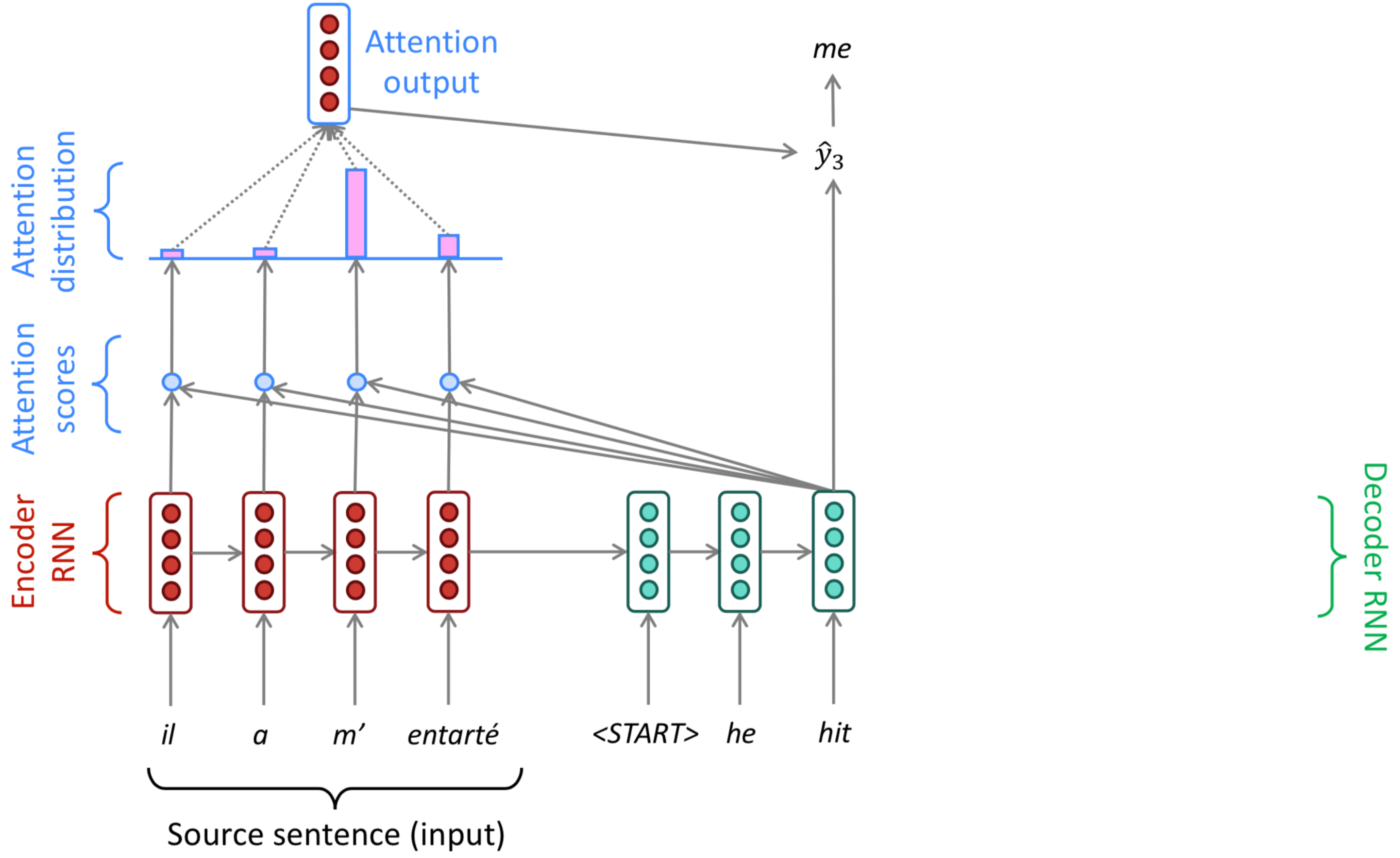
Attention на текущем элементе можно либо сконкатенировать с выходом RNN или добавить в текущий hidden



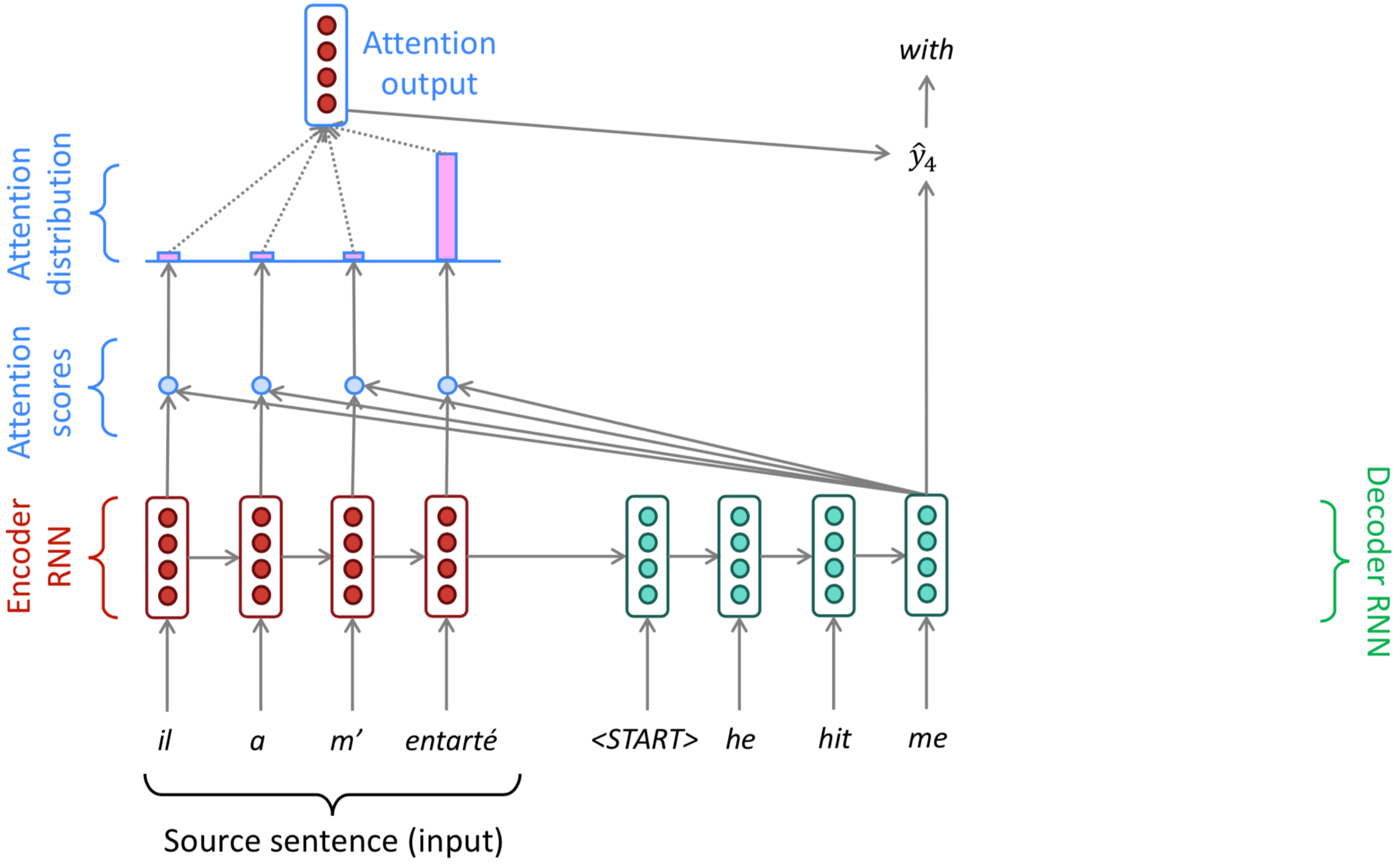
Механизм внимания(attention)



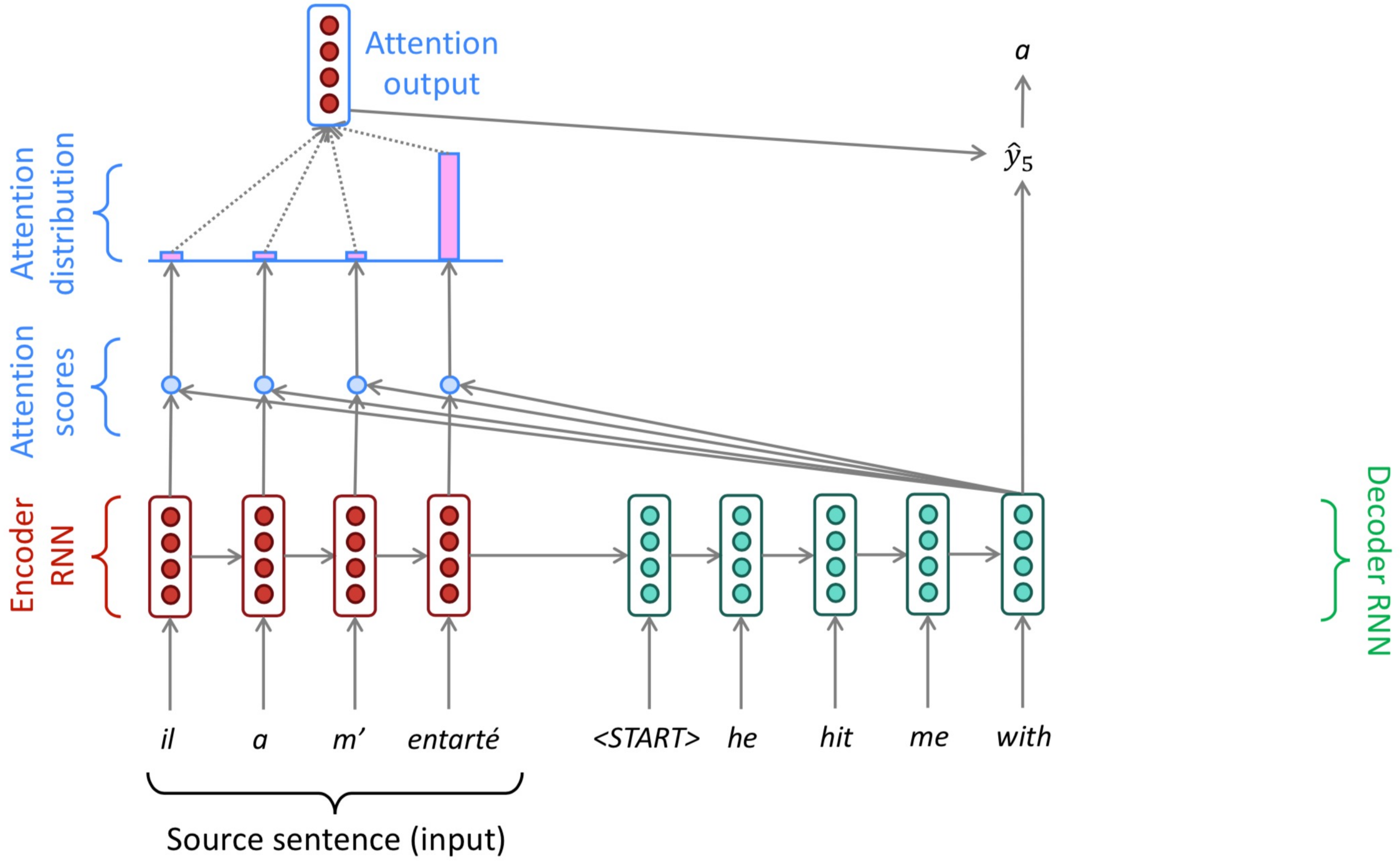
Механизм внимания(attention)



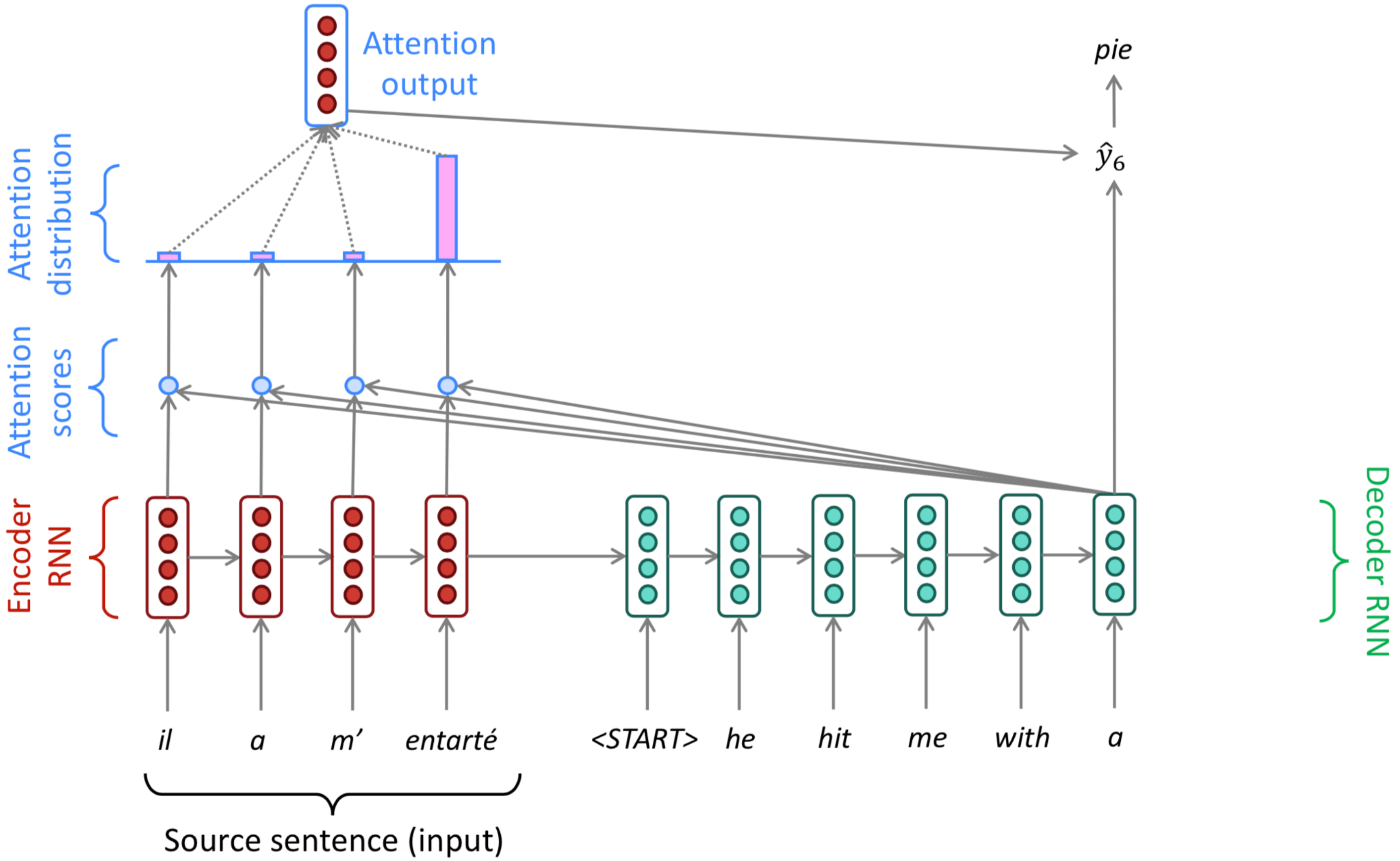
Механизм внимания(attention)



Механизм внимания(attention)



Механизм внимания(attention)



Алгоритм расчета attention

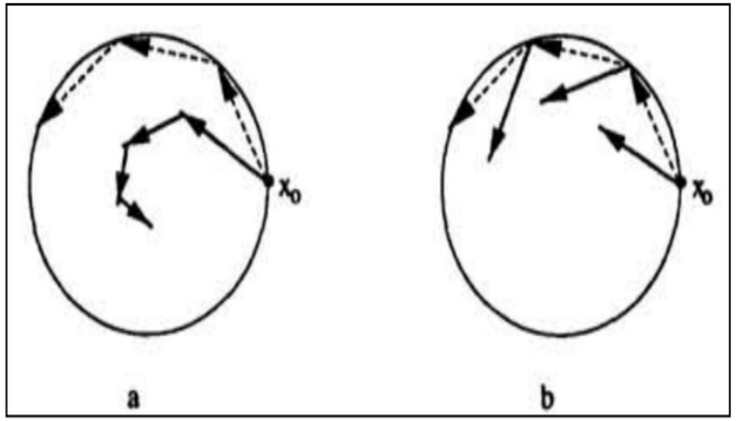
1. У нас есть N hidden states энкодера $h_1, \dots h_n \in R^h$
2. На шаге t есть текущий hidden state декодера $s_t \in R^h$
3. Мы считаем, насколько текущий state s_t декодера “похож” на каждый из state h_n энкодера с помощью метрики скалярного произведения – получаем вектор чисел $e^t = [s_t^T h_1, \dots, s_t^T h_n] \in R^N$
4. Считаем вектор α_i - это $softmax$ от полученного вектора, чтобы получить распределение по словам для текущего hidden state декодера. Это распределение значит – насколько на текущее слово, которое мы пытаемся раскодировать, влияют другие слова из входного предложения
5. Считаем α_i средневзвешенный вектор для текущего s_t по всем h_i

$$\alpha_t = \sum_{i=0}^{N} \alpha_i^T h_i \in R^h$$

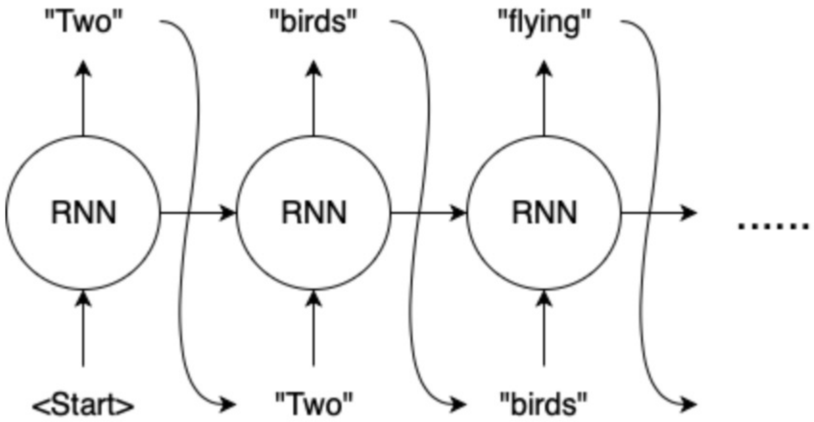
Посчитанный α_t конкатенируем/суммируем с hidden state/выходом декодера. Не забываем проецировать в вектор размером нашего словаря – $\mathrm{W_{out}}([a_t; s_t]) -> R^V$

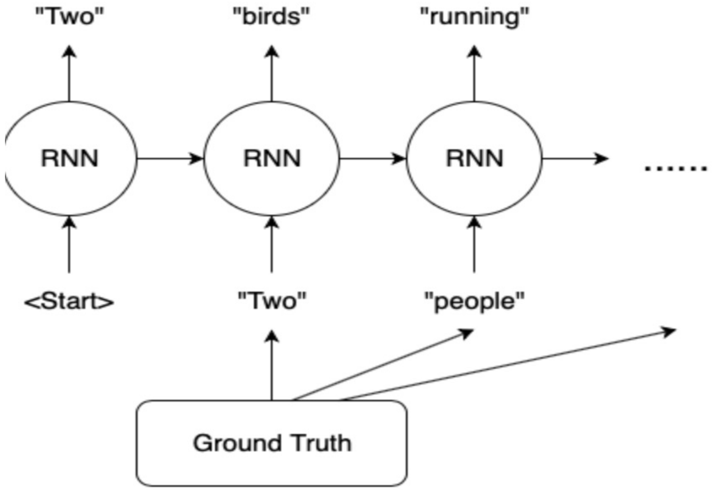
Teacher forcing

- На этапе обучения t, модель получает на вход истинное значение y_t вместо предсказанного на этапе $\hat{y}_{t-1}$
- В таком случае, $\boldsymbol{p}(\boldsymbol{y_1}, \boldsymbol{y_0}|\boldsymbol{h_0}) = \boldsymbol{p}(\boldsymbol{y_0}|\boldsymbol{h_0}) * \boldsymbol{p}(\boldsymbol{y_1}|\boldsymbol{h_1})$ вместо $\boldsymbol{p}(\boldsymbol{y_1}, \boldsymbol{y_0}|\boldsymbol{h_0}) = \boldsymbol{p}(\boldsymbol{y_0}|\boldsymbol{h_0}) * \boldsymbol{p}(\boldsymbol{y_1}|\widetilde{\boldsymbol{y}}_0, \boldsymbol{h_1})$
- Во время тестирования, мы следуем классическому процессу декодирования(y_{t-1} как вход в RNN_t)
- Teacher forcing позволяет держаться корректной траектории обучения.
- Но если распределение данных на этапе тестирования сильно отличается от распределения на этапе обучения **– teacher forcing not a silver bullet.**
  - Включать TF с некоторой вероятностью на каждом этапе(+ curriculum learning technique)
  - Professor forcing



a) Без TF, траектория обучения(сплошная линия) уходит от истинной(пунктир)
b) С TF, на каждом этапе обучения происходит корректировка траектории обучения.





Beam search

$$y_t = argmax\ P(y|y_1, \dots, y_{t-1}, \theta)$$

Time step	1	2	3	4
A	**0.5**	0.1	0.2	0.0
B	0.2	**0.4**	0.2	0.2
C	0.2	0.3	**0.4**	0.2
<eos>	0.1	0.2	0.2	**0.6**

$$0.5 \times 0.4 \times 0.4 \times 0.6 = \mathbf{0.048}$$

Жадный поиск – на каждой итерации берем наиболее вероятное слово

Time step	1	2	3	4
A	**0.5**	0.1	0.1	0.1
B	0.2	0.4	**0.6**	0.2
C	0.2	**0.3**	0.2	0.1
<eos>	0.1	0.2	0.1	**0.6**

Что если на 2ей итерации взять не самое вероятное слово?

$$0.5 \times 0.3 \times 0.6 \times 0.6 = \mathbf{0.054}$$

Что если перебирать все возможные исходы? $|V| = 10000$, длина предложения = 10. $\mathbf{10000^{10} = 10^{40}}$**!**

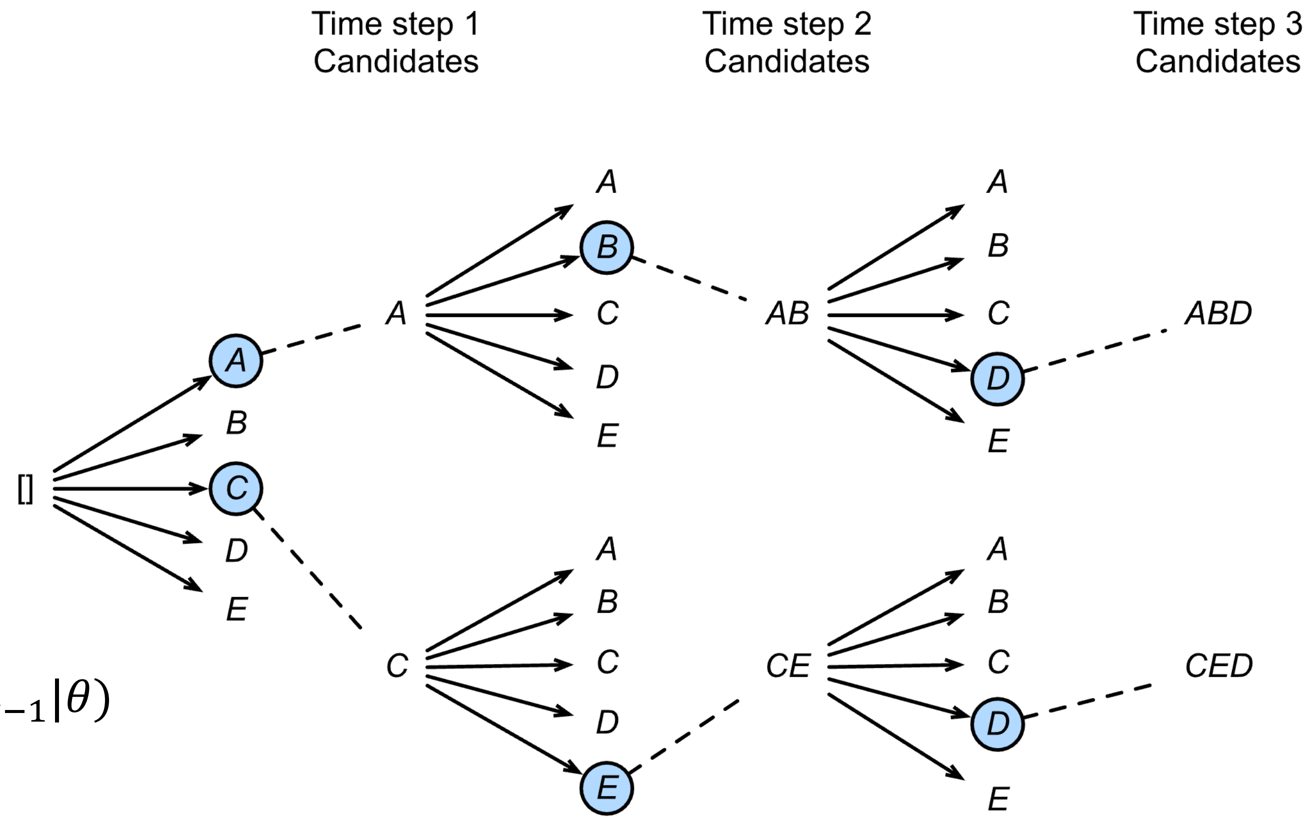
Beam search

Алгоритм Beam Search:

На этапе тестирования модели

0) Фиксируем размер луча $\beta = \mathrm{N}$

1) На t_0, берем β кандидатов
2) На t_1 каждый из кандидатов с t_0 подаем в свою сеть и вместе получаем $\beta * |V|$ кандидатов(из β сетей). Сортируем кандидатов и берем β самых вероятных.
3) Повторяем до конца генерации или *END* символа.
4) Задача - $\max \frac{1}{L^\alpha} \log(P(y_1, \dots, y_T | \theta) = \frac{1}{L^\alpha} \sum_{t=1}^{T} \log P(y_t | y_1, \dots, y_{t-1} | \theta)$



Проблемы:

- **Exposure bias** – ошибки на тесте и обучении разные.
- **Loss evaluation mismatch** – на этапе обучения мы смотрим на значение функции ошибки, в то время как на тесте мы пытаемся оптимизировать метрики перевода(BLEU и тд.)
- **Label bias** – так как на каждом шаге t происходит локальная нормализация, некорректные предсказания, получают такой вес, как и корректные

Это не BFS – оптимальное решение не гарантируется!

https://arxiv.org/pdf/1606.02960.pdf

Метрики

- Perplexity – мера того, насколько хорошо распределение предсказывает класс конкретного примера.

$$PP(p) = 2^{-\frac{1}{N}\sum_{i=0}^{N}\log_2 q(x_i)}$$

Чем больше число получается в степени дроби, тем выше качество модели(модель дает большую вероятность на верных семплах) → Чем больше число в степени дроби, тем ниже значение perplexity(сомнения модели) → Чем ниже значение perplexity, тем больше уверена модель в правильности своего выбора.

- BLEU – оценка качества перевода с одного языка на другой. Основная идея – **чем ближе машинный перевод к переводу эксперта, тем лучше.**

$$BLEU = \min\left(1, e^{1-\frac{refLen}{candLen}}\right)\left(\prod_{i=1}^{4} precision_i\right)^{\frac{1}{4}}$$

- Считается по всему корпусу
- Служебные и лексически важные слова считаются одинаково(a vs NASA)
- Не учитывается значение и грамматическая корректность предложения
- Зависит от нормализации и токенизации

Reference: The NASA Opportunity rover is battling a massive dust storm on Mars .
Candidate 1: The Opportunity rover is combating a big sandstorm on Mars .
Candidate 2: A NASA rover is fighting a massive storm on Mars .

Metric	Candidate 1	Candidate 2
$precision_1$ (1gram)	8/11	9/11
$precision_2$ (2gram)	4/10	5/10
$precision_3$ (3gram)	2/9	2/9
$precision_4$ (4gram)	0/8	1/8
Brevity-Penalty	0.83	0.83
BLEU-Score	**0.0**	**0.27**

Заключение

- Языковое моделирование – способность модели понимать текст и генерировать из него схожие последовательности.
- Контекстно-независимые эмбеддинги позволяют понимать лексическое и грамматические значения слов. Далее эти эмбеддинги используются для других задач NLP
  - Модель Word2Vec учится определять значение слова по его контексту.
  - GLOVE улучшает качество word2vec добавляя в процесс обучения матрицу частотности встречаемости слов.
- Рекуррентные модели позволяют определить весь контекст предложения.
  - Одна из главных технологий NLP – seq2seq моделирование. Оно состоит из энкодера, кодирующего целое предложение в один вектор и декодера, который раскодирует это предложение в новую последовательность(например, в другой язык).
  - Для улучшения seq2seq моделей есть много эвристик:
    - Attention
    - Teacher Forcing
    - Beam Search
- Оценка известными метриками типа BLEU позволяет определить качество основных аспектов моделей, для качественного продукта надо пользоваться собственными метриками.