

Рекуррентные сети и работа с текстами

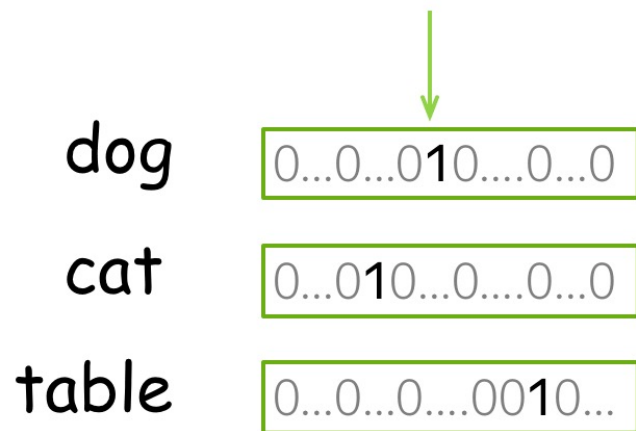
О чем мы сегодня поговорим?

- Векторное представление текста, word2vec
- Классификация текста с помощью нейронных сетей
- Рекуррентные модели для классификации
 - RNN
 - LSTM
 - GRU

Как можно представить текст?

- One-hot encoding

One is 1, the rest are 0



Embedding dimension =
vocabulary size

Как улучшить?

1. There is a bottle of _dkfhvbd___ on the table.
2. It's hard to pick a good _ dkfhvbd ___.
3. ___ dkfhvbd ___ makes you drunk.

	Контекст 1	Контекст 2	Контекст 3	Контекст n
dkfhvbd	1	0	1	1
масло	0	0	0	1
вино	1	1	1	1

Как улучшить?

1. There is a bottle of _dkfhvbd___ on the table.
2. It's hard to pick a good _ dkfhvbd ___.
3. ___ dkfhvbd ___ makes you drunk.

	Контекст 1	Контекст 2	Контекст 3	Контекст n
dkfhvbd	1	0	1	1
масло	0	0	0	1
вино	1	1	1	1



Похожие строки
означают что
смысл слов тоже
схож

Как улучшить?

Главная идея: хотим
использовать контекст для
представления слов

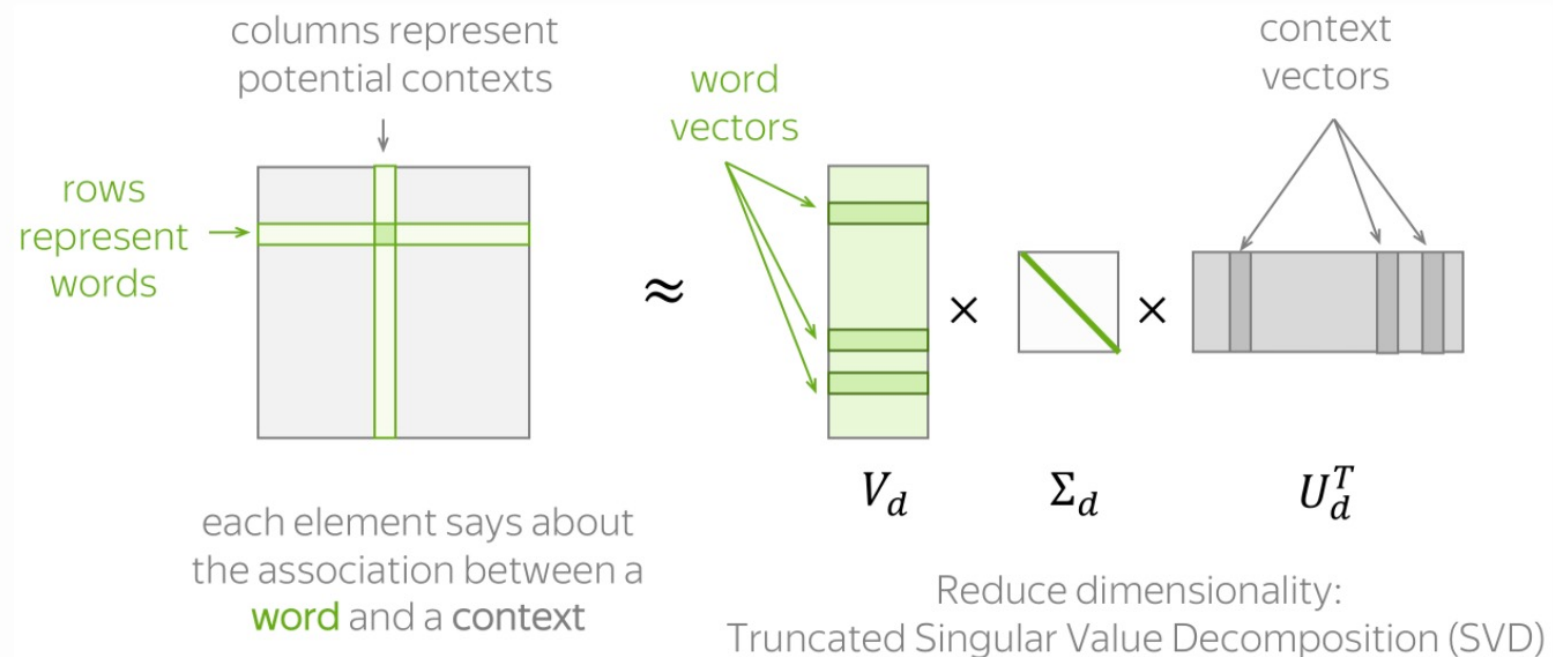
Гипотеза - слова, которые часто
встречаются в одном контексте,
имеют похожее значение

1. На столе стоит бутылка ____.
2. Сложно выбрать хорошее ____.
3. ____ делает вас пьяным.



По контексту можем догадаться о
чем идет речь

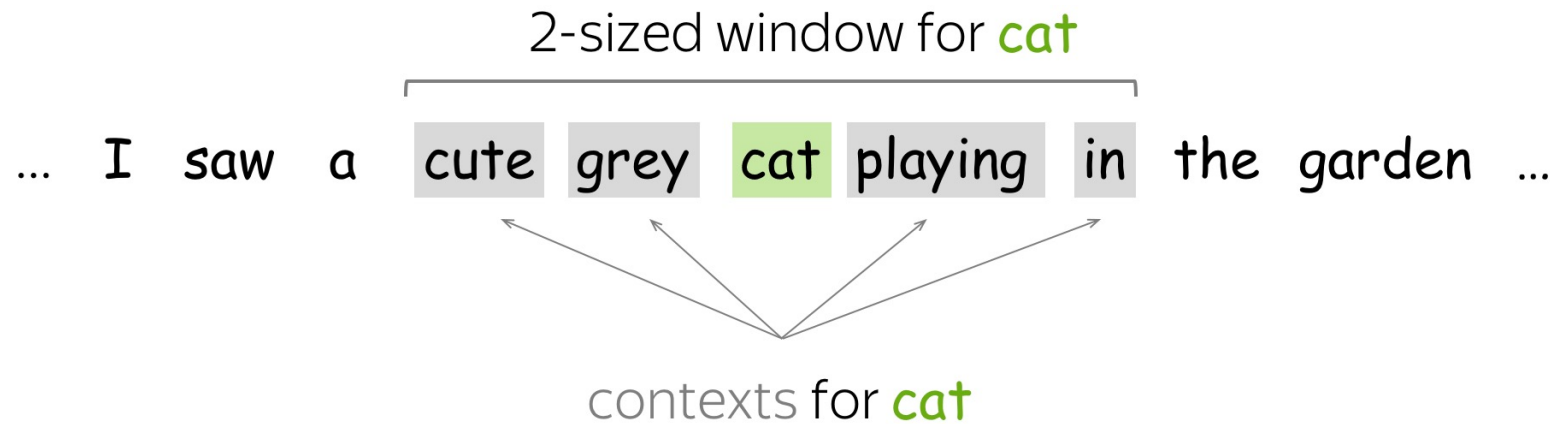
Посчитаем статистики!



Хотим определить:

- Что считать контекстом
- Как считать элементы матрицы

Co-occurrence counts



Контекст – встречающиеся слова в окне размера L

Элемент матрицы – кол-во раз которое встретится слово w в контексте c

Latent Semantic Analysis (LSA)

Контекст – документ из коллекции документов

Элемент матрицы – tf-idf

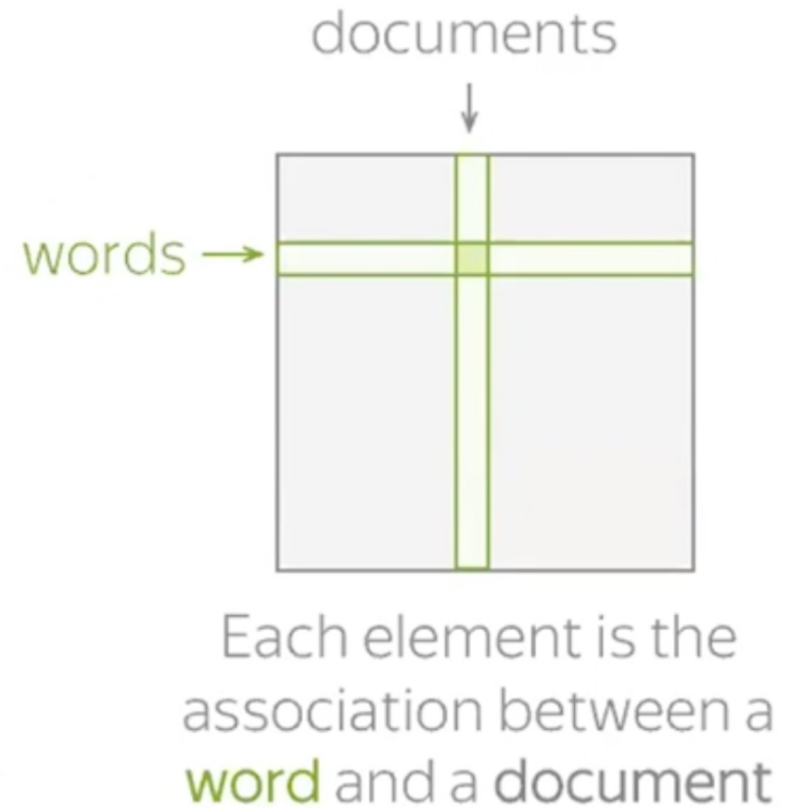
$$\text{tf-idf}(w, d, D) = \text{tf}(w, d) \cdot \text{idf}(w, D)$$

$N(w, d)$

term frequency

$$\log \frac{|D|}{|\{d \in D: w \in d\}|}$$

inverse document frequency



Word2Vec – prediction based метод

Параметры модели – векторы слов

Цель – сделать так, чтобы каждый вектор “знал” о контекстах, в которых он встречается

Как – будем обучать вектора, чтобы предиктить возможный контекст по слову

Word2Vec

Итеративно:

1. Берем большой текстовый корпус
2. Скользящим окном идем по тексту
– на каждом шаге будет
центральное слово и контекст
3. Для центрального слова считаем
вероятности контекста
4. Пересчитаем вектора, чтобы
увеличить эту вероятность

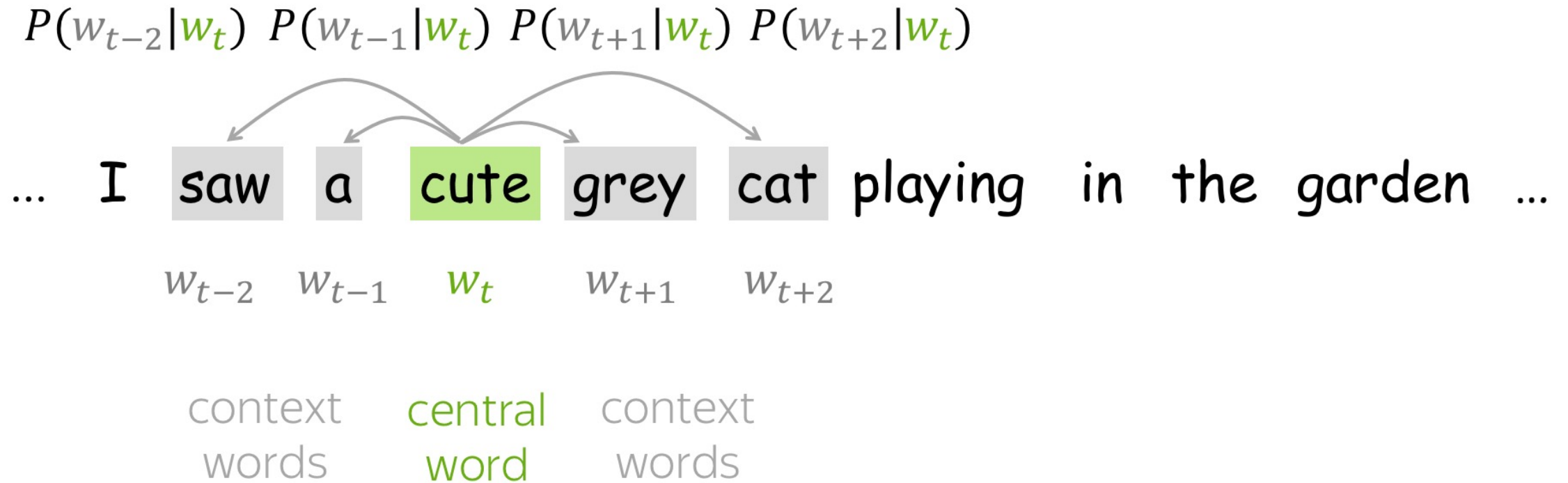
Word2Vec

$$P(w_{t-2}|w_t) \quad P(w_{t-1}|w_t) \quad P(w_{t+1}|w_t) \quad P(w_{t+2}|w_t)$$

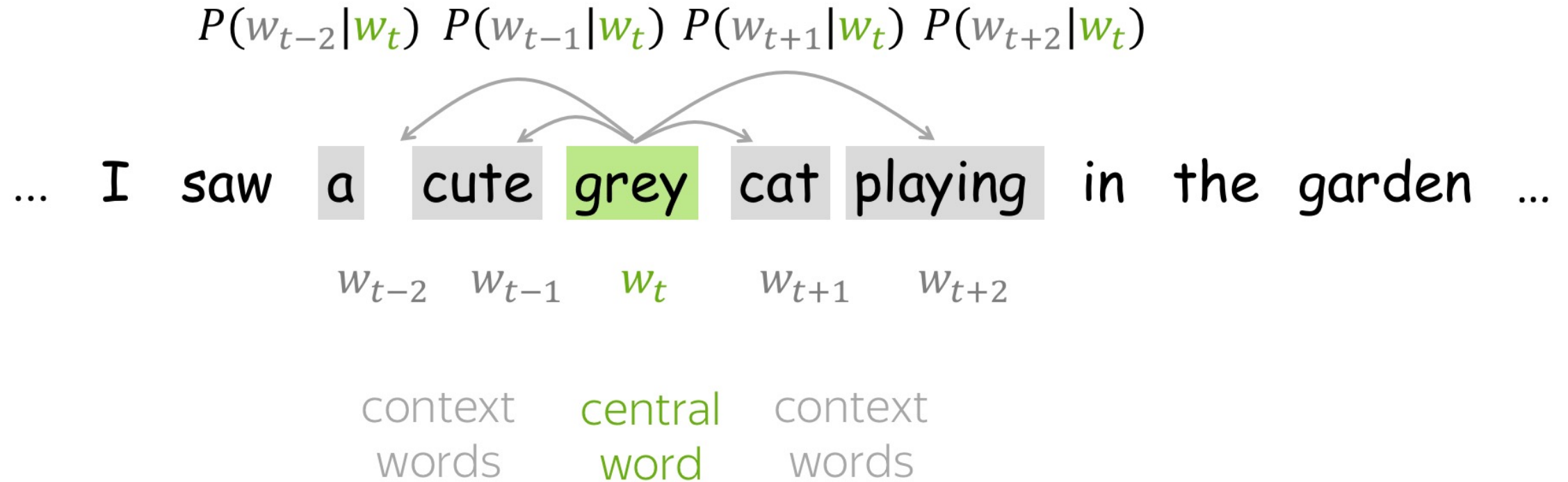


context words central word context words

Word2Vec



Word2Vec



Что оптимизируем?

$$\text{Likelihood} = L(\theta) = \prod_{t=1}^T \prod_{-m \leq j \leq m, j \neq 0} P(w_{t+j} | w_t, \theta),$$

Хотим максимизировать
правдоподобие – текст который мы
встретили более правдоподобен, чем
тот который мы не встречали

$$\text{Loss} = J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^T \sum_{\substack{-m \leq j \leq m, \\ j \neq 0}} \log P(w_{t+j} | w_t, \theta)$$

agrees with our
plan above



go over text



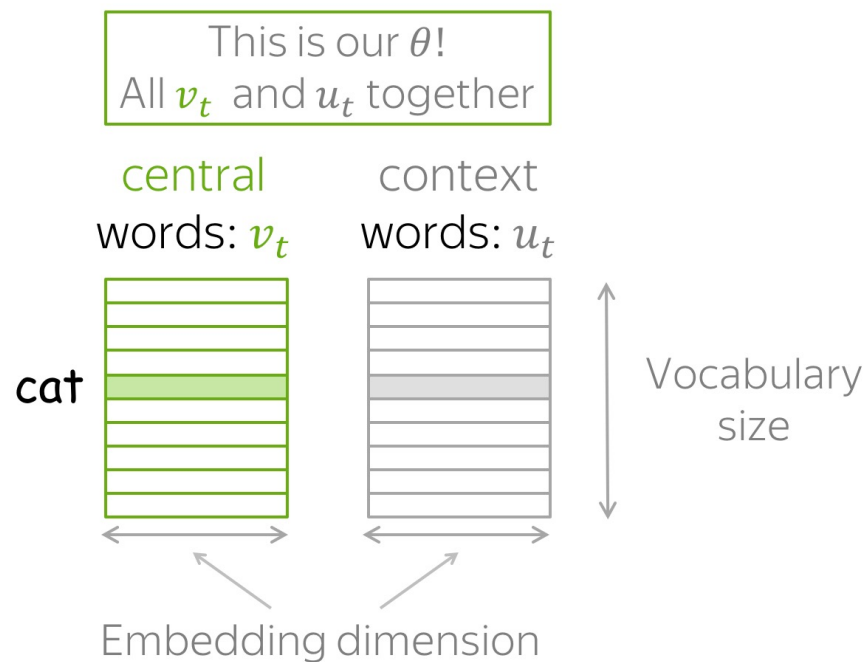
with a sliding
window



compute probability of the
context word given the central

Как считать вероятности?

Два вектора для каждого слова



Посчитаем скалярное произведение векторов

$$P(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$$

Dot product: measures similarity of o and c
Larger dot product = larger probability

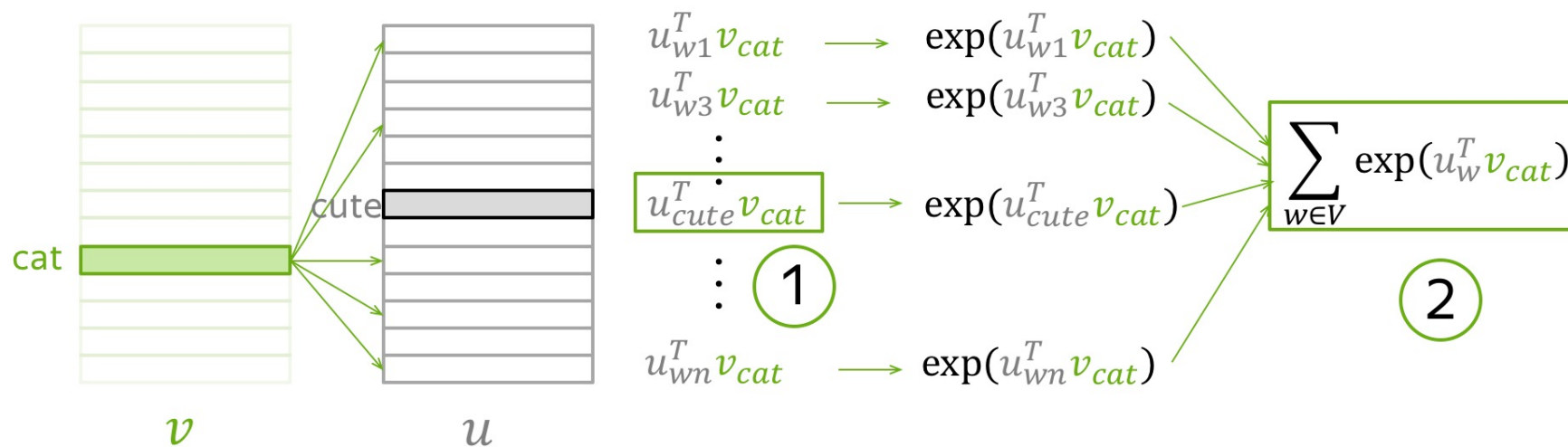
Normalize over entire vocabulary to get probability distribution

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)}.$$

1. Take dot product of v_{cat} with all u

2. exp

3. sum all



4. get loss (for this one step)

5. evaluate the gradient, make an update

$$J_{t,j}(\theta) = \underbrace{-u_{cute}^T v_{cat}}_{1} + \log \underbrace{\sum_{w \in V} \exp(u_w^T v_{cat})}_{2}$$

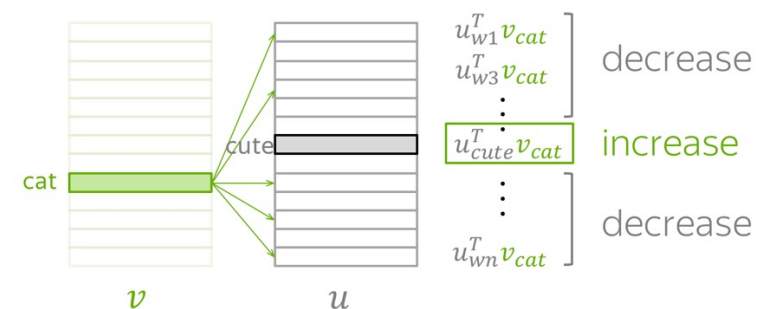
$$v_{cat} := v_{cat} - \alpha \frac{\partial J_{t,j}(\theta)}{\partial v_{cat}}$$

$$u_w := u_w - \alpha \frac{\partial J_{t,j}(\theta)}{\partial u_w} \quad \forall w \in V$$

Negative Sampling

Dot product of v_{cat} :

- with u_{cute} - increase,
- with all other u - decrease



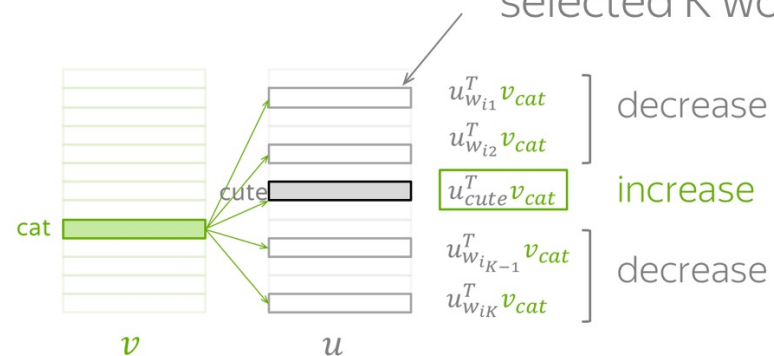
Parameters to be updated:

- v_{cat}
- u_w for all w in the vocabulary $|V| + 1$ vectors

Dot product of v_{cat} :

- with u_{cute} - increase,
- with a subset of other u - decrease

Negative samples: randomly selected K words

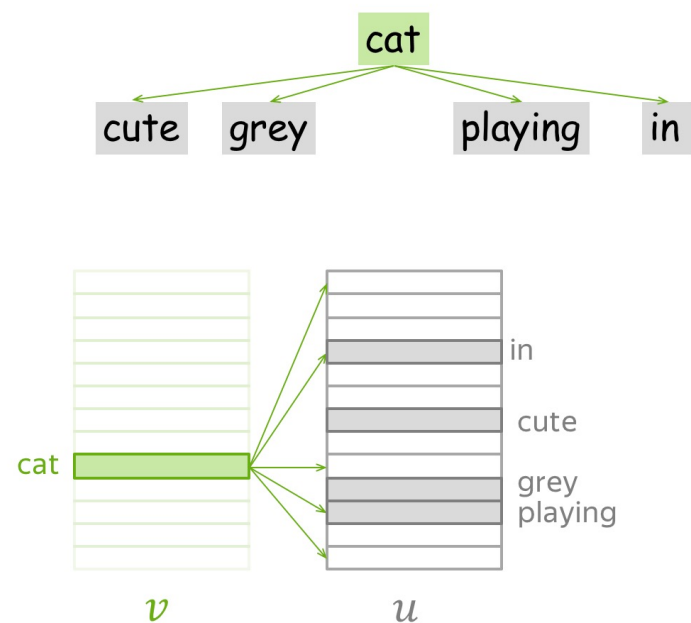


Parameters to be updated:

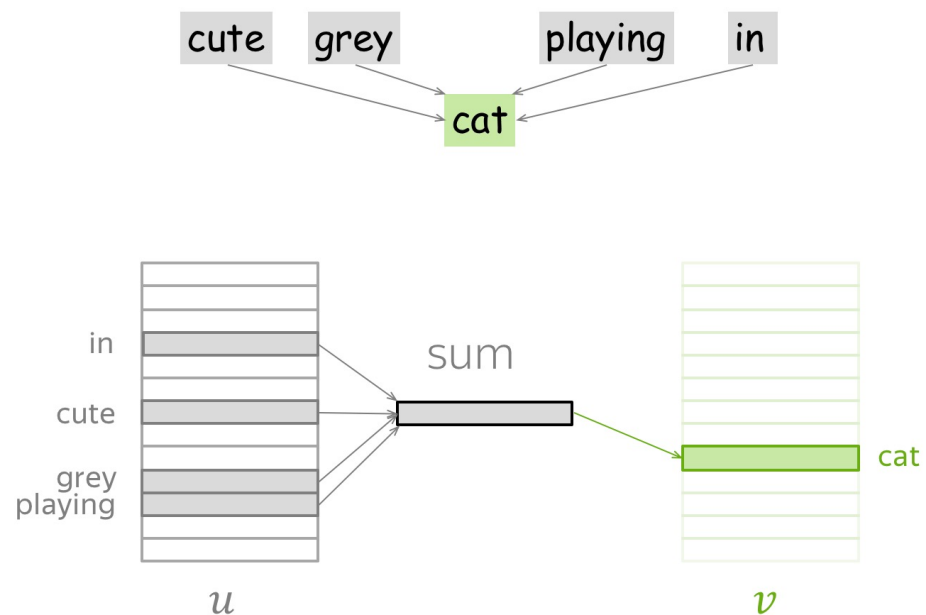
- v_{cat}
- u_{cute} and u_w for w in K negative examples $K + 2$ vectors

Вариации Word2Vec

... I saw a cute grey cat playing in the garden ...



Skip-Gram: from **central** predict context
(one at a time)

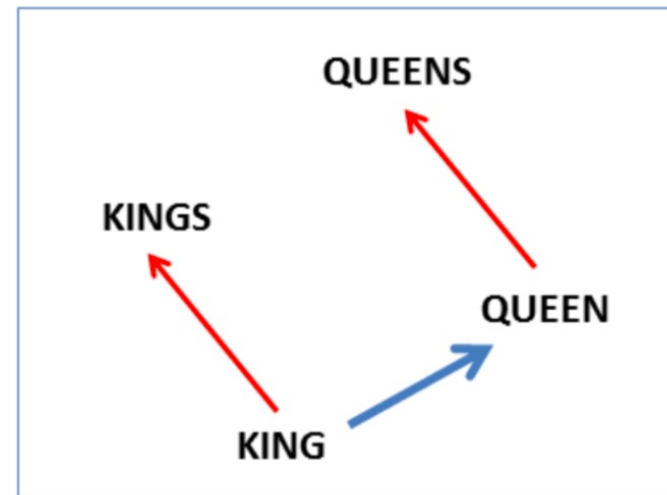
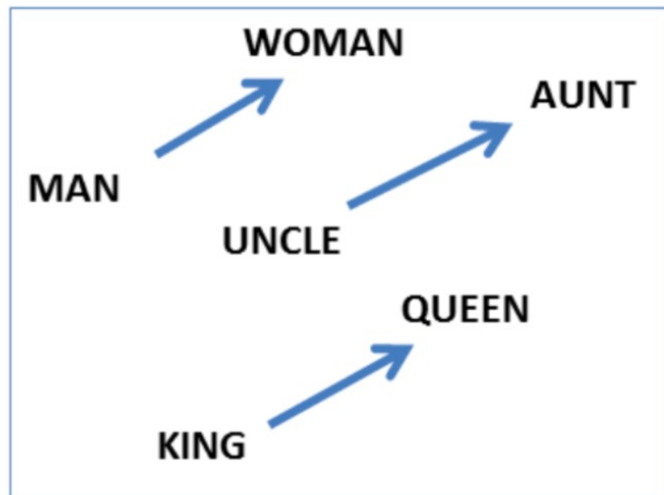


CBOW: from sum of context predict **central**

Word2Vec – линейная структура

semantic: $v(\text{king}) - v(\text{man}) + v(\text{woman}) \approx v(\text{queen})$

syntactic: $v(\text{kings}) - v(\text{king}) + v(\text{queen}) \approx v(\text{queens})$



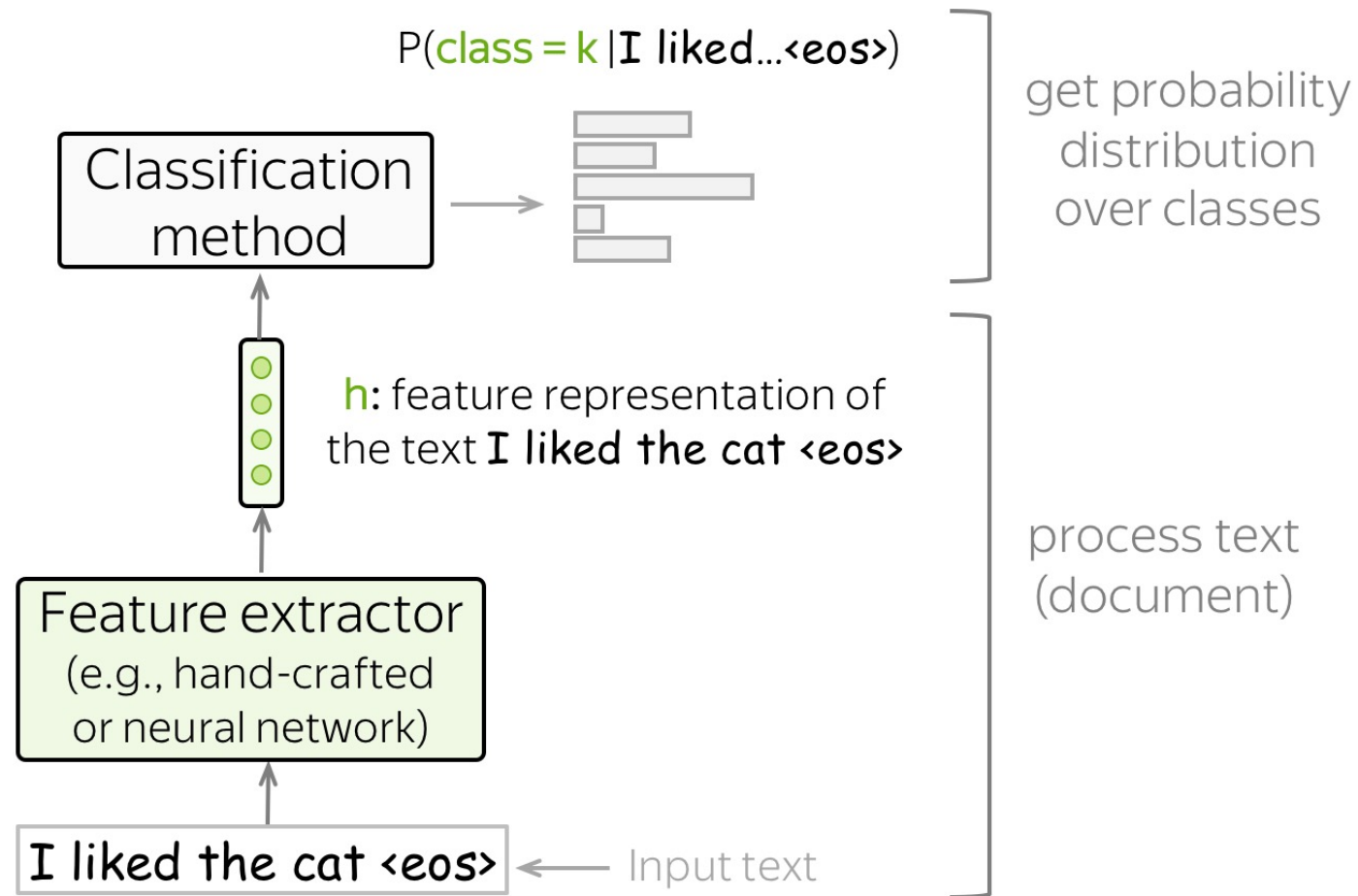
Классификация текстов

Бинарная –
два класса, один правильный

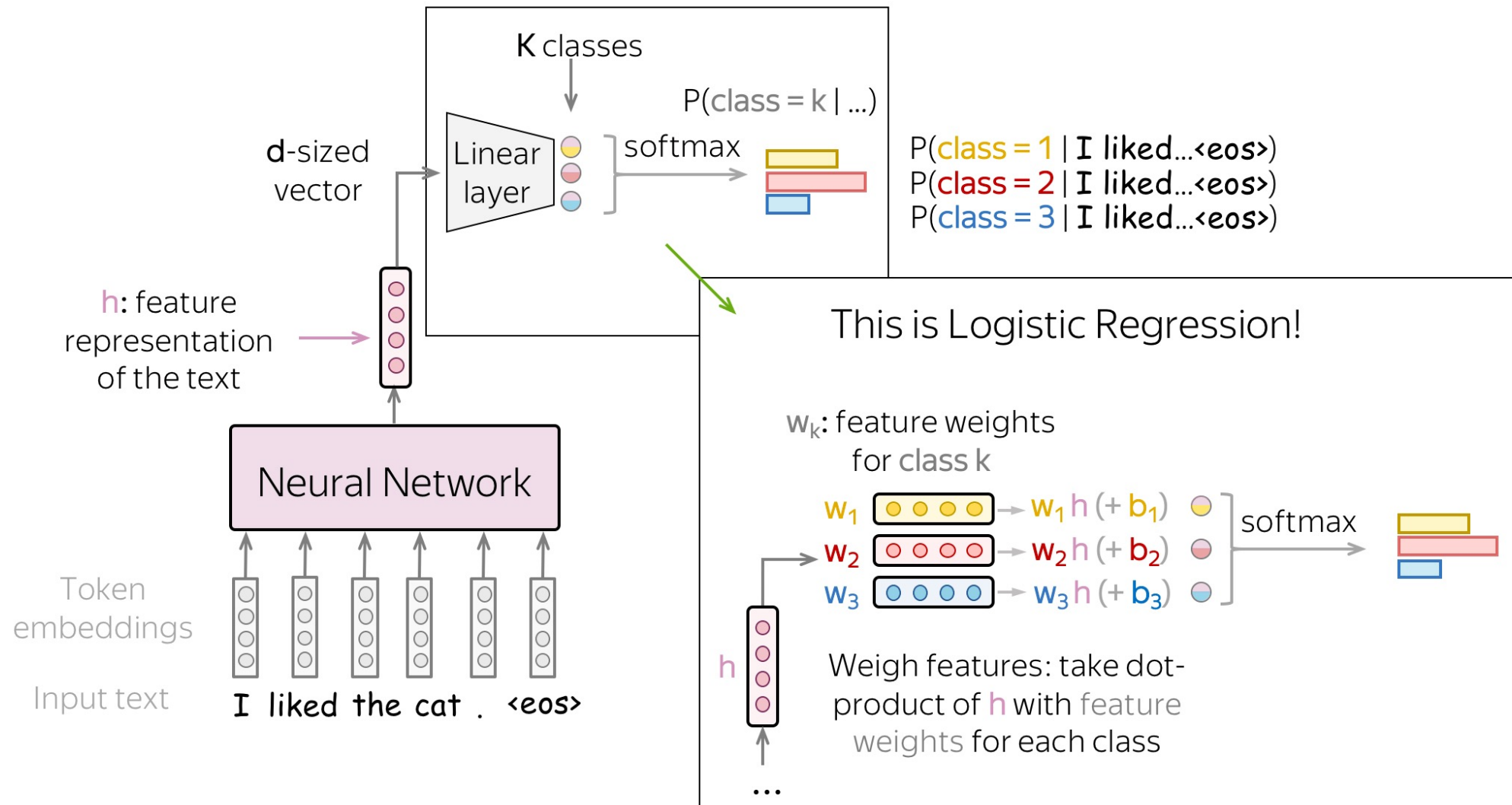
Мультиклассовая –
много классов, один правильный

Мультилейбл –
много лейблов, сразу несколько могут быть правильными

Классификация текстов



Классификация текстов



Как обучать?

Training example: **I liked the cat on the mat** <eos>

Label: **k**
↑
target

Model prediction:

$P(\text{class} = i | \text{I liked...<eos>})$



Target:

p^*



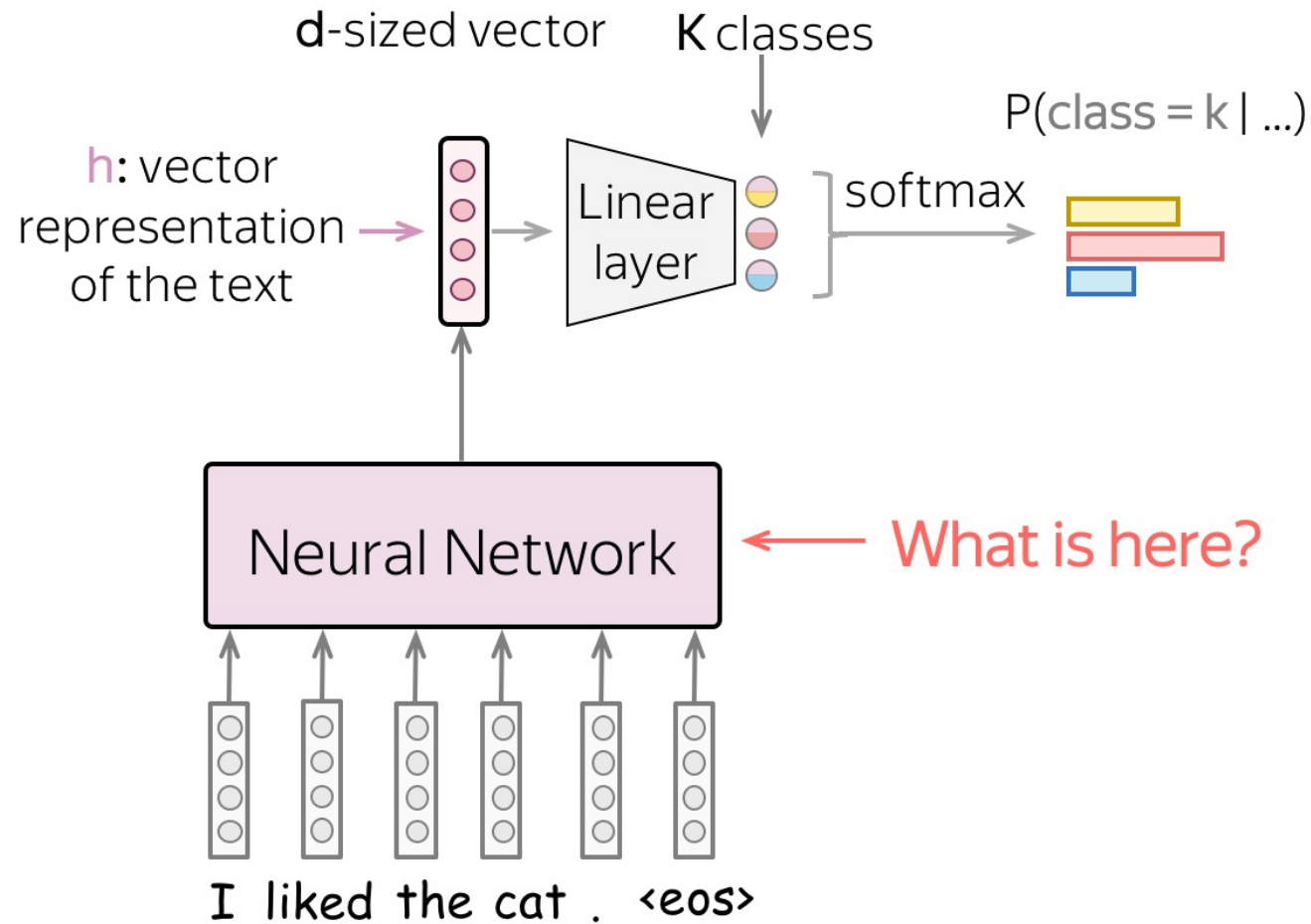
Cross-entropy loss:

$$-\sum_{i=1}^K p_i^* \cdot \log P(y = i|x) \rightarrow \min \quad (p_k^* = 1, p_i^* = 0, i \neq k)$$

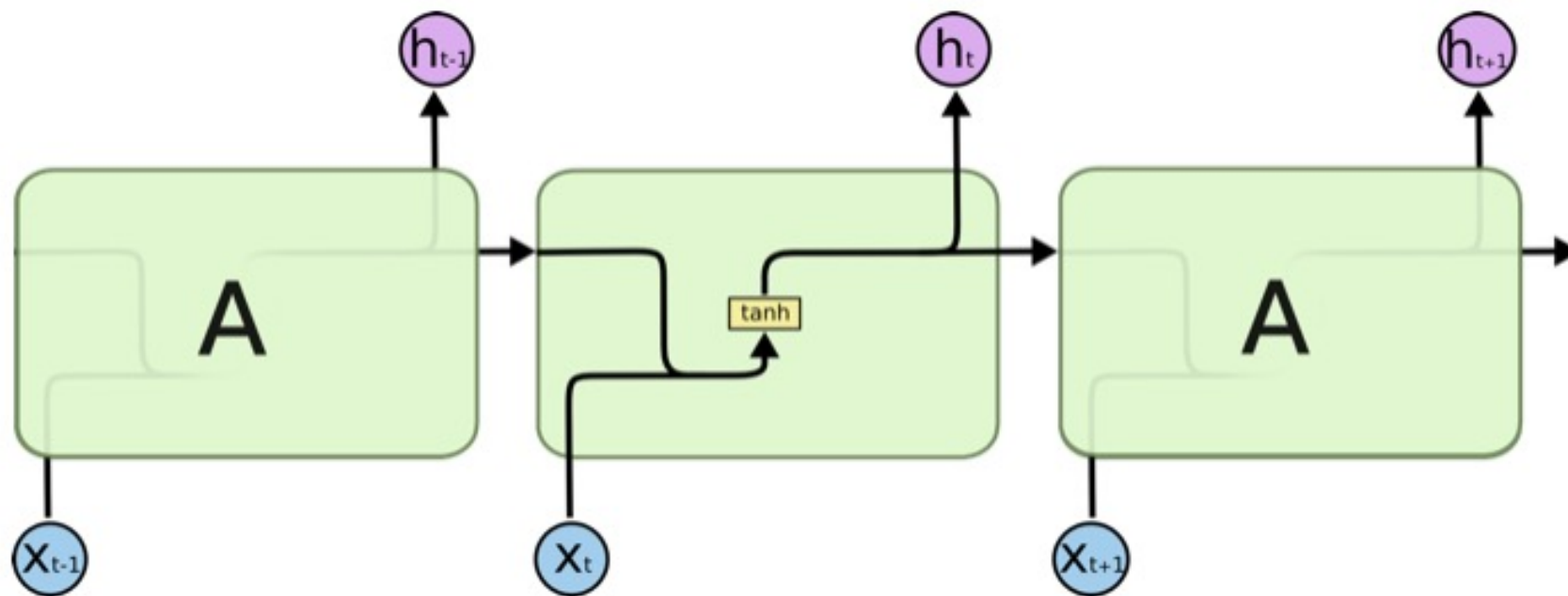
For one-hot targets, this is equivalent to

$$-\log P(y = k|x) \rightarrow \min$$

Классификация текстов



RNN

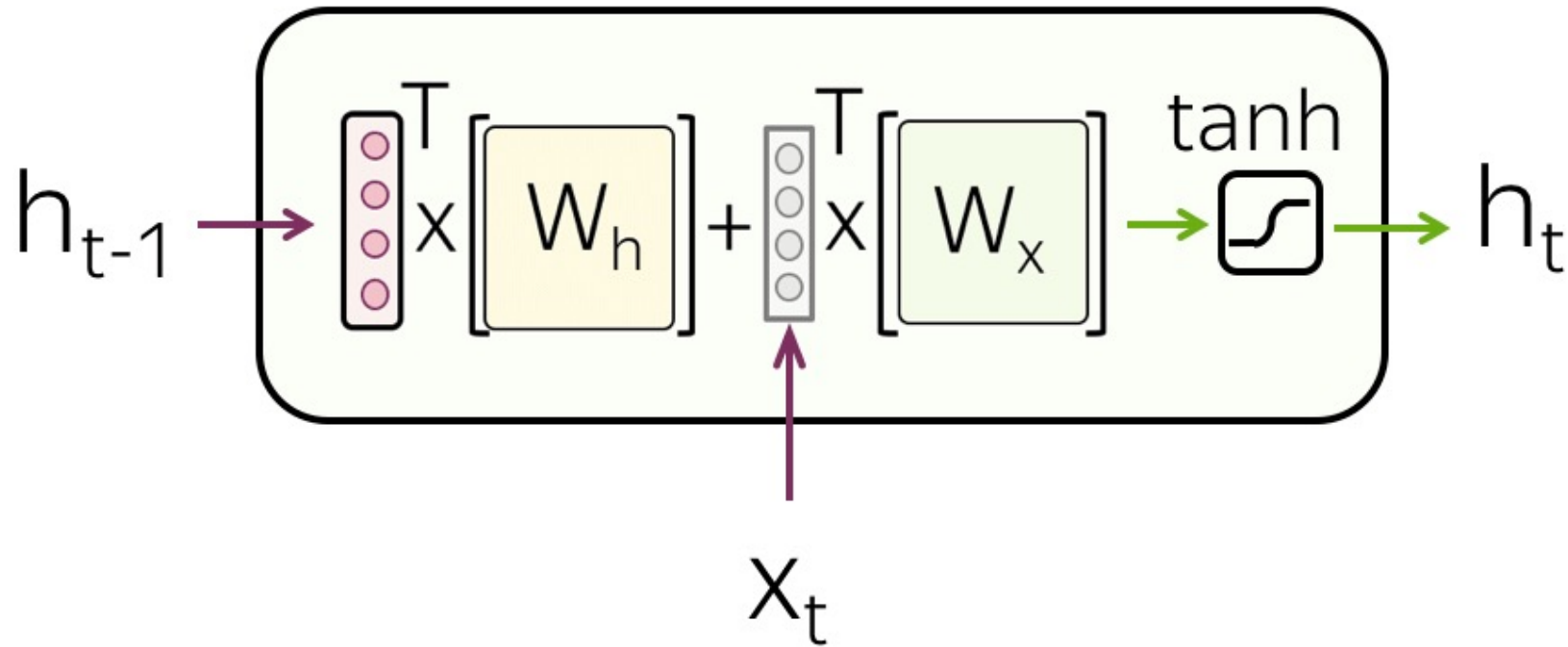


$$h_{i+1} = \tanh(W_x \cdot X_{i+1} + W_y \cdot h_i)$$

RNN

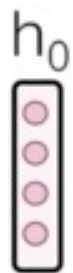
Vanilla RNN

$$h_t = \tanh(h_{t-1}W_h + x_tW_x)$$



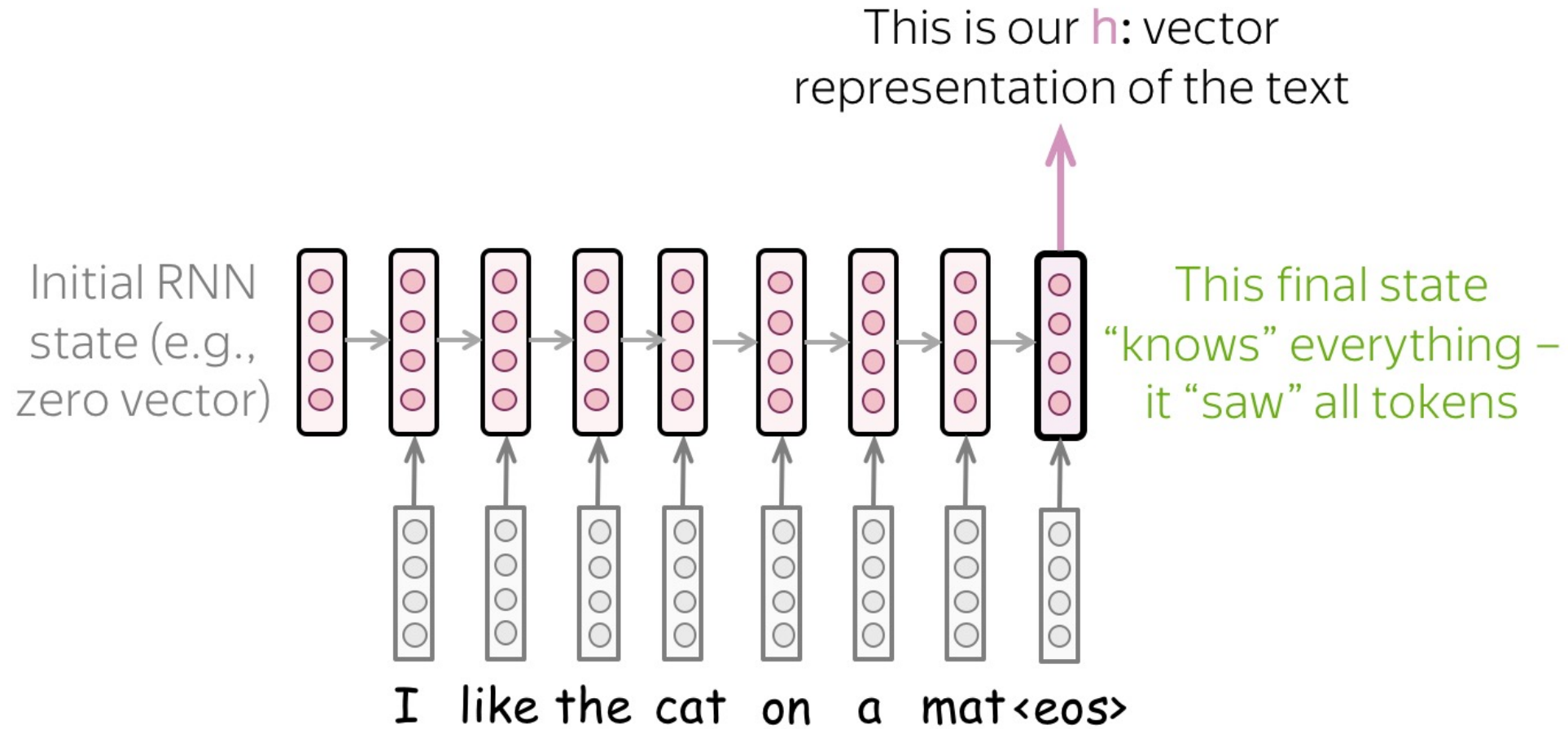
RNN

Initial RNN
state (e.g.,
zero vector)

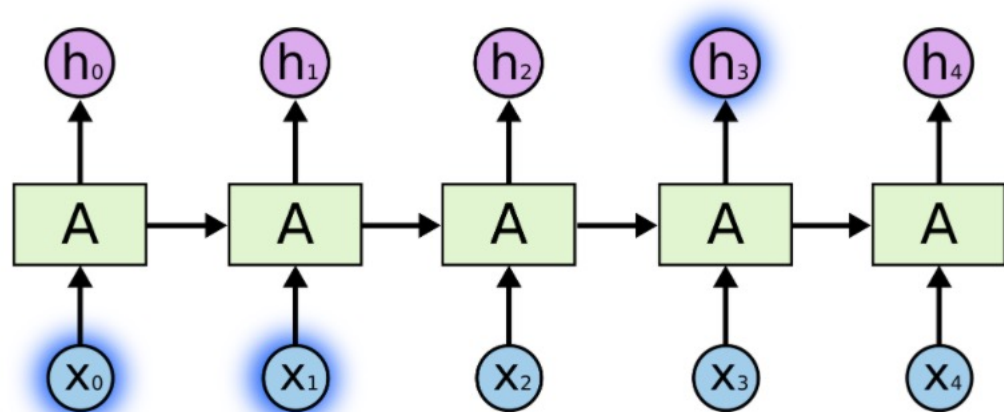


Text: I like the cat on a mat <eos>
not read yet

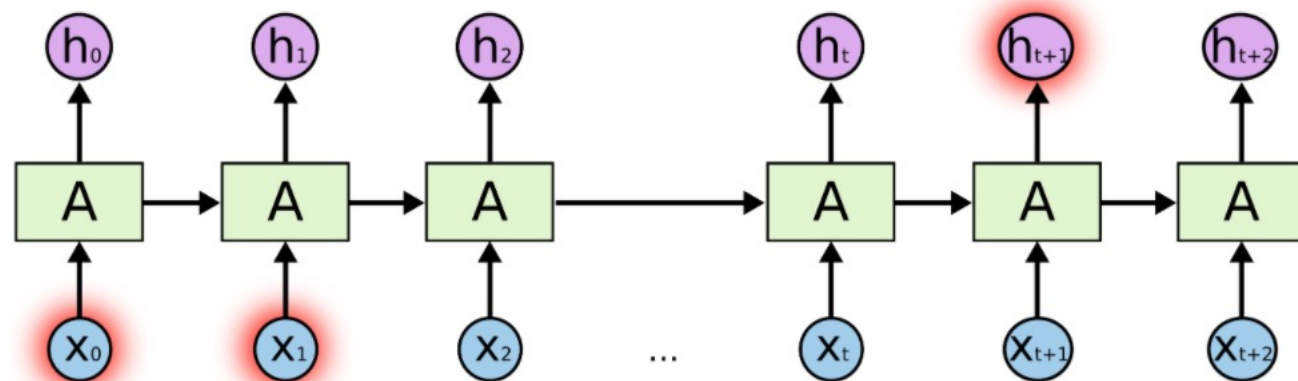
RNN



RNN - минусы

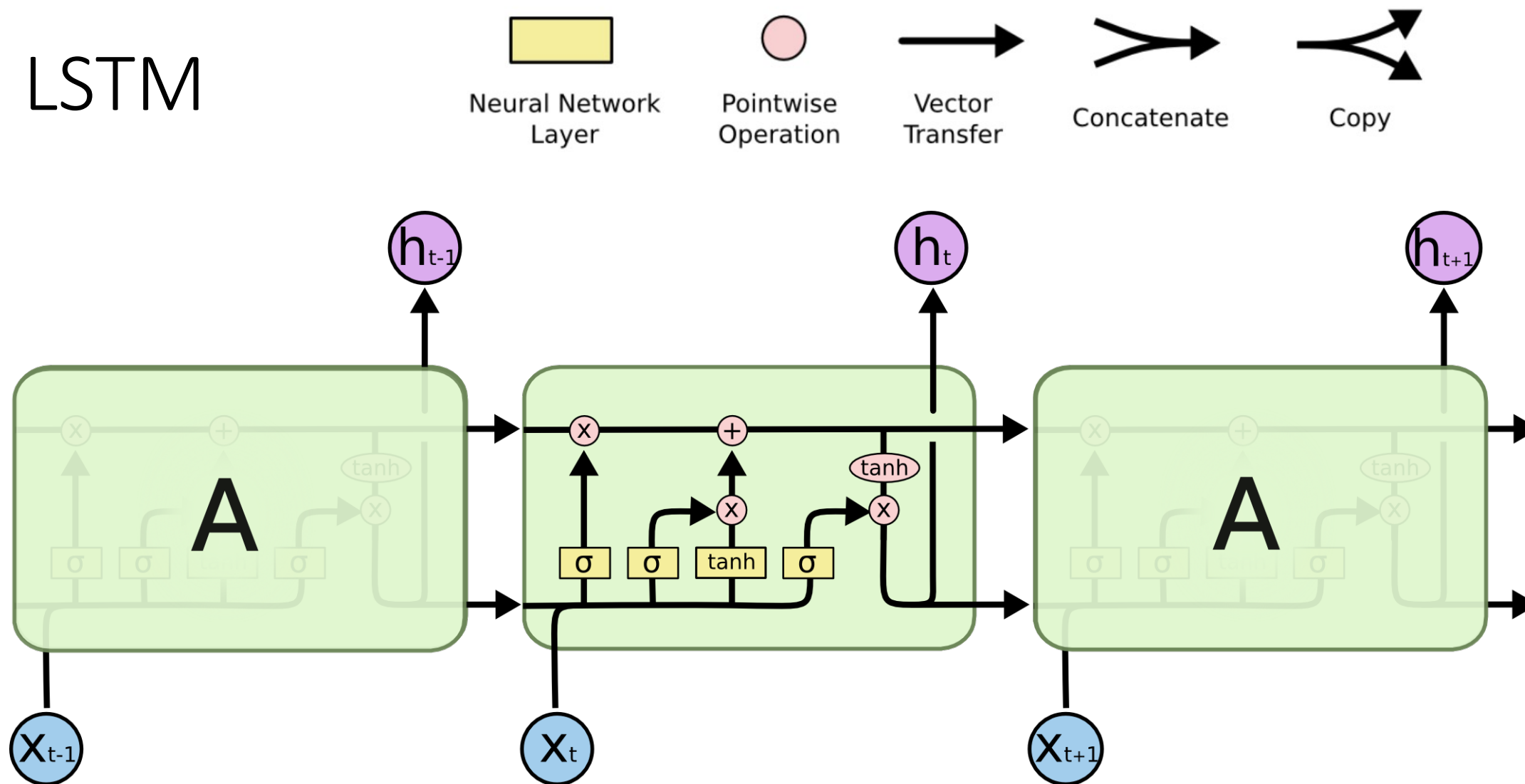


“the clouds are in the *sky*”




“I grew up in France...
I speak fluent *French*”

LSTM





[Hochreiter & Schmidhuber \(1997\)](#)

LSTM

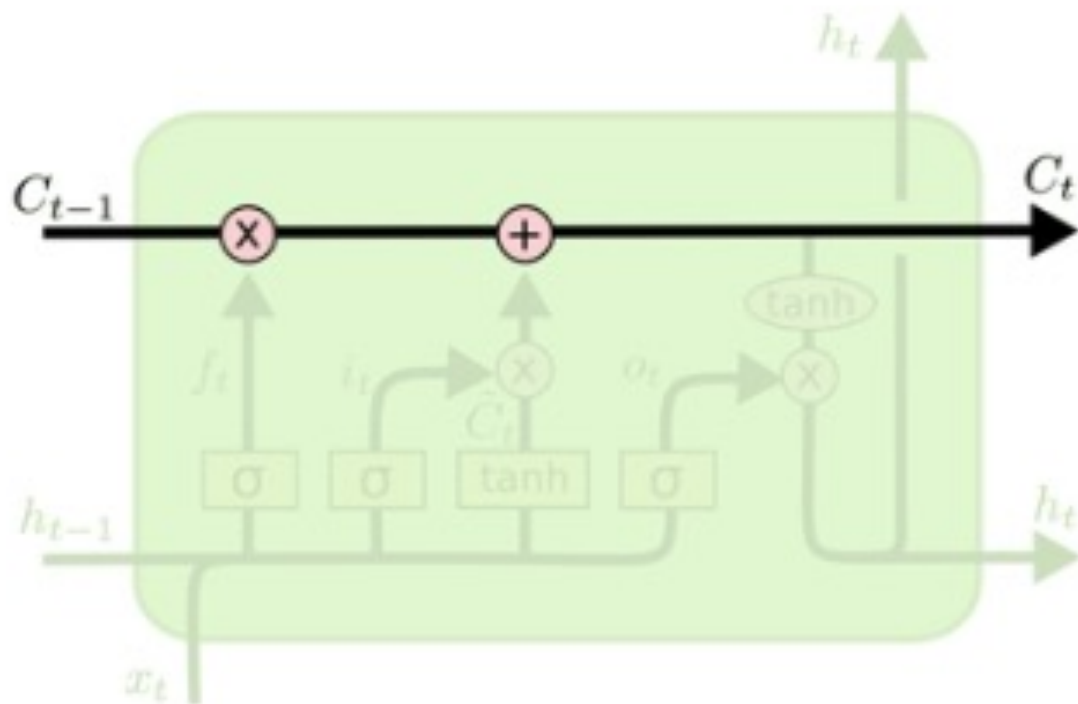

Neural Network
Layer


Pointwise
Operation


Vector
Transfer


Concatenate

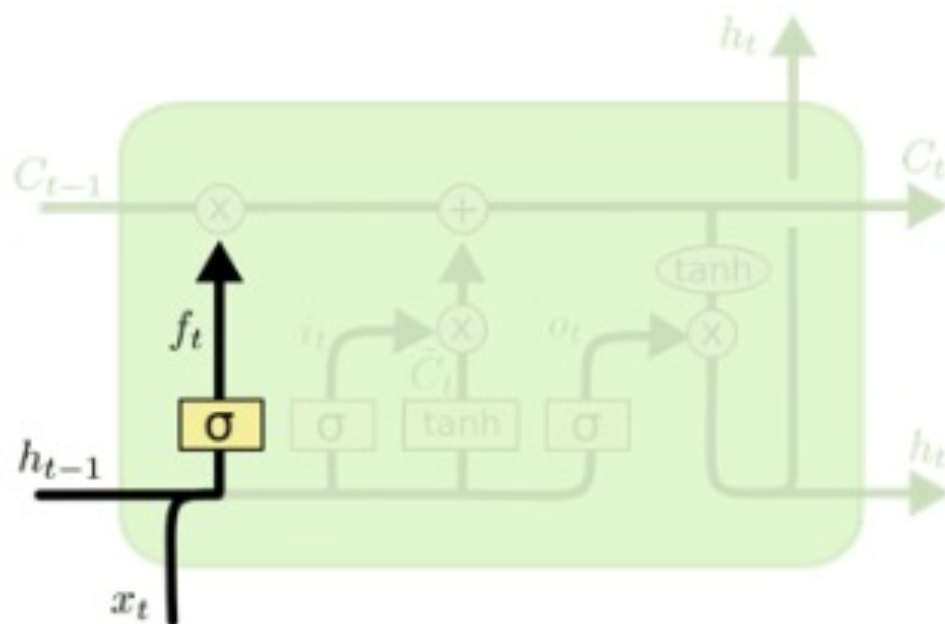
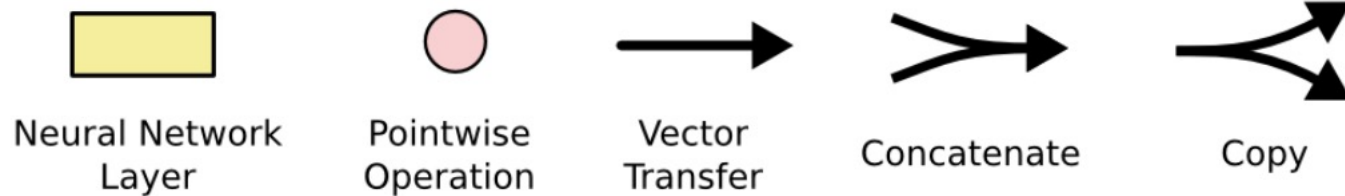

Copy



Cell state:

- "труба" которая позволяет проходить информации
- LSTM может с помощью гейтов на каждом шаге регулировать, какую информацию оставить и какую добавить
- Гейты – это сигмоида, которая принимает значения от 0 до 1
- 0 – не пропускаем информацию, 1 – пропускаем все
- LSTM содержит три таких гейта

LSTM



Forget gate layer:

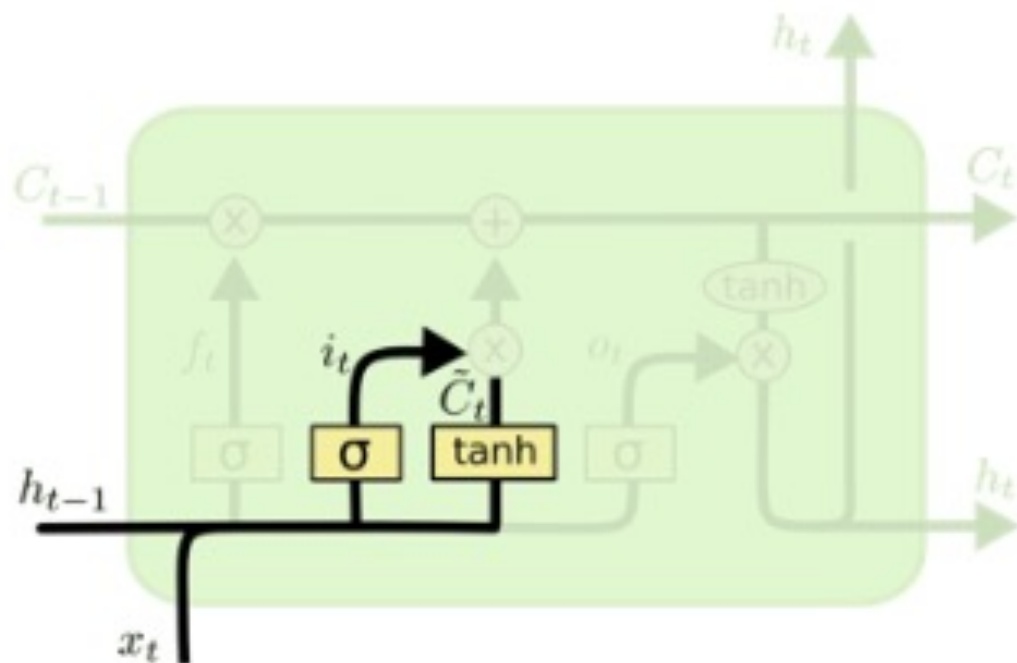
- Сколько информации мы выкинем из cell state

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

Пример – встретили объект нового пола, надо забыть старый

LSTM

- Какую новую информацию мы будем хранить в cell state
- Input gate layer решает что именно мы обновляем
- C_t – вектор кандидатов, которые можно будет добавить к cell state

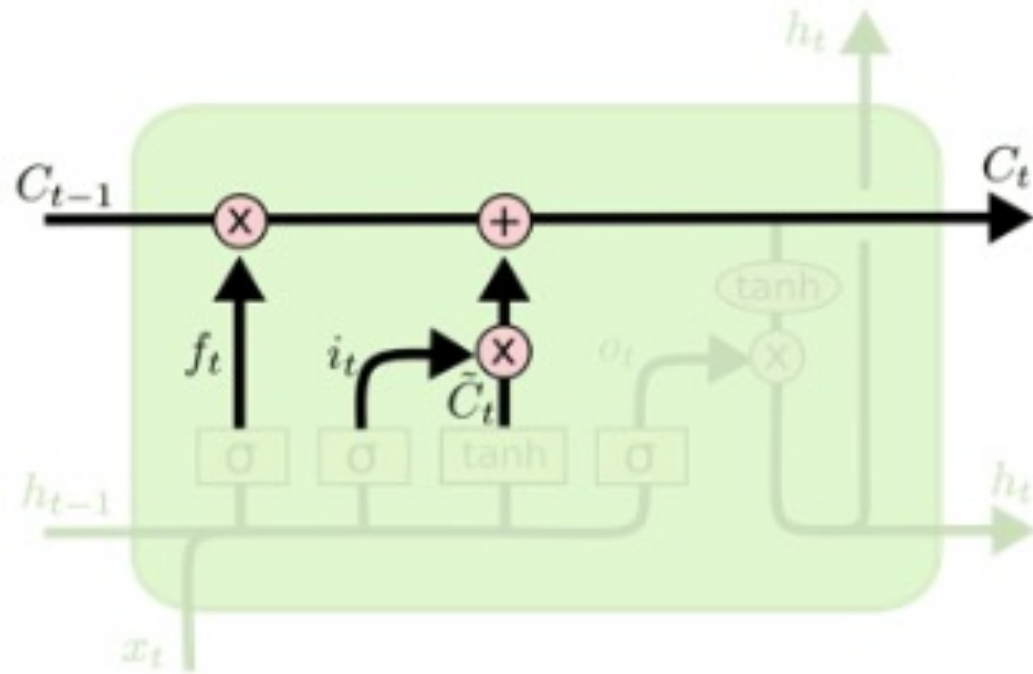


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

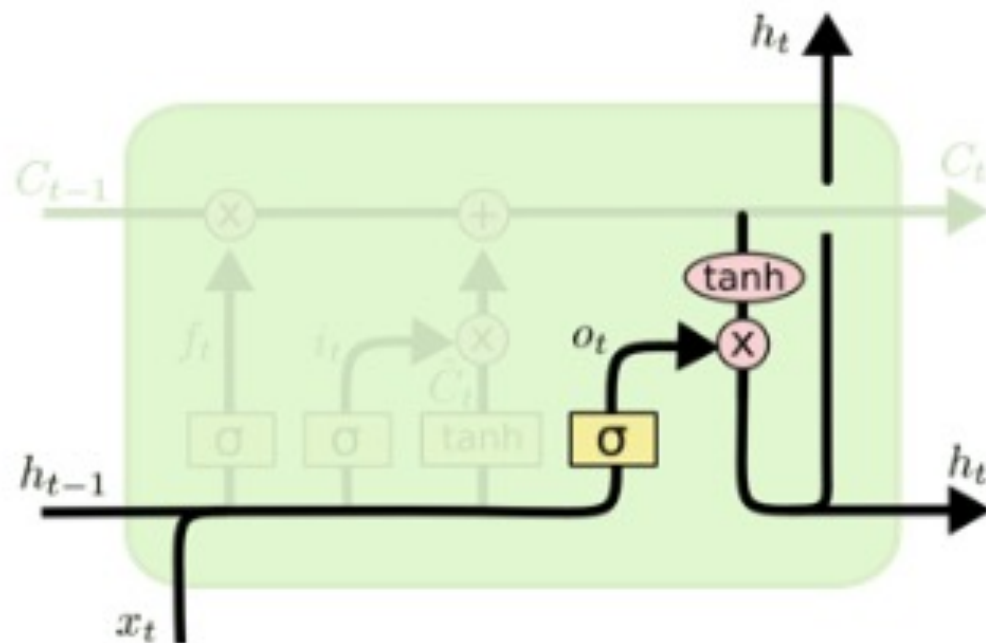
Пример – встретили объект нового пола, выделяем пол в векторе и заменяем на новый

LSTM



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

LSTM



А что на выходе?

- Сигмоида чтобы понять, какую часть cell state мы хотим получить на выходе

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

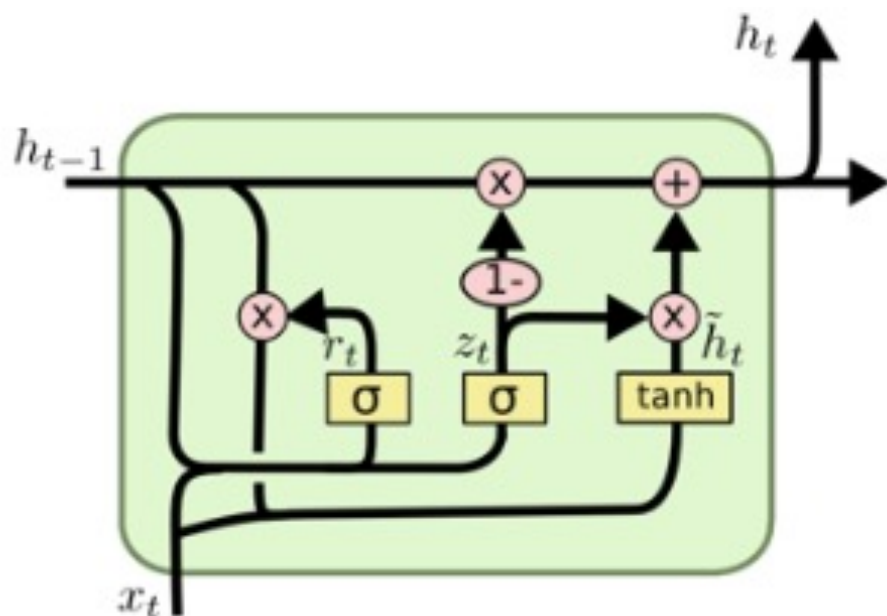
$$h_t = o_t * \tanh (C_t)$$

Пример – увидели объект, хотим вывести информацию, относящуюся к глаголу, на случай, если это будет нужно дальше.

Например, можно выводить существительное в единственном или множественном числе, чтобы мы знали, в какую форму следует спрягать глагол в будущем

GRU

- Объединим forget и input гейты в один update gate
- Соединим cell state и hidden state



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

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

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

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